Draft date: February 2, 2024

Stochastic Simulation and Monte Carlo Methods

Pierre L'Ecuyer

Département d'Informatique et de Recherche Opérationnelle
Université de Montréal

This is an unfinished draft of a book, currently used for the course IFT6561.
This book is free. You can use it and cite it as you wish.

The orange text is temporary; it will be hidden in the final version.

Table of Contents

Preface

The real preface is not yet written. Meanwhile, I collect here some comments and remarks that could be useful.

General Comments

This book started as class notes for the graduate class "IFT6561, Simulation: Aspects Stochastiques", at Université de Montréal, around 1992. The first versions were partly inspired by Bratley, Fox, and Schrage (1987) and Law and Kelton (1991). A draft of the book in current format was distributed to the students already in 2003, and made available online privately every year after that.

The results of the numerical examples were obtained by using the *Stochastic Simulation in Java* (SSJ) library (L'Ecuyer, Meliani, and Vaucher 2002, L'Ecuyer and Buist 2005, L'Ecuyer 2023). The Java programs used for this can be found on the GitHub site of SSJ (L'Ecuyer 2023). Several examples are related to computational finance, because master students from that field often formed the majority in the IFT6561 class.

Chapter 1 gives an introduction to stochastic simulation. Its main goal is to provide an intuitive understanding of the key ideas and principles via simple examples and numerical illustrations. It aims to give the reader a proper *insight* into the most elementary and important concepts, and develop *motivation* for the further chapters by pointing out various types of difficulties that can arise and techniques that can be used to address these difficulties. Detailed analysis of these techniques, advanced methods, theory, and special cases, are left out for the other chapters. This chapter can serve for a short introductory course just by itself.

Appendix A offers a condensed review of key topics and results in probability, statistics, and stochastic modeling, that can be useful to understand the material of the book. It gives short elementary introductions to Markov chains and queuing theory, in particular.

Notation

What follows is a partial list of the notation, abbreviations, and symbols often used in the book. As a general rule, we use uppercase letters for random variables and lowercase for the values they take (their realizations). Vectors and matrices are usually in boldface. By default, all vectors are column vectors. [1]

$\mathbf{0}$	A vector whose coordinates are all 0.
$\mathbf{1}$	A vector whose coordinates are all 1.
$\lceil x \rceil$	Ceiling of x; i.e., smallest integer $\geq x$.
$\lfloor x \rfloor$	Floor of x; i.e., largest integer $\leq x$.
$(0,1)^t$	The t-dimensional unit hypercube.
A_i	Often used for time between arrivals of customers i and $i+1$.
A_n^2	Anderson-Darling statistic.
AR(p)	Autoregressive process of order p.
$B(\cdot)$	Used to denote a standard Brownian motion.
C_i	Cost (or reward) incurred at the ith event epoch.
cdf	Cumulative distribution function.
CLT	Central limit theorem.
CMC	Conditional Monte Carlo.
Corr	Linear correlation.
Cov	Covariance.
CRN	Common random numbers.
$C(X)$	Can be the expected computing cost for the estimator X.
CTMC	Continuous-time Markov chain.
CV	Control variate.
D_n	Kolmogorov-Smirnov statistic.
$D_n(P_n)$	The extreme discrepancy.

[1] From Pierre: This list needs to be updated.

$D_n^*(P_n)$	The star discrepancy.
$D_n^{*(p)}(P_n)$	The $\mathcal{L}_p$-star discrepancy.
det	Determinant.
DTMC	Discrete-time Markov chain.
$\mathbb{E}$	Mathematical expectation.
e_i	The ith event of the simulation.
$\boldsymbol{e}_i$	The ith unit vector, with a 1 in position i and zeros elsewhere.
E_n	Integration error.
Eff	Efficiency (of an estimator).
f	Used to denote a density or the integrand in MC integration.
F	Cumulative distribution function (cdf).
F^{-1}	Inverse cdf.
$\hat{F}_n$	Empirical cdf.
$\mathbb{F}_2$	The finite field with two elements, 0 and 1, in which all operations are performed modulo 2.
$\mathcal{F}_i$	The σ-field generated by the history up to time t_i.
FCFS	First-come first-served.
FIFO	First-in first-out.
GFSR	Generalized feedback shift register.
$GI/GI/1$	A single-server queue with independent interarrival times and service times, with general distributions.
HRMS	Highly-reliable Markovian system.
i	Often denotes the number of a replication or the index of a point for QMC methods. May also be an event number in discrete-event simulation.
i.i.d.	Independent and identically distributed.
$\mathbb{I}$	The indicator function: $I(x) = 1$ when x is true; $I(x) = 0$ otherwise. Sometimes denote the interval $[0, 1]$ in random number generation.
IS	Importance sampling.
j	Often denotes a regenerative cycle or the number of a replication.
L	Used to denote the likelihood ratio.
ln	Natural logarithm.
LCG	Linear congruential generator.
LFSR	Linear feedback shift register.
LHS	Latin hypercube sampling.
LRS	Linear recurring sequence.

MC	Monte Carlo.
MCMC	Markov chain Monte Carlo.
mgf	Moment generating function.
$M/M/1$	A single-server FIFO queue with independent exponential interarrival times and service times.
$M/M/s$	An s-server FIFO queue with independent exponential interarrival times and service times.
MRG	Multiple recursive generator.
MSE	Mean square error.
n	Number of simulation runs (sample size) or number of points in QMC point set.
$N(t)$	Number of events that have occurred by time t.
$N_c(t)$	Number of customers having started their service by time t.
$N(0,1)$	The standard normal distribution.
$N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate normal with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
NORTA	Normal to anything. Models a multivariate distribution using a normal copula.
$o(f(x))$	We write $g(x) = o(f(x))$ to mean that $g(x)/f(x) \to 0$ when $x \to \infty$, or when $x \to 0$, depending on the context.
$O(f(x))$	The set of all functions g such that there is a constant K for which $\|g(x)\| \le K\|f(x)\|$ for all x sufficiently large (when we are interested in $x \to \infty$), or sufficiently small (when we are interested in $x \to 0$). For historical reasons, the standard practice is to write $g(x) = O(f(x))$ instead of $g(x) \in O(f(x))$ to mean that g belongs to $O(f(x))$.
$\underline{O}(f(x))$	We write $g(x) = \underline{O}(f(x))$ to mean that there is a constant K for which $\|g(x)\| \ge K\|f(x)\|$ for all x sufficiently large (when we are interested in $x \to \infty$), or sufficiently small (when we are interested in $x \to 0$).
$\mathbb{P}$	Probability.
P_n	A point set of cardinality n, for QMC methods.
$P_n(I)$	The projection of P_n on the lower-dimensional subspace determined by the coordinates that belong to I.
P-P	Probability-probability (plot).
pdf	Probability density function.
pmf	Probability mass function.
$Q(t)$	The length of a queue at time t.
$\bar{Q}_T$	Average queue length from time 0 to time T.
Q-Q	Quantile-quantile (plot).
QMC	Quasi-Monte Carlo.
$\mathbb{R}$	The real numbers.

$\boldsymbol{R}$	Rank correlation matrix.
RE	Relative error.
RNG	Random number generator.
RQMC	Randomized quasi-Monte Carlo.
r.v.	Random variable.
S_n^2	Sample variance of n observations (usually $X_1, \ldots X_n$).
S_i	Service time of customer i.
$\mathbb{S}$	State space of the simulation model.
$\mathcal{S}_i$	State of the simulation model at time t_i.
SA	Stochastic approximation.
SA	Sample average approximation.
SSJ	Stochastic simulation in Java.
σ^2	The (theoretical) variance.
t	Transposition operator, for vectors and matrices.
t	Often denotes the dimension of an integral.
t_i	Epoch of occurrence of the i-th event e_i.
T	Simulation time horizon.
U, U_j	Often denotes a uniform random variable over $(0, 1)$.
$\boldsymbol{U}$, $\boldsymbol{U}_j$	Usually denotes a random vector uniformly distributed over $(0, 1)^t$.
u_n	The nth output value of a uniform RNG.
$U(0, 1)$	The uniform distribution over $(0, 1)$.
V_N	Cumulative cost for the first N events.
$V_{\rho,N}$	Cumulated discounted cost for the first N events, with discount rate ρ.
V_ρ^∞	Total discounted cost over an infinite horizon.
$\bar{v}$	Average cost per unit of time over an infinite horizon.
Var	The variance.
vol	The volume.
VRF	Variance reduction factor.
VRT	Variance reduction technique.
X	Usually denotes a random variable, often the output of a simulation.
X_i	A copy of X.
$\bar{X}_n$	Sample average of $X_1, \ldots X_n$.
W_i	Waiting time (in queue) for customer i.
$\bar{W}_n$	Average waiting time of the first n customers.

w.p.1	With probability 1.
Z, Z_j	Often denotes a standard normal random variable.
β	The bias of an estimator.
$\Gamma(\cdot)$	The gamma function; see Eq. (5.35).
$\lambda(t)$	Rate (or intensity) of a Poisson process.
$\Lambda(t)$	Cumulative rate of a Poisson process.
$\Phi(\cdot)$	Used to denote the standard normal distribution function.
Ψ_t	The set of all t-dimensional vectors of successive output values of a RNG.
Ψ_I	The set of all vectors of output values of a RNG, for lacunary index set I.
μ	Usually denotes a theoretical mean (expectation) to be estimated by simulation.
ω	A sample point (an element of Ω); represents the underlying randomness.
Ω	Sample space.
$\rho(X, Y)$	Linear correlation between X and Y.
$\rho_s(X, Y)$	Rank (or Spearman's) correlation between X and Y.
$\Theta(f(x))$	We write $g(x) = \Theta(f(x))$ to mean that we have both $g(x) = O(f(x))$ and $g(x) = \underline{O}(f(x))$.

1. Introduction

1.1 Systems, Models, and Simulation

1.1.1 Modeling and simulation

In this book, *simulation* means running a computer program to imitate (simulate) the behavior of a system. Simulation permits one to try out various kinds of actions and even make bad decisions without paying the real price for one's mistakes. Think of flight simulators and computer games. Simulation has thousands of serious applications, for planning and decision making in manufacturing, material handling systems, computer and communication systems, logistics, transportation, health care, military operations, security, economics and finance, and so on. It is used in sciences (physics, chemistry, biology, medicine, etc.) to better understand how things behave, in computer graphics to generate images and movies by simulating light paths, and we could go on. In most applications, simulation involves random sampling from carefully selected probability distributions.

What we actually simulate on a computer is a *model* of the system of interest, which is a simplified abstract (conceptual, mathematical) representation of something that may or may not exist. Its purpose is to better understand the behavior of the system that it represents, often to improve its design or control or to make external decisions about it. Models are used when experimenting with the system itself is impossible (it may not even exist), or too costly, or just inconvenient. Simulation means reproducing and observing the behavior of the model by running a computer program that behaves like the model.

Generally speaking, *conceptual* or *mathematical* models are abstract constructions that may exist only in the mind. Mathematical models used for simulation must be precisely defined, by a set of logical relationships, mathematical equations, and probability laws used to represent the randomness (uncertainty) in the system. All mathematical models simplify reality. In most cases, the model is better defined than the real system. It must be defined well enough so that all its relevant behavior and properties can be determined in a practical way: analytically, numerically, or by simulation, i.e., by running the model on a computer with certain (typically random) inputs and observing the output. A good abstract model often greatly improves our understanding of the system's behavior.

Simulation can also be performed with *physical models*, which are scaled-down versions of the system (e.g., miniature representations). Examples include an airplane wing in a wind-tunnel, artificial waves produced in a swimming pool to test the effect of a storm on different shapes of breakwaters, toy dinosaurs used in movie production, or a miniature representation of a city block. In the first example, the model may be used to obtain relatively

precise measures of performance expressed as numerical values, whereas in the last one, the goal could be to evaluate (visually) how well a new building to be constructed would fit its environment. In recent decades, physical models in practically all areas have been replaced by mathematical models simulated on computers. We will not discuss them any further in this book.

Simulation is also used for other purposes than imitating the behavior of systems by simulating models. For example, it is a very powerful tool to estimate high-dimensional integrals, to optimize complicated functions, and to solve large systems of equations, by using random sampling, as we shall see later. Using random sampling in such settings is often called *Monte Carlo* simulation. This is widely used in engineering, numerical analysis, computational statistics, machine learning, operations research, and many other areas.

1.1.2 Types of mathematical models

Figure 1.1 crudely classifies mathematical models in three types. The left side corresponds to simplification and the right side represents better realism.

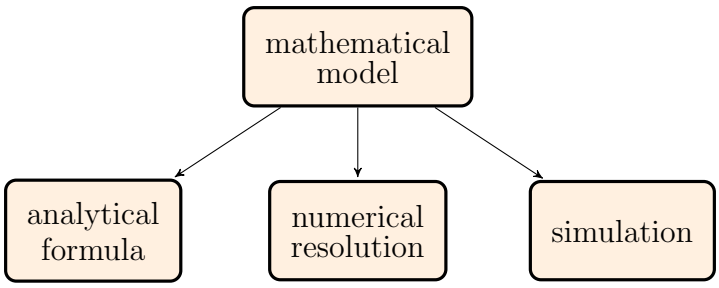


Fig. 1.1. Classes of models

The simplest and easiest-to-handle models lend themselves to *analytic solutions*, i.e., the performance measure of interest can be quickly evaluated by a simple mathematical formula.

Example 1.1 Suppose you know the position, angle, and speed of the shot when it leaves the hand of your favorite Olympic shot-putter. Then you can compute the distance of the throw pretty accurately without having to simulate, by using classical ballistic formulas. Assuming that wind and air resistance are negligible, that the shot is launched at angle θ from the horizontal, speed v (in meters per second), and at x_0 meters ahead and y_0 meters above the starting point of measurement on the ground (the front of the throwing circle), then the distance in meters is given by the analytic (balistic) formula: ♣ Picture to be added.

$$d = x_0 + \frac{v\cos\theta}{g}\left(v\sin\theta + \sqrt{(v\sin\theta)^2 + 2gy_0}\right), \tag{1.1}$$

where $g \approx 9.8$ meters per squared second ($\mathrm{m/s^2}$) is the downward vertical acceleration due to gravity. We will not prove or derive this formula here. It has been derived using the laws of classical mechanics, by assuming that the horizontal speed is constant, the vertical speed

changes at constant rate $-g$, and the measurement is made at the point where the center of the shot hits the ground. In practice, the measurement is made at the closest mark left on the ground by the shot, and we could make an adjustment to the formula for this. Computing d with this formula is very easy once we know (x_0, y_0, v, θ).

If we replace the shot by a javelin or a discus or a football, the effect of the wind, object orientation, and rotation are no longer negligible, so the simple balistic formula (1.1) is no longer sufficient. □

Analytical formulas are available for certain (simple) inventory models, reliability models, queuing models, and so on (see, e.g., Hillier and Lieberman 2021, Wolff 1989). *Using* an available formula is much easier than developing and programming a simulation model. *Deriving* an analytical formula, on the other hand, is generally more difficult, involves higher mathematical sophistication, and often requires making strong and unrealistic assumptions on the behavior of the real-life system.

In other situations, the relevant quantities can be computed or closely approximated by *deterministic numerical algorithms* such as those used in linear, nonlinear, and dynamic programming, differential-equation solvers, and so on. Solving ordinary or partial differential equations by deterministic numerical methods such as finite differences or finite elements is often called *simulation* in the literature. In this book, we focus on situations where the model or the solution method (or both) involves uncertainty and random sampling. The class of techniques that rely on random sampling for solving problems are also known (collectively) as the *Monte Carlo method* (Fishman 1996, Hammersley and Handscomb 1964, Liu 2001, Rubinstein 1981, Sobol' 1994, and Section 1.4 below). This name was coined by physicists working on nuclear weapons at Los Alamos (USA) during World War II, at a time when the prime spot for games of chance was Monte Carlo, on the French *Côte d'Azur*.

As a model gets too complicated, analytical or numerical solutions become impossible to obtain, and one must rely on simulation. More realistic models are usually be more complicated. Simulation models typically require fewer simplifying assumptions, and hence tend to be more credible because they have the flexibility to better capture the real system. Simulation is often easier to understand and justify, for the (non-specialist) managers and customers, than analytical formulas or mathematical programming algorithms. On the other hand, a simple analytic model may give much more insight into what is most important, e.g., could make you understand why policy A is better than policy B. Simulation can also be used to *validate* a simple analytical model. If the simple formula and the (more detailed and realistic) simulation give similar results, one will be more willing to rely on the approximation provided by the simple formula.

Example 1.2 Suppose we want to study a service system that operates from 8:00 to 18:00 (10 hours in total). Customers arrive at random, one by one, and their arrival rate may depend on the time of the day. In our model, we decide to assume that the customers have independent (random) service times, whose probability distribution can be estimated and is the same for all customers. There are s servers and the customers are served by order of arrival. We may be interested in the average waiting time in the queue per customer, or the fraction of customers whose waiting time exceeds a given number, or perhaps the entire distribution of the customer's waiting times, in the long run. This could model the queue for

tellers in a bank, or at an ice-cream shop, or the waiting times of calls in a telephone call center, for example.

We could develop a detailed model of this situation using estimates of all the probability distributions involved, which are possibly time-dependent, and then simulate the model for a large number of days to estimate the performance measures of interest with sufficient accuracy. If we wish, we could also simulate customer abandonments, servers taking a break for a few minutes, and other fine details.

An alternative widely-used by managers is to partition the day into p time periods of equal length, often 15 or 30 minutes each, and assume that over each such period we have an $M/M/s$ queueing system in steady-state, as defined in Section A.19. For this type of model, we assume that over each period $j = 1, \ldots, p$, the arrivals times follow a stationary Poisson process with rate λ_j customers per minute (i.e., the times between successive arrivals are independent and exponential with mean $1/\lambda_j$), the service times are independent and exponential with rate μ_j (i.e., with mean $1/\mu_j$), there are $s = s_j > \lambda_j/\mu_j$ identical servers, and the system has been evolving for an infinite amount of time with these parameters. This model neglects the boundary effects across periods (the fact that the queue is more likely to be smaller at the beginning of a period if the previous period had a smaller arrival rate, for example). Obviously, there is no chance that all these assumptions are exactly true; the simplified model is used only as an approximation for which performance measures of interest can be computed exactly and easily for each period, using the $M/M/s$ queueing formulas.

Under the $M/M/s$ model for period j, the waiting time $W^{(j)}$ of a "random" customer in that period is a random variable whose exact probability distribution is given by Equations (A.12) and (A.14) of the appendix. The probability $\gamma_j(x) = \mathbb{P}[W^{(j)} > x]$ under this model can be computed by these formulas. A customer selected at random over a long period (a large number of days) will arrive in period j with probability $p_j = \lambda_j/(\lambda_1 + \cdots + \lambda_p)$, and then the probability that its waiting time exceeds x can be computed by the weighted average

$$\gamma(x) = \sum_{j=1}^{p} p_j \gamma_j(x).$$

With the $M/M/s$ approximation, the quantities of interest are much faster to compute than with simulation. This advantage could be particularly significant if we want to evaluate several configurations of the system, for example to optimize the value of s_j in each period. On the other hand, the $M/M/s$ approximation might be too unrealistic for our needs. To assess this, we may first validate the approximation by comparing the values it gives with those observed in the real system or those obtained by simulation, to see if we can use the less expensive approximation with confidence. □

A mathematical model is called *stochastic* if its specification involves randomness, and *deterministic* otherwise. For example, linear programming or ordinary differential equation models are deterministic, while queuing or reliability models are stochastic because the arrival times and service times in the queuing model and the lifetimes of components in the reliability model are usually random. A simulation model of ambulances is a city must be stochastic, because the times and physical origins (addresses) of ambulance calls and the travel times of the ambulances are random. Inventory and supply chain models are stochastic as soon as

demands or delivery times are random. Simulating stochastic models involve the generation of random variables on the computer and can benefit from special tools for proper statistical analysis of the results and for reducing the variance of statistical estimators. Models may also be classified as static or dynamic. In a *static* model, the time plays no significant role. For example, we might want to estimate the probability that all the nodes in a given network are connected, given that we know the probability, for any pair of nodes, that there is a direct link between these nodes, and that these direct links are independent. *Dynamic* models are those that evolve in time, and for which we are interested in quantities that depend on this evolution. Examples include queuing networks, inventory models, and so on. Most models examined in this book are dynamic and stochastic.

The time-evolution (or space-evolution, in some cases) of a model is sometimes described by a set of differential equations. The state then changes continuously with time (or space), and simulating this process is called *continuous simulation*. In other cases, as far as the logical relationships and equations are concerned, everything happens only at discrete points in time, which are not necessarily evenly spaced, and are typically random. These points are called the *event times*, and what happens at any of these time points is called an *event*. Such models are called *discrete-event* models; examples will be examined in Sections **??** and 1.12, for instance. When the event times are deterministic and evenly spaced (e.g., the ith event occurs at time $t_i = i$), we have a *discrete-time* model. There also exists *hybrid* models, for which the state may change abruptly at certain event times and would evolve continuously between these events according to differential equations (either deterministic or stochastic).

Discrete-event stochastic simulation is widely used to study the performance on complex systems encountered in manufacturing, logistics, communication, transportation, health-care, economics, finance, etc. It can be used for example to compare the performances of different configurations of a factory, or a communication network, or the ambulances in a large city, or each restaurant in a fast-food chain, etc. It can be used to estimate the probability of a delay that exceeds a given threshold in a construction project, or for a given train in a large rail network, or for an airline flight or a delivery truck, etc. More examples are given in the rest of the book.

1.1.3 Advantages and disadvantages of simulation

Playing with a mathematical model has many advantages over experimenting with the real system:

- It is non-destructive: One can make mistakes for free. One can test different layouts, designs, operating policies, schedules, etc., to answer "what-if" questions without committing real resources or causing costly damage.

- The system of interest does not have to exist: It may be totally virtual (as in computer games) or planned to be built in the future. A material handling system, for example, can be tested by simulation in the context where it is going to operate, before being built or installed.

- Similar experiments (e.g., with the same inputs but slightly different policies, or with either the same or different realizations of the underlying random variables in the case

of a stochastic simulation) can be repeated at will, under the same conditions. This can be used to improve accuracy when estimating the differences between the performances of two or more systems, and for optimization via simulation.

- A simulation model can often be run at a much faster rate than the real system. For example, two years of operation of a manufacturing plant may be simulated in a few minutes on a computer. It can also be the opposite. For example, a realistic simulation of a human hearth for one second can take hours of computing with millions of processors (Fedele et al. 2023).

- Simulation can be coupled with graphical animation, so one can visualize how the system evolves, either in accelerated mode or in slow motion. This permits one to better understand the general behavior of the system, or observe special phenomena in greater details, and gain insight into how things actually work.

- Specially designed simulation tools can be used for teaching and training students, managers, pilots, etc.

Building and running simulation models also offers some challenges:

- The modeling and programming often requires a lot of time and money. The quality and utility of a model depends highly on the skill of the modeler and on the quality and volume of the available data, which is often not what we would like. Building realistic stochastic models and estimating their parameters from available data can be a very difficult task in general, more difficult than programming and running simulations (Macal et al. 2013).

- In their plain forms, stochastic simulation and Monte Carlo methods produces only point estimates for a fixed set of model parameters rather than the "optimal" solution. Developing effective methods for *stochastic optimization by simulation* provides additional challenges. This is typically more difficult than, say, with traditional mathematical programming tools.

1.1.4 Models and programs

Nowadays, most simulations are performed on *computers*; the models are implemented as computer programs. One should distinguish between the real system, the model, and the simulation program (also called computerized model; see Figure 1.2). The model is only a conceptual (abstract) mathematical construct which, in a sense, is itself *simulated* on the computer by the simulation program. It is important to make the distinction because the model must be precisely defined "separately" from the program, in the sense that the abstract model still exists regardless of whether it is implemented in a simulation program.

In some cases, the program may only approximate the model. For example, if some inventory model specifies that the size of the demand for a given day is a Poisson random variable with mean λ, then the simulation program will determine the demand size using some (deterministic) algorithm that produces numbers that "look like" random variates from the Poisson distribution (see Chapter 4). As another example, a model may be defined over

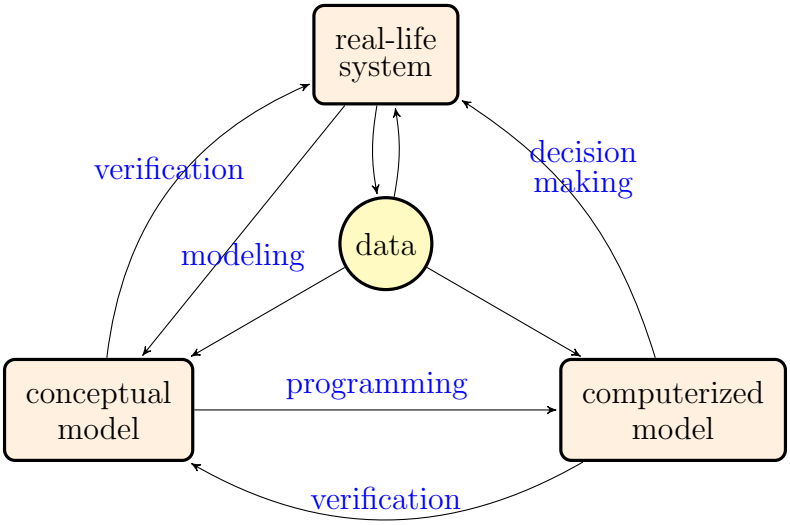


Fig. 1.2. Simulation models and programs

an infinite time horizon, and the performance measure of interest defined as a long-term (steady-state) average, whereas the program can only run over a finite horizon.

Building a mathematical model of a real-life system is called *modeling*. It involves abstraction, simplification, approximation, assumptions, and the like. Writing a simulation program from the model is *programming*. Making sure that the program is correct (i.e., really corresponds to the model) is called *verification* and has to do with software engineering. Checking whether the model is a realistic representation of the real-life system is called *validation*. A correct program for a valid model should behave in essentially the same manner as the real-life system *for the purpose of the intended application of the simulation model* when both are fed with the same inputs.

In general, proper modeling and validation is much more difficult than programming and verification. Modeling is an art for which there are guidelines and principles, but no set of universal procedures that can be followed systematically. Part of it must be learned by practice and experience. The notion of validity is *relative*, because a model will never correspond exactly to the real-life system. Details must be neglected, often on a priori grounds, to keep the model tractable, even when there is no rigorous proof that such neglect is justified. As a result, there is no such thing as an "absolutely valid" model. One must be pragmatic: a model is considered valid if it is *accurate enough for its target application*. So, validity depends on the model's purpose. A model valid for studying a given property of a certain system might no longer be valid to estimate a different property of the same system later on. For a given performance measure of interest, the difference between the value of the performance measure for the model and that for the real system is the *modeling error*. It depends of course on the performance measure that we consider.

Making a model more realistic is costly. Validation is also costly by itself. There should be a compromise between modeling and validation costs on the one hand, and costs due to the consequences of an invalid (or not realistic enough) model on the other hand, i.e., the

costs of modeling error. It is pointless to spend more than the cost of modeling error to correct for the modeling error.

Useful simulation models and programs tend to be often modified and expanded. Both the modeling activity and the results from the simulation program reveal new information along the way, so the modeling process has to be incremental and flexible. Often, certain aspects of the system (probability laws, operating procedures, etc.) change with the passing of time (sometimes because of decisions made as a consequence of the simulation study itself) and the model must be regularly updated. For instance, one may decide to have a more detailed model of a certain subsystem. Pritsker (1998) discusses these issues.

Example 1.3 As an illustration, suppose one wishes to build a model to simulate the operations of a large airport over a period of one year. Clearly, one cannot model *everything* that happens in the airport: the model would be too large, too costly to develop and run, and the required detailed data would most likely not be available anyway. The aspects that are important to model faithfully would depend on the model purpose. They would not be the same if, for example, the model is to be used by an airline carrier to study the distribution of waiting times at registration desks, or by border control services to determine their staffing as a function of time, or by the airport management to study the delays in takeoffs and landings. □

Examples of simplified models will be given in this book; many others can be found in the Proceedings of the annual *Winter Simulation Conference*.

1.1.5 Monte Carlo methods

Monte Carlo methods refer to a general family of techniques that involve random sampling for solving a problem. It is not only used to imitate (simulate) the behavior of a given system. These techniques have a wide range of applications where there is no system to be simulated in the original problem formulation, but a purely artificial stochastic model is constructed whose simulation permits one to solve (approximately) a problem of interest. They are heavily used for example in computer graphics, machine learning, computational statistics, computational biology, computational physics, finance, and many other areas.

The plain vanilla Monte Carlo methods are to estimate integrals, expressed as mathematical expectations (Section 1.4), which may or may not correspond to the average of a performance measure in some system of interest. The method is also used to solve linear systems of equations and partial differential equations, to estimate the cardinality of large sets of combinatorial objects (counting), to optimize a function with respect to certain parameters (see Section 1.15), and to estimate parameters in Bayesian statistics, for example (Asmussen and Glynn 2007, Kroese, Taimre, and Botev 2011, Liu 2001, Robert and Casella 2004).

In optimization applications of the Monte Carlo method, there are situations where the objective function to be optimized can be evaluated exactly at any point, and the evaluation points are selected by Monte Carlo sampling. In other situations, only noisy evaluations of the function and/or of some constraints are available. Sometimes, these evaluations are obtained themselves by Monte Carlo. Often, the function is hard to optimize because it has

a multitude of local optima and we inject randomness in the optimization algorithm to be able to visit (at random) a larger and better selection of local optima than with our best deterministic algorithm.

An important type of approach having applications in several areas is *Markov chain Monte Carlo (MCMC)* (Asmussen and Glynn 2007, Chib 2004, Robert and Casella 2004, Häggström 2002). It applies to situations where we need to generate samples from a probability measure known only up to a multiplicative constant, and for which we do not know how to generate samples directly. The idea of MCMC is to construct an artificial ergodic Markov chain whose steady-state distribution is the probability distribution of interest, and to simulate that chain long enough to get samples having approximately the desired distribution. The main technique for constructing this chain is the Metropolis-Hasting algorithm. See Section 4.9.

1.2 Examples of Simple Simulation Models

Example 1.4 A *stochastic activity network* is a directed acyclic graph $(\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is a set of *nodes* that contains one origin (source) and one destination (sink), and $\mathcal{A}$ is a set of arcs corresponding to *activities*. Each activity $j \in \mathcal{A}$ has a random duration Y_j with cdf $F_j(\cdot)$, so $\mathbb{P}[Y_j \le x] = F_j(x)$ for all x. This duration Y_j can be viewed as the length of arc j. Figure 1.3 gives an illustration with a network of 9 nodes and 13 arcs, taken from Elmaghraby (1977). Such networks are used in PERT-type methodologies for planning and scheduling activities in large complex projects, often with thousands or more activities. They are static stochastic models. The *network (or project) completion time* T is the length of the *longest path* from the source to the sink. This longest path is also called the *critical path*. Its length T is a random variable. In Figure 1.3, there are 6 different paths, whose lengths are

$$\begin{aligned} W_1 &= Y_2 + Y_6 + Y_{11}, \\ W_2 &= Y_1 + Y_3 + Y_6 + Y_{11}, \\ W_3 &= Y_1 + Y_5 + Y_{11}, \\ W_4 &= Y_1 + Y_4 + Y_8 + Y_{10} + Y_{11}, \\ W_5 &= Y_1 + Y_4 + Y_8 + Y_9 + Y_{13}, \\ W_6 &= Y_1 + Y_4 + Y_7 + Y_{12} + Y_{13}, \end{aligned}$$

and we have $T = \max(W_1, \ldots, W_6)$.

For a numerical illustration, we take the same cdf's F_j as in Avramidis and Wilson (1998), Section 4.1. For the activities $j = 1, 2, 4, 11, 12$, we have $Y_j = \max(0, \tilde{Y}_j)$ where $\tilde{Y}_j$ has a normal distribution, $\tilde{Y}_j \sim \mathsf{N}(\theta_j, \sigma_j^2)$ with mean θ_j and standard deviation $\sigma_j = \theta_j/4$. The other Y_j's are exponential with mean θ_j. The values of the mean activity durations $\theta_1, \ldots, \theta_{13}$ are 13.0, 5.5, 7.0, 5.2, 16.5, 14.7, 10.3, 6.0, 4.0, 20.0, 3.2, 3.2, 16.5, respectively.

To get a first crude estimate of the project duration T, one may just replace each Y_j by its expectation θ_j, then compute $W_1, \ldots, W_6$ and their maximum, which is the length T of the longest path. For our network, this gives 48.2, which is the length W_6 of the path that

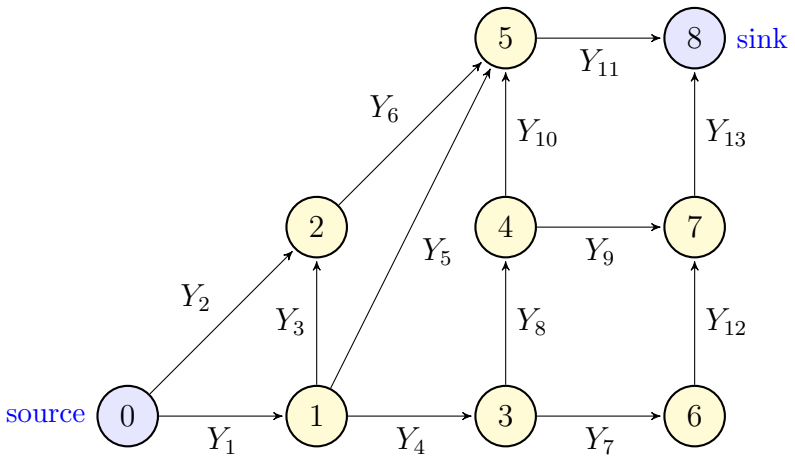


Fig. 1.3. Example of a stochastic activity network, from Elmaghraby (1977)

follows the arcs 1, 4, 7, 12, 13. This simplistic type of approximation is often used in practice, but it is very naive and tells us little about the real behavior of the random variable T. First of all, it generally differs from the true mathematical expectation of T, which turns out to be significantly larger than 48.2, as we will see in a moment. It also says nothing about the likelihood that T takes a value larger than a given threshold x, which could be the true event of interest for instance in case we must pay a large fine if the completion time exceeds a given deadline.

We may be interested in estimating the expected project duration $\mu = \mathbb{E}[T]$, and/or the probability $p_x = \mathbb{P}[T > x] = \mathbb{E}[\mathbb{I}[T > x]]$ that the project completion time exceeds a given time x, for example. (Recall that $\mathbb{I}$ denotes the indicator function.) No efficient numerical method is available for computing these expectations exactly in general, for large complex networks. To estimate the true values of μ and p_x, and perhaps other quantities, we can simulate the model on a computer n times, independently, and take empirical averages. The random variables Y_j are simulated by generating pseudorandom numbers that are transformed to imitate independent realizations of these random variables, as explained in Section 1.3. In our case, we need functions that return normal random variates with a given mean and variance and exponential random variates with a given mean. Once we have generated all the random variates Y_j, we can easily compute the length T of the longest path by taking the maximum over the six paths as shown earlier. For larger networks, there are efficient standard algorithm that can compute T in $\mathcal{O}(m)$ time, where m is the number of activities (Hillier and Lieberman 2021). We repeat this n times, independently, and compute $T_1, \ldots, T_n$, the n realizations of T. The averages

$$\bar{T}_n = \frac{1}{n}\sum_{i=1}^{n} T_i \quad \text{and} \quad \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[T_i > x]$$

are unbiased estimators of μ and p_x, respectively.

We simulated this model $n = 20$ times. The average of the 20 realizations of T was $\bar{T}_n = 65.32$, and 2 values out of 20 (which is 10%) exceeded $x = 90$. How accurate are these

estimates of μ and p_{90}? By repeating the experiment 5 times, we obtained the following values:

average	65.32	67.25	61.47	60.60	59.14
number above 90	2	2	1	3	1

The numbers vary, which illustrates the importance of assessing the accuracy of estimators in stochastic simulation. We also see that all values are largely above 48.2, which suggests that the true mean μ is significantly larger than 48.2. See Exercise 1.5 to understand why.

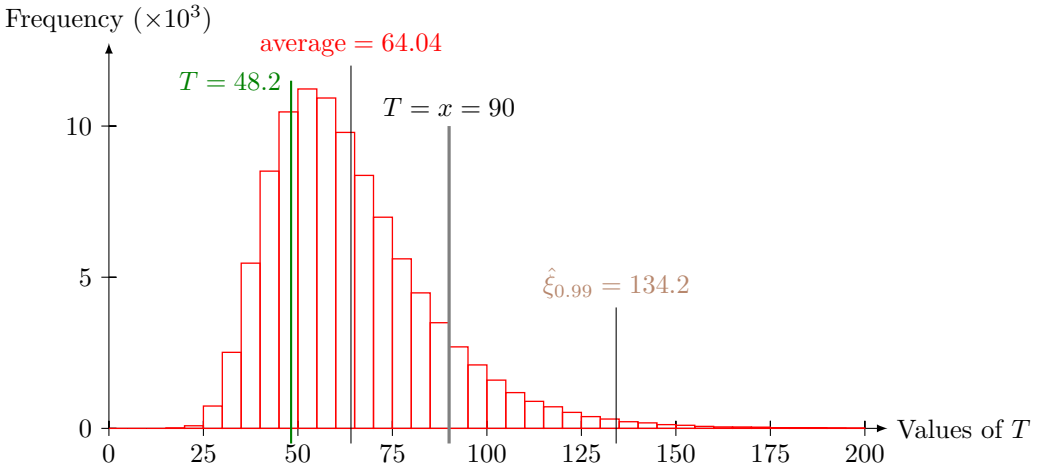


Fig. 1.4. A histogram of the $n = 100,000$ realizations of T in Example 1.4.

To get a better idea of the distribution of T, let us simulate a much larger number n of realizations of T, and look at their empirical distribution. We simulated the model $n = 100,000$ times. Figure 1.4 gives a histogram of the n realizations of T, which ranged from 15.7 to 282.7, with an average of $\bar{T}_n = 64.04$. This average is again significantly larger than the deterministic approximation of 48.2 given earlier. To measure the accuracy of this estimate, we can compute a confidence interval on μ in the standard way as follows. The empirical variance of $T_1, \ldots, T_n$ was $S_n^2 = \frac{1}{n-1}\sum_{i=1}^n (T_i - \bar{T}_n)^2 = 471.6$. Then a 95% confidence interval on μ (see Sections 1.4.3 and A.13) is given by $(64.04 \pm 1.96\, S_n/\sqrt{n}) = (63.90, 64.17)$.

To estimate p_x for any given $x > 0$, we can compute $Y(x) = \sum_{i=1}^n \mathbb{I}[T_i > x]$, the number of realizations for which $T > x$, and estimate $p_x = \mathbb{P}[T > x]$ by $Y(x)/n$. In our experiment, the number of realizations with $T > 90$ was $Y(90) = 11,385$, so our estimate of $\mathbb{P}[T > 90]$ is $Y(90)/n = 0.1138$, or 11.38%. The variance of $Y(x)/n$ is $\mathrm{Var}[\mathbb{I}[T > x]]/n = p_x(1 - p_x)/n$, which admits the unbiased estimator $S_n^2/n \stackrel{\text{def}}{=} (Y(x)/n)(1 - Y(x)/n)/(n-1)$, and a 95% confidence interval on p_x would be $(Y(x)/n \pm 1.96\, S_n/\sqrt{n})$, where S_n depends implicitly on x. In our experiment, for $x = 90$, we have $S_n^2 = 0.1009$ and the confidence interval on p_{90} is then $(0.1138 \pm 1.96\sqrt{0.1009/n}) = (0.1119, 0.1158)$.

We also had 1% of the realizations giving $T > 134.2$, so the latter value provides an estimation of the 99% quantile of the distribution of T. Quantile estimation is discussed further in Example 1.12 and Section 5.7.

Finally, the histogram in the figure gives an estimation of the density of the random variable T. A good estimate of this density gives much more information than all the averages and confidence intervals mentioned so far, since it tells us about the entire distribution of T. Better (more accurate) density estimators than the histogram are examined in Section 2.9.4. In the forthcoming sections and chapters, we will also see how to construct more efficient estimators for μ and p_x.

Other variants of this static simulation model occur in applications. For example, one may be interested in the length T of the *shortest path* in a transportation or communication network with random arc lengths. The indicator $\mathbb{I}[T > x]$ could be replaced by another (general) function of the Y_j's, for which we want to estimate the expectation or perhaps even the entire probability distribution. □

Example 1.5 *Static reliability of a multicomponent system.* The reliability of complex systems is often analyzed in the following framework, which covers reliability networks, reliability diagrams, and fault trees as special cases (Ball, Colbourn, and Provan 1995, Gertsbakh 1989, Gertsbakh and Shpungin 2010, Rubino 1998). This applies to a wide variety of systems, such as communication networks, computer systems in banks, power plants, production systems, aircrafts, military devices, etc. We consider a model in which the system has m components, which can be in state 0 (failed) or 1 (operational). In a *static* model, the time plays no role. This could mean that the system is observed at a fixed point in time, for example, or that components are never repaired and a component is declared "failed" if it fails sometimes during a mission of the system. Let $\boldsymbol{Y} = (Y_1, \ldots, Y_m)^\mathsf{t}$ represent the states of the m components (the vector $\boldsymbol{Y}$ is transposed). A *structure function* $\Phi : \{0,1\}^m \to \{0,1\}$ maps each possible configuration $\boldsymbol{Y}$ to either 0 (the system is failed) or 1 (the system is operational). The *reliability* of the system is defined as $r = \mathbb{P}[\Phi(\boldsymbol{Y}) = 1]$, the probability that the system is operational. If we assume that the components are independent and component j has reliability $r_j = \mathbb{P}[Y_j = 1]$, for all j, then

$$r = \sum_{\boldsymbol{Y} \in \mathcal{U}} \prod_{\{j:Y_j=1\}} r_j \prod_{\{j:Y_j=0\}} (1 - r_j),$$

where $\mathcal{U} = \{\boldsymbol{Y} : \Phi(\boldsymbol{Y}) = 1\}$. Each term in this sum represents the probability of the given $\boldsymbol{Y}$, which is the product of probabilities r_j for the components that are operational ($Y_j = 1$) and the probabilities $1 - r_j$ for the failed components ($Y_j = 0$). In principle, r can be computed directly by this formula. But since there are 2^m different possible values for the vector $\boldsymbol{Y}$, the number of terms in the sum typically increases exponentially in m, which means that computing this expression requires a time that also increases exponentially with m and is therefore impractical in general when m is large. The Monte Carlo method gets around this drawback by sampling a random subset of the terms of the sum instead of enumerating them all, and estimating r from that. Here we assume that $\Phi(\boldsymbol{Y})$ can be computed efficiently for any $\boldsymbol{Y} \in \{0,1\}^m$.

This gives the following simple way of estimating r by Monte Carlo: Generate n independent copies of $\boldsymbol{Y}$, say $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$, compute $X_i = \Phi(\boldsymbol{Y}_i)$ for each i, and estimate r by the average $\bar{X}_n = (X_1 + \cdots + X_n)/n$.

For many applications, the structure function Φ is defined in terms of a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with a set of nodes $\mathcal{N}$ and a set of edges (undirected arcs) $\mathcal{A}$. Each edge j connects two

nodes and can be either operational or failed; edge j corresponds to the jth component. Let $\mathcal{G}(\boldsymbol{Y})$ denote the subgraph of $\mathcal{G}$ that contains only the edges j for which $Y_j = 1$. In some situations, two special nodes s_0 (the source) and s_1 (the destination) are selected in $\mathcal{N}$ and we have $\Phi(\boldsymbol{Y}) = 1$ (the system is operational) if and only if nodes s_0 and s_1 are connected by at least one path in $\mathcal{G}(\boldsymbol{Y})$. This requirement can also be made with directed arcs. More generally, a subset of special nodes $\mathcal{S} \subseteq \mathcal{N}$ is specified a priori and $\Phi(\boldsymbol{Y}) = 1$ if and only if all pairs of nodes in $\mathcal{S}$ are connected by at least one path in $\mathcal{G}(\boldsymbol{Y})$. Sometimes, the paths must be directed, sometimes not. These types of structure functions are easy to evaluate, for any given $\boldsymbol{Y}$, using standard graph algorithms. For example, to check if all nodes in $\mathcal{S}$ are connected, it suffices to construct a *minimal spanning tree* for $\mathcal{S}$ using the edges of $\mathcal{G}(\boldsymbol{Y})$ only, and there are well-known efficient algorithms to do that (Hillier and Lieberman 2021).

In many reliability applications, another important difficulty arises from the fact that r is very close to 1, and one is interested in estimating the *unreliability* $1 - r$, which is the probability of system failure and is very small. For example, if $1 - r = 10^{-8}$, we will observe on average only one failure every 100 million simulation runs. Thus, to estimate the failure probability with reasonable accuracy with standard Monte Carlo, we would need a huge number of simulation runs, and this is not always practical, especially for complex systems where each run requires significant computing time. Fortunately, efficient *rare event simulation* techniques have been developed for this (frequent) type of situation (Juneja and Shahabuddin 2006, Rubino and Tuffin 2009, L'Ecuyer et al. 2011, Botev et al. 2013, Botev, L'Ecuyer, and Tuffin 2016). We will return to this in Sections 1.6 and 6.12.

For a numerical illustration, consider the graph of Figure 1.3 and suppose that the system is operational if and only if there is a directed path going from node 0 to node 8. For example, if links 1, 3, 4, 7, and 10 are failed, the system is still operational, whereas if links 11 and 13 are failed, it is not. We want to estimate the *unreliability*, which is the probability $1-r$ that the system is not operational. Suppose each link j has reliability $r_j = 0.9$, for $1 \le j \le 13$. We simulated this model $n = 10^5$ times and nodes 0 and 8 were disconnected for 4786 of those realizations, giving an unreliability estimate of 0.04786 (approximately 4.8%). Then we made the same experiment with $r_j = 0.998$ for all j and the two nodes were disconnected for 3 of the 10^5 realizations, giving an unreliability estimate of 3×10^{-5}. Finally, we tried $r_j = 0.999$ for all j and the two nodes were never disconnected, giving an unreliability estimate of 0. Needless to say, these last two estimates are inaccurate, because system failure occurs too rarely to obtain a reliable estimator of its probability with our current sample size. The true unreliability in the last case cannot be exactly zero. To get better estimates in the (frequent) situations where the r_j are close to 1, we must either increase n substantially or (preferably) construct more clever estimators. □

Example 1.6 *Approximate counting.* The Monte Carlo method is useful for counting (approximately) the number of elements in a large finite set $\mathcal{S}$, in situations where the set is too large to enumerate its elements explicitly and there is no practical way to compute its exact cardinality. The idea is to find a larger finite set $\mathcal{R}$ that contains $\mathcal{S}$, whose cardinality $|\mathcal{R}|$ is known, and such that an efficient algorithm is available to sample random elements uniformly in $\mathcal{R}$ (i.e., in a way that each element of $\mathcal{R}$ has the same probability $1/|\mathcal{R}|$ of being selected, each time). We assume that it is easy to check if a given element of $\mathcal{R}$ belongs to $\mathcal{S}$ or not. Because $|\mathcal{R}|$ is known, to estimate $|\mathcal{S}|$ it suffices to estimate $p = |\mathcal{S}|/|\mathcal{R}|$ and then

multiply by $|\mathcal{R}|$. For this, we just sample n random elements uniformly and independently from $\mathcal{R}$, count the number Y of those elements that belong to $\mathcal{S}$, and estimate $|\mathcal{S}| = |\mathcal{R}|\,p$ by $|\mathcal{R}|\,Y/n$. Note that Y has the $\mathsf{Binomial}(n, p)$ distribution. We will see in Section 4.4 that the elements that fall in $\mathcal{S}$ also have the uniform distribution over $\mathcal{S}$.

As a simple illustration, consider the set $\mathcal{R}$ of all four-letter words, and suppose we want to know how many of those words contain exactly three different letters (i.e, twice the same letter and two other different letters). This small problem can be solved exactly by combinatoric calculations, but let us ignore that and see how the number can be estimated by Monte Carlo. Here, $\mathcal{R}$ is the set of 26^4 four-letter words and $\mathcal{S}$ is the subset of those words that satisfy the property. To estimate $|\mathcal{S}|$, we sample n words from $\mathcal{R}$ independently, count the number Y of them that have three different letters, and the (unbiased) estimator is $26^4 Y/n$. To generate an element of $\mathcal{R}$, it suffices to generate four random letters independently of each other. This can be generalized to L-letter words for an arbitrary positive integer L, and the property of "exactly three different letters" can be replaced by any other property that is easy to check.

For a less trivial example, consider the set of two-way tables with r rows and c columns. Let $a_1, \ldots, a_r, b_1, \ldots, b_c$ be fixed positive integers such that $a_1 + \cdots + a_r = b_1 + \cdots + b_c$. Let $\mathcal{S}$ be the set of two-way tables with integer entries $e_{i,j} \geq 0$ and for which the sum of entries over row i is a_i and the sum over column j is b_j, for all i and j. A tiny illustration that can be solved easily by hand is given below, for a 2×3 table with row sums $a_1 = 4$, $a_2 = 2$, and column sums $b_1 = b_2 = b_3 = 2$.

$e_{1,1}$	$e_{1,2}$	$e_{1,3}$	$a_1 = 4$
$e_{2,1}$	$e_{2,2}$	$e_{2,3}$	$a_2 = 2$
$b_1 = 2$	$b_2 = 2$	$b_3 = 2$	

Here, $e_{1,1}$ cannot take other values than 0, 1, and 2, because the first column sum is 2. If $e_{1,1} = 0$, we must have $e_{2,1} = 2$, then $e_{2,2} = e_{2,3} = 0$, and then $e_{1,2} = e_{1,3} = 2$. If $e_{1,1} = 1$, then $e_{2,1} = 1$, and either $e_{2,2} = 0 = 1 - e_{2,3}$ or $e_{2,2} = 1 = 1 - e_{2,3}$, and the other values are fixed automatically in each case. If $e_{1,1} = 2$, then $e_{2,1} = 0$ and $e_{2,2}$ has three possible values: 0, 1, or 2. In each case, there is a single choice for the other values. This gives a total of $|\mathcal{S}| = 6$ possible solutions. This case is easy, but for larger tables and larger sums, the number of possibilities explodes very quickly. How can we compute or estimate $|\mathcal{S}|$ then, when it is very large?

Let $\mathcal{R}$ be the set of tables whose sum in row i is a_i for each i. It is easy to generate a random table uniformly from $\mathcal{R}$: For each row i, construct an array with $a_i + c$ entries, with a_i entries equal to 0 and c entries equal to 1. Then generate a random permutation of these $a_i + c$ elements (see Exercise 1.3). Here, the 1's represent separators between the columns, the number $e_{i,1}$ of 0's before the first 1 gives the count in the first column, and the number $e_{i,j}$ of 0's between the $(j-1)$th and jth 1 gives the count in column j. The reader can verify that this gives equal probability to any distinct allocation of the a_i entries to the columns, in row i, because all permutations have the same probability of being selected. Once the table has been generated, it is easy to check the column sums to see if this table belongs to $\mathcal{S}$ or not. One drawback of this simple algorithm, however, is that for large tables, the probability p that the generated table belongs to $\mathcal{S}$ is likely to be extremely small, in which case we will have a rare-event problem and our estimator of $|\mathcal{S}|$ will be very unreliable, similar to the

numerical illustration of Example 1.5. This comes from the fact that the set $\mathcal{R}$ selected here is too large. More efficient sampling schemes for this problem are examined by Blanchet and Rudoy (2009) and later in this book. [1] See Exercise 1.9 for another example. □

Example 1.7 *Estimating volume.* The sampling scheme of Example 1.6 extends to infinite sets and continuous spaces. Suppose we want to estimate the volume $\text{vol}(\mathcal{S})$ of some bounded set $\mathcal{S} \subset \mathbb{R}^d$. We assume that this volume is too difficult or costly to compute exactly, but that for any point $\boldsymbol{y} \in \mathbb{R}^d$ it is easy to verify if $\boldsymbol{y} \in \mathcal{S}$ or not. The idea is to select a larger set $\mathcal{R}$ that contains $\mathcal{S}$ and such that it is easy to compute $K = \text{vol}(\mathcal{R})$ and also to sample points uniformly in $\mathcal{R}$. To estimate $\mu = \text{vol}(\mathcal{S})$, one samples n independent points $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$ uniformly in $\mathcal{R}$, count the fraction of those points that belong to $\mathcal{S}$, and multiply this fraction by $\text{vol}(\mathcal{R})$ to estimate $\text{vol}(\mathcal{S})$. Denoting $B_i = \mathbb{I}[\boldsymbol{Y}_i \in \mathcal{S}]$, this gives the unbiased estimators

$$\hat{B}_n = \frac{1}{n}\sum_{i=1}^{n} B_i$$

for $p = \text{vol}(\mathcal{S})/\text{vol}(\mathcal{R})$ and $K\hat{B}_n$ for μ. The B_i are i.i.d. Bernoulli random variables with parameter p, so $\mathbb{E}[B_i] = \mathbb{E}[B_i^2] = p$,

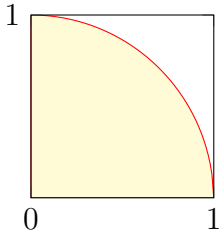
$$\mathbb{E}[K\hat{B}_n] = K\,\mathbb{E}[B_i] = Kp = \mu,$$

and

$$\text{Var}[K\hat{B}_n] = K^2\text{Var}[B_i]/n = K^2 p(1-p)/n = \mu(K-\mu)/n.$$

Section 4.4 will also tell us that the points that fall in $\mathcal{S}$ have the uniform distribution over $\mathcal{S}$, so the method just described provides a simple way to sample independent points uniformly over the set $\mathcal{S}$. It is called the *rejection method*.

A simple (well-known) illustration is the following. Since we know that π is the surface of a disc of radius 1, we can estimate π by estimating this surface via Monte Carlo. This can be done by taking $\mathcal{S}$ as the unit disc centered at $(0, 0)$, $\mathcal{R}$ as the 2×2 square that contains the disc (with the same center), and apply the estimation method just described. To simplify, one can also take $\mathcal{S}$ as the positive quadrant of the disc and $\mathcal{R}$ as the unit square $[0, 1]^2$ that contains this quadrant, as shown below, and apply the same method to estimate $\pi/4$.



Each point (U_1, U_2) is generated in $[0, 1]^2$ by generating two independent random numbers uniformly over $[0, 1]$, and the point belongs to $\mathcal{S}$ if and only if $U_1^2 + U_2^2 \leq 1$. The proportion of the n points that satisfy this condition, multiplied by 4, gives an unbiased estimator of π.

[1] From Pierre: To do. Also give other applications for this.

Moreover, the points that fall in $\mathcal{S}$ are uniformly distributed over $\mathcal{S}$. See also Exercises 1.18 and 1.19. □

Example 1.8 *Distribution of the number of collisions in a hashing scheme. Hashing* is a storage technique allowing fast retrieval of information. To illustrate the principle, suppose we want to store and access information about different types of items in an inventory system, where each item type has a unique identifier (either a name or a number) and there are too many potential identifiers to assign one physical memory location to each possibility and guarantee a fast direct access (with no need for a search). If identifiers are 48-bit numbers, for example, then there are 2^{48} (more than 250 trillions) possibilities. Using a list that contains only the identifiers in use would require searching the list each time we want to access the information on one item, making the access too slow.

A *hashing function* maps each potential identifier to an *address* in $\{0, \ldots, k-1\}$ in a random-looking way, where k represents the number of physical memory locations actually reserved for storage (i.e., the size of the hashing table). Typically, k is much less than the number of possibilities. Each identifier is thus mapped to a specific storage location that can be computed very quickly. But collisions may happen. We say that there is a *collision* whenever a new identifier is mapped to a storage location already occupied by another identifier. There are several ways of handling collisions in hashing tables; see any good book on data structures in computer science.

Suppose that M distinct identifiers are *used*, i.e., there are M types of items in the inventory, where M is a random variable having the Poisson distribution with mean λ_{m}. That is, $\mathbb{P}[M = j] = e^{-\lambda_{\mathrm{m}}}\lambda_{\mathrm{m}}^j/j!$ for $j = 0, 1, 2, \ldots$. The addresses of the M different identifiers are assumed to be independent random variables, uniformly distributed over $\{0, \ldots, k-1\}$. Let C be the number of collisions. Note that $C = M - D$ where D is the number of distinct addresses to which the identifiers are mapped. The following figure illustrates a simple case where $k = 25$, $M = 10$, $C = 3$, and $D = 7$. Each square represents an address and the red points represent collisions.

If k is large and $\lambda_{\mathrm{c}} = \lambda_{\mathrm{m}}^2/(2k)$ is small, it can be shown (see, e.g., L'Ecuyer, Simard, and Wegenkittl 2002) that the random variable C has approximately the Poisson distribution with mean λ_{c}. This also holds if M is fixed to the constant λ_{m} (instead of being random). The precise result says that in both cases, if $k \to \infty$ and $\lambda_{\mathrm{m}}^2/(2k) \to \lambda_{\mathrm{c}}$ simultaneously, then the distribution of C converges to a Poisson distribution with mean λ_{c}. This implies in particular that when k is large and λ_{c} is small, the expected number of collisions is approximately λ_{c}, the probability of no collision is approximately $e^{-\lambda_{\mathrm{c}}}$, the probability of more than c collisions is approximately $1 - \sum_{i=0}^{c} \lambda_{\mathrm{c}}^i e^{-\lambda_{\mathrm{c}}}/i!$, and so on. This is an example of an analytic model.

For example, if the identifiers are 48-bit numbers, there are 2^{48} possibilities. If $\lambda_{\mathrm{m}} = 2^{15}$ identifiers are used on average, and if we take $k = 2^{22}$ physical addresses, then $\lambda_{\mathrm{c}} = 2^{30}/2^{23} = 2^7 = 128$, so we expect around 128 collisions.

However, the Poisson distribution is only an approximation. If k is small, or if we want to assess the quality of the Poisson approximation for moderate values of k, the true distribution of C can be estimated by simulation. For this, we first generate M from the Poisson

Table 1.1. Observed frequency Y_c of each number c of collisions for an experiment with $n = 10^7$ and $k = 10,000$, when M is fixed at $\lambda_m = 400$ (M fixed), when M is Poisson with mean 400 (M Poisson), and expected frequency if C is Poisson with mean 8 (Expected). The expected frequencies in blue are smaller than those predicted by the (simplified) Poisson approximation, whereas those in red are larger.

c	M fixed	M Poisson	Expected
0	3181	4168	3354.6
1	25637	32257	26837.0
2	105622	122674	107348.0
3	288155	316532	286261.4
4	587346	614404	572522.8
5	948381	957951	916036.6
6	1269427	1247447	1221382.1
7	1445871	1397980	1395865.3
8	1434562	1377268	1395865.3
9	1251462	1207289	1240769.1
10	978074	958058	992615.3
11	688806	692416	721902.0
12	442950	459198	481268.0
13	260128	282562	296164.9
14	141467	162531	169237.1
15	71443	86823	90259.7
16	33224	43602	45129.9
17	14739	20827	21237.6
18	5931	9412	9438.9
19	2378	3985	3974.3
20	791	1629	1589.7
21	310	592	605.6
22	79	264	220.2
23	26	83	76.6
24	5	33	25.5
25	5	10	8.2
26	0	2	2.5
≥ 27	0	3	1.0

Table 1.2. Same as in Table 1.1, but with $k = 100$ and $\lambda_{\mathrm{m}} = 40$.

c	M fixed	M Poisson	Expected
0	1148	17631	3354.6
1	14355	100100	26837.0
2	84046	294210	107348.0
3	302620	600700	286261.4
4	744407	951184	572522.8
5	1340719	1238485	916036.6
6	1828860	1379400	1221382.1
7	1941207	1353831	1395865.3
8	1634465	1194915	1395865.3
9	1103416	956651	1240769.1
10	602186	705478	992615.3
11	267542	485353	721902.0
12	97047	309633	481268.0
13	29161	187849	296164.9
14	7195	107189	169237.1
15	1363	58727	90259.8
16	224	30450	45129.9
17	35	15115	21237.6
18	4	7271	9438.9
19	0	3311	3974.3
20	0	1468	1589.7
21	0	607	605.6
22	0	258	220.2
23	0	110	76.6
24	0	44	25.5
≥ 25	0	30	11.7

distribution (or take M equal to its constant value, if M is assumed to be a constant), then we generate M independent random integers, uniformly distributed over $\{0, \ldots, k-1\}$, and let C be the number of times we get an integer already obtained before (the number of collisions). We repeat this simulation n times, independently, and let $C_1, \ldots, C_n$ be the n values of C thus obtained. If n is large, the expected number of collisions, $\mathbb{E}[C]$, can be estimated by the average $\bar{C}_n = (C_1 + \cdots + C_n)/n$, and for each $c \geq 0$, the probability $p_c = \mathbb{P}[C = c]$ that there are exactly c collisions can be estimated by Y_c/n where $Y_c = \sum_{i=1}^{n} \mathbb{I}[C_i = c]$ is the number of times that we have observed exactly c collisions. These Y_c/n provide the empirical distribution of $C_1, \ldots, C_n$, which estimates the entire probability distribution of the random variable C.

For a numerical illustration, we simulated this model $n = 10^7$ times with $k = 10,000$ and $\lambda_{\mathrm{m}} = 400$, both for M constant and M having the Poisson distribution. The 10 million observed realizations of C ranged from 0 to 25 for $M = 400$ and from 0 to 29 when M had the Poisson distribution. Table 1.1 reports the frequency Y_c of each value of c in the experiment, for each case, and also gives $n\lambda_{\mathrm{c}}$, the expected number of occurrences of c under the assumption that C is a Poisson random variable with mean $\lambda_{\mathrm{c}} = \lambda_{\mathrm{m}}^2/(2k) = 8$. We show a table with numbers instead of a plot because the differences are harder to see on a plot. We find that when M is fixed, the values of C from 3 to 9 occur more frequently than expected, whereas all values smaller then 3 or larger than 9 occur less frequently. Thus, C has less variance than a Poisson random variable with mean 8. When M has the Poisson distribution, we observe the opposite: the values of C from 0 to 7 or larger than 18 occur more frequently than expected, while the other values occur slightly less frequently (with a few exceptions due to random noise). The empirical mean and variance are 7.87 and 7.47 for the fixed case, and 7.89 and 8.10 for the Poisson case. These numbers are actually equal to the exact mean and variance up to the given digits, for the respective models. For comparison, the Poisson random variable used for the approximation has mean and variance both equal to 8. The fact that the true variance is larger when M is random makes perfect sense, because taking M random must increase the variance.

For smaller values of k, the approximation error by the Poisson model is likely to be larger. To illustrate this, we made the same experiment with $k = 100$ and $\lambda_{\mathrm{m}} = 40$, which also gives $\lambda_{\mathrm{c}} = \lambda_{\mathrm{m}}^2/(2k) = 8$. The results with $n = 10^7$ are in Table 1.2. The Poisson approximation is definitely worse in this case. For $M = \lambda_{\mathrm{m}} = 40$ (fixed), the empirical mean and variance of the n realizations of C are 6.90 and 4.10. When M has the Poisson distribution with mean 40, they are 7.03 and 8.48. The approximation error is smaller in the latter case, but still significant. In both cases, the Poisson approximation overestimates $\mathbb{E}[C]$. The approximation may nevertheless be good enough if we only need a very rough estimate of $\mathbb{E}[C]$.

In a careful study, we would also compute a confidence interval on the probability of each value of c, for each case, to assess the accuracy of the estimates. □

♣ Add an example of a discrete-time Markov chain

Example 1.9 *Stochastic path tracing in computer graphics.* Image synthesis by computer is a very important activity nowadays. Sequences of high-quality images must be produced when making special effects in movies and in video games (in real time), for exam-

ple. Computer-generated images are also important in architectural design to view buildings, landscapes, and other structures before they are built, to test interior lighting designs, to view the appearance of various types of objects before they are fabricated, and for many other purposes. An image represented as a two-dimensional array of pixels is defined by specifying the color of each pixel. Ideally, this color should be determined by summing up all the light that reaches the eye of the observer (or camera) by passing through the area that corresponds to that pixel. This sum can be written as an integral which in all but extremely simplified configurations is impossible to evaluate exactly, but can be estimated via Monte Carlo by sampling incoming directions at random, and tracing backward to find all (or most) important sources of light coming from that direction. Several refined techniques must be used to make this procedure efficient. The complete details are not simple, but here we sketch the key ideas. See also Dutré, Bala, and Bekaert (2006) and Pharr, Jakob, and Humphreys (2016).

♣ Add picture: draw camera (point) and pixel (square), plus a few objects and lines that go through the pixel.

We assume a static three-dimensional scene in which nothing depends on time, with three-dimensional objects and some light sources. Let x be a point on the surface of an object in the scene. Put a sphere of unit radius centered at x, and consider the unit hemisphere obtained by cutting this sphere in two with the plane tangent to the surface at x and taking the piece that contains $\boldsymbol{n}_x$, the normal vector of unit length pointing outwards of the surface at x. This normal vector ends at the pole of the retained hemisphere. The direction Θ of a ray of light outgoing from point x corresponds to a point on this hemisphere. A surface area on this hemisphere is called a *solid angle*. The *radiance* $L(x \to \Theta)$ going from x in direction Θ is the power (in watts) transmitted in this direction per unit of solid angle and per unit of surface area projected in this direction. The radiance $L(x \leftarrow \Theta)$ coming to x from direction Θ is defined similarly. One can decompose

$$L(x \to \Theta) = L_{\mathrm{e}}(x \to \Theta) + L_{\mathrm{r}}(x \to \Theta) \tag{1.2}$$

where $L_{\mathrm{e}}(x \to \Theta)$ is the radiance emitted from point x (e.g., if x is at the surface of a light source) and $L_{\mathrm{r}}(x \to \Theta)$ is the radiance coming from other sources and surfaces and reflected at x in direction Θ. The *rendering equation* in computer graphics is the combination of Eq. (1.2) with

$$L_{\mathrm{r}}(x \to \Theta) = \int_{\Omega_x} f_{\mathrm{r}}(x, \Psi \to \Theta) L(x \leftarrow \Psi) \cos(\boldsymbol{n}_x, \Psi) \mathrm{d}\omega_\Psi, \tag{1.3}$$

where Ω_x is the unit hemisphere for point x, $\mathrm{d}\omega_\Psi$ represents the solid angle on the hemisphere that corresponds to an infinitesimal neighborhood of the direction Ψ, $f_{\mathrm{r}}(x, \Psi \to \Theta)$ is the *bidirectional reflectance distribution function* (BRDF) which specifies the proportion of the light coming at x from direction Ψ that will be reflected from x in direction Θ, and $\cos(\boldsymbol{n}_x, \Psi)$ is the cosine of the angle between the normal vector $\boldsymbol{n}_x$ and the direction Ψ. This cosine factor accounts for the fact that light that arrives from a direction close to the normal has more impact than light that arrives at a larger angle. The BRDF is the most complicated part and there are many models for it, depending on the properties of the surface on which x lies. For example, a mirror will reflect most of the light from one direction as a narrow beam

in a single direction, whereas a fabric or a piece of wood would diffuse the light in a wider range of directions. Here we have implicitly assumed that all surfaces are opaque, so the light can only be reflected in one hemisphere. For semi-transparent surfaces and materials, on which refraction or transmission can also occur, one can replace the hemisphere by a full sphere. We also did not mention the light frequencies (or colors). Ideally, we would like to have the radiance for each frequency, so $L(x \to \Theta)$ should actually be a function of the frequency, which may be approximated by partitioning the range of visible frequencies into a finite number of intervals and computing the radiance for each interval. A less costly (and more popular) approach is to express the radiance as a three-dimensional vector giving the contribution to each color (red, green, blue) in the RBG system. This model also makes other simplifying assumptions not discussed here (Dutré, Bala, and Bekaert 2006, Pharr, Jakob, and Humphreys 2016).

Solving the integral equation given by (1.2) and (1.3) would (in principle) give the radiance $L(x \to \Theta)$ at all surface points x and in all directions Θ, but this is usually much too difficult. Approximate solutions can sometimes be computed using finite element schemes often called *radiosity methods*. But the best quality images are produced via Monte Carlo sampling combined with variance reduction methods.

In its basic form, the *backward path tracing* Monte Carlo scheme works as follows. A given pixel of the image corresponds to a small rectangular surface. Ideally, the color of the pixel would be determined by averaging over all points p in the pixel, the radiance from p in the direction of the eye, for each frequency (or base color). This average is an integral that is much too difficult to compute, but we can estimate it by Monte Carlo by sampling say n_0 random points p over the pixel rectangle. For any given p, let Θ be the direction from p to the eye, and let x be the first surface point that is encountered when moving from p in the opposite direction $-\Theta$. The radiance $L(p \leftarrow \Theta)$ that hits p in direction Θ is the same as that leaving x in direction Θ, $L(x \to \Theta)$. This $L(x \to \Theta)$ is expressed by the rendering equation, which contains another integral, this time with respect to the incoming direction Ψ to x. Monte Carlo can be applied to estimate this integral as well, for each sample point p, say by sampling n_1 random incoming directions for each. For each of the $n_0 \times n_1$ pairs (x, Ψ), we find the nearest surface point y from x in direction $-\Psi$, estimate the radiance $L(x \leftarrow \Psi) = L(y \to \Psi)$, compute the corresponding integrand in (1.3). For each x (i.e., each p) in the sample, these integrands are averaged over the n_1 realizations of Ψ, and the radiance emitted at x is added, to obtain an estimate of $L(x \to \Theta) = L(p \leftarrow \Theta)$. These values are averaged over the n_0 sampled realizations of p to estimate the average radiance of the pixel. But to do this, we need estimates of the $L(y \to \Psi)$ that appear in the integrands. They are obtained simply by applying the procedure recursively. At level ℓ of the recursion, we estimate each integral by averaging over say n_ℓ samples. The recursion is stopped at some level $\ell_{\max}$ and the remaining terms (beyond that level) are neglected. This corresponds to assuming that light becomes negligible after being reflected $\ell_{\max}$ times. Of course, this introduces bias.

This simple path tracing method is in fact too inefficient, because the number of paths grows exponentially with the number of recursion levels, and it may take many levels for any given path to hit a light source and have a nonzero contribution. As an illustration, if we take $2^8 = 256$ samples for each integral at each recursion level, and we use ℓ levels of recursion, the total number of paths at level ℓ will be $256^{\ell+1}$. For $\ell = 10$, this gives 2^{88} paths,

which is impractical. One may of course decrease the number of sample paths by taking smaller values of n_ℓ (possibly $n_\ell = 1$) for $\ell > 0$. But reducing the n_ℓ values increases the variance while reducing $\ell_{\max}$ increases the bias by removing all contributions that come after several reflections. Effective implementations use various types of modifications to improve efficiency.

Key improvement methods include importance sampling, Russian roulette, quasi-Monte Carlo, bidirectional path tracing, and direct illumination, which we now briefly outline. There are many more. The idea of *importance sampling* is that instead of sampling p uniformly over the pixel area, and sampling each direction Ψ uniformly over the hemisphere, one can sample using a nonuniform density that is larger in areas or directions where the radiance contribution is larger, and multiply the estimator by an appropriate factor to make it unbiased again. For more on importance sampling, see Sections 1.6 and 6.12. Ideally, the density should be proportional to the radiance contribution, but this is impossible to implement exactly, so in practice one uses heuristics to roughly approximate this density. *Russian roulette* gives an effective way of reducing the number of sample paths at higher levels by stopping any given path with a probability that decreases with the anticipated radiance contribution of that path. The contribution of each surviving path is multiplied by a weight larger than 1 to make up for this discarding and recover an unbiased estimator. See Section 6.13 for more on Russian roulette. With *quasi-Monte Carlo*, the rectangular pixels and hemisphere surfaces are sampled more evenly than with independent random points over these surfaces, so the points cover the surfaces more uniformly. See Sections 1.5.3 and 6.10.

When a scene has few and/or small light sources, it may take a huge number of paths started from the eye before a reasonable number of them hit a light source within the $\ell_{\max}$ levels. The paths that do not hit a light source, which may form the great majority, have no contribution to the estimator and are therefore useless in some sense. In *bidirectional path tracing*, paths are also started from the light sources and are combined in some way with the paths started from the eye. One simple example of this is *direct illumination*, which estimates the radiance contribution that comes exclusively from the light sources at level $\ell = 1$, i.e., directly from the source to the first surface visible in the pixel. The contribution of each light source can be estimated as follows. Sample a given number of points on the surface of the light source, and for each of these points, compute the radiance contribution emitted at this point in the direction of the first point visible to the eye on a pixel. The light sources that are not directly visible from a point visible to the eye (at least partially) can be discarded a priori. The number of samples would generally vary across the light sources and depend on the importance of the source, related to its power. Uniform sampling can be used to generate the points on the surface of the source, but it is often worth doing importance sampling. Direct illumination is a very effective way of estimating the level-one contribution to the incoming radiance at each pixel. One may think of applying a similar technique for each level ℓ separately, with different numbers of samples across the different levels (the higher levels may be allocated fewer samples because their contribution is usually less important), and then sum up the estimated contributions. One could also think of optimizing the number of samples at each level to minimize the variance for a given total computing budget. This would be an application of *multilevel Monte Carlo*, discussed in Section **??**. [2] However,

[2] From Pierre: Not yet written.

things are more complicated at levels $\ell > 1$ than at the first level. At level 1, sampling one path reduces to sampling one point on the surface of a light source, but at a higher level ℓ, directions must also be sampled at the intermediate levels in $\{1, \ldots, \ell - 1\}$, so the space of path possibilities is harder to cover. □

Example 1.10 *A tandem queue.* Queueing systems, in which entities have to wait for certain resources to be available, are common in many areas. People wait for their turn at call centers, health clinic, ticket boots, banks, etc., vehicles wait at intersections, parts wait for their turn to be processed at various types of machines in a manufacturing plant, packets of information wait for their turn to be transmitted in communication networks, and so on. Those types of systems are often simulated for performance evaluation and improvement. Here we give an illustration with a particular type of queueing system: a tandem queue with the possibility of blocking.

Figure 1.5 represents a system of m service stations arranged in tandem. Each station is a *single-server queue*; it has one server and a waiting line (queue) of customers, possibly with limited (finite) capacity. Customers arrive randomly to the first station, where they are served one by one in *first come first served (FCFS)* order. Since there is a single server at each station, the customers also exit the stations in *first in first out (FIFO)* order. If the server at the first queue is idle when a customer arrives, the server starts serving this customer immediately, otherwise the customer joins the back of the queue. When the server completes a service at a station, say station j, if the queue at station $j + 1$ is not full, the customer just served at station j joins it and the server at station j starts serving the first customer in queue if there is one. If the queue at $j + 1$ is full, the customer just served is blocked at station j and also blocks server j until a space becomes available at queue $j + 1$. Arrivals at queue j correspond to departures at queue $j - 1$, for $j = 2, \ldots, m$. At station m, customers just leave the system upon completing their service.

One may want to estimate the expected wait time per customer at specific stations, or the expected total waiting time per customer, or the expected total sojourn time in the system, or the percentage of customers that wait more than a given time x in the long run, etc. These quantities can be estimated by simulation.

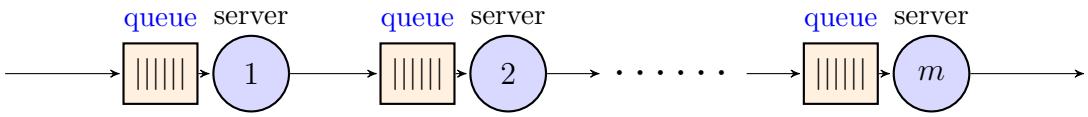


Fig. 1.5. A tandem queue.

Suppose that station j has capacity $c_j \geq 1$ (including the customer in service), for $j = 1, \ldots, m$, with $c_1 = \infty$ (there is no blocking at the first queue). To simplify the model (perhaps on first reading), one may just assume that $c_j = \infty$ for each j and ignore all the terms and sentences that involve blocking, because there is no blocking in that case.

The model evolves as follows. Let $T_0 = 0$. For $i \geq 1$, let T_i be the arrival time of customer i at the first queue and $A_i = T_i - T_{i-1}$ the time between arrivals of customers $i - 1$ and i. For the ith arriving customer, let $W_{j,i}$ be its waiting time in queue j, $S_{j,i}$ its service time

at station j, $B_{j,i}$ its blocked time at server j (if any), and $D_{j,i}$ its departure time from station j. The first customer arrives at time $T_1 = A_1$, leaves the first service station at time $D_{1,1} = T_1 + S_{1,1}$ to join queue 2 and starts its service immediately at station 2, then leaves the second station at time $D_{2,1} = D_{1,1} + S_{2,1}$ to join queue 3, and so on. The second customer arrives at time $T_2 = T_1 + A_2$. If $T_2 < D_{1,1}$, this customer must wait in the first queue for $D_{1,1} - T_2$ time units, starts its service at time $D_{1,1}$, and can leave to join the second queue at time $D_{1,1} + S_{1,2}$ if there is a space available there, and so on. Otherwise it starts its service at time T_2 and can leave at time $D_{1,2} = T_2 + S_{1,2}$. In the case where $c_2 = 1$ (no waiting space at station 2) and $D_{2,1} > T_2 + S_{1,2}$, the second customer is blocked at the first queue and we have $D_{1,2} = D_{2,1}$.

All quantities defined above can be computed by a rather simple recurrence as we shall now explain (see Buzacott and Shantikumar 1993) provided that we know how to generate the successive interarrival times A_i and the service times $S_{j,i}$. These random variables can be independent with known distributions, but not necessarily. From the A_i's, we readily obtain the arrival times T_i, and the departure times $D_{j,i}$ can be computed from the T_i and the $S_{j,i}$ by setting $D_{0,i} = T_i$, $D_{j,i} = 0$ for $i \le 0$, $D_{m+1,i} = 0$ for all i, and using the recurrence:

$$D_{j,i} = \max[D_{j-1,i} + S_{j,i},\ D_{j,i-1} + S_{j,i},\ D_{j+1,i-c_{j+1}}] \tag{1.4}$$

for $1 \le j \le m$ and $i \ge 1$. In this recurrence, the max is attained by the first term when customer i does not wait at station j, by the second term if it waits at station j and starts its service as soon as customer $i-1$ leaves station j (the customer is not blocked), and by the third term if customer i gets blocked and cannot enter the next station at the end of its service (it enters station $j+1$ at the time when customer $i - c_{j+1}$ leaves that station). When there is no blocking, this last term is zero and can be removed.

Customer i arrives at queue j at time $D_{j-1,i}$, and starts its service at the time $D_{j,i-1}$ when the previous customer leaves queue j, so its waiting time is $D_{j,i-1} - D_{j-1,i}$, unless this quantity is negative (the previous customer has already left), in which case the waiting time is zero. That is:

$$W_{j,i} = \max[0,\ D_{j,i-1} - D_{j-1,i}]. \tag{1.5}$$

Customer i arrives at station j at time $D_{j-1,i}$ and leaves at time $D_{j,i}$, so its sojourn time at station j is $D_{j,i} - D_{j-1,i} = W_{j,i} + S_{j,i} + B_{j,i}$, and its blocked time is then

$$B_{j,i} = D_{j,i} - D_{j-1,i} - W_{j,i} - S_{j,i}. \tag{1.6}$$

For infinite buffer sizes (*no blocking*), we always have $B_{j,i} = 0$ and (1.4) simplifies to

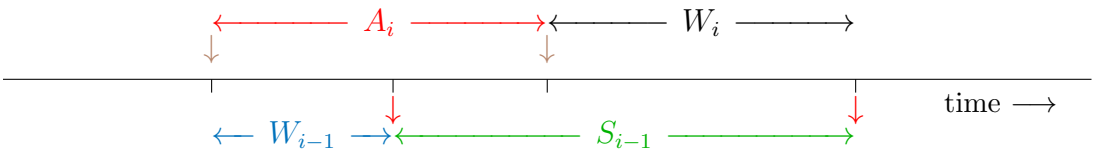
$$D_{j,i} = \max[D_{j-1,i},\ D_{j,i-1}] + S_{j,i}.$$

The special case of a single queue with no blocking is the $GI/G/1$ model (see Section A.19) for which we have

$$W_{1,i} = \max[0,\ D_{1,i-1} - T_i] = \max[0,\ W_{1,i-1} + S_{1,i-1} - A_i]. \tag{1.7}$$

This is the well-known *Lindley recurrence* (Lindley 1952, Kleinrock 1975), illustrated in the following diagram. (We removed the index $j = 1$ to simplify.) The two down arrows above the horizontal line represent the arrival times of customers $i-1$ and i to the system. The red

down arrows below the horizontal line represent the times when these two customers start their service. We see in the figure that $A_i + W_i = W_{i-1} + S_{i-1}$. But when $A_i > W_{i-1} + S_{i-1}$, it means that customer i arrives after customer $i-1$ has left, and then $W_i = 0$. This gives the formula (1.7).



The type of blocking described above is known as *production blocking*. Another type is *communication blocking*, where service at a given station starts only when the queue at the next station is not full. In that case, we have

$$D_{j,i} = \max[D_{j-1,i},\, D_{j,i-1},\, D_{j+1,i-c_{j+1}}] + S_{j,i}.$$

Algorithm 1 : Simulating a tandem queue with production blocking

Let $T_0 = 0$ and $D_{0,0} = D_{1,0} = 0$;
for $j = 2$ **to** m **do**
    for $i = -c_j + 1$ **to** 0 **do**
        let $D_{j,i} = 0$;
for $i = 1$ **to** N_c **do**
    generate A_i from its distribution;
    let $D_{0,i} = T_i = T_{i-1} + A_i$;
    let $W_i = B_i = 0$;
    for $j = 1$ **to** m **do**
        generate $S_{j,i}$ from its distribution;
        let $D_{j,i} = \max[D_{j-1,i} + S_{j,i},\, D_{j,i-1} + S_{j,i}\,,\, D_{j+1,i-c_{j+1}}]$;
        let $W_{j,i} = \max[0,\, D_{j,i-1} - D_{j-1,i}]$ and $W_i = W_i + W_{j,i}$;
        let $B_{j,i} = D_{j,i} - D_{j-1,i} - W_{j,i} - S_{j,i}$ and $B_i = B_i + B_{j,i}$;
Compute and return the averages:
    $\bar{W}_{N_c} = (W_1 + \cdots + W_{N_c})/N_c$ and $\bar{B}_{N_c} = (B_1 + \cdots + B_{N_c})/N_c$.

Algorithm 1 simulates the first N_c customers in the tandem queue with production blocking, via (1.4), and computes the waiting time and blocking time of each customer and their averages across customers. The scope of the "**for**" loops in the algorithms (all over this book) is indicated by the indentation. When there is no blocking ($c_j = \infty$ for all j) the expression shaded in light brown must be replaced by "$i = 0$" only, and the expressions shaded in light violet must be removed.

If instead of fixing N_c, we want to fix a time horizon T and simulate all customers that arrive before time T, it suffices to replace the loop "**for** $i = 1$ **to** N_c" by "**for** ($i = 1$, $T_i < T$, i++)", in C language notation.

The recurrence equations obtained here are rather simple and easy to simulate via Algorithm 1, because each station has a single server and the FIFO property. In case some

stations have more than one server, with random service times, in general customers may arrive at station $j+1$ in a different order than at station j. Customer $i+1$ may arrive after customer i at a given station and leave this station earlier due to a shorter service time, so the stations are not FIFO in general even if the customers are served by order of arrival (FCFS). The recurrences then become much more complicated, since we have to find the earliest departure event for each departure at each stage. These models are typically simulated using *discrete-event simulation*, as explained in Section 1.9.

♣ This is implemented in TandemQueue.java and TandemQueue0.java (infinite capacities). Could run simulations and display histograms. □

Example 1.11 *Pricing a financial option.* We consider a financial *asset* (for example, one share of a stock of a company, or one barrel of oil, or one ounce of gold) whose market price at time t is denoted by $S(t)$. We assume that this price evolves as a stochastic process $\{S(t),\, t \geq 0\}$ with known probability law (in practice this law can be estimated from data). Suppose that the owner of a *financial contract* (called a *financial derivative* or *option*) receives a *net payoff* of $g(S(t_1), \ldots, S(t_d))$ at time T, where $g : \mathbb{R}^d \to \mathbb{R}$ is a given function and $t_1 \ldots, t_d$ are predetermined observation times of $S(t)$, which satisfy $0 \leq t_1 < \cdots < t_d = T$. That is, the payoff depends only on the values of the asset at these observation times.

Suppose also that money left in the owner's bank account yields interest at the continuously compounded rate r (called the *short rate*). This means that one dollar placed in the account at time 0 is worth e^{rt} dollars at time t. Equivalently, this means that the present value of an amount to be received in t units of time is that amount multiplied by the *discount factor* e^{-rt}. Our aim here is to estimate the fair (present) value of the financial contract. Financial institutions who offer these contracts do that on a daily basis.

Economists and financial engineers compute this value based on the assumption that the financial market should always be in a form of equilibrium under which it is not possible to make money with positive probability without taking risks. A financial market that satisfies this "no free lunch" condition is called a *no-arbitrage* market. To explains what this means, suppose that starting with a given amount of money at time 0, one would follow a strategy that holds $w(t)$ shares of the stock at time t for $0 < t \leq T$, where $w(t)$ may be a function of $\{S(\zeta),\, 0 \leq \zeta \leq t\}$ and $\{w(\zeta),\, 0 \leq \zeta < t\}$ (the history so far) and can take arbitrary real values (possibly negative). The rest of the money (also possibly negative) would be in the bank account. Then, no arbitrage means that there is no way to design such a strategy whose net return at time T is never smaller than if all the money was left in the bank account, and is strictly larger with positive probability.

Under the setting of a no-arbitrage market, it turns out that the present value (or fair price) of the financial contract at time 0, when $S(0) = s_0$, can be written as

$$v(s_0, T) = \mathbb{E}^* \left[e^{-rT} g(S(t_1), \ldots, S(t_d)) \right],$$

where $\mathbb{E}^*$ denotes the mathematical expectation under a certain probability measure $\mathbb{P}^*$ called the *risk-neutral measure*. This measure $\mathbb{P}^*$ generally differs from the true measure under which the process $\{S(t),\, t \geq 0\}$ evolves in real life. Under it, the process $\{e^{-rt} S(t),\, t \geq 0\}$ is a *martingale*, which means that for any $t \geq 0$ and $0 < \delta \leq T - t$, $\mathbb{E}^*[e^{-r\delta} S(t+\delta)] = S(t)$. For further details on this (which are not essential here), see for example Duffie (1996). Note

that a risk-neutral measure does not always exist, and is not always unique. Here, we assume that it exists.

Except for a few simple models, there is no efficient way of computing $v(s_0, T)$ exactly, and Monte Carlo becomes the method of choice for its estimation, provided that we have an efficient algorithm to generate a trajectory $S(t_1), \ldots, S(t_d)$ from $\mathbb{P}^*$ and to compute g. We repeat this n times, independently. Let X_i be the outcome of $e^{-rT} g(S(t_1), \ldots, S(t_d))$ for the ith replication. The final estimator is the average $\bar{X}_n = (X_1 + \cdots + X_n)/n$.

In the popular model of Black and Scholes (1973), $S(t)$ is assumed to evolve as a geometric Brownian motion (GBM), which means that its logarithm is a Brownian motion. This implies that under $\mathbb{P}^*$, one must have

$$S(t) = S(0) e^{(r - \sigma^2/2)t + \sigma B(t)} \tag{1.8}$$

where r is the *short rate*, σ is a constant called the *volatility*, and $B(\cdot)$ is a *standard Brownian motion* (see Section 2.14.1 for the details). The latter means that $B(0) = 0$ and for any fixed $t_2 > t_1 \geq 0$, $B(t_2) - B(t_1)$ is a normal random variable with mean 0 and variance $t_2 - t_1$, independent of the behavior of $B(\cdot)$ outside the interval $[t_1, t_2]$. It is then easy to simulate the values of $B(t)$ at successive times $0 < t_1 < t_2 < t_3 < \cdots$ by simulating independent standard normal random variables, and then compute the value of $S(t)$ at those times via (1.8). We do this in Algorithm 2.

An *European call option* is a simple financial contract that gives the owner the right to buy one unit of the asset at price K (the *strike price*) at time T (the *expiration date*). For example, the owner may have an option to buy 12,000 barrels of crude oil for $K = 1.1$ million dollars on July 1 of next year. If we assume that the asset provides no dividend, the net payoff at time T is $g(S(T)) = \max[0, S(T) - K]$. In other words, if $S(T) > K$, the owner exercises the option by buying at price K and may resell the asset immediately at the market price $S(T)$, thus making a profit $S(T) - K$. If $S(T) \leq K$, the owner will not exercice the option, so the payoff is zero and the option is worthless. In this case, if the asset price obeys (1.8), it can be derived (see, e.g., Taylor and Karlin 1998 or Duffie 1996) that

$$v(s_0, T) = s_0 \Phi(z_0 + \sigma\sqrt{T}) - K e^{-rT} \Phi(z_0), \tag{1.9}$$

where

$$z_0 = \frac{\ln(s_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}}$$

and Φ is the *standard normal cdf*. Eq. (1.9) is the celebrated *Black-Scholes formula*.

When g is more complicated, there is often no analytic formula for $v(s_0, T)$. For instance, the payoff at time T of a discretely-monitored *Asian call option* is

$$g(S(t_1), \ldots, S(t_d)) = \max\left(0, \frac{1}{d}\sum_{j=1}^{d} S(t_j) - K\right), \tag{1.10}$$

and no simple closed-form formula is available for the expectation of this payoff under the GBM setup. Algorithm 2 estimates $v(s_0, T)$ by Monte Carlo for a general payoff function g, under the GBM model (1.8) for the asset price. The algorithm generates d independent and identically distributed (i.i.d.) standard normal random variables $Z_1, \ldots, Z_d$ and computes

the payoff as a function of their outcomes. Note that $B(t_j) - B(t_{j-1}) = \sqrt{t_j - t_{j-1}}Z_j$ is a normal random variate with mean 0 and standard deviation $\sqrt{t_j - t_{j-1}}$ (variance $t_j - t_{j-1}$). More efficient estimators for this model are discussed in Chapter 6.

Algorithm 2 : Option pricing under a GBM model by Monte Carlo

for $i = 1$ **to** n **do**
  let $t_0 = 0$ and $B(t_0) = 0$;
  for $j = 1$ **to** d **do**
    generate $Z_j \sim \mathsf{N}(0, 1)$;
    let $B(t_j) = B(t_{j-1}) + \sqrt{t_j - t_{j-1}}Z_j$;
    let $S(t_j) = s_0 \exp\left[(r - \sigma^2/2)t_j + \sigma B(t_j)\right]$;
  compute $X_i = e^{-rT} g(S(t_1), \ldots, S(t_d))$;
plot a histogram of the X_i's if desired;
return the average $\bar{X}_n = (X_1 + \cdots + X_n)/n$ as an estimator of $v(s_0, T)$.

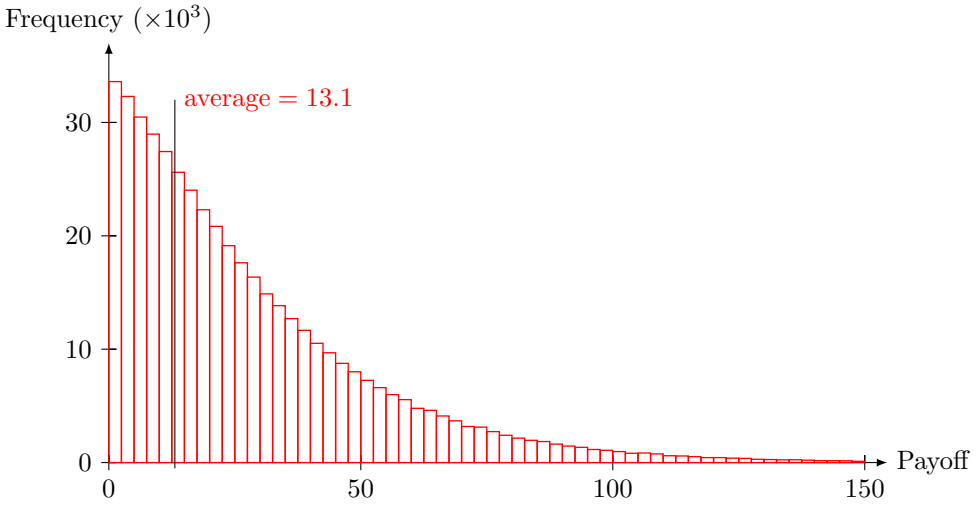


Fig. 1.6. A histogram of the 465,332 positive discounted payoffs among $n = 10^6$ realizations under the risk-neutral measure, in Example 1.11.

For a numerical illustration, suppose that time is measured in years and let $d = 12$, $T = 1$, $t_j = j/12$ for $j = 0, \ldots, 12$, $K = 100$, $s_0 = 100$, $r = 0.05$, $\sigma = 0.5$, and the payoff at time T is given by (1.10). We performed $n = 10^6$ (one million) independent simulation runs of this model. Figure 1.6 gives a histogram of the n realizations of the discounted payoff, $X_1, \ldots, X_n$. Only the positive values are shown on the histogram. In 53.47% of the cases, the average in (1.10) was less than K so the payoff was zero. The average discounted payoff over the one million runs was $\bar{X}_n = 13.1$. It provides an estimate of the fair value of the option. There is, however, a significant variability in the observed payoffs: They range from 0 to 390.8. We recall that these simulations are under the risk-neutral measure $\mathbb{P}^*$, so this empirical distribution of the payoffs does not reflect the true distribution of the payoff in the real life.

It is important to recognize that despite its widespread adoption by financial institutions, the GBM model is not necessarily very realistic. Several more refined models in which the volatility changes randomly with time, the process $S(\cdot)$ can have jumps, etc., have been proposed in the mathematical finance literature. Other types of stochastic processes can also be used to model the evolution of the interest rate, the exchange rate between two currencies, or the market price of a commodity such as oil, for example. Often, the payoff of a financial derivative is based on many different assets, whose price evolutions are not independent. This is more difficult to model. There are also cases where the payoff depends on the entire sample path $\{S(t),\, t \geq 0\}$. Moreover, one is frequently interested in estimating not only the expected discounted payoff (the option price), but also its derivative (or sensitivity) with respect certain model parameters such as the initial value $S(0)$, the strike price K, or the volatility σ. These derivatives are collectively known as the *Greeks*, because many have standard names given by Greek letters. For example, the sensitivities of $v(s_0, T)$ with respect to s_0, σ, T, and r, are called the *delta*, the *vega*, the *theta*, and the *rho* of the option, respectively. These sensitivities can be more difficult to compute than the option price itself. Often (but not always), they can be written as expectations, which can themselves be estimated by Monte Carlo as usual.

All of this gives rise to a large variety of situations where simulation is a handy tool. Monte Carlo methods in finance are surveyed by Boyle, Broadie, and Glasserman (1997a), Glasserman (2004), Jäckel (2002), and Staum (2009), for example. □

Example 1.12 *Estimating a quantile.* In the examples seen so far, the quantities we wanted to estimate were expressed as mathematical expectations. One exception was the 99% quantile of the distribution of T in Example 1.4. In general, for $0 < p < 1$ and p fixed, the p-quantile of a random variable X (or of its distribution) is defined as

$$\xi_p = \inf\{x : \mathbb{P}[X \leq x] \geq p\}. \tag{1.11}$$

See Figure 1.7. Quantiles are used as performance or risk measures in several area. For example, the performance of certain emergency systems such as ambulances is often measured by a p-quantile of the response time distribution, say for $p = 0.90$ or $p = 0.95$. In Example 1.4, we looked at an estimate of the 0.99 quantile of T, which is defined as the project duration $\xi_{0.99}$ that has exactly 1% chance of being exceeded.

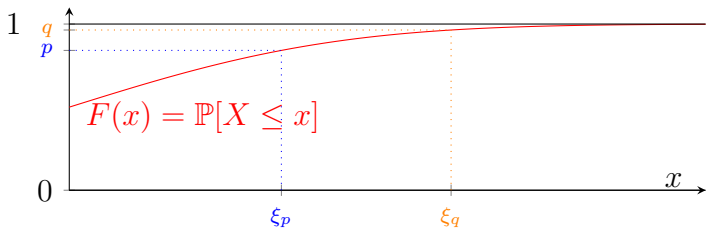


Fig. 1.7. Illustration of ξ_p and ξ_q for p and q close to 1.

In finance, if X is the net profit for an investor over a given time period, the *value-at-risk* at level $\alpha \in (0, 1)$ for X, denoted $\mathrm{VaR}_\alpha(X)$, is the $(1-\alpha)$-quantile of the distribution of the

net loss $L = -X$ (Artzner et al. 1999). It is also often called the *$(1-\alpha)$-value-at-risk* or *$(1-\alpha)$-VaR for L* instead (Glasserman 2004, Hong, Hu, and Liu 2014). Portfolio managers often have a constraint on their $\text{VaR}_\alpha(X)$ for a given α. If $\text{VaR}_{0.01}(X)$ for the profit (or equivalently the 99%-VaR for the loss) over the next day is two million dollars, it means that the chance of losing more than two million dollars over the next day does not exceed 1%. See Hong, Hu, and Liu (2014) for an overview of VaR and its estimation in simulation settings. In actuarial science, $\text{VaR}_\alpha(X)$ is defined as the α-quantile of X instead, where X represents the loss (Asmussen and Glynn 2007).

For many models, the distribution of X is too complicated for the desired quantile ξ_p to be computable exactly, but an independent random sample $X_1, \ldots, X_n$ from the distribution of X can be generated by simulation. More generally, $X_1, \ldots, X_n$ can just be a data set from an unknown distribution. One simple estimator of ξ_p, in this case, is

$$\hat{\xi}_{p,n} = \inf\left\{x \in \mathbb{R} : \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[X_i \le x] \ge p\right\} = X_{(\lceil np \rceil)},$$

where $X_{(1)}, \ldots, X_{(n)}$ are the observations $X_1, \ldots, X_n$ sorted in increasing order, also called the *order statistics*. This estimator $\hat{\xi}_{p,n}$ is simply the smallest observation whose rank is at least np. This is the p-quantile of the empirical distribution of $X_1, \ldots, X_n$, It is biased for finite n, but strongly consistent. For the experiment reported in Example 1.4, the empirical quantile for $p = 0.99$ was $\hat{\xi}_{0.99,n} = 131.8$. Thus, there is approximately one percent chance that T exceeds 131.8. More refined quantile estimators used in popular software are given in Hyndman and Fan (1996). Quantile estimation is further discussed in Section 5.7. □

♣ Perhaps an example in radiotherapy or nuclear physics.

Examples of discrete-event dynamic simulation models and programs will be given in Section 1.9.

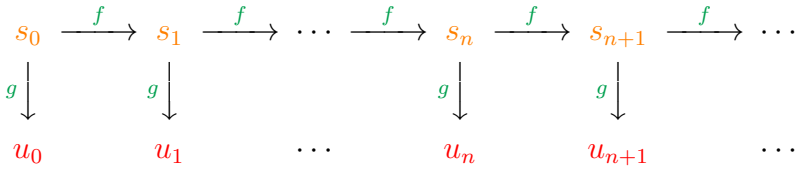
1.3 Introduction to Random Number Generation

1.3.1 The concept of a random number generator

Stochastic models are built over the notions of random variable and stochastic process, which are well-defined in the abstract mathematical framework of probability theory, but cannot be implemented concretely and exactly on a conventional computer. In simulation, the so-called *random number generators* are in fact *deterministic* algorithms that produce periodic sequences of numbers. This seems to have little to do with the idea of random variables. However, well-designed algorithms *simulate* sequences of independent random variables well enough for most practical purposes. They also have several advantages over physical devices that produce *genuinely* random sequences of numbers (e.g., using noise diodes or drawing balls from a container), as we shall see later on.

In this book, a random number generator (RNG, for short) is a small computer program (or algorithm) that can produce deterministic sequences of numbers that behave very much

like realizations of independent random numbers from the uniform distribution. The classical constructions are *recurrence-based RNGs* (L'Ecuyer 1990a, L'Ecuyer 1994b). This type of algorithmic RNG has a finite set of *states*, and goes from state to state according to a deterministic *transition function* f. At each step, the next state is a function of the current state only. There is also an *output function* g which assigns to each state a real number between 0 and 1. From any given starting state, the generator always produce exactly the same output sequence. This is illustrated in following diagram, where the state at step n is s_n and the output random number at step n is u_n.



Since the number of states is finite, the output sequence is *periodic*, with a period length that cannot exceed the total number of states. That is, as soon as a state is visited for a second time, the output sequence from the time of this second visit is exactly the same as the output sequence from the time of the first visit. In practice, good RNGs are constructed so that their period is long enough (e.g., 2^{200} or more) to make sure that it will never be exhausted. Moreover, a certain amount of randomness can be introduced by selecting the initial state, called the *seed*, at random. The RNG can then be viewed as an *extensor of randomness*, transforming a short random seed into a long "random" output sequence. The latter sequence is sometimes called *pseudorandom*.

The aim of the RNG is to imitate a sequence of independent random variables $U_0, U_1, U_2, \ldots$ uniformly distributed over the real interval $(0, 1)$ (i.i.d. $U(0, 1)$, for short). Note that in theory, we could as well take the uniform distribution over the closed interval $[0, 1]$, because the endpoints 0 and 1 would have probability zero anyway. However, an RNG with a finite state set can only produce a *finite* set of numbers, and we must make sure that 0 and 1 do not belong to that set, because these values will cause trouble when fed to certain non-uniform variate generators. For example, the standard method to generate an exponential computes $-\ln(1-u)$ for the uniform variate u returned by the RNG; for $u = 1$, this quantity is infinite.

In a simulation, the successive numbers in the output sequence of the RNG, loosely called the *random numbers*, are *assumed* to be the realizations of i.i.d. $U(0, 1)$ random variables. They are transformed as needed to simulate random variables from other distributions such as the normal, exponential, geometric, Poisson, etc., or to simulate random vectors with more complicated distributions. Formally, we know that this assumption is false, but we hope that the output of our simulation program will behave as if the assumption was true. Fortunately, well-designed RNGs do fulfill this hope, at least for practical purposes.

Example 1.13 *A linear congruential generator.* A simple and well-known type of RNG, perhaps too simplistic but historically important, is the *linear congruential generator (LCG)* (Lehmer 1951, Knuth 1981). The state at step n is an integer x_n and the transition function is defined by the recurrence

$$x_n = (ax_{n-1} + c) \bmod m, \tag{1.12}$$

where $m > 0$, $a > 0$, and c are integers called the *modulus*, the *multiplier*, and the *additive constant*, respectively, and "mod m" means taking the remainder of the division by m. The

state x_n always belongs to the finite set $\{0, \ldots, m-1\}$. The output function is usually defined by

$$u_n = x_n/m, \tag{1.13}$$

which is always strictly between 0 and 1 provided that we never have $x_n = 0$. It is customary to take $c = 0$, in which case we have a *multiplicative LCG*, whose period length cannot exceed $m - 1$, because when hitting state 0 we never leave it again, so we must remove this absorbing state from the set $\{0, \ldots, m-1\}$. The period length can actually be equal to $m-1$ if m is a prime number and a is properly chosen (see Chapter 3). Then, x_n is always strictly positive.

For a concrete illustration of how an LCG behaves, let $m = 101$ (a prime number), $a = 12$, and $c = 0$. If $x_0 = 10$, then we can compute

$$\begin{aligned} x_0 &= 10 & u_0 &= 10/101 \approx 0.09901, \\ x_1 &= 12 \times 10 \bmod 101 = 19, & u_1 &= 19/101 \approx 0.18812, \\ x_2 &= 12 \times 19 \bmod 101 = 26, & u_2 &= 26/101 \approx 0.25742, \\ x_3 &= 12 \times 26 \bmod 101 = 9, & u_3 &= 9/101 \approx 0.08911, \end{aligned}$$

and so on. The output sequence is periodic with period length 100: we have $x_{100} = x_0 = 10$ and $x_0, \ldots, x_{99}$ are all distinct; they are the numbers $\{1, \ldots, 100\}$ permuted in a different order. Then, $x_{100+n} = x_n$ for all $n > 0$. Moreover, over each subsequence of 100 steps, the state visits each number from 1 to 100 exactly once, so the output takes each value in $\{1/101, 2/101, \cdots, 100/101\}$ exactly once. This is a very good approximation of uniformity given that we only have 100 possible values.

Of course, this mini-LCG is not to be taken seriously for simulation use. It is only to illustrate the basic ideas. An acceptable LCG would need to use a very large m, say $m > 2^{200}$, in which case computing the product modulo m must be done via decomposition methods on standard computers, and the generator may then be too slow, unless the parameters are selected in a way that a clever fast implementation is possible. LCGs with moduli $m = 2^{31}-1$, $m = 2^{32}$, and $m = 2^{48}$ have been popular in the past and are still used in some software libraries and products, but they are too small for the requirements of today's simulations. These LCGs should be discarded and replaced by more robust generators such as those recommended in Chapter 3. □

1.3.2 Quality criteria

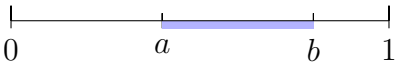
In probability theory, the requirement that $\{U_n,\ n \geq 0\}$ is a sequence of i.i.d. $U(0,1)$ random variables can be recast into the following *uniformity property*: For each integer $s > 0$ and each $n \geq 0$, the random vector $\boldsymbol{U}_{n,s} = (U_n, \ldots, U_{n+s-1})$ of s successive random variables has the uniform distribution over the s-dimensional unit hypercube

$$(0,1)^s = \{(u_0, \ldots, u_{s-1}) : 0 < u_j < 1 \text{ for each } j\}.$$

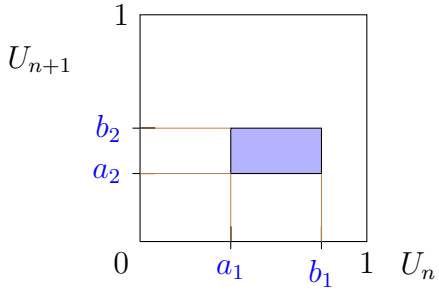
If this holds for $n = 0$ and all $s > 0$, then this also holds for all $n > 0$. This property captures both uniformity and independence. It means that for every integer $s > 0$ and any $n \geq 0$, whenever $0 \leq a_j < b_j \leq 1$ for all j, we must have

$$\mathbb{P}[a_j \le U_{n+j} \le b_j \text{ for } j = 0, \ldots, s-1] = (b_0 - a_0) \cdots (b_{s-1} - a_{s-1}).$$

That is, the probability that the point $\boldsymbol{U}_{n,s}$ falls in some rectangular box in the unit cube should be equal to the volume of that box. For $s = 1$, this means that whenever $0 \le a < b \le 1$, we have $\mathbb{P}[a \le U_n \le b] = b - a$:



The following picture illustrates the case $s = 2$: The probability that the point falls in the blue box should be equal to the surface of that box.



Now, suppose we replace the random vector $\boldsymbol{U}_{n,s}$ by the vector $\boldsymbol{u}_{n,s} = (u_n, \ldots, u_{n+s-1})$ of s output values produced by the RNG starting at step n, and assume that the seed is random and uniformly distributed over the state space. To approximate the above uniformity property we will require that the (finite) set Ψ_s of all s-dimensional vectors $\boldsymbol{u}_{n,s}$ of successive output values produced by the generator, from all possible initial states, is uniformly spread over $(0, 1)^s$. Here we interpret Ψ_s as a *multiset*, which means that the vectors are counted as many times as they appear, and the cardinality of Ψ_s is exactly equal to that of the state space. For our LCG example with $m = 101$ and $a = 12$, this cardinality is $|\Psi_s| = 101$ for all s, and we have for example

$$\Psi_3 = \{(0, 0, 0), (1/101, 12/101, 43/101), (2/101, 24/101, 38/101), \ldots, (100/101, 89/101, 58/101)\}.$$

The set Ψ_s can be viewed in a way as a *sample space* from which the s-dimensional output vectors are drawn at random, by selecting a random seed for the RNG. Since we want to approximate the uniform distribution over $(0, 1)^s$, it makes sense to construct the RNG so that Ψ_s is *huge* and *very evenly spread* over this unit hypercube. The cardinality of Ψ_s should be several orders of magnitude larger than the maximum number of points that we are likely to draw from it in any simulation, firstly because with more points we can cover the unit hypercube more uniformly, and secondly because if $|\Psi_s|$ is too small, we may draw nearly all the points of Ψ_s, and then these points will look more uniform than independent random points (they will be too uniform).

The sets Ψ_s are comprised of vectors of *successive* output values. We may also want to look at the uniformity of the sets of vectors of *non-successive* output values, of the following form. Let $0 \le i_1 < \cdots < i_s$ be s distinct integers and consider the set $\Psi_{\{i_1,\ldots,i_s\}}$ of all the vectors $(u_{i_1}, \ldots, u_{i_s})$ that can be produced by the generator, from all possible initial states. We would also like each of these sets to be evenly spread over the s-dimensional unit hypercube. For example, $\Psi_{\{0,2\}}$ is the set of all vectors (u_n, u_{n+2}),

i.e., formed by values that are two steps apart, whereas $\Psi_{\{0,3,6,9\}}$ is the set of all vectors of the form $(u_n, u_{n+3}, u_{n+6}, u_{n+9})$, i.e., formed by 4 values that are 3 steps apart, for all possible seeds of the generator. For our small LCG example, we have for example $\Psi_{\{0,2\}} = \{(0,0), (1/101, 43/101), (2/101, 38/101), \ldots, (100/101, 58/101)\}$.

To implement these ideas, we need: (1) a precise definition of "evenly spread", in the sense of computable measures of (non-)uniformity of the point sets Ψ_s and $\Psi_{\{i_1,\ldots,i_s\}}$ over the unit hypercube, (2) efficient ways of computing these measures without generating all the points (because there are much too many), at least for the classes of RNGs in which we have interest, even when the state space is huge, and (3) practical methods for finding RNGs with very long periods, for which fast and relatively simple implementations are available, and with good s-dimensional uniformity at least for s up to a certain dimension s_1 and for a certain class of sets $\{i_1, \ldots, i_s\}$ deemed important. Obviously, the s-dimensional uniformity cannot be checked for all s up to infinity and for all subsets $\{i_1, \ldots, i_s\}$. Such quality criteria, together with a few additional requirements such as portability and jumping ahead facilities, and empirical statistical testing of RNGs, are discussed in Chapter 3. In that chapter, we also provide concrete implementations of RNGs that we recommend.

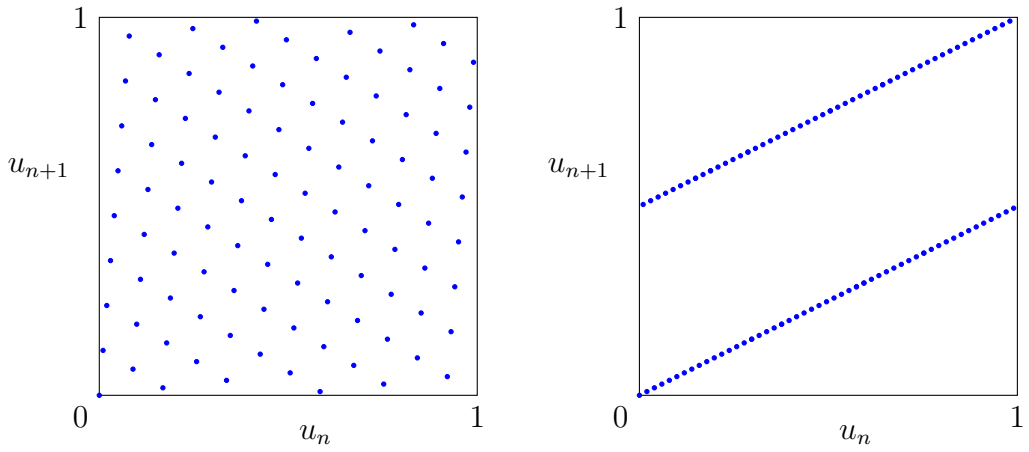


Fig. 1.8. The set Ψ_2 of all pairs (u_n, u_{n+1}) for the LCG with $m = 101$ and $a = 12$ (left) and $a = 51$ (right).

Example 1.14 RNGs based on linear recurrences are highly structured. This can be seen as a weakness because too much structure means less random-looking, but on the other hand it permits one to study and measure their uniformity. As an illustration, Figure 1.8 (left) shows the set Ψ_2 of all pairs of successive output values produced by the LCG with modulus $m = 101$ and multiplier $a = 12$, over its entire period length of 100, plus the zero vector which corresponds to the case where the initial state is 0. These 101 points turns out to have a regular *lattice* structure. They do not look like random points, but they are quite evenly distributed in the unit square, which is what we are looking for. Of course, in practice, the cardinality of Ψ_s should be much larger than 101. We recommend at least 2^{200} or more. For an LCG with carefully selected a and prime $m > 2^{200}$, so that Ψ_2 covers the unit square very

evenly, selecting a point at random from Ψ_2 is almost the same as generating it at random from the uniform distribution over the unit square $(0,1)^2$.

Figure 1.8 (right) shows all the points of Ψ_2 for an LCG with the same modulus $m = 101$, but this time with $a = 51$. In this case, all the points are contained in only two lines, with large empty spaces between the lines. The uniformity is much worse than with $a = 12$. This represents a bad choice of the parameter a. □

Example 1.15 We now give an example of a recommendable RNG for simulation, proposed by L'Ecuyer (1999a) and used in several software products. It is named *MRG32k3a*. The algorithm works as follows. The initial state (the seed) consists of six integer:

$$x_{-2}, x_{-1}, x_0 \in \{0, 1, \ldots, 4294967086\}, \text{ not all 0, and}$$
$$y_{-2}, y_{-1}, y_0 \in \{0, 1, \ldots, 4294944442\}, \text{ not all 0.}$$

The recurrence is

$$\begin{aligned} x_n &= (1403580x_{n-2} - 810728x_{n-3}) \bmod 4294967087, \\ y_n &= (527612y_{n-1} - 1370589y_{n-3}) \bmod 4294944443, \\ u_n &= [(x_n - y_n) \bmod 4294967087]/4294967087. \end{aligned}$$

The state at step n is $s_n = (x_{n-2}, x_{n-1}, x_n, y_{n-2}, y_{n-1}, y_n)$. It turns out that the vector (x_{n-2}, x_{n-1}, x_n) visits each of the $4294967087^3 - 1$ possible nonzero values that it can take exactly once before restarting the same sequence again, and similarly (y_{n-2}, y_{n-1}, y_n) visits each of the $4294944443^3 - 1$ possible nonzero values that it can take exactly once before cycling again. The sequence $u_0, u_1, u_2, \ldots$ is periodic, with 2 cycles of period near $2^{191} \approx 3.1 \times 10^{57}$. This combined RNG turns out to be equivalent (approximately) to a linear generator with a large modulus. The uniformity of the point set Ψ_s for the combined generator has been analyzed (and found to be excellent) in up to 45 dimensions. □

Example 1.16 One form of *subtract-with-borrow (SWB)* generator with parameters (b, r, k) is defined by the recurrence

$$\begin{aligned} x_n &= (x_{n-r} - x_{n-k} - c_{n-1}) \bmod b, \\ c_n &= \mathbb{I}[x_{n-r} - x_{n-k} - c_{n-1} < 0], \quad \text{and} \\ u_n &= x_n/b, \end{aligned}$$

where $k > r > 0$, $x_n \in \{0, \ldots, b-1\}$, and $c_n \in \{0, 1\}$ for each n. The state at step n is $s_n = (x_{n-k+1}, \cdots, x_n, c_n) \in \{0, \ldots, b-1\}^k \times \{0, 1\}$. This type of generator was proposed by Marsaglia and Zaman (1991) and has been widely used in several software libraries. For example, in Mathematica, version 5.2 and earlier, the default RNG used two steps of the recurrence of a SWB generator with parameters $(b, r, k) = (2^{31}, 8, 48)$ to produce each uniform random number in (0,1), via $u_n = x_{2n}/2^{62} + x_{2n+1}/2^{31}$. This generator is very fast, and its period is approximately 2^{1479}, which is extremely long. However, a long period is not sufficient for good quality. Tezuka, L'Ecuyer, and Couture (1993) have shown that the output values u_n produced by the SWB generator defined above are almost the same (the values differ by less than $1/b = 2^{-31}$) as those produced by an LCG with modulus

$m = b^k - b^r + 1$ and multiplier a that satisfies $ab \bmod m = 1$. Moreover, Couture and L'Ecuyer (1994) have shown that all the nonzero points of $\Psi_{\{0,k-r,k\}}$ for this LCG, i.e., the set of all three-dimensional points of the form $(u_n, u_{n+k-r}, u_{n+k})$ produced by this LCG from a nonzero state, is contained in only two parallel planes in the unit cube $[0,1)^3$. Those two planes are defined by $u_{n+k} - u_{n+k-r} + u_n = q$ for $q = 0$ and 1, Thus, the SWB generator has extremely bad uniformity for this particular three-dimensional projection. This important defect may have a huge impact on simulation results, as observed empirically by Ferrenberg, Landau, and Wong (1992) for applications in computational physics. See also Exercise 1.4. □

Other examples of widely-used poor generators are unveiled in L'Ecuyer and Simard (2007) and in Chapter 3.

1.3.3 Multiple streams and substreams

In modern simulation software, RNGs are often seen as objects that can be created at will, in practically unlimited number, and can be viewed as independent sources of random numbers (L'Ecuyer et al. 2002, L'Ecuyer and Buist 2005, L'Ecuyer 2010, L'Ecuyer et al. 2017). This is typically implemented by partitioning the cycle of a large-period backbone RNG into long disjoint segments of a given length ν, and using a function that can jump ahead quickly by ν steps in the RNG sequence to be able to jump from the beginning of a segment to the beginning of the next one. The starting points of several successive segments can then be computed sequentially. Each RNG object is mapped internally to one of these segments, often called a *stream* of random numbers, and provides a virtual RNG. In some popular implementations, the streams are also partitioned into multiple substreams, and each stream object has methods to generate the next number, to rewind to the beginning of the stream, or to the beginning of the current substream, or to the beginning of the next substream. The streams and subtreams must be long enough to make sure that they would not overlap, at least in reasonable time. These multiple streams and substreams are extremely useful for the correct implementation of common random numbers for comparing systems and for sensitivity analysis, for example. We give concrete examples of that in Section 1.7 and in Chapter 6. They are also useful for simulations running on parallel processors, where each processor can produce its own streams of random numbers without having to care about what the other processors are doing.

1.3.4 Counter-based generators

The classical recurrence-based RNGs have been designed to operate in a sequential fashion: the random numbers in each stream are generated sequentially by using a transition function and the streams are created sequentially by a jump-ahead function. This sequential design is not ideal for massively-parallel computers. *Counter-based generators* provide a way to do all of this in parallel (Salmon et al. 2011, L'Ecuyer et al. 2021). The idea is to make the transition function f extremely simple and leave all the complicated transformations (most of the work) to the output function g. In counter-based RNGs, the state at step n is simply

n, and the transition function just increases the counter n by 1. The output function g is also parameterized by another integer k called the *key*, so it is a function of the pair (k, n), which can be viewed as a two-dimensional counter. The function g is usually taken as a cryptographic hashing function, often simplified to increase the speed, and selected in a way that for each k, the set of values of $g(k, n)$ for all admissible n covers the interval $(0, 1)$ very uniformly. Multiple streams can be obtained simply by assigning one stream to each value k of the key, with its counter starting at 0. Then both the creation of multiple streams and the generation of arrays of random numbers can be done easily in parallel. The counter and the key may be 128-bit integers, for example, in which case there are 2^{128} different keys and 2^{128} admissible values of n for each key.

1.3.5 The inversion method for non-uniform random variate generation

Random variables from non-uniform distributions are generated by applying certain transformations to the output values of a uniform RNG. This is easily achieved for certain distributions, but not for all.

Recall that a random variable X has cdf F if $F(x) = \mathbb{P}[X \leq x]$ for all $x \in \mathbb{R}$. The simplest way of generating X with cdf F is to apply the inverse of F to a $U(0, 1)$ random variable U, as in Figure 1.9:

$$X = F^{-1}(U) \stackrel{\text{def}}{=} \min\{x \mid F(x) \geq U\}. \tag{1.14}$$

With this definition of X, we have

$$\mathbb{P}[X \leq x] = \mathbb{P}[F^{-1}(U) \leq x] = \mathbb{P}[U \leq F(x)] = F(x),$$

so X has distribution F, exactly. This *inversion* method requires the availability of F^{-1}, or a good approximation of it.

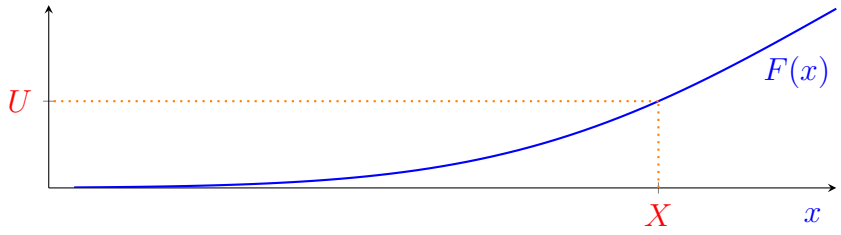


Fig. 1.9. Generating X by inversion from U.

Example 1.17 A random variable X has the *Weibull distribution* with parameters $\alpha > 0$ and $\lambda > 0$ if F has the form $F(x) = 1 - \exp[-(\lambda x)^\alpha]$ for $x > 0$ and $F(x) = 0$ for $x \leq 0$. In this case, one has (Exercise 1.6) $X = F^{-1}(U) = [-\ln(1 - U)]^{1/\alpha}/\lambda$, so X is easy to generate by inversion. As a special case, if $\alpha = 1$, X has the exponential distribution with mean $1/\lambda$. An exponential random variable can thus be generated by taking $X = F^{-1}(U) = -\ln(1 - U)/\lambda$. □

Example 1.18 Suppose that $\mathbb{P}[X = i] = p_i$ where $p_0 = 1/2$, $p_2 = 3/8$, $p_4 = 1/8$, and $p_i = 0$ elsewhere. The corresponding cdf F is a piecewise-constant right-continuous function having a jump of size p_i at $x = i$ for $i = 0, 2, 4$. The inversion method here will return 0 if $U < 1/2$, 2 if $1/2 \le U < 7/8$, and 4 if $U \ge 7/8$. In Figure 1.10, we have $U = 0.6$ and this gives $X = 2$. □

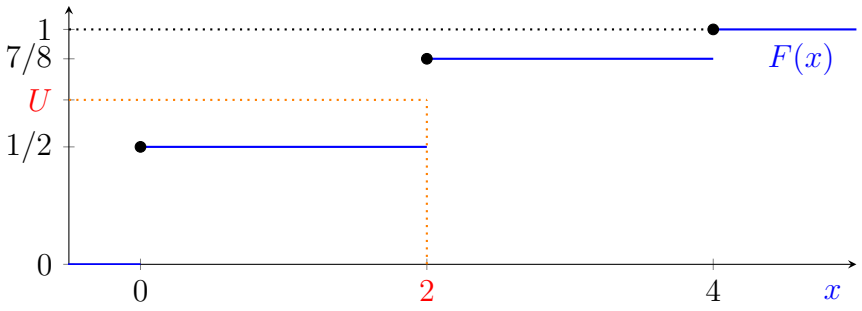


Fig. 1.10. Inversion for a discrete random variable with three possible outcomes.

For a general discrete distribution, suppose $\mathbb{P}[X = x_i] = p_i$ for $i = 0, 1, \ldots$. We have $F(x_i) = p_0 + \cdots + p_i$, and the inversion method returns $X = x_i$ if and only if $F(x_i) \ge U > F(x_{i-1})$, where $F(x_{-1}) = 0$ by convention. See Figure 1.11. To generate X, we first generate U, then we must find the smallest i for which $F(x_i) \ge U$. A simple way to find this i is to check the condition for $i = 0, 1, 2, \ldots$ sequentially until it is satisfied. But when this i is very large, this sequential search method is too inefficient. This will happen for example if X has a binomial or Poisson distribution with a large mean. More efficient search methods for this situation are discussed in Section 4.1.1. A direct inversion formula is available for certain distributions, as shown in Example 1.19.

Example 1.19 A random variable X has the *geometric distribution* with parameter p, $0 < p < 1$, if $\mathbb{P}[X = x] = p(1-p)^x$ for $x = 0, 1, 2, \ldots$. This is a discrete distribution and it represents the number of failures before the first success in a sequence of Bernoulli trials

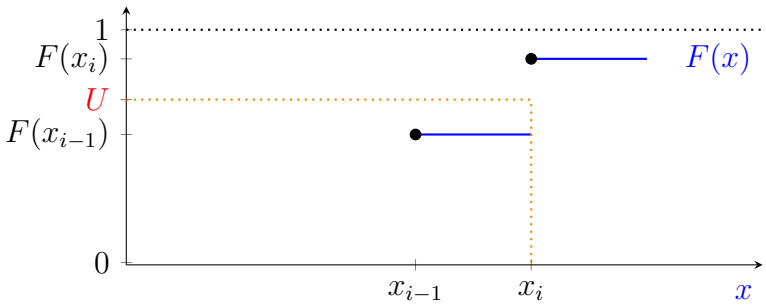


Fig. 1.11. Inversion for a general discrete random variable.

(see Chapter 2). The cdf of X is given by $F(x) = 1 - (1-p)^{\lfloor x+1 \rfloor}$ for $x \geq 0$ and $F(x) = 0$ for $x < 0$. When x is a non-negative integer, we have $F(x) = 1 - (1-p)^{x+1}$. Here, inversion should return $X = x$ if and only if $F(x) \geq U > F(x-1)$, which can be rewritten as

$$\begin{aligned} 1 - (1-p)^{x+1} \;\geq\; U \;>\; 1 - (1-p)^{x}, & \qquad \text{i.e.,} \\ -(x+1)\ln(1-p) \;\geq\; -\ln(1-U) \;>\; -x\ln(1-p), & \qquad \text{i.e.,} \\ (x+1) \;\geq\; \ln(1-U)/\ln(1-p) \;>\; x, \quad (\text{since } -\ln(1-p) > 0) & \end{aligned}$$

which gives $x = \lceil \ln(1-U)/\ln(1-p) \rceil - 1$. With probability 1, this is the same as returning $X = \lfloor \ln(1-U)/\ln(1-p) \rfloor$. □

For some distributions (e.g., the normal, Student, chi-square, etc.), F^{-1} cannot be written in close form but reasonably good numerical approximations are often available. For most simulations, inversion should be the method of choice when it is applicable, because it transforms U into X *monotonously*, which makes it compatible with quasi-Monte Carlo methods (Section 1.5.3) and with variance-reduction techniques such as common random numbers and antithetic variates (Chapter 6). In situations where *speed* is a real issue and monotonicity is no real concern, non-inversion methods are sometimes more appropriate. See Chapter 4 for further details.

We think that modern software for random number generation should be built with the following types of components (or objects): *random streams*, which provide sequences of independent uniform random numbers, *probability distributions*, which specify the target distributions, and (non-uniform) *random variate generators*, which generate random variates from a given distribution, using a given random stream. Thus, a generator can be constructed by putting together a distribution, a stream, and in some cases a generating method as well. Any stream can be used for any distribution and any generating method. Eventually, a random stream can also be replaced by a sequence of quasi-random points (see Section 6.10). This type of organization has been adopted in SSJ (L'Ecuyer and Buist 2005, L'Ecuyer 2023), for example. In older software, sometimes there is a single random stream for everything and the user has little control over it, and some non-uniform generators have a hidden uniform RNG embedded in them. We think this is bad design. Later in this book (e.g., in Section 1.7), we elaborate on the need for multiple streams of random numbers and provide illustrations.

1.4 Monte Carlo Integration

1.4.1 Estimating an integral by Monte Carlo

Stochastic simulation is frequently used to estimate an unknown mathematical expectation, formally defined as an integral with respect to a probability measure. *Monte Carlo integration*, in its simplest (crude) form, draws an independent random sample of size n from this measure and estimates the integral by averaging the n values taken by this integrand over this sample.

A random variable X defined over a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ can be seen as a measurable function $X : \Omega \to \mathbb{R}$ and its mathematical expectation (or mean) can be written as the integral (see Section A.5 of the Appendix)

$$\mu = \mathbb{E}[X] = \int_{\Omega} X(\omega)\mathbb{P}(\mathrm{d}\omega). \tag{1.15}$$

In this section, we assume that our purpose is to estimate μ. Section 1.2 gave examples of that.

In general, the probability space can be defined in many different ways for the same μ. In the context of simulation it is often convenient to interpret ω as a sequence of s independent $U(0,1)$ random variables $U_1, \ldots, U_s$. Then $\mathbb{P}$ represents the uniform distribution over the unit hypercube $(0,1)^s$ and $X = X(\omega)$ can be written as $X = f(U_1, \ldots, U_s)$, where f is the function computed by the computer program that simulates realizations of X. This interpretation of ω is natural because this is how randomness is actually generated in virtually all stochastic simulations. We will also use this setting to define quasi-Monte Carlo methods (see Section 1.5.3). The function f depends on how the simulation is implemented; there are many possibilities for any given model. Note that s can be huge and in some cases random and unbounded. To cover this unbounded case, one can allow s to be infinite, with the understanding that only a finite (perhaps random) number of the uniforms in the sequence is actually used by the simulation. With this interpretation, one can rewrite

$$\mu = \mathbb{E}[X] = \int_0^1 \cdots \int_0^1 f(u_1, \ldots, u_s)\mathrm{d}u_1 \cdots \mathrm{d}u_s = \int_{(0,1)^s} f(\boldsymbol{u})\mathrm{d}\boldsymbol{u} = \mathbb{E}[f(\boldsymbol{U})], \tag{1.16}$$

so $X = f(\boldsymbol{U})$ is an unbiased estimator of μ, where $\boldsymbol{u} = (u_1, \ldots, u_s)$ represents a point in $(0,1)^s$, and $\boldsymbol{U} \sim U(0,1)^s$ is a random point uniformly distributed over the unit hypercube. We take this unit hypercube *open* to account for the fact that the function f sometimes becomes infinite when a coordinate u_j equals 0 or 1.

For small s, say up to 3 or 4, and appropriate smoothness conditions on f, there are efficient classical deterministic numerical integration techniques to approximate μ by some $\tilde{\mu}$, and deterministic bounds on the approximation error $|\tilde{\mu} - \mu|$ can be obtained in terms of measures of smoothness of f, such as a bound on its derivative of a given order; see Section 1.5.3. However, the convergence rate (and effectiveness) of these methods degrades rapidly with s. For large s, they are typically useless and Monte Carlo is often the best (or only) alternative.

Example 1.20 In the stochastic activity network considered in Example 1.4, there are 13 independent random variables to generate for simulating the network, so μ can be written as an integral over the 13-dimensional unit hypercube $(0,1)^{13}$, as follows. To each point $\omega = \boldsymbol{U} = (U_1, \ldots, U_{13}) \in (0,1)^{13}$ there corresponds a vector $\boldsymbol{Y} = (Y_1, \ldots, Y_{13}) = (F_1^{-1}(U_1), \ldots, F_{13}^{-1}(U_{13}))$ of activity durations, and to each such vector $\boldsymbol{Y}$ there corresponds a length T of the longest path in the network. Thus, we can write $T = f_1(\boldsymbol{U})$ for some function $f_1 : (0,1) \to [0,\infty)$ which is computed by the simulation program. If the goal is to estimate $\mu = \mathbb{E}[T]$, then $X = T$ and the function f in (1.16) can be taken as $f = f_1$. If the goal is to estimate $\mu = \mathbb{P}[T > x] = \mathbb{E}[\mathbb{I}(T > x)]$ instead, then we can define $X = f(\boldsymbol{U}) = \mathbb{I}(T > x) = \mathbb{I}(f_1(\boldsymbol{U}) > x)$, which is 1 if $T > x$ and 0 otherwise. In both cases, μ is written as in (1.16) and computing μ amounts to computing a 13-dimensional integral, which is difficult, but we can easily use simulation to *estimate* μ. Note that if we generate one or more of the Y_j's by another method than inversion, this changes the function

f but not μ. One can also interpret ω as $\boldsymbol{Y}$ instead of $\boldsymbol{U}$, and redefine $\mathbb{P}$ accordingly, as the product of the densities of $Y_1, \ldots, Y_{13}$. □

Example 1.21 We return to the Asian option valuation problem introduced in Example 1.11, under the GBM model, with payoff given in (1.10). To simulate the payoff of that option, each Z_j can be generated by inversion, i.e., $Z_j = \Phi^{-1}(U_j)$ where the U_j's are i.i.d. $U(0,1)$. The option price can be written as an integral over the s-dimensional unit hypercube as in (1.16), with $s = d$:

$$\begin{aligned} \mu &= v(s_0, T) = \int_{(0,1)^s} f(\boldsymbol{u}) \mathrm{d}\boldsymbol{u} \\ &= e^{-rT} \int_{(0,1)^s} \max\left(0, \frac{1}{s}\sum_{i=1}^{s} s_0 \exp\left[(r - \sigma^2/2)t_i \right.\right. \\ &\qquad \left.\left. + \sigma \sum_{j=1}^{i} \sqrt{t_j - t_{j-1}}\Phi^{-1}(u_j)\right] - K\right) \mathrm{d}u_1 \ldots \mathrm{d}u_s \end{aligned} \tag{1.17}$$

where $f(\boldsymbol{u})$ is the expression inside the integral multiplied by the discount factor e^{-rT}. This integral can also be written with respect to the density ϕ of the standard normal distribution instead:

$$\begin{aligned} v(s_0, T) &= e^{-rT} \int_{\mathbb{R}^s} \max\left(0, \frac{1}{s}\sum_{i=1}^{s} s_0 \exp\left[(r - \sigma^2/2)t_i \right.\right. \\ &\qquad \left.\left. + \sigma \sum_{j=1}^{i} \sqrt{t_j - t_{j-1}} z_j\right] - K\right) \phi(z_1) \cdots \phi(z_s) \mathrm{d}z_1 \cdots \mathrm{d}z_s. \end{aligned} \tag{1.18}$$

This corresponds to the fact that the integral can be estimated by generating independent standard normals $Z_1, \ldots, Z_s$ by any method, and computing the discounted payoff as in the algorithm of Example 1.11. Eq. (1.17) can be obtained from Eq. (1.18) via the change of variable $u_j = \Phi(z_j)$, which gives $\mathrm{d}u_j = \phi(z_j)\mathrm{d}z_j$, for $j = 1, \ldots, s$. This highlights the fact that using the inversion method corresponds exactly to making this change of variable. This interpretation applies whenever a continuous random variable is generated by inversion. □

The simulation approach outlined in Examples 1.4 and 1.11 is equivalent to estimating these integrals by the Monte Carlo method that we now describe.

1.4.2 Crude Monte Carlo

The *(crude) Monte Carlo (MC)* estimator of the integral μ in (1.15) is

$$\hat{\mu}_n = \bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{1.19}$$

where n is the sample size (a fixed constant) and $X_1, \ldots, X_n$ are independent realizations of $X = X(\omega)$ where ω is generated from the probability measure $\mathbb{P}$. One has

$$\mathbb{E}[\hat{\mu}_n] = \mathbb{E}[X_i] = \mu$$

and

$$\mathrm{Var}[\hat{\mu}_n] = \frac{1}{n}\left[\mathbb{E}[X_i^2] - \mu^2\right] = \frac{\sigma^2}{n}$$

where

$$\sigma^2 \stackrel{\text{def}}{=} \mathbb{E}[X_i^2] - \mu^2 = \mathrm{Var}[X_i].$$

The estimator $\hat{\mu}_n$ is unbiased for μ, with variance $\sigma^2/n = \mathcal{O}(n^{-1})$.

In the setting where $X = f(\boldsymbol{U})$ where $\boldsymbol{U} \sim U(0,1)^s$, we have $X_i = f(\boldsymbol{U}_i)$ for each i where $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n$ are independent with $\boldsymbol{U}_i \sim U(0,1)^s$, and

$$\sigma^2 \stackrel{\text{def}}{=} \int_{(0,1)^s} f^2(\boldsymbol{u})\mathrm{d}\boldsymbol{u} - \mu^2.$$

Then the variance depends on the shape of f through $\int_{(0,1)^s} f^2(\boldsymbol{u})\mathrm{d}\boldsymbol{u}$. It is finite if and only if f is square-integrable over the unit hypercube. When f is bounded, f^2 is also bounded and the variance is always finite. If f is a constant, the variance is zero.

Example 1.22 For an example of an infinite-variance estimator, let $s = 1$ and $f(u) = u^{-1/2}$. Then $\mu = 2$ and $\int_{(0,1)^s} f^2(x)\mathrm{d}x = \int_0^1 x^{-1}\mathrm{d}x = \infty$, so $\mathrm{Var}[\hat{\mu}_n] = \mathrm{Var}[X_i] = \infty$. □

Example 1.23 We now give a simple example to illustrate the fact that the same integral can often be formulated in different ways, with different probability measures $\mathbb{P}$. Let Z be a standard normal random variable and suppose we want to estimate

$$\mu = \int_0^2 \phi(z)\mathrm{d}z = \mathbb{P}[0 \le Z \le 2] = \mathbb{E}[\mathbb{I}[0 \le Z \le 2]]$$

by MC, where ϕ is the standard normal density. Of course, this integral can be approximated more accurately by numerical quadrature methods than by MC; we just use this simple example to illustrate MC ideas.

A direct MC method proceeds as follows. For $i = 1, \ldots, n$, generate a standard normal Z_i by inversion via $Z_i = \Phi^{-1}(U_i)$, where $U_i \sim U(0,1)$, and compute $X_i = \mathbb{I}[0 \le Z_i \le 2]$. Then compute $\hat{\mu}_n$ as in (1.19). This is the proportion of Z_i values that fall into the interval $[0, 2]$.

For an alternative MC estimator of the same integral, we can make the change of variable $z = 2u$ and define $f(u) = 2\phi(2u)$ for all u, we obtain

$$\mu = \int_0^2 \phi(z)\mathrm{d}z = \int_0^1 2\phi(2u)\mathrm{d}u = \int_0^1 f(u)\mathrm{d}u.$$

Thus, to estimate μ, we can generate $U_1, \ldots, U_n$ i.i.d. $U(0,1)$, compute $X_i = 2\phi(2U_i)$ for $i = 1, \ldots, n$, and compute the estimator $\hat{\mu}_n$ as in Eq. (1.19).

These two estimators have the same expectation (both are unbiased), but not the same variance. □

1.4.3 Convergence

The *strong law of large numbers* and the *central limit theorem (CLT)*, stated in Section A.12 of the Appendix and restated below, are two fundamental results on the convergence of $\hat{\mu}_n = \bar{X}_n$ to μ. The CLT implies that for large n, asymptotically, the error $\bar{X}_n - \mu$ divided by $\sigma/\sqrt{n}$ has an invariant distribution. This means that the size of the error is $\mathcal{O}(\sigma/\sqrt{n})$ in a probabilistic sense.

Theorem 1.1 *Assume that* $\sigma^2 < \infty$.
(i) (Strong law of large numbers.) *One has* $\lim_{n\to\infty} \bar{X}_n = \mu$ *with probability 1.*
(ii) (Central limit theorem.)

$$\frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma} \;\Rightarrow\; \mathsf{N}(0,1) \tag{1.20}$$

(convergence in distribution to the standard normal) as $n \to \infty$. *This can be rewritten as*

$$\lim_{n\to\infty} \mathbb{P}\left[\frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma} \le x\right] = \Phi(x) \stackrel{\text{def}}{=} \mathbb{P}[Z \le x] \tag{1.21}$$

for all $x \in \mathbb{R}$, *where* Z *is a* $\mathsf{N}(0,1)$ *random variable.*

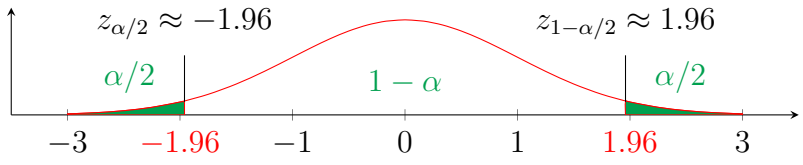
The CLT permits one to construct a confidence interval for μ, based on the normal distribution, and valid asymptotically as $n \to \infty$. The variance σ^2 in the CLT (1.20) is usually unknown, but the theorem still holds if σ^2 is replaced by its unbiased estimator (the empirical variance)

$$S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}_n)^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i^2 \; - n(\bar{X}_n)^2\right), \tag{1.22}$$

because $\lim_{n\to\infty} S_n^2 = \sigma^2$ with probability 1. For $0 < \alpha < 1$, a confidence interval at confidence level $1-\alpha$ (also called a $100(1-\alpha)\%$ confidence interval) on μ is then given by

$$(\bar{X}_n \pm z_{1-\alpha/2} S_n/\sqrt{n}),$$

where z_q denotes the q-quantile of the standard normal distribution: $z_q = \Phi^{-1}(q)$. In other words, $\alpha/2$ is the probability that a standard normal random variable exceeds $z_{1-\alpha/2}$. For example, to compute a 95% confidence interval, we take $\alpha = 0.05$ and we have $z_{1-\alpha/2} \approx 1.96$ and $z_{\alpha/2} = -z_{1-\alpha/2} \approx -1.96$, as illustrated here:



When n is large, $S_n \approx \sigma$, so the width of the confidence interval is approximately proportional to $\sigma/\sqrt{n}$. When n is small, invoking the CLT is not justified, but if we assume that the X_i are normally distributed, then $\sqrt{n}(\bar{X}_n - \mu)/S_n$ has the *Student distribution*

with $n-1$ degrees of freedom (see Section A.13). In this case, a valid confidence interval can be computed using the Student distribution instead of the normal. On the other hand, confidence intervals based on either the normal or Student distribution become invalid when the distribution of X_i is highly skewed, or is concentrated on just a few discrete values. Chapter 5 covers these issues in further detail and discusses alternative methods for these situations.

Example 1.24 For the stochastic activity network in Examples 1.4 and 1.20, we already gave 95% confidence intervals for $\mu = \mathbb{E}[T]$ and for $p_x = \mathbb{P}[T > x] = \mathbb{E}[\mathbb{I}[T > x]]$ for a given x, based on a sample size of $n = 10^5$. When estimating p_x, we have $X_i = \mathbb{I}[T_i > x]$, $\bar{X}_n = Y(x)/n$, and $S_n^2 = Y(x)(1-Y(x)/n)/(n-1)$. These X_i's are Bernoulli (binary) random variables, so they are far from being normally distributed, but if p_x is not too close to 0 or 1, then it is reasonable to assume that $\bar{X}_n$ is approximately normal, because n is large. □

Example 1.25 For another example, if we perform $n = 1000$ independent runs of the simulation for a given x and we observe $Y(x) = 882$, then $\bar{X}_n = 882/1000 = 0.882$, $S_n^2 \approx 0.1042$, and the normal approximation makes sense. It gives the 95% confidence interval $(\bar{X}_n \pm 1.96 S_n/\sqrt{n}) \approx (0.882 \pm 0.020) = (0.862, 0.902)$. This says (roughly) that our estimator of p_x has only two meaningful decimal digits: $\mu \approx 0.88$. The "2" in 0.882 is not significant. When reporting results of simulation or statistical experiments, it is customary to report only significant digits. Reporting several meaningless digits is bad style and can be misleading. For example, if we measure $\bar{X}_n = 31.376324384$ and $S_n/\sqrt{n} = 0.2$, then we should report something like $\bar{X}_n = 31.4 \pm 0.4$.

Suppose now that we observe $Y(x) = 998$ instead. Then, $\bar{X}_n = 0.998$ and $S_n^2 \approx 0.002$, which would give the 95% confidence interval (0.998 ± 0.0028) for p_x if we use the normal approximation. However, if p_x is near 0.998, then most of the probability mass of $Y(x)$ is concentrated on a few integer values, say from 995 to 1000, and therefore the distribution of $\bar{X}_n$ is clearly far from normal in this case, so we should not use the normal approximation to compute a confidence interval. One way to handle this when p_x is close to 1 is use the fact that $n - Y(x)$ has a $\mathsf{Binomial}(n, q)$ distribution where $q = 1 - p_x$ is very small. This distribution is well approximated by a $\mathsf{Poisson}$ distribution with mean nq, so the problem becomes that of computing a confidence interval for the mean of a Poisson distribution when this mean is small. Section 5.2.3 show how to do it. Example 1.29 is related to the present example. □

Example 1.26 *Confidence interval for the price of the Asian option.* For the numerical illustration at the end of Example 1.11, the estimator X_i is clearly far from normally distributed. It has a probability mass of approximately 0.535 at 0, and a density that resembles the histogram of Figure 1.6 over the positive real axis. This density is highly skewed. Thus, computing a confidence interval for the option price based on the normal distribution is reasonable only if n is large (we can then invoke the CLT). □

MC integration has the following important advantages over the classical deterministic numerical integration methods such as trapezoidal and Simpson rules (Section 1.5.3). These

advantages also stand to some extent when MC is compared with deterministic quasi-Monte Carlo methods (Section 1.5.3).

1. MC requires only a very weak condition on the integrand f, namely square integrability. The classical numerical methods give better convergence rates in small dimensions, but only under smoothness assumptions that are stronger than square integrability.

2. The convergence rate of the error does not depend on the dimension s (the convergence speed depends on s only indirectly via σ^2). With the classical methods, the convergence rate deteriorates quickly when s increases.

3. In most cases, one can easily estimate the error, in a probabilistic sense, and compute confidence intervals. The classical methods give deterministic error bounds, but these bounds are typically much too hard to compute explicitly or too loose to be useful.

4. When the integral represents the expectation of a random variable X, MC provides more information than just a point estimate of the expectation. It can also provide an estimate of the entire distribution of X.

One the other hand, the convergence rate of $\mathcal{O}(\sigma/\sqrt{n})$ is very slow. For each additional decimal digit of accuracy in the estimator, i.e., to divide the width of a confidence interval by 10, the sample size n (and the work) must be multiplied by 100. Thus, computing high-precision MC estimators becomes rapidly very costly. But in many cases, when s is large, there is just no alternative to MC. In Chapter 6, we will study efficiency improvement methods whose aim is mostly to reduce the constant σ^2, although some of the methods can also improve the convergence rate.

Our discussion of convergence and confidence intervals for MC estimators so far assumes that the model is exact, in the sense that the input probability distributions used in the simulation are perfectly known. We only account for the *sampling (or statistical) error*. But in real-life applications, the input distributions and their parameters are typically unknown; they are selected (or "learned") based on available data. This brings a second source of error, called the *modeling error*. Ideally, confidence intervals should account for both types of errors, otherwise they can give an overly optimistic confidence. This is further discussed in Chapter 5. [3]

1.4.4 Efficiency of simulation estimators

It is useful to have a notion of efficiency for simulation (or MC) estimators that takes into account both the work and the noise. Suppose that an estimator X is available to estimate some unknown quantity μ. The *bias* β, *variance* σ^2, *mean square error (MSE)*, *root mean square error (RMSE)*, and *relative error (RE)* of X are defined by

[3] From Pierre: To do...

$$\begin{aligned} \beta &= \mathbb{E}[X] - \mu; \\ \sigma^2 &= \mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]; \\ \mathrm{MSE}[X] &= \mathbb{E}[(X-\mu)^2] = \beta^2 + \sigma^2; \\ \mathrm{RMSE}[X] &= \sqrt{\mathrm{MSE}[X]}; \\ \mathrm{RE}[X] &= \mathrm{RMSE}[X]/|\mu|, \quad \text{for } \mu \neq 0. \end{aligned}$$

RMSE$[X]$ is also called the *absolute error*; it is an absolute measure of statistical accuracy of the estimator X, whereas RE$[X]$ is a measure of that accuracy relative to the size of the mean μ, which is typically more relevant than the absolute value when μ is very small (see Example 1.29).

We assume that the size of the computational effort required to compute X (e.g., in terms of CPU time) is a random variable (typically correlated with X) and we denote its mathematical expectation by $C(X)$. We call the product $C(X)\cdot\mathrm{MSE}(X)$ the *work-normalized MSE* of X and we define the *efficiency* of X by its inverse:

$$\mathrm{Eff}(X) = \frac{1}{C(X)\cdot \mathrm{MSE}(X)}. \tag{1.23}$$

An estimator Y is said to be *more efficient* than another estimator X if $\mathrm{Eff}(Y) > \mathrm{Eff}(X)$. *Efficiency improvement* means finding a more efficient estimator Y than the currently used estimator X in the above sense. The ratio $\mathrm{Eff}(Y)/\mathrm{Eff}(X)$ is the *efficiency improvement factor*. Often, both estimators are unbiased and are assumed to have roughly the same computational costs; then, improving the efficiency is equivalent to reducing the variance. For this reason, one often speaks of a *variance reduction technique (VRT)*. However, efficiency can sometimes be improved by *increasing* the variance slightly while reducing the computing cost more significantly.

If the computing cost is not taken into account, which makes sense when the two estimators X and Y under consideration require about the same computing effort, we call $\mathrm{Var}[X]/\mathrm{Var}[Y]$ the *variance reduction factor (VRF)* of Y with respect to X. It represents the factor by which the variance is reduced when using Y instead of X. In what follows, we give a few simple examples of variance reduction. Much more is given in Chapter 6.

- ♣ Example based on Example 1.23?
- ♣ Static reliability example?

Example 1.27 *A control variate.* [4] For the Asian call option model based on a geometric Brownian motion in Example 1.11, if we replace the *arithmetic average* $\bar{S} = (1/d)\sum_{j=1}^{d} S(t_j)$ in the payoff by the *geometric average* $Y = \prod_{j=1}^{d} S(t_j)^{1/d}$, then there is an explicit formula for $\mathbb{E}[e^{-rT}\max(0, Y-K)]$, which is the exact value of the option. (Since Y has a lognormal distribution, one can easily adapt the Black-Scholes formula; see Example 6.17 for the details.) Therefore, it suffices to estimate the difference between the values of the options based on the arithmetic and geometric averages, and add the exact (precomputed) value of the option based on the geometric average. An unbiased estimator of this difference is

[4] From Pierre: Move this to the CRN section.

$$e^{-rT}\left[\max\left(0,\,\bar{S}-K\right)-\max\left(0,\,Y-K\right)\right].$$

Here, the two averages $\bar{S}$ and Y can be obtained either from two independent sample paths $S(t_1),\ldots,S(t_d)$, or by using the same sample path for both. The variance is typically much smaller when using the same sample path, i.e., computing both $\bar{S}$ and Y from the same values of $S(t_1),\ldots,S(t_d)$. In realistic applications, the variance of this modified estimator can be up to a thousand times and sometimes up to a million times smaller than the variance of the standard MC estimator $e^{-rT}\max\left(0,\,\bar{S}-K\right)$ used in example Example 1.11, and it is not much more expensive to compute. Thus, this alternative estimator provides a significant efficiency improvement in this case. It is a special case of a control variate estimator (see Example 6.17), with the control variate coefficient equal to 1. □

Example 1.28 *Conditional Monte Carlo.* [5] Returning to the numerical illustration at the end of the reliability Example 1.5, here we give a simple way of building a better estimator for r. Suppose that in our simulation, we generate only $Y_3,\ldots,Y_{13}$, but not Y_1 and Y_2. From the values of $Y_3,\ldots,Y_{13}$, we can compute the indicator random variables I_1 and I_2, where I_j is 1 if node j is connected to node 8, and 0 otherwise. Given I_1 and I_2, the probability that nodes 0 and 8 are connected via node 1 is $I_1 r_1$, while the probability that they are not connected via node 1 but connected via node 2 is $(1-I_1r_1)I_2r_2$. These two events are disjoint. Therefore, the probability that these nodes are connected conditional on (I_1, I_2) is their sum

$$X_{\mathrm{e}} = \mathbb{P}[\Phi(\boldsymbol{Y}) = 1 \mid I_1, I_2] = I_1 r_1 + (1-I_1 r_1) I_2 r_2.$$

We have

$$\mathbb{E}[X_{\mathrm{e}}] = \mathbb{E}[\mathbb{E}[\Phi(\boldsymbol{Y}) \mid I_1, I_2]] = \mathbb{E}[\Phi(\boldsymbol{Y})] = r,$$

so X_{e} is an unbiased estimator of r. It is a *conditional Monte Carlo (CMC)* estimator. We will see in Section 6.6 that the variance of the CMC estimator is guaranteed to be smaller than that of the crude MC estimator $X = \mathbb{I}[\Phi(\boldsymbol{Y}) = 1]$.

To test this empirically, we simulated $n = 10^8$ independent replicates of $1-X$ and of $1-X_{\mathrm{e}}$ with $r_1 = r_2 = 0.95$ and $r_j = 0.999$ for $j > 2$, to compare the variances. Our estimates were $1-r \approx 2.5969\times10^{-3}$, $\mathrm{Var}[X] = \mathrm{Var}[1-X] \approx 2.59\times10^{-3}$, and $\mathrm{Var}[X_{\mathrm{e}}] = \mathrm{Var}[1-X_{\mathrm{e}}] \approx 6.45\times10^{-6}$. Thus, using CMC reduces the variance by a factor of about 400. This reduces the width of confidence intervals (for the same n) by a factor of about $\sqrt{400} = 20$. We computed 95% confidence intervals on $1-r$ based on a normal approximation with different values of n, for both estimators. Here are some of the intervals (multiplied by 10^3) that we obtained:

n	$10^3\times(1-\bar{X}_n)$	$10^3\times(1-\bar{X}_{\mathrm{e},n})$
10^3	$(1.205, 10.79)$	$(2.504, 2.876)$
10^6	$(2.519, 2.719)$	$(2.593, 2.605)$
10^8	$(2.592, 2.612)$	$(2.596, 2.597)$

The CMC confidence intervals on the right are much narrower than the MC ones on the left. The CMC estimator $1-\bar{X}_{\mathrm{e},n}$ is clearly more accurate than $1-\bar{X}_n$. The gain is significant here mostly because Y_1 and Y_2 are the two variables that contribute most to the variance

[5] From Pierre: Move this to separate intro-CMC section?

(they have a more important influence on the outcome), and we remove this variance contribution by computing explicitly the probability with respect to these two variables instead of generating them. If we choose two other variables instead, the gain would be more modest. □

The efficiency criterion (1.23) is not the only possibility, but it is widely agreed upon. Under the assumption that X is an unbiased estimator of μ, it is standard and was already proposed by Hammersley and Handscomb (1964). The following gives some justification for the definition (1.23).

Let $X_1, \ldots, X_n$ be an i.i.d. sample of size n, and let

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

One has $\mathrm{Var}[\bar{X}_n] = \mathrm{Var}[X_i]/n$, so the variance of $\bar{X}_n$ can be cut in half by doubling the sample size n, for example. Assuming that $C(\bar{X}_n)$ is proportional to n, doubling n also doubles $C(\bar{X}_n)$, so it does not change $\mathrm{Eff}(\bar{X}_n)$ if there is no bias. In general, if the sample size n is multiplied by some constant a, the variance is divided by the same constant a and the efficiency remains unchanged. Thus, for unbiased estimators, the efficiency measure (1.23) is invariant with respect to the number n of replications, which is a convenient property and suggests that this definition makes sense.

More generally, in case there might be a bias $\beta = \mathbb{E}[X_i] - \mu$, suppose that κ is the expected cost for computing each X_i in (1.19), so $C(\bar{X}_n) = \kappa n$, and that $\mathrm{Var}[X_i] = \sigma^2$. Then we have

$$\mathrm{Eff}[\bar{X}_n] = \frac{1}{\kappa n(\beta^2 + \sigma^2/n)} = \frac{1}{\kappa(n\beta^2 + \sigma^2)}. \tag{1.24}$$

If $\beta \neq 0$, then $\mathrm{Eff}(\bar{X}_n)$ decreases as a function of n. The definition (1.23) is based on the idea that variance can be traded off for squared bias, and vice-versa, without essentially altering the statistical accuracy of the estimator. The fact that $\mathrm{Eff}[\bar{X}_n]$ depends on the sample size in the biased case is somewhat annoying. This difficulty can be addressed by considering the asymptotic behavior of the efficiency when comparing estimators, as we do in Chapter 6.

♣ To be done in Chapter 6...

In practice, estimator efficiencies are typically unknown, because σ^2 and β (when $\neq 0$) are unknown. When significant bias is present, it can usually *not* be estimated precisely, because otherwise it would have been removed by simple subtraction from the estimator. The variance σ^2 can be estimated by the sample variance S_n^2 defined in (1.22) if $X_1, \ldots, X_n$ are i.i.d. copies of the estimator X. One must be careful, however; the variance of S_n^2 itself sometimes happens to be huge. The following pathological example may look exaggerated, but similar situations do happen, for example in the context of rare important events (Chapter 6). The example indicates that the cost, variance, and bias of an estimator do not tell the whole story; other properties such as skewness, higher moments, etc., may also be important (see Exercise 1.17). The definition of efficiency in (1.23), adopted here, is a compromise.

Example 1.29 Suppose we want to estimate $p = \mathbb{P}\{A\}$, the probability of occurrence of some rare event A for a complex model. The event A may correspond to having $A = \{T > x\}$

for some large x in Example 1.4, or a non-operational system in Example 1.5, or a loss of information due to buffer overflow in a communication switch, or an investor losing more than a certain (large) amount of money on a given day, etc. Let $X = \mathbb{I}[A]$, where $\mathbb{I}[A] = 1$ if the event A occurs during the simulation, $\mathbb{I}[A] = 0$ otherwise. The binary random variable X is a straightforward estimator of p, with $\mathbb{E}[X] = p$ and $\text{MSE}[X] = \text{Var}[X] = p(1-p)$. If p is very small, $\text{MSE}[X] \approx p$ is also very small. But obtaining an MSE smaller than p is trivial here. For example, just by taking $Y = 0$ as an estimator, we get $\text{Var}[Y] = 0$ and $\text{MSE}[Y] = p^2$, so a more useful estimator must have its MSE at least smaller than p^2. The reader can verify that $\bar{X}_n$ satisfies this condition if and only if $n > (1-p)/p$. More generally, to have a square relative error smaller than ϵ, we need $n > (1-p)/(p\epsilon)$.

For small p (and more generally when $\mathbb{E}[X]$ is near zero), instead of the MSE, it is more meaningful to consider the *relative MSE*, defined as $\text{MSE}[X]/(\mathbb{E}[X])^2 = \text{MSE}[X]/p^2$, or the *relative error* $\text{RE}[X] = \sqrt{\text{MSE}[X]}/p$. The idea is that the estimation error on p (or the width of a meaningful confidence interval on p) should not be larger than p, i.e., the square error should not be larger than p^2, and this square error can be measured roughly by $\text{MSE}[X]$. For the current example, $\text{RE}[X] = \sqrt{(1-p)/p}$, which increases to infinity as p approaches zero. This relative error could of course be divided by $\sqrt{n}$ by making n independent replications of the simulation, but keeping it under control when p is very small can be too costly. For instance, if $p \approx 10^{-10}$, we need $n \approx 10^{12}$ for a 10% relative error. Unless the average computing cost per run is extremely small, an alternative estimator, more efficient than X, is required.

The large relative error is only one facet of the problem here. Suppose that the average $\bar{X}_n$ of n i.i.d. copies of X is taken as an estimator of p and that a confidence interval for p is computed by assuming that $\bar{X}_n$ follows approximately the normal distribution, as is often done (see Section 1.4.3 and Chapter 5). If np is small, there is a large probability that $X_1 = \cdots = X_n = 0$, in which case both the sample mean $\bar{X}_n$ and the sample variance S_n^2 are 0, as we saw in Example 1.5 for $r_j = 0.999$, and the confidence interval has zero width. That probability is

$$\mathbb{P}[\bar{X}_n = S_n^2 = 0] = (1-p)^n \approx 1 - np$$

if np is small and n is large. Thus, confidence intervals based (naïvely) on normality are very unreliable in this context.

In fact, here, $n\bar{X}_n$ has the binomial distribution with parameters (n, p). This is well approximated by the normal distribution when both n and np are large. But if n is large and np is small, $n\bar{X}_n$ follows approximately the Poisson distribution with mean np, which is far from the normal. One can use this Poisson approximation to compute a confidence interval; see Section 5.2.3. The two main approaches for dealing with rare-event problems are importance sampling and splitting; the first is introduced in Section 1.6 and both are discussed in Chapter 6. □

Fortunately, for the great majority of simulation models, useful confidence intervals are not so difficult to obtain.

1.4.5 The hit-or-miss estimator

Suppose that $f : (0,1)^s \to [0,K]$ for some constant $K > 0$. Note that $\mu = \int_{(0,1)^s} f(\boldsymbol{u})\mathrm{d}\boldsymbol{u}$ is the volume of the region $\mathcal{S} = \{(\boldsymbol{u}, v) \in \mathcal{R} : v \le f(\boldsymbol{u})\}$ where $\mathcal{R}$ is the $(s+1)$-dimensional rectangle $\mathcal{R} = (0,1)^s \times [0,K]$. We can therefore estimate μ using the volume estimation method of Example 1.7 with this pair $(\mathcal{R}, \mathcal{S})$. For each i, we generate a point $(\boldsymbol{U}_i, V_i)$ uniformly in $(0,1)^{s+1}$, then $\boldsymbol{Y}_i = (\boldsymbol{U}_i, K\,V_i)$ is uniform over $\mathcal{R}$, and we put

$$B_i = \mathbb{I}[\boldsymbol{Y}_i \in \mathcal{S}] = \mathbb{I}[V_i K \le f(\boldsymbol{U}_i)] = \begin{cases} 1 & \text{if } V_i K \le f(\boldsymbol{U}_i); \\ 0 & \text{otherwise.} \end{cases}$$

The estimator

$$\tilde{\mu}_n = K\,\bar{B}_n = \frac{K}{n}\sum_{i=1}^{n} B_i \tag{1.25}$$

is called a *hit-or-miss* estimator for μ (Hammersley and Handscomb 1964).

We can compare this estimator with the standard MC estimator $\hat{\mu}_n$ in (1.19) with $X_i = f(\boldsymbol{U}_i)$, whose variance is $\mathrm{Var}[\hat{\mu}_n] = (\mathbb{E}[f^2(\boldsymbol{U})] - \mu^2)/n$. Since $0 \le f(\boldsymbol{u}) \le K$, one has $\mathbb{E}[f^2(\boldsymbol{U})] = \int_{(0,1)^s} f^2(\boldsymbol{u})\mathrm{d}\boldsymbol{u} \le \int_{(0,1)^s} K f(\boldsymbol{u})\mathrm{d}\boldsymbol{u} = K\mu$, and therefore

$$\mathrm{Var}[\hat{\mu}_n] \le \mathrm{Var}[\tilde{\mu}_n].$$

Furthermore, if $0 < f(\boldsymbol{u}) < K$ on a set $A \subseteq (0,1)^s$ of strictly positive size, then $\int_A f^2(\boldsymbol{u})\mathrm{d}\boldsymbol{u} < K\int_A f(\boldsymbol{u})\mathrm{d}\boldsymbol{u}$ and the variance of $\hat{\mu}_n$ is *strictly* smaller than that of $\tilde{\mu}_n$. However, the expected cost to compute B_i may be less than the expected cost to compute $f(\boldsymbol{U}_i)$, for instance if easily computed bounds are available for f and if the condition $V_i K \le f(\boldsymbol{U}_i)$ can be verified indirectly at a cheaper cost than that of computing $f(\boldsymbol{U}_i)$ explicitly. For example, if f is a square root, then computing B_i is equivalent to checking whether $V_i^2 K^2 \le f^2(\boldsymbol{U}_i)$, which bypasses the (costly) computation of the square root. (In this expression, the square should be computed by a *multiplication*, not by a more costly power function.) Thus, it is not always true that $\hat{\mu}_n$ is more efficient than $\tilde{\mu}_n$.

Example 1.30 When estimating π in Example 1.7, what we were doing was in fact using the hit-or-miss method to estimate the area μ of the yellow surface $\mathcal{S}$ shown there, using the unit square for $\mathcal{R}$ and $K = 1$. This area is also the integral $\mu = \int_0^1 \sqrt{1-u_1^2}\mathrm{d}u_1 = \mathbb{E}[f(U)]$ where $f(U) = \sqrt{1-U^2}$, so it can be estimated by MC via (1.19) with $X = \sqrt{1-U^2}$ where $U \sim U(0,1)$. We have just shown that the latter estimator has smaller variance than the hit-or-miss estimator. However, its computation involves a square root, so it is unclear if it is more efficient. This may depend on both the hardware and software that is used. □

1.5 Numerical Integration and Quasi-Monte Carlo

1.5.1 Deterministic integration rules

In this section, we assume that the goal is to estimate the integral of a function f over $(0,1)^s$ as in (1.16). The MC method evaluates f at n independent random points $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n$

uniformly distributed in $(0,1)^s$, and takes the average. But choosing the evaluation points completely at random is not necessarily the most intelligent way of sampling the space. There are better ways, at least in small dimensions. One general approach replaces the random points by a set of more carefully selected deterministic points $P_n = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\}$, where $\boldsymbol{u}_i \in [0,1]^s$ for each i, and approximates μ in (1.16) by

$$\bar{\mu}_n = \sum_{i=0}^{n-1} w_i f(\boldsymbol{u}_i), \tag{1.26}$$

where the w_i are real-valued *weights* that sum to 1. In this section, the points are indexed from 0 to $n-1$ instead of from 1 to n, and their coordinates are allowed to take the values 0 and (sometimes) 1: this is for compatibility with standard notation for QMC integration and popular quadrature rules such as those in (1.28) and (1.29). For QMC methods, the weights w_i are all equal to $1/n$, and the coordinates take their values in $[0,1)$. The integration error is $E_n = \bar{\mu}_n - \mu$. Here, both $\bar{\mu}_n$ and E_n are deterministic, not random variables.

1.5.2 One-dimensional numerical integration

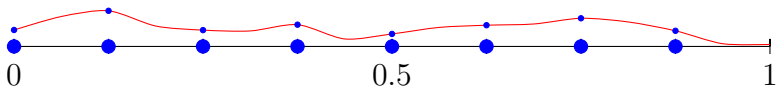
To start with a simple case, suppose $f : [0,1] \to \mathbb{R}$ is a one-dimensional bounded and n is fixed, so we just want to cover the unit interval $[0,1]$ evenly with n points. A simple solution takes $u_i = i/n$. This gives

$$P_n = \mathbb{Z}_n/n \stackrel{\text{def}}{=} \{0,\, 1/n,\, \ldots, (n-1)/n\} \tag{1.27}$$

and the corresponding approximation is

$$\bar{\mu}_n = \frac{1}{n} \sum_{i=0}^{n-1} f(i/n).$$

The integration points of P_n are illustrated here in blue for $n = 8$. The true function is in red.

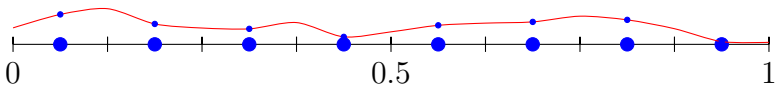


Suppose f is differentiable and has a bounded derivative $|f'(u)| \le K$ for $[0,1]$. A first-order Taylor expansion of $f(u)$ for $u \in [i/n, (i+1)/n]$ gives

$$f(u) = f(u_i) + f'(\xi)(u - u_i)$$

for some $\xi \in [i/n, u]$, and therefore $|f(u) - f(u_i)| \le K(u - u_i)$. By integrating both sides, we get $\int_{i/n}^{(i+1)/n} |f(u) - f(u_i)| \mathrm{d}u \le K(1/n)^2/2 = K/(2n^2)$, and by summing over all i, we find $|E_n| \le K/(2n) = \mathcal{O}(1/n)$. This convergence rate for the worst-case error is better than the probabilistic convergence rate of $\mathcal{O}(n^{-1/2})$ provided by standard MC.

A better strategy is to shift the points by $1/(2n)$ so there is a point $u_i = (2i+1)/n$ in the middle of each interval $[i/n, (i+1)/n]$:



This gives the *midpoint rule*, with integration points

$$P'_n \stackrel{\text{def}}{=} \{1/(2n),\, 3/(2n),\, \ldots,\, (2n-1)/(2n)\}.$$

If f is twice differentiable and has a bounded second derivative $|f''(u)| \leq K$ for $u \in [0,1]$, then a second-order Taylor expansion of f around u_i gives

$$f(u) = f(u_i) + f'(u_i)(u-u_i) + f''(\xi)(u-u_i)^2/2$$

for some ξ between u and u_i. By integrating with respect to u over $[i/n, (i+1)/n]$ and noting that $\int_{i/n}^{(i+1)/n} f'(u_i)(u-u_i)\mathrm{d}u = 0$ and $\int_{i/n}^{(i+1)/n}(u-u_i)^2/2 \leq 1/(24n^3)$, we have

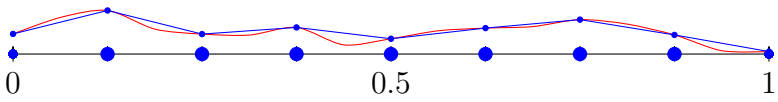
$$\int_{i/n}^{(i+1)/n} |f(u) - f(u_i)|\mathrm{d}u \leq \frac{K}{2}\int_{i/n}^{(i+1)/n}(u-u_i)^2\mathrm{d}u = \frac{K}{24n^3}.$$

Summing over $i = 0, \ldots, n-1$ gives $|E_n| \leq K/(24n^2) = \mathcal{O}(n^{-2})$. Thus, centering the points gives a better convergence rate.

If we allow *different weights* on the $f(\boldsymbol{u}_i)$, then we may use the well-known *trapezoidal rule*, defined as

$$\bar{\mu}_n = \frac{1}{n}\sum_{i=1}^{n}\frac{f(i/n) + f((i-1)/n)}{2} = \frac{1}{n}\left[\frac{f(0)+f(1)}{2} + \sum_{i=1}^{n-1} f(i/n)\right], \tag{1.28}$$

based on the $n+1$ points $P''_{n+1} = \{0, 1/n, \ldots, 1\}$ with weights $w_0 = w_n = 1/(2n)$ and $w_i = 1/n$ for $1 \leq i \leq n-1$. This rule evaluates f at the $n+1$ points, takes a linear interpolation over each interval $[i/n,\, (i+1)/n]$, for $i = 0, \ldots, n-1$, and integrates this piecewise linear function. This is illustrated below for $n = 8$; the piecewise-linear approximation is in blue.



With a Taylor expansion, one can show that this method has $|E_n| \leq K/(12n^2) = \mathcal{O}(n^{-2})$ if $|f''(u)|$ is bounded by a constant K. This is the same rate as for the midpoint rule, but the error bound is twice larger.

The midpoint and trapezoidal rules remove (integrate exactly) the linear terms in the Taylor expansion of E_n. Higher-order methods integrate exactly some higher-order terms. For example, *Simpson's rule* integrates exactly both the linear and quadratic terms, by using $P''_{n+1} = \{0, 1/n, \ldots, 1\}$ as before, taking a quadratic interpolation at the three evaluation points over each interval $[2i/n,\, 2(i+1)/n]$, $i = 0, \ldots, n/2-1$ (assuming that n is even), and integrating the piecewise quadratic interpolation. In the end, it just corresponds to using a different choice of weights than the trapezoidal rule. It is defined as

$$\begin{aligned}\bar{\mu}_n &= \frac{f(0) + 4f(1/n) + 2f(2/n) + \cdots + 2f((n-2)/n) + 4f((n-1)/n) + f(1)}{3n} \\ &= \frac{1}{3n}\left[f(0) + f(1) + 2\sum_{i=1}^{n/2-1} f(2i/n) + 4\sum_{i=1}^{n/2} f((2i-1)/n)\right].\end{aligned} \tag{1.29}$$

By using again a Taylor expansion, one can show that this gives $|E_n| \le K/(180n^4) = \mathcal{O}(n^{-4})$ if the fourth derivative of f is bounded in absolute value by K (Davis and Rabinowitz 1984, Press et al. 1992).

The trapezoidal and Simpson rules are special cases of a general class of methods known as composite, closed, *Newton-Cotes quadrature rules*, which approximate the integral by a weighted average of function evaluations at equally spaced points. The weights are determined in a way that Lagrange polynomials up to a given order, say α, are integrated with zero error (Press et al. 1992). This is achieved by interpolating the $\alpha + 1$ evaluations of f over each interval $[\alpha i/n,\, \alpha(i+1)/n]$, $i = 0, \ldots, n/\alpha - 1$ (assuming that α divides n), and integrating this piecewise polynomial interpolation. For example, Simpson's rule and Boole's rule provide convergence rates of $\mathcal{O}(n^{-4})$ and $\mathcal{O}(n^{-6})$ for $\alpha = 3$ and $\alpha = 4$, respectively, when f has bounded derivatives of order α. Error bounds can be obtained in general by doing a Taylor expansion of f, with an error term that involves the derivative of order α, over each subinterval of the form $[\alpha i/n,\, \alpha(i+1)/n]$.

Gaussian rules and *Clenshaw-Curtis* rules use unequally spaced points instead, with a higher density of points near the extremities of the interval. They can provide a convergence rate of $\mathcal{O}(n^{-2\alpha})$ when f has bounded derivatives of order 2α. They are generally more stable and accurate than the Newton-Cotes rules, but also more complicated to implement, and not always convenient, because of the unequal spacing. For all these quadrature rules, the actual error is typically very difficult to estimate, and the error bounds are often too hard to compute.

♣ Other methods: adaptive rules, etc.

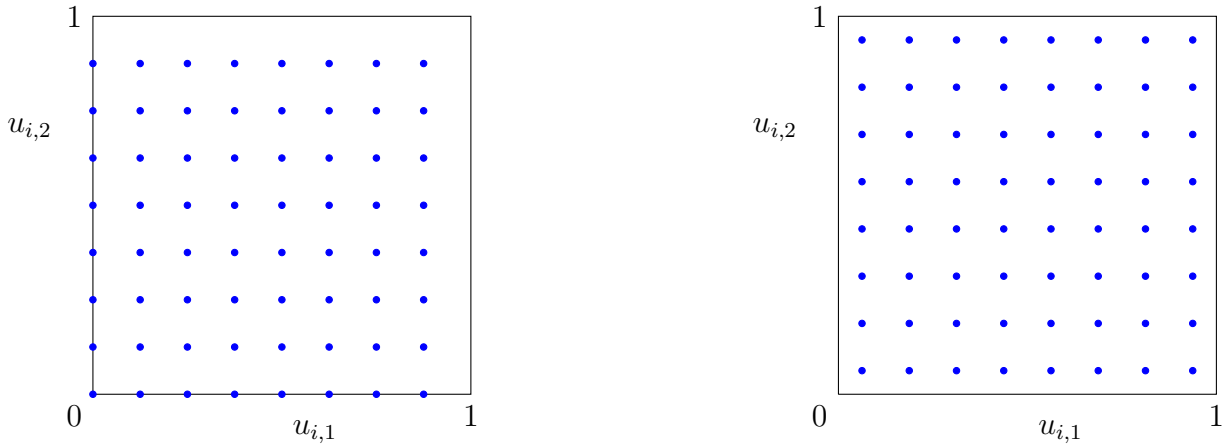


Fig. 1.12. A rectangular grid in $s = 2$ dimensions with $d_1 = d_2 = 8$, raw on the left and centered on the right.

The above discussion is for one-dimensional functions only. For functions f in $s > 1$ dimensions, a straightforward extension of $P_n = \mathbb{Z}_n/n$ is a *rectangular grid* of cardinality $n = d_1 \cdots d_s$ defined by

$$P_n = \{(i_1/d_1, \ldots, i_s/d_s) \text{ such that } 0 \le i_j < d_j \text{ for each } j\},$$

for some positive integers $d_1, \ldots, d_s$ (see the left panel of Figure 1.12 for $s = 2$ and $d_1 = d_2 = 8$). This grid can be centered by adding $(1/(2d_1), \ldots, 1/(2d_s))$ to all the points, to produce an *s-dimensional midpoint rule*, as shown in the right panel of Figure 1.12. For this rule, we have the following error bound:

Proposition 1.2 *Suppose that* $f : [0,1)^s \to \mathbb{R}$ *satisfies*

$$\sup_{\boldsymbol{u}\in[0,1]} \left| \frac{\partial^2 f(\boldsymbol{u})}{\partial u_j^2} \right| \leq K_j < \infty$$

for $j = 1, \ldots, s$. *Then the integration error for this* f *with the* s*-dimensional midpoint rule satisfies*

$$|E_n| \leq \sum_{j=1}^{s} \frac{K_j}{24 d_j^2}. \tag{1.30}$$

If we take $d_j = d$ *for all* j, *so* $n = d^s$, *and put* $K = K_1 + \cdots + K_s$, *this gives*

$$|E_n| \leq K/(24d^2) = K n^{-2/s}/24 = \mathcal{O}(n^{-2/s}). \tag{1.31}$$

Proof. To prove (1.30), first note that the n points partition $[0,1)^s$ into n subrectangles of dimensions $d_1^{-1} \times \cdots \times d_s^{-1}$, with one point at the center of each subrectangle. Let R_i be one of these subrectangles, with center $\boldsymbol{u}_i = (u_{i,1}, \ldots, u_{i,s})$. We want to bound $|f(\boldsymbol{u}) - f(\boldsymbol{u}_i)|$ for all $\boldsymbol{u} = (u_1, \ldots, u_s) \in R_i$. We can write

$$\begin{aligned} |f(\boldsymbol{u}) - f(\boldsymbol{u}_i)| \;\leq\; & |f(\boldsymbol{u}) - f(u_{i,1}, u_2, \ldots, u_s)| \\ & + |f(u_{i,1}, u_2, \ldots, u_s) - f(u_{i,1}, u_{i,2}, \ldots, u_s)| + \cdots \\ & + |f(u_{i,1}, \ldots, u_{i,s-1}, u_s) - f(u_{i,1}, \ldots, u_{i,s-1}, u_{i,s})|. \end{aligned}$$

We can bound the jth term on the right of this expression by noting that only coordinate j changes in this term, so from the bound for the one-dimensional midpoint rule over the interval of length $1/d_j$, the integral of this term with respect to u_j in the box R_i is bounded by $K_j/(24d_j^3)$. Integrating this bound over R_i with respect to the other coordinates and summing up then gives

$$\int_{R_i} |f(\boldsymbol{u}) - f(\boldsymbol{u}_i)| \mathrm{d}\boldsymbol{u} \leq \sum_{j=1}^{s} \frac{K_j}{24 d_j^2 n}.$$

Since we have n such rectangles, we multiply by n and obtain (1.30).

In $s = 4$ dimensions, this rate is the same as for MC. In $s < 4$ dimensions, it is better, while in $s > 4$ dimensions it is worse.

One can define a trapezoidal s-dimensional rule with $n = d^s$ points in a similar way, and the error bound also converges as $\mathcal{O}(n^{-2/s})$. But those rectangular grids are obviously impractical for large s. If $s = 100$, for example, even with $d_j = 2$ for all j (the smallest non-trivial value for the midpoint rule) we already have $n = 2^{100}$ points. No current computer can enumerate that number of points even with millions of years of CPU time. Another

drawback of rectangular grids is that when the points are projected to a single coordinate or to a small subset of the s coordinates, many points are projected onto each other. This usually translates into a loss of efficiency, as show in the next example. [6]

Example 1.31 Consider a function $f : [0,1)^s \to \mathbb{R}$ of the form

$$f(u_1, \ldots, u_s) = \sum_{j=1}^{s} f_j(u_j) + \text{other terms that all depend on more than one coordinate.}$$

This is a sum of s univariate functions, plus other terms. The integration error for this function is the sum of the error for the s univariate terms, plus the error for the other terms. With the s-dimensional midpoint rule, the terms of the first sum are evaluated only at the d distinct values $\{0, 1/d, \ldots, (d-1)/d\}$ (the projections of the points over one coordinate), so the error for these terms will be $\mathcal{O}(d^{-2}) = \mathcal{O}(n^{-2/s})$ instead of the $\mathcal{O}(n^{-2})$ that we would get with the one-dimensional midpoint rule. And if we use the unshifted points as in the left panel of Figure 1.12, the error with the s-dimensional rule is $\mathcal{O}(d^{-1}) = \mathcal{O}(n^{-1/s})$, compared with $\mathcal{O}(n^{-1})$ for the one-dimensional rule. Therefore, we clearly want to avoid losing points in the projections. □

1.5.3 Quasi-Monte Carlo

Quasi-Monte Carlo (QMC) methods are high-dimensional quadrature rules that avoid the superposition of points in the projections. These methods normally use equal weights $w_i = 1/n$ and a carefully selected deterministic point set $P_n = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\}$ in $[0,1)^s$ that cover the unit hypercube $(0,1)^s$ very uniformly without the gaps and clusters that occur with random points. They approximate μ in (1.16) by

$$\bar{\mu}_n = \frac{1}{n} \sum_{i=0}^{n-1} f(\boldsymbol{u}_i). \tag{1.32}$$

The points of P_n are known as *quasi-random* points. This name may be misleading, because no attempt is made to imitate randomness, but it has become a *de facto* standard term in the literature. In this book, we restrict ourselves to QMC methods with equal weights as in (1.32).

A necessary and sufficient condition for not losing points in the projections is that all one-dimensional projections contain n distinct points. One simple way to achieve this is to construct the points so that each one-dimensional projection is the set $\mathbb{Z}_n/n = \{0, 1/n, 2/n, \ldots, (n-1)/n\}$. Of course, these values must be enumerated in a different order for the different coordinates, otherwise all the points will be on the main diagonal of the unit hypercube (all coordinates of each point will be the same). Thus, one must have a different permutation of $\mathbb{Z}_n/n$ for each of the s coordinates. The goal is to select these permutations in a way that the resulting point set P_n also has high uniformity over $[0,1)^s$. We will look at two types of constructions that can achieve this: lattice rules and digital nets.

[6] From Pierre: We could give a numerical example with a comparison with MC, either here or in Section 1.5.3.

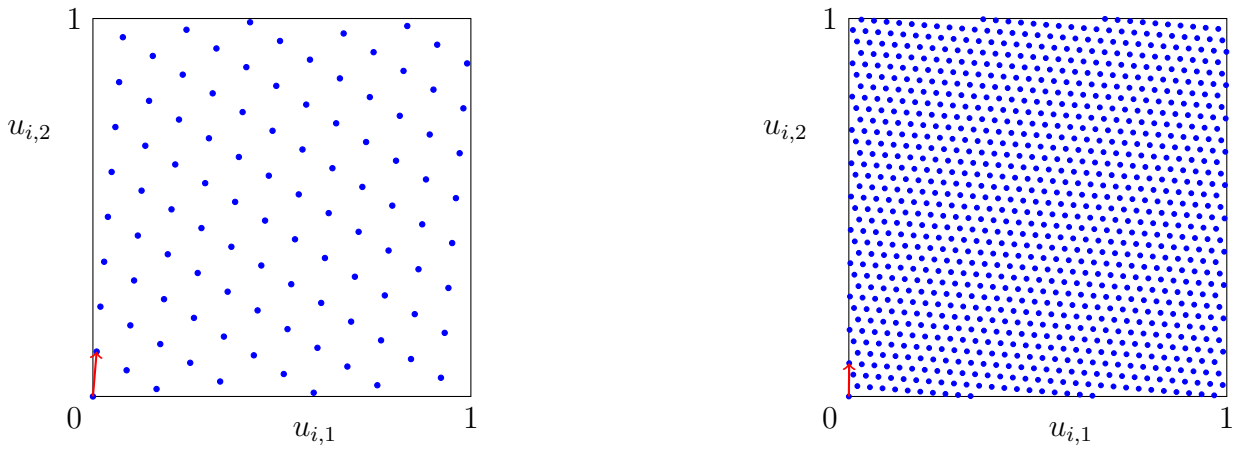


Fig. 1.13. Lattice points set in $s = 2$ dimensions with $n = 101$ (left) and $n = 1021$ (right).

1.5.4 Lattice Rules

Example 1.32 We start with a simple example in two dimensions. Suppose we want to integrate f over the unit square $(0, 1)^2$. A simple idea is to take P_n as the set of $n = m = 101$ points displayed in Figure 1.8 (left), which cover the unit square quite evenly. We see those points again in the left panel of Figure 1.13. These 101 points are all the pairs of two successive values produced by the LCG of Example 1.13, from all 101 possible initial states (including 0). Recall that when the LCG is in state x, the next state is $ax \bmod m = 12x \bmod 101$, and the corresponding pair in Figures 1.8 and 1.13 is $(x/m, (ax \bmod m)/m) = (x/m, (ax/m) \bmod 1)$. This means that P_n can be written as follows (here we use i and n in place of x and m to follow the QMC notation):

$$\begin{aligned} P_n &= \{\boldsymbol{u}_i = (u_{i,1}, u_{i,2}) = (i/m,\ (ai/m) \bmod 1) : i = 0, \ldots, n-1\} \\ &= \{\boldsymbol{u}_i = (i/101,\ (12i/101) \bmod 1) : i = 0, \ldots, 100\} \\ &= \{(0,0),\ (1/101, 12/101), (2/101,\ 24/101), \ldots, (100/101,\ 89/101)\}. \end{aligned}$$

These points are the intersection of a *integration lattice* with the unit square $[0, 1)^2$ (the lattice itself extends the same pattern of points to infinity in all directions). We may approximate the integral μ by the average (1.32) using this P_n. This type of integration rule is called a *lattice rule*. If f is bounded and smooth over the unit square $[0, 1)^2$, this is likely to give a more accurate estimate of μ than an average over $n = 101$ independent random points.

The point $(0, 0)$ may cause a problem in some situations. For example, for the Asian option in Example 1.11 with $d = 2$ observation times, if we evaluate the estimator at $(u_1, u_2) = (0, 0)$, we obtain $(z_1, z_2) = (\Phi^{-1}(u_1), \Phi^{-1}(u_2)) = (-\infty, -\infty)$ for the pair of normal random variates, and the simulation program may not execute correctly to completion when encountering such values (even though the payoff is zero at that point). A simple solution is to remove the point $(0, 0)$ from P_n and use only the $n = m - 1 = 100$ other points.

A larger two-dimensional lattice point set P_n is illustrated on the right panel of Figure 1.13. This one also comes from a LCG, but now with $n = m = 1021$ and $a = 90$. It covers the unit square very nicely. Here one has

$$P_n = \{\boldsymbol{u}_i = (i/1021,\ (90i/1021) \bmod 1) : i = 0, \ldots, 1020\}.$$

Note that for both point sets, if we project the points on a single coordinate, either the first or the second, we get the one-dimensional point set $\mathbb{Z}_n/n$ as in (1.27). There is no superposition of points. One can also think of "centering" the lattice points by adding $(1/(2n), 1/(2n))$ to all of them. Each one-dimensional projection then gives a midpoint rule in one dimension. Another advantage of this centering is there is no point at $(0, 0)$ anymore, so no need to remove a point. □

Lattice rules can also be defined in $s > 2$ dimensions. One can select an integer $n > 0$ and a s-dimensional vector $\boldsymbol{a} = (a_1, \ldots, a_s)$ with coordinates in $\{2, \ldots, n-1\}$, and define

$$P_n \ = \ \{\boldsymbol{u}_i = (i\boldsymbol{a} \bmod n)/n = i\boldsymbol{v} \bmod 1 :\ i = 0, \ldots, n-1\}, \tag{1.33}$$

where $\boldsymbol{v} = \boldsymbol{a}/n$. This is called a *lattice rule of rank-1* (all the points are multiples of the same vector, modulo 1). As a special case, one can take $a_j = a^j \bmod n$ for $j = 1, \ldots, s$. This gives what is called a *Korobov lattice rule*. For this special case, the point set P_n is the same as the set Ψ_s of all vectors of s successive values that can be produced by an LCG with modulus $m = n$ and multiplier a, from all possible initial states (including 0), as defined in Section 1.3.

For the lattice point set P_n in (1.33), the jth coordinate of $\boldsymbol{u}_i$ takes each value in $\mathbb{Z}_n/n$ exactly once when i goes from 0 to $n-1$ if and only if $\gcd(a_j, n) = 1$, i.e., a_j and n have no common factor. For a Korobov rule, this holds for all $j \geq 1$ if $\gcd(a, n) = 1$. This was the case for our two examples in Figure 1.13. To prove that this holds in general, suppose the points number i and i' have the same coordinate j, where $0 \leq i \leq i' < n-1$. That is, $0 = (ia_j - i'a_j) \bmod n = (i - i')a_j \bmod n$. Since a_j has no common factor with n, $i - i'$ must be a multiple of n, which implies that $i = i'$. In other words, each coordinate visits the same set of numbers as in (1.27), but these numbers may be visited in a different order by the different coordinates. If this holds and we add $1/(2n)$ to each coordinate of each point, then each coordinate visits each number in the set P'_n used for the midpoint rule instead of $P_n = \mathbb{Z}_n/n$. With the random shift modulo 1 that we will add in Section 1.5.6, these two sets will become equivalent. Lattice rules are examined further in Section 6.10.9.

Example 1.33 Here we try lattice rules on the Asian option example given at the end of Example 1.11. We use the same parameter values as in the numerical example given there, but we first try a two-dimensional version with $d = 2$, $t_1 = 1/2$, and $t_2 = T = 1$, before looking at the original example in $d = 12$ dimensions.

For $d = 2$, the exact value (computed via extensive simulations) is $\mu \approx 17.0958$, and the MC estimator of μ has variance $\mathrm{Var}[X_i] \approx 934.0$ (per simulation run). Using QMC with the $n = 100$ nonzero lattice points in the left panel of Figure 1.13, we obtain $\bar{\mu}_{100} = 17.6302$, so the squared error is $\mathrm{MSE}[\bar{\mu}_{100}] = (17.6302 - 17.0958)^2 = 0.2856$. With MC, if we use the same number of independent simulation runs, we obtain $\mathrm{MSE}[\hat{\mu}_{100}] = \mathrm{Var}[\hat{\mu}_{100}] \approx 9.34$.

Equivalently, we can multiply the MSE of the QMC estimator by 100, for a fair comparison with $\mathrm{Var}[X_i]$: 28.56 vs 934. The CPU times for computing $\bar{\mu}_{100}$ and $\hat{\mu}_{100}$ are approximately the same (the QMC estimator is actually a bit faster to compute). Therefore, QMC improves the efficiency by a factor of $934/28.56 \approx 32$.

Encouraged by this success, we tried a Korobov rule with a larger n. With $n = 65521$ and $a = 944$, whose corresponding two-dimensional lattice also has good uniformity, we obtain $\bar{\mu}_{65520} = 17.0963$. This gives a MSE of 2.5×10^{-7}. If we assume again that the computing times per run are about the same, the efficiency is improved by a factor of approximately $934.0/(65520 \times 2.5 \times 10^{-7}) \approx 5.7 \times 10^4$. We are 57 thousand times more efficient!

For the case where $d = 12$, the exact value is $\mu \approx 13.122$ and the MC estimator X_i has variance $\mathrm{Var}[X_i] = 516.3$. For the Korobov lattice points with $n = 101$ and $a = 12$, we obtain $\bar{\mu}_{100} = 12.4116$, whereas with $m = 65521$ and $a = 944$, we have $\bar{\mu}_{65520} = 13.1193$. The corresponding square errors, multiplied by the numbers of evaluation points (for a fair comparison) are $100\,\mathrm{MSE}[\bar{\mu}_{100}] \approx 50.5$ and $65520\,\mathrm{MSE}[\bar{\mu}_{65520}] \approx 0.477$, respectively. Although less spectacular than for $d = 2$, we still have significant reductions of the MSE compared to MC, by factors of about 10 and 1000, respectively. □

1.5.5 Digital Nets

Digital nets offer another way to construct P_n so that each coordinate visits each value in $\mathbb{Z}_n/n$ exactly once when we enumerate the points, and the visiting order is different for the different coordinates. Digital nets can be defined in any prime base $b \geq 2$, although $b = 2$ is by far the most popular value because the computer implementation is particularly fast in that base. In this section, we assume $b = 2$.

To define a *digital net in base 2*, we choose two integers $w \geq k > 0$ and s *generating matrices* $\boldsymbol{C}_1, \cdots, \boldsymbol{C}_s$, which are binary matrices having w rows and k columns, and whose first k rows are linearly independent. The matrix $\boldsymbol{C}_j$ is used to define coordinate j of all the points $\boldsymbol{u}_i = (u_{i,1}, \ldots, u_{i,s})$, as follows. For point i, let $\boldsymbol{a}_i = (a_{i,0}, \ldots, a_{i,k-1})^\mathsf{t}$ be the vector that contains the k bits of i written in base 2, starting with the least significant one. That is,

$$i = a_{i,0} + a_{i,1}2 + \cdots + a_{i,k-1}2^{k-1}.$$

The bits $u_{i,j,\ell}$ of the binary expansion of $u_{i,j}$ are obtained by multiplying $\boldsymbol{a}_i$ by $\boldsymbol{C}_j$:

$$u_{i,j} = \sum_{\ell=1}^{w} u_{i,j,\ell} 2^{-\ell}$$

where

$$(u_{i,j,1} \ \ldots \ u_{i,j,w})^\mathsf{t} \ = \ \boldsymbol{C}_j \boldsymbol{a}_i \bmod 2.$$

This defines the point set $P_n = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\}$ for $n = 2^k$.

Suppose for now that $w = k$. Since the first k rows are linearly independent, $\boldsymbol{C}_j$ is an invertible matrix, and therefore it defines an invertible linear mapping (a bijection) from $\mathbb{Z}_n$ to $\mathbb{Z}_n$. Such a bijection corresponds to a permutation of the elements of $\mathbb{Z}_n$, or equivalently a permutation of $\mathbb{Z}_n/n$. In other words, the role of $\boldsymbol{C}_j$ is to determine the order in which the jth coordinate $u_{i,j}$ will visit all the values of $\mathbb{Z}_n/n$ when i goes from 0 to $n-1$.

If $w > k$, the first k bits of $u_{i,j}$ will be the same as for $w = k$, but there will be $w - k$ additional bits, whose effect is to add a non-negative number smaller than $2^{-k} = 1/n$ to this coordinate. Regardless of w, we will always have one value of $u_{i,j}$ in each interval $[i/n, (i+1)/n)$, for $i = 0, \ldots, n-1$. This holds for each j. Note also that since $\boldsymbol{a}_0 = \mathbf{0}$, we always have $\boldsymbol{C}_j \boldsymbol{a}_0 = \mathbf{0}$, so $u_{0,j} = 0$ for all j, and therefore $\boldsymbol{u}_0 = \mathbf{0}$ for any choice of the generating matrices. This summarizes the uniformity behavior of the one-dimensional projections.

The uniformity in more than one dimension will depend on how the $\boldsymbol{C}_j$'s interact with each other. We want different matrices $\boldsymbol{C}_j$ because we want a different order for the different coordinates j, and we want to select them in a way that the points $\boldsymbol{u}_i$ cover $[0,1)^s$ as evenly as possible. In s dimensions, the intervals $[i/n, (i+1)/n)$ are replaced s-dimensional intervals, which are rectangular boxes. We can partition the hypercube $[0,1)^s$ in n rectangular boxes of the same size, and try to have exactly one point $\boldsymbol{u}_i$ in each box. We can do that only for partitions of a specific type, for which the size of the box in each dimension is a negative power of 2.

In *two dimensions*, in particular, if $k = k_1 + k_2$ for $k_1, k_2 \geq 0$, and if we partition the first axis in 2^{k_1} equal intervals and the second axis in 2^{k_2} equal intervals, we obtain $n = 2^k$ rectangles of the same size, $2^{-k_1} \times 2^{-k_2}$. The first k_1 bits of $u_{i,1}$ and the first k_2 bits of $u_{i,2}$ determine in which box the point $\boldsymbol{u}_i$ will fall. There will be exactly one point in each box if and only if the vector formed by those $k_1 + k_2 = k$ bits takes each of the 2^k possible values when i goes from 0 to $n-1$. This holds if and only if the matrix formed by the first k_1 rows of $\boldsymbol{C}_1$ and the first k_2 rows of $\boldsymbol{C}_2$ is invertible. To obtain this property for all choices of k_1, it suffices to choose an arbitrary invertible $k \times k$ matrix $\boldsymbol{C}_1$, and reverse the order of its rows to obtain $\boldsymbol{C}_2$. Then the matrix formed by the first k_1 rows of $\boldsymbol{C}_1$ and the first $k_2 = k - k_1$ rows of $\boldsymbol{C}_2$ will have the same rows as $\boldsymbol{C}_1$, and will therefore be invertible.

Example 1.34 *Hammersley points.* For a simple illustration, let $s = 2$, $\boldsymbol{C}_1$ be the reflected identity matrix, and $\boldsymbol{C}_2$ the identity:

$$\boldsymbol{C}_1 = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & \iddots & 0 & 0 \\ 1 & \cdots & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{C}_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

$\boldsymbol{C}_2$ has the same rows as $\boldsymbol{C}_1$, but in reverse order. The resulting P_n is called the *Hammersley point set in base 2*. For $k = 8$, the $n = 2^8 = 256$ points $\boldsymbol{u}_i = (u_{i,1}, u_{i,2})$ are given in Figure 1.14, in binary. Here, $u_{i,1} = i/n$ whereas $u_{0,2}, u_{1,2}, \ldots, u_{255,2}$ are the first 256 terms of the *van der Corput sequence* en base 2. The first n numbers in this (infinite) sequence fill the interval $[0,1)$ quite evenly for any integer n large enough. Observe that the binary expansion of $u_{i,2}$ is the mirror image of that of $u_{i,1}$ (the bits are in reverse order). The 256 values are $\mathbb{Z}_n/n$ in both cases; they are enumerated in increasing order for $u_{i,1}$ and in a different order for $u_{i,2}$.

7 The corresponding point set is illustrated in the upper half of Figure 1.15. On the left, the unit square is partitioned into 256 rectangles of size 1/8 by 1/32 obtained by dividing the

[7] From Pierre: It might be better to take only 64 or 128 points here, to improve visibility.

i	$u_{i,1}$	$u_{i,2}$
0	.00000000	.0
1	.00000001	.1
2	.00000010	.01
3	.00000011	.11
4	.00000100	.001
5	.00000101	.101
6	.00000110	.011
⋮	⋮	⋮
254	.11111110	.01111111
255	.11111111	.11111111

Fig. 1.14. The Hammersley point set in base 2 with $n = 2^8 = 256$.

two axes into 8 and 32 equal intervals, respectively, and we see that each of these rectangles contains exactly one point. For this particular point set, the same is true if we partition the unit square into rectangles of size 2^{-k_1} by 2^{-k_2} where k_1 and k_2 are non-negative integers that sum to 8 (so we have $2^8 = 256$ rectangles). On the left side of the figure, we have $k_1 = 3$ and $k_2 = 5$. On the right side, we have $k_1 = 6$ and $k_2 = 2$. □

In s dimensions, if $k_1 + \cdots + k_s = k - t$ for $t \geq 0$ and if we partition the jth axis in 2^{k_j} intervals of equal sizes for $j = 1, \ldots, s$, we obtain 2^{k-t} rectangular boxes of equal sizes. We have exactly 2^t points in each box if and only if the matrix formed by the first k_j rows of $\boldsymbol{C}_j$ for each j has full rank (i.e., its $k - t$ rows are linearly independent). When this holds, we say that the points are $(k_1, \ldots, k_s)$-equidistributed in base 2. If this holds for $t = 0$, we have exactly one point per box. When this holds for all choices of $k_1, \ldots, k_s$ for which $k_1 + \cdots + k_s = k - t$, we say that the point set is a *digital (t, k, s)-net in base 2*. The smallest t for which this property holds is called the *t-value* of the net. We want it to be as small as possible. However, digital $(0, k, s)$-nets in base 2 do not exist in more than 3 dimensions. The smallest possible value of t increases with the dimension (Schmid and Schürer 2005). The projection of a digital net over a subset of the coordinates is also a digital net having its own t-value, which can be smaller than the t value of the whole net. One can therefore measure the quality by looking also at the t-values of several low-dimensional projections like pairs of coordinates, triples, etc.

Sobol' (1967) proposed a specific method to construct an infinite sequence of upper-triangular binary matrices $\boldsymbol{C}_1^\infty, \boldsymbol{C}_2^\infty, \boldsymbol{C}_3^\infty, \ldots$ each with an infinite number of rows and columns, and ones everywhere on the diagonal. For each $j \geq 1$ and any $k \geq 1$, the first k rows and k columns of $\boldsymbol{C}_j^\infty$ form an invertible matrix. The bits above the diagonal in the first few columns of each $\boldsymbol{C}_j$ are parameters that can be selected (their choice has an impact on the quality of the nets), and the other columns are determined by a recurrence, using a different recurrence for each j. These matrices can be used to construct a digital net in base 2 with $n = 2^k$ points for any k, in arbitrary dimension s. This construction effectively

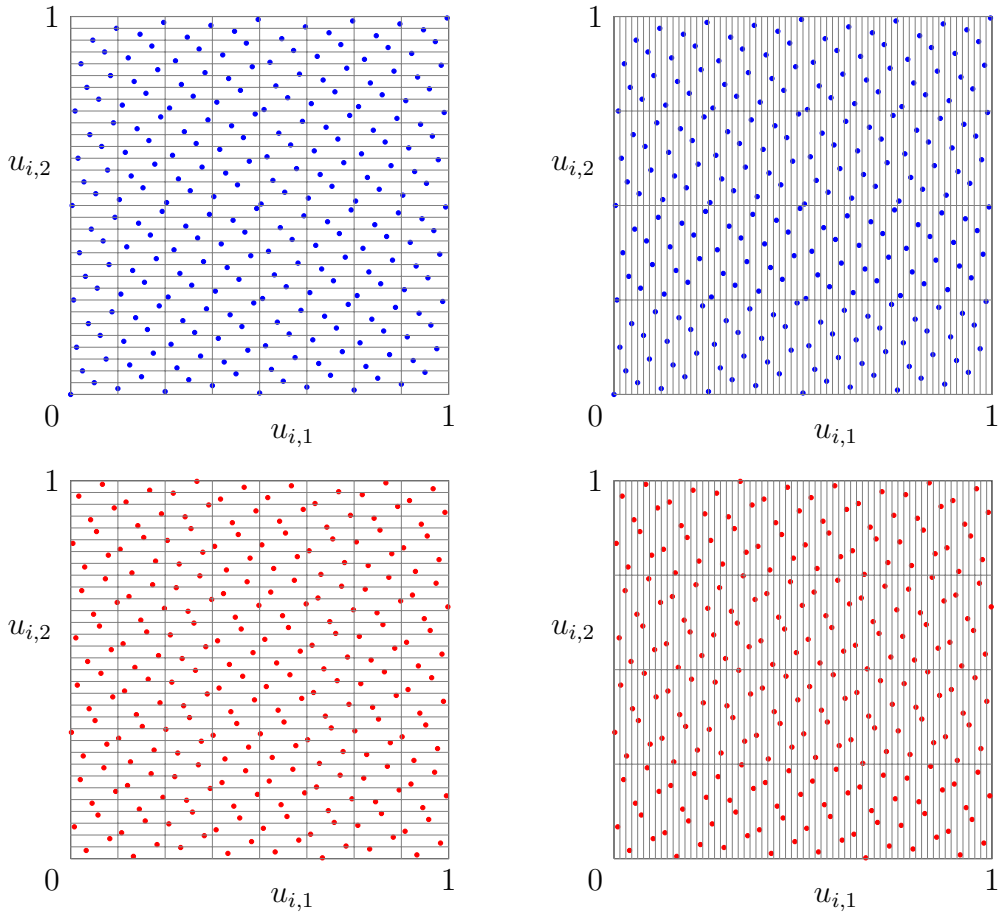


Fig. 1.15. The two-dimensional Hammersley point set in base 2, with cardinality $n = 2^8 = 256$ (upper panels) and the same point set after a random digital shift (bottom panels).

provides an *infinite sequence* of points and we can take the first $n = 2^k$ points for any k. When we increase k by 1, we double n by adding new points while keeping the previous ones. A popular choice for the parameters are those provided by Joe and Kuo (2008), who computed a table of parameter values for up to $s = 21,201$ dimensions, based on the t-values of the two-dimensional projections of the points. Section 6.10.5 gives more details on Sobol' sequences.

There are also other ways to construct digital nets and other ways to measure their quality, based on bounds on the integration error or on the variance for certain classes of functions f. See Section 6.10.

1.5.6 Randomized Quasi-Monte Carlo

The QMC approximation (1.32) is purely *deterministic*. The variance is zero, the integration error $E_n = \bar{\mu}_n - \mu$ can be seen as a bias, and the MSE is simply the square error E_n^2

(the square bias). But this error or bias is generally very hard to estimate. In the previous examples, we were able to compute the bias because we knew the answer, but this is not the case in practice. With the MC method, on the other hand, the error can be estimated (in a probabilistic sense) by computing a confidence interval.

A standard way of addressing this error estimation problem for QMC is to randomize the QMC point set P_n in a way that

(a) it retains its high uniformity over $(0,1)^s$ when taken as a set and
(b) each point $\boldsymbol{U}_i$ of the randomized point set has the uniform distribution over $(0,1)^s$ when taken individually.

This approach is known as *randomized QMC* (RQMC). We call a point set P_n that satisfies the conditions (a) and (b) an *RQMC point set*. Condition (b) ensures that $\mathbb{E}[f(\boldsymbol{U}_i)] = \mu$ for each i, by definition of μ. This implies that the *RQMC estimator*

$$\hat{\mu}_{n,\text{rqmc}} = \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i)$$

is an unbiased estimator of μ. If the randomization also preserves the high uniformity of P_n, then $\hat{\mu}_{n,\text{rqmc}}$ should also have smaller variance than the standard MC estimator $\hat{\mu}_n$. RQMC can then be viewed as a variance reduction method. Under certain conditions on f, it can be proved that the variance converges at a faster rate than the MC rate of $\mathcal{O}(1/n)$, as a function of n. The preferred randomization method depends on the type of point set. Most randomizations are designed to be compatible mainly with a specific class of point set constructions.

With a RQMC rule, the $\boldsymbol{U}_i$ are not independent, and therefore we *cannot* estimate $\text{Var}[\hat{\mu}_{n,\text{rqmc}}]$ by using the empirical variance S_n^2 of the $X_i = f(\boldsymbol{U}_i)$ defined in (1.22) and dividing it by n, as we did for MC. To estimate this variance and eventually compute a confidence interval on μ, we can randomize the same point set r times independently, and compute the sample mean $\bar{X}_r$ and the sample variance S_r^2 of the r corresponding (independent) realizations of $\hat{\mu}_{n,\text{rqmc}}$. Then, $\mathbb{E}[\bar{X}_r] = \mu$ and $\mathbb{E}[S_r^2] = \text{Var}[\hat{\mu}_{n,\text{rqmc}}] = r\text{Var}[\bar{X}_r]$ (Owen 1998, L'Ecuyer and Lemieux 2000). If r is large enough, we may assume that $\bar{X}_r$ is approximately normally distributed, but not if r is small, even if n is very large (L'Ecuyer, Munger, and Tuffin 2010). The computation of confidence intervals in this situation is discussed in Section 6.11.

Random shift modulo 1. One simple randomization method is a *random shift modulo 1*, proposed by Cranley and Patterson (1976) for lattice points: Generate a *single* point $\boldsymbol{U}$ uniformly distributed over $(0,1)^s$ and add it to each point of P_n, coordinate-wise, modulo 1. All points of P_n are shifted by the same amount. Moreover, for any fixed point $\boldsymbol{u}_i$, the randomized point $(\boldsymbol{u}_i + \boldsymbol{U}) \bmod 1$ has the uniform distribution over $(0,1)^s$. This implies that for any type of point set P_n, Condition (b) holds, and therefore the RQMC estimator $\hat{\mu}_{n,\text{rqmc}}$ is unbiased. When P_n corresponds to a lattice rule, the random shift preserves much of the structure and uniformity: it just shifts the lattice. The resulting integration rule is then called a *randomly-shifted lattice rule*. With a random shift, there is no need to discard the point $(0,0)$ from P_n.

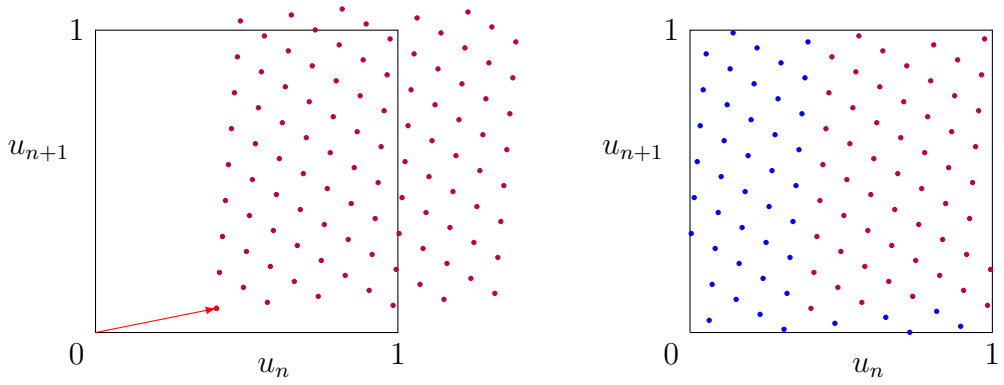


Fig. 1.16. Applying a shift modulo 1 to the point set P_n of Figure 1.8 (left).

Example 1.35 Figure 1.16 shows the point set P_n of Example 1.32, to which we have applied a shift of $\boldsymbol{U} = (0.40, 0.08)$, modulo 1. The points that fall outside the unit square after the shift, on the left panel of the figure, are brought back into the square by the modulo 1 operation, as seen on the right panel. □

Example 1.36 We applied a randomly-shifted lattice rule to the two-dimensional Example 1.32, with the same point sets, and $r = 1000$ independent randomizations in each case. Here, we keep the point $(0, 0)$, so we have $n = m$ points. The sample mean $\bar{X}_r$ and sample variance per run nS_r^2 were 17.076 and 77.9 for $n = 101$, and 17.095 and 4.03 for $n = 65521$. Compared with the MC variance of 934.0, the variance (and the MSE) is reduced by the factors 12.0 and 232, respectively. These factors are more modest than for the deterministic rule in Example 1.32. On the other hand, it is important to observe that they are estimated without knowing the exact mean. In general, the MSE is not always smaller with QMC; it depends on the interaction between the deterministic points and the function f.

We made the same experiment for the case of $d = 12$, examined in Example 1.33. Here, the sample mean and the sample variance per run are 13.089 and 94.9 for $n = 101$, and 13.122 and 23.0 for $n = 65521$. Compared with the MC variance of 516.3, the variance is reduced by the factors 5.4 and 22.4, respectively. □

Random digital shift. If we apply a random shift modulo 1 to a digital net like that of Figure 1.15, for which each box of a given partition contains exactly the same number of points (i.e., equidistribution holds for this partition), the random shift will generally destroy the equidistribution property. But if P_n is a digital net in base 2, and if the partition is defined by dividing each axis j into 2^j intervals for some integers j, then a slightly different type of randomization, called a *random digital shift in base 2*, preserves the equidistribution property. This randomization generates a single random point $\boldsymbol{U}$ in $(0, 1)^s$ and adds it to every other point as in the random shift modulo 1, but this time the addition modulo 1 is replaced by a bitwise addition modulo 2 (i.e., a bitwise exclusive-or), coordinate by coordinate. As an illustration in $s = 2$ dimensions, the randomized version of $\boldsymbol{u}_i$ below is $\boldsymbol{u}_i \oplus \boldsymbol{U}$:

$$\begin{aligned} \boldsymbol{u}_i &= (0.01100100\ldots,\ 0.10011000\ldots)_2 \\ \boldsymbol{U} &= (0.00100000\ldots,\ 0.01010001\ldots)_2 \\ \boldsymbol{u}_i \oplus \boldsymbol{U} &= (0.01000100\ldots,\ 0.11001001\ldots)_2. \end{aligned}$$

If we assume that $\boldsymbol{U}$ is uniform over $(0,1)^s$, which means that each coordinate of $\boldsymbol{U}$ has an infinite number of random bits, then for any fixed $\boldsymbol{u}_i$, $\boldsymbol{u}_i \oplus \boldsymbol{U}$ is also uniform over $(0,1)^s$. That is, each point has the uniform distribution after a random digital shift. In software implementations, the number of bits for each coordinate of $\boldsymbol{U}$ is restricted by machine precision; e.g., 53 bits, which is usually sufficient for practical purposes. We now explain why for a digital net in base 2, this randomization also preserves equidistribution. Saying that each box contains the same number of points in the upper left panel of Figure 1.15 is equivalent to saying that if we concatenate the first 3 bits of the first coordinate and the first 5 bits of the second coordinate of $\boldsymbol{u}_i$, then each of the 256 possibilities for this 8-bit string occurs exactly once when i runs through $\{0, 1, \ldots, 255\}$. This is because these 8 bits determine the box in which the point falls. Now, when we perform a bitwise exclusive-or of each point $\boldsymbol{u}_i$ with the same $\boldsymbol{U}$, whenever a given bit of $\boldsymbol{U}$ is 1, we flip the corresponding bit of all the $\boldsymbol{u}_i$'s (represented by a red $*$ below), whereas if this bit is 0, the corresponding bit of all the $\boldsymbol{u}_i$'s remains unchanged.

$$\begin{aligned} \boldsymbol{u}_i &= (0.***,\ 0.*****) \\ \boldsymbol{U} &= (0.001,\ 0.01010)_2 \\ \boldsymbol{u}_i \oplus \boldsymbol{U} &= (0.**{*},\ 0.*{*}***) \end{aligned}$$

But since a given bit is changed in the same way for all the points, each of the 256 possibilities for the 8-bit string that determines the box number will still appear exactly once, and so we will still have exactly one point per box, regardless of what $\boldsymbol{U}$ turns out to be. This also holds more generally if the partition is in 2^{k-t} boxes that contain 2^t points each. In essence, each time we flip one bit in one coordinate for all the points, we just exchange two sets of rectangular boxes in the partition. Flipping the first bit of the first coordinate exchanges the left half of the unit hypercube with the right half. Flipping the second bit exchanges the first quarter with the second quarter and the third quarter with the fourth quarter. And so on. This holds for each coordinate. If the boxes have width 2^{-k_j} along axis j and we flip bit ℓ of coordinate j where $\ell > k_j$, this flipping will only move the points inside their own boxes and not across different boxes. Therefore, if the digital net already has $(k_1, \ldots, k_s)$-equidistribution for given values of $k_1, \ldots, k_s$, the digital shift will always preserve this equidistribution.

The bottom panels of Figure 1.15 show the same point set as on the upper panels, but after a digital shift by

$$\begin{aligned} \boldsymbol{U} &= (0.1270111220,\ 0.3185275653) \\ &= (0.00100000100000111100,\ 0.01010001100010110000)_2, \end{aligned}$$

where the latter is the representation in base 2. This shift flips the bits number 3, 9, 15, 16, 17, 18 in the first coordinate, and bits number 2, 4, 8, 9, 13, 15, 16 in the second coordinate, for all the points

Applying a digital random shift to a digital net is sufficient to provide an unbiased estimator while preserving all the equidistribution properties that already exist for the net. But often, the equidistribution properties can be further improved by changing the generating matrices themselves. For instance, for the Sobol' points, certain low-dimensional projections over coordinates of higher indices are known to have poor uniformity. One approach to improve this behavior is to just randomize the Sobol' generating matrices in a way that their upper $k \times k$ parts remain invertible, and perhaps adding more rows at the same occasion. Matoušek (1999) proposed to do this via a *left matrix scramble (LMS)* as follows. Suppose the initial generating matrices are $k \times k$ and select $w \geq k$. Then for each j, generate a random $w \times k$ binary matrix $\boldsymbol{L}_j$ whose upper $k \times k$ part is lower triangular with ones on the diagonal. This upper part of $\boldsymbol{L}_j$ is invertible, and therefore the upper $k \times k$ part of $\boldsymbol{L}_j \boldsymbol{C}_j$ is also invertible. The matrices $\tilde{\boldsymbol{C}}_j = \boldsymbol{L}_j \boldsymbol{C}_j$ are the new "scrambled" generating matrices. This method is commonly implemented with $w = 31$ or 32, so each column of $\boldsymbol{L}_j$ and $\tilde{\boldsymbol{C}}_j$ can be represented as a 32-bit integer. It tends to do significantly better than using the Sobol' points directly. Note that LMS alone *does not* give an RQMC estimator, since each point does not have the uniform distribution over $[0,1)^s$ after the scramble. The point $\mathbf{0}$ remains $\mathbf{0}$, for instance. LMS must be followed by a random digital shift to provide an unbiased estimator. Other scrambling methods are discussed in Owen (2003).

Example 1.37 We made the same experiment as in Example 1.36, but now with P_n taken as the Hammersley point set in base 2, randomized by a LMS followed by a random digital shift in base 2. We tried $n = 2^{10} = 1024$ and $n = 2^{16} = 65536$. The sample mean and sample variance per run were 17.096 and 1.815 for $n = 1024$, and 17.096 and 0.034 for $n = 65536$. The variance reduction factors with respect to MC are 515 and 27,120, respectively. This beats the randomly-shifted lattice rules.

For $d = 12$, we tried a Sobol' net in 12 dimensions, constructed by taking i/n as the first coordinate and the first n points of a Sobol' sequence in 11 dimensions for the next coordinates, with the same randomization and same values of n as in the previous example. The sample mean and sample variance per run were 13.122 and 6.2 for $n = 1024$, and 13.122 and 1.7 for $n = 65536$. Compared with MC, the variance is reduced by the factors 84 and 304, respectively. □

We will examine RQMC more extensively in Section 6.11. We will see in particular that for any given integer $\alpha > 0$, for the class of functions f whose partial derivative of order α with respect to any subset of coordinates is square integrable over $[0,1)^s$, it is *proved* that one can construct RQMC point sets of various sizes n for which $\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}] = \mathcal{O}(n^{-2\alpha+\delta})$ for any $\delta > 0$. We will also point out software that can construct such point sets.

Note that for the Asian option example, the integrand f is not differentiable at points where the two terms inside the max are equal. For this reason, the proof of the faster rates mentioned earlier does not apply to this example. Nevertheless, in further experiments reported in Section 6.11 for this example with certain RQMC point sets and schemes, we will find (empirically) that the RQMC variance $\mathrm{Var}[\hat{\mu}_{n,\mathrm{rqmc}}]$ converges approximately as $\mathcal{O}(n^{-2})$, at least for the range of n that we can observe. That is, we observe a VRF of RQMC vs MS

that increases linearly with n, for this example. L'Ecuyer (2018) gives a short tutorial on RQMC. [8] [9]

1.6 Choice of Sampling Distribution

At first sight, it may appear essential that when we simulate a stochastic model, the basic random variates must be generated from the "correct" distributions specified in the model. For example, it seems natural that each Y_j in the stochastic activity network of Example 1.4 must be generated from its distribution F_j, and that the Z_j in the Asian option example must be generated necessarily from the $\mathsf{N}(0,1)$ distribution. The purpose of this section is to explain that this is *not* the case. In fact, we often have a lot of freedom in selecting the density or probability mass from which we sample a random variable. We are allowed to change the distributions, provided that we also modify the estimator in the appropriate way to avoid introducing bias. In certain situations, changing the distributions may improve the efficiency by huge factors. To explain the principle, we start with very simple illustrations, for which we know the answer in advance.

1.6.1 Examples and heuristics

Example 1.38 Let Y be an exponential random variable with rate parameter λ, whose density is $\pi(y) = \lambda e^{-\lambda y}$ for $y > 0$, and suppose we want to estimate $p = \mathbb{P}[Y \geq y_0]$, where $y_0 > 0$ is a constant. In this example, we know already that $p = e^{-\lambda y_0}$, so this is purely an academic illustration, but its simplicity makes it perfect to understand the key ideas. Applying the MC method naively, we can generate Y from density π and estimate p by the indicator $X = \mathbb{I}[Y \geq y_0]$. Normally, we would average n independent copies of X as usual to divide the variance by n, but let us focus on just a single realization of X. This X is a Bernoulli random variable with parameter p, whose variance is $\mathrm{Var}[X] = p(1-p)$.

At first we consider changing only the parameter λ of the exponential distribution. For an arbitrary $\lambda_0 > 0$ (to be selected later), let $\pi_0(y) = \lambda_0 e^{-\lambda_0 y}$ for $y > 0$. This is the exponential density with rate λ_0. We will sample from density π_0 instead of π. We can write

$$p = \int_0^\infty \mathbb{I}[y \geq y_0]\pi(y)\mathrm{d}y = \int_0^\infty \mathbb{I}[y \geq y_0]\frac{\pi(y)}{\pi_0(y)}\pi_0(y)\mathrm{d}y = \mathbb{E}[X_{\mathrm{is}}], \tag{1.34}$$

where

$$X_{\mathrm{is}} = \mathbb{I}[Y_0 \geq y_0]\frac{\pi(Y_0)}{\pi_0(Y_0)} = \mathbb{I}[Y_0 \geq y_0]\frac{\lambda}{\lambda_0}\exp[-(\lambda - \lambda_0)Y_0]$$

and Y_0 is an exponential random variable with rate λ_0. In the second integral of Eq. (1.34), we have just multiplied and divided by $\pi_0(y)$. Then we interpret this integral as an expectation with respect to the density π_0 and this gives $\mathbb{E}[X_{\mathrm{is}}]$. That is, X_{is} is an unbiased estimator of p, regardless of how we choose $\lambda_0 > 0$. The ratio of densities $\pi(Y_0)/\pi_0(Y_0)$ by which we

[8] From Pierre: Add a few lines (short preview) about Koksma-Hlawka inequalities.

[9] From Pierre: Add a discussion and a simple example showing that more variation (e.g., oscillation) degrades the RQMC efficiency, but not MC efficiency. Perhaps in Chapter 6.

multiply the original estimator (the indicator) to recover an unbiased estimator is called the *likelihood ratio*.

This new estimator has variance

$$\begin{aligned}
\mathrm{Var}[X_{\mathrm{is}}] &= \mathbb{E}[X_{\mathrm{is}}^2] - p^2 \\
&= \int_{y_0}^{\infty} \frac{\pi^2(y)}{\pi_0^2(y)} \pi_0(y) \mathrm{d}y \; - p^2 \\
&= \int_{y_0}^{\infty} \frac{\pi^2(y)}{\pi_0(y)} \mathrm{d}y \; - p^2 \\
&= \int_{y_0}^{\infty} \frac{\lambda^2}{\lambda_0} \exp[-(2\lambda - \lambda_0)y] \mathrm{d}y \; - p^2 \\
&= \begin{cases} \dfrac{\lambda^2}{\lambda_0(2\lambda - \lambda_0)} \exp[-(2\lambda - \lambda_0)y_0] - p^2 & \text{if } 0 < \lambda_0 < 2\lambda, \\ \infty & \text{otherwise.} \end{cases}
\end{aligned}$$

For $\lambda_0 = \lambda$ (no change in the sampling distribution), the variance remains equal to $p(1-p)$, where $p = e^{-\lambda y_0}$. For λ_0 slightly smaller than λ, we get a smaller variance. When λ_0 approaches 0 or 2λ, on the other hand, the variance increases to infinity. Thus, a good choice of λ_0 can reduce the variance, but a bad choice may increase it by an arbitrarily large factor. In Exercise 1.30 the reader is asked to show that the variance is minimized by taking $\lambda_0 = \lambda + 1/y_0 - (\lambda^2 + 1/y_0^2)^{1/2} < \lambda$, and to find the range of values of λ_0 that reduce the variance.

For a numerical illustration, let $\lambda = 1$ and $y_0 = 4$. Figure 1.17 shows the variance ratio $\mathrm{Var}[X_{\mathrm{is}}]/\mathrm{Var}[X]$ as a function λ_0. Here, the variance is minimized with $\lambda_0 \approx 0.2192$ and $\mathrm{Var}[X_{\mathrm{is}}]/\mathrm{Var}[X] \approx 0.0962$ for that λ_0. That is, to estimate $p = \mathbb{P}[Y > 4]$, we change the mean of Y to the value $1/\lambda_0 \approx 1/.2192 \approx 4.56$, which is not far from 4, which is the threshold for the probability that we want to estimate. □

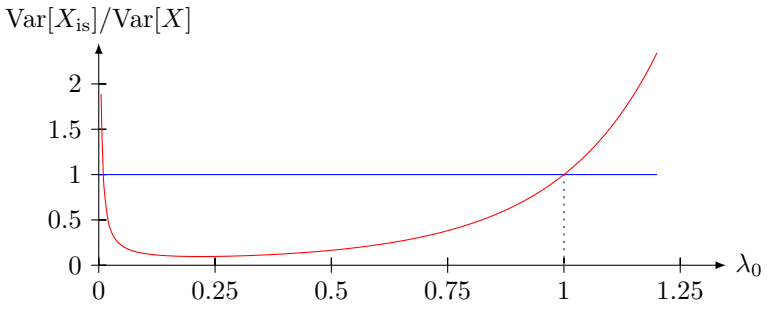


Fig. 1.17. Variance ratio $\mathrm{Var}[X_{\mathrm{is}}]/\mathrm{Var}[X]$ as a function λ_0 for Example 1.38, with $\lambda = 1$ and $y_0 = 4$. The dotted line is at $\lambda_0 = \lambda$.

Example 1.39 As an alternative sampling density for Example 1.38, consider the exponential with rate λ (unchanged), but truncated to the interval $[y_0, \infty)$. This density is

$$g(y) = \frac{\lambda \exp[-\lambda y]}{\int_{y_0}^{\infty} \lambda \exp[-\lambda y] \mathrm{d}y} = \frac{\lambda \exp[-\lambda y]}{\exp[-\lambda y_0]} = \lambda \exp[-\lambda(y - y_0)] \qquad \text{for } y \geq y_0,$$

and $g(y) = 0$ elsewhere. Sampling from this density is equivalent to taking $Y_0 = Y + y_0$ where Y has density π; i.e., a right shift of the density π. With this density, we avoid sampling in the area where the estimator is zero. We can write

$$p = \int_{y_0}^{\infty} \lambda \exp[-\lambda y] \mathrm{d}y = \int_{y_0}^{\infty} \frac{\lambda \exp[-\lambda y]}{g(y)} g(y) \mathrm{d}y = \int_{y_0}^{\infty} \exp[-\lambda y_0] g(y) \mathrm{d}y = \mathbb{E}[X_{\mathrm{is}}]$$

with

$$X_{\mathrm{is}} = \mathbb{I}[Y_0 \geq y_0] \exp[-\lambda y_0]$$

where Y_0 has density g. Since Y_0 cannot be smaller than y_0, we always have $X_{\mathrm{is}} = \exp[-\lambda y_0] = p$. Thus, this X_{is} is an unbiased estimator with zero variance! In other words, we have managed to cleverly change the sampling distribution so that the estimator becomes a constant, always equal to p. We will see later that such "magical" zero-variance sampling schemes do exist for most simulation models encountered in practice. Unfortunately, the corresponding sampling distributions are typically much too difficult to find and to sample from. On the other hand, they can be approximates and their approximation can often provide sampling schemes with much smaller variance than crude MC (L'Ecuyer and Tuffin 2008). □

Example 1.40 In Example 1.38, suppose we want to estimate $p = \mathbb{P}[Y \leq y_0]$ instead, where $y_0 \ll 1/\lambda$. The crude MC estimator is $X = \mathbb{I}[Y \leq y_0]$. As in Example 1.38, we apply IS by sampling Y_0 from an exponential density with rate λ_0 and the estimator is

$$X_{\mathrm{is}} = \mathbb{I}[Y_0 \leq y_0] \frac{\lambda}{\lambda_0} \exp[-(\lambda - \lambda_0) Y_0].$$

This estimator has variance

$$\begin{aligned}
\mathrm{Var}[X_{\mathrm{is}}] &= \mathbb{E}[X_{\mathrm{is}}^2] - p^2 \\
&= \int_0^{y_0} \frac{\lambda^2}{\lambda_0^2} \exp[-2(\lambda - \lambda_0) y] \lambda_0 \exp[-\lambda_0 y] \mathrm{d}y - p^2 \\
&= \int_0^{y_0} \frac{\lambda^2}{\lambda_0} \exp[-(2\lambda - \lambda_0) y] \mathrm{d}y - p^2 \\
&= \frac{\lambda^2}{\lambda_0(2\lambda - \lambda_0)} (1 - \exp[-(2\lambda - \lambda_0) y_0]) - p^2.
\end{aligned}$$

Note that this expression remains finite and positive for all $\lambda_0 > 0$, including at $\lambda_0 = 2\lambda$ (by taking the limit). Figure 1.18 shows the ratio $\mathrm{Var}[X_{\mathrm{is}}]/\mathrm{Var}[X]$ as a function λ_0, for $\lambda = 1$ and $y_0 = 0.15$. The variance is minimized with $\lambda_0 \approx 11.0$, and this minimal variance is about 13 times smaller than $\mathrm{Var}[X]$. If λ_0 is increased beyond about 36.5, then $\mathrm{Var}[X_{\mathrm{is}}]$ becomes larger than $\mathrm{Var}[X]$. This happens because if λ_0 is very large, the likelihood ratio takes huge values when Y_0 is just below y_0.

One can verify that the "magical" zero-variance IS here consists in sampling Y_0 from the exponential density with rate λ, but truncated to the interval $[0, y_0]$; i.e., from the density $g(y) = \lambda e^{-\lambda y}/(1 - e^{-\lambda y_0})$ for $0 \leq y \leq y_0$. □

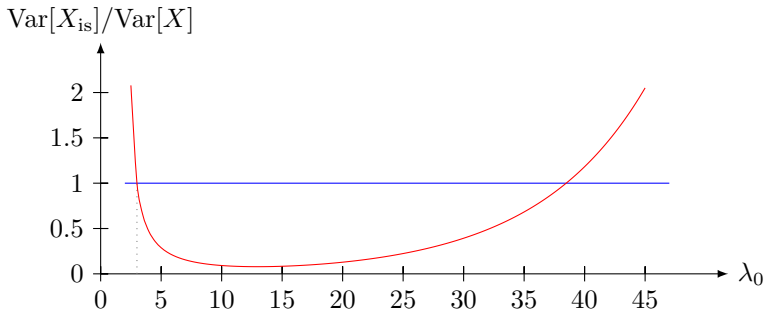


Fig. 1.18. Variance ratio $\mathrm{Var}[X_{\mathrm{is}}]/\mathrm{Var}[X]$ as a function λ_0 for Example 1.40, with $\lambda = 1$ and $y_0 = 0.15$.

Example 1.41 Let Y_1 and Y_2 be two independent random variables with densities π_1 and π_2, and cdfs F_1 and F_2, respectively, over $\mathbb{R}$. Suppose that we receive a payoff

$$X = \begin{cases} Y_1 + Y_2 - K & \text{if } Y_1 \le a \text{ and } Y_1 + Y_2 \ge b, \\ 0 & \text{otherwise,} \end{cases}$$

where $K > 0$, and a and b are fixed constants. We want to estimate $\mu = \mathbb{E}[X]$. This is a simplified version of a type of model that occurs when pricing barrier options in finance (Glasserman 2004, Hull 2000). With standard MC, we generate Y_1 and Y_2 from their original normal distributions and compute X. We repeat this n times and take the average and the empirical variance of the n independent realizations of X to compute a confidence interval on μ.

Inspired by the previous examples and based on the observation that it is worthless to sample in the areas where the payoff X is zero, the following approach seems reasonable: Generate Y_1 from its density conditional on $Y_1 \le a$, then generate Y_2 from its density conditional on $Y_1 + Y_2 \ge b$, i.e., truncated to the interval $[b - Y_1, \infty)$. The new density of Y_1 is

$$g_1(y) = \frac{\pi_1(y)}{\mathbb{P}[Y_1 \le a]} = \frac{\pi_1(y)}{F_1(a)}$$

for $y \le a$ and 0 elsewhere, and the new density of Y_2 conditional on $Y_1 = y_1$ is

$$g_2(y \mid y_1) = \frac{\pi_2(y)}{\mathbb{P}[Y_2 \ge b - y_1]} = \frac{\pi_2(y)}{1 - F_2(b - y_1)}$$

for $y \ge b - y_1$ and 0 elsewhere. We have

$$\begin{aligned} \mu &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X\,\pi_2(y_2)\pi_1(y_1)\mathrm{d}y_2\mathrm{d}y_1 \\ &= \int_{-\infty}^{a}\int_{b-y_1}^{\infty} X\frac{\pi_2(y_2)\pi_1(y_1)}{g_2(y_2 \mid y_1)g_1(y_1)} g_2(y_2 \mid y_1)g_1(y_1)\mathrm{d}y_2\mathrm{d}y_1 \\ &= \int_{-\infty}^{a}\int_{b-y_1}^{\infty} X\,F_1(a)\,(1 - F_2(b - y_1))g_2(y_2 \mid y_1)g_1(y_1)\mathrm{d}y_2\mathrm{d}y_1 \\ &= \mathbb{E}_0[X_{\mathrm{is}}], \end{aligned}$$

where

$$X_{\text{is}} = X\, F_1(a)\,(1 - F_2(b - Y_1))$$

and the expectation $\mathbb{E}_0$ is under the densities g_1 and g_2. This is a case of *dynamic importance sampling*, where the sampling distribution of each random variable may depend on the realizations of the previous ones. This can be done whenever a sequence of random variables is generated in the model and permits one to design adaptive IS strategies.

For concreteness, suppose that both π_1 and π_2 are the normal density with mean 1 and variance 1. In that case, $F_1(a) = \mathbb{P}[Y_1 < a] = \mathbb{P}[Y_1 - 1 < a - 1] = \Phi(a-1)$. To generate Y_1 from the conditional (truncated) distribution, it suffices to generate $U_1 \sim \mathsf{Uniform}(0, \Phi(a-1))$ and put $Y_1 = 1 + \Phi^{-1}(U_1)$ (see Exercise 2.16). Then, we want to generate Y_2 conditional on $Y_2 - 1 \geq b - Y_1 - 1$. We will have $1 - F_2(b - Y_1) = \mathbb{P}[Y_2 > b - Y_1] = \mathbb{P}[Y_2 - 1 > b - 1 - Y_1] = 1 - \Phi(b - 1 - Y_1)$. To generate Y_2 from its conditional density, generate $U_2 \sim \mathsf{Uniform}(\Phi(b - 1 - Y_1), 1))$ and put $Y_2 = 1 + \Phi^{-1}(U_2)$. To generate U_1 and U_2, it suffices to generate $V_1 \sim \mathsf{Uniform}(0, 1)$ and $V_2 \sim \mathsf{Uniform}(0, 1)$, and define $U_1 = \Phi(a-1)V_1$ and $U_2 = \Phi(b-1-Y_1) + (1 - \Phi(b-1-Y_1))V_2$. The new estimator is $X_{\text{is}} = X\Phi(a-1)(1 - \Phi(b-1-Y_1))$. This is illustrated in Figure 1.19. We can interpret this strategy either as a change of densities on the uniforms that drive the simulation, or as a change of the normal densities to truncated ones. The estimator is the same in both cases.

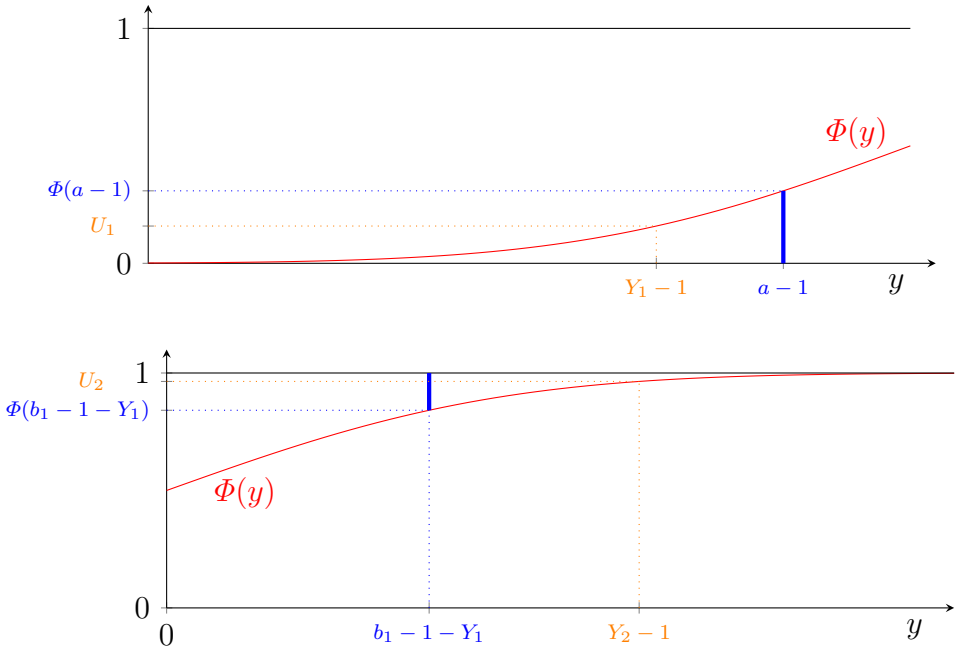


Fig. 1.19. Sampling from conditional densities in Example 1.41. First, U_1 is sampled uniformly over the thick blue line of the upper figure to produce Y_1, then U_2 is sampled uniformly over the tick blue line in the lower figure to produce Y_2. The red curve is the standard normal cdf.

For a numerical illustration, let $K = 1$, $b = 2$, and $a = 1/2$. We applied the MC method and the modified sampling strategy just described, with $n = 10^5$ in both cases. Table 1.3 reports the empirical mean and variance, and a 95% confidence interval for the mean, for

the two methods. The empirical variance S_n^2 is about 43 times smaller with the estimator X_{is} than with the crude MC estimator X. This means that with X, we need about 43 times more runs than with X_{is} to reach the same accuracy.

Table 1.3. Empirical results for Example 1.41 with $n = 10^5$, for X and X_{is}.

Estimator	$\hat{\mu}_n$	S_n^2	95% confidence interval
X	0.0733	0.1188	(0.071, 0.075)
X_{is}	0.0742	0.0027	(0.074, 0.075)

□

Example 1.42 Suppose we want to estimate the integral of a function f over an unbounded region instead of over the unit hypercube $(0,1)^s$. One approach is to make a change of variables to transform the integral into an integral over the unit hypercube, as we have seen earlier. But this is not always convenient. We illustrate an alternative approach. Suppose we want to estimate

$$\mu = \int_{-\infty}^{\infty} f(x)\mathrm{d}x < \infty,$$

where $f : \mathbb{R} \to [0, \infty)$. Let g be the density of some random variable Y that can be generated efficiently, and such that $g(x) > 0$ whenever $f(x) > 0$. We can write

$$\mu = \int_{-\infty}^{\infty} [f(y)/g(y)]g(y)\mathrm{d}y = \mathbb{E}_g[f(Y)/g(Y)]$$

where $\mathbb{E}_g$ denotes the expectation with respect to the density g. This means that to obtain an unbiased estimator of μ, we can generate n independent copies of Y from density g, say $Y_1, \ldots, Y_n$, and take the average

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} f(Y_i)/g(Y_i).$$

If $f(x) > 0$ for all $x \in \mathbb{R}$, then g could conceivably be a normal density, for example. We have $\mathrm{Var}[\hat{\mu}_n] = \mathrm{Var}[f(Y_i)/g(Y_i)]/n$, where

$$\mathrm{Var}\left[\frac{f(Y_i)}{g(Y_i)}\right] = \mathbb{E}_g\left[\frac{f^2(Y)}{g^2(Y)}\right] - \mu^2 = \int_{-\infty}^{\infty} \frac{f^2(y)}{g^2(y)} g(y)\mathrm{d}y - \mu^2 = \int_{-\infty}^{\infty} \frac{f^2(y)}{g(y)}\mathrm{d}y - \mu^2.$$

This expression highlights the fact that if $g(y) \ll f(y)$ on some interval, or if $f^2(y)/g(y)$ does not decrease quickly enough when $y \to \pm\infty$, then the variance will be very large, perhaps even infinite. This means that the choice of g has a crucial importance for this method to be effective. This applies in general when we change a sampling density. □

1.6.2 Outline of the importance sampling methodology

In general, the random variable $f(\boldsymbol{U})$ in (1.16) is naturally expressed as $h(\boldsymbol{Y}) = h(\tau(\boldsymbol{U})) = f(\boldsymbol{U})$ for some functions h and τ, where $\boldsymbol{Y} = \tau(\boldsymbol{U})$ is a random vector having a non-uniform distribution. This $\boldsymbol{Y}$ represents the basic input random variables whose distributions are specified in the model. In the stochastic activity network of Example 1.20, for instance, we would take $\boldsymbol{Y} = (Y_1, \ldots, Y_{13})$ and $h(\boldsymbol{Y}) = h(Y_1, \ldots, Y_{13}) = T$ if we want to estimate $\mathbb{E}[T]$, or $h(\boldsymbol{Y}) = h(Y_1, \ldots, Y_{13}) = \mathbb{I}[T > x]$ if we want to estimate $\mathbb{P}[T > x]$, where T is the length of the longest path. The MC estimator $f(\boldsymbol{U})$ would normally be generated by generating $\boldsymbol{Y}$ from its distribution and then computing $h(\boldsymbol{Y})$.

There are generally many ways of decomposing $f(\boldsymbol{U})$ as $h(\boldsymbol{Y}) = h(\tau(\boldsymbol{U}))$, giving different distributions for $\boldsymbol{Y}$. For any given selection, one may change the distribution $\boldsymbol{Y}$ as we saw in the previous examples. When used as a variance reduction tool, this technique based on a change of distribution is known as *importance sampling (IS)*.

We now explain the principle of IS in a setting where $\mu = \mathbb{E}[h(\boldsymbol{Y})]$ for a function $h : \mathbb{R}^d \to \mathbb{R}$, and $\boldsymbol{Y}$ is a continuous random vector with density $\pi(\boldsymbol{y})$ over the d-dimensional real space $\mathbb{R}^d$, for some integer $d > 0$. Let g be another density, such that $g(\boldsymbol{y}) > 0$ whenever $h(\boldsymbol{y})\pi(\boldsymbol{y}) \neq 0$ (this assumption is to make sure that we never divide by zero). We can write

$$\begin{aligned} \mu &= \mathbb{E}_\pi[h(\boldsymbol{Y})] = \int_{\mathbb{R}^d} h(\boldsymbol{y})\pi(\boldsymbol{y})\mathrm{d}\boldsymbol{y} = \int_{\mathbb{R}^d} [h(\boldsymbol{y})\pi(\boldsymbol{y})/g(\boldsymbol{y})]g(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \\ &= \mathbb{E}_g[h(\boldsymbol{Y})\pi(\boldsymbol{Y})/g(\boldsymbol{Y})] \end{aligned} \tag{1.35}$$

where $\mathbb{E}_\pi$ denotes the mathematical expectation when $\boldsymbol{Y}$ has a distribution with density π, and $\mathbb{E}_g$ denotes the expectation when the density of $\boldsymbol{Y}$ is g. Eq. (1.35) tells us that if $\boldsymbol{Y}$ is generated from density g,

$$X_{\text{is}} = h(\boldsymbol{Y})\pi(\boldsymbol{Y})/g(\boldsymbol{Y}) \tag{1.36}$$

is an unbiased estimator of μ. In other words, to account for the fact that the density has been changed from π to g, the old estimator $X = h(\boldsymbol{Y})$ must be multiplied by the *likelihood ratio* $L(\boldsymbol{Y}) = \pi(\boldsymbol{Y})/g(\boldsymbol{Y})$, i.e., the ratio of the likelihoods (or densities) $\pi(\boldsymbol{Y})$ and $g(\boldsymbol{Y})$ of the observed value of $\boldsymbol{Y}$ under the original and modified distributions, respectively. For $g \equiv \pi$, we get the standard MC estimator and the likelihood ratio is one.

The application of IS for discrete random variables is similar; just replace the densities by probabilities (mass functions) and the integrals by sums.

Why would we want to select a g different than π? There are two main motivations:

(1) it may be too difficult or too costly to sample directly from π and an "easier" density g is available;

(2) if g is selected properly, sampling from the alternative density may provide an estimator with much smaller variance than the original one.

1.6.3 The zero-variance simulation and its approximation

Suppose the function h is never negative and we take $g(\boldsymbol{y})$ proportional to $h(\boldsymbol{y})\pi(\boldsymbol{y})$, i.e., $g(\boldsymbol{y}) = h(\boldsymbol{y})\pi(\boldsymbol{y})/K$ for some constant K. Since g must be a probability density, its integral

must be equal to 1. For that, we must have $K = \int_{\mathbb{R}^d} h(\boldsymbol{y})\pi(\boldsymbol{y}) = \mu$. Then, the IS estimator is always equal to $h(\boldsymbol{y})\pi(\boldsymbol{y})/g(\boldsymbol{y}) = K = \mu$, so it has zero variance! We already saw this "magic" in Example 1.39. This looks too good to be true and there is indeed a catch: To sample from this zero-variance density in practice, we need to know μ, which is the quantity we want to estimate in the first place. But even if the zero-variance IS estimator cannot be implemented exactly in practice, it can sometimes be approximated. At a minimum, it gives a rough guideline on how to modify the density π to obtain a good g: we want to inflate π where $h(\boldsymbol{y})$ is large and deflate it where $h(\boldsymbol{y})$ is small, by a factor roughly proportional to $h(\boldsymbol{y})$. Moreover, we know that the optimal sampling density is *proportional* to $h(\boldsymbol{y})\pi(\boldsymbol{y})$, and there are method that permit one to do this approximately even when the normalizing constant K is unknown.

Example 1.43 In Example 1.29, we wanted to estimate $p = \mathbb{P}\{A\}$, the probability of an event A, where p was assumed to be small. Suppose the indicator of A can be written as a function h of a continuous random vector $\boldsymbol{Y}$ with density π. That is, $\mathbb{I}[A] = h(\boldsymbol{Y})$. The optimal IS density for $\boldsymbol{Y}$ is then

$$g(\boldsymbol{y}) = \frac{h(\boldsymbol{y})\pi(\boldsymbol{y})}{p} = \begin{cases} \pi(\boldsymbol{y})/p & \text{when } h(\boldsymbol{y}) = 1, \\ 0 & \text{elsewhere.} \end{cases}$$

This is the conditional density of $\boldsymbol{Y}$ given that the event A has occurred:

$$\pi(\boldsymbol{y} \mid A) = \frac{\pi(\boldsymbol{y})\mathbb{I}[A]}{\mathbb{P}[A]} = \frac{h(\boldsymbol{y})\pi(\boldsymbol{y})}{p} = g(\boldsymbol{y}).$$

This density reduces the variance to 0, but can be computed explicitly only in very simple situations. For example, if Y is a one-dimensional random variable with density $\pi(y)$ over the real line and A is the interval $[y_0, \infty)$ for some constant y_0, then g should be the density π truncated to the interval $[y_0, \infty)$ and rescaled: $g(y) = \pi(y)/\mathbb{P}[Y \geq y_0]$ for $y \geq y_0$, and zero elsewhere. This is what we had in Example 1.39. In more complex simulation models, sampling exactly from the conditional density $\pi(\boldsymbol{y} \mid A)$ is usually impractical, because it is too complicated and unknown. But the knowledge of its general shape provides a guideline for selecting g. □

We saw that replacing the density π by another density g is not always profitable. The method is often extremely sensitive to the choice of g. With a bad choice, the variance can increase substantially, and even become infinite in some cases, as shown in Example 1.38. To a certain extent, applying IS is like performing a critical medical operation: It is important to understand what one is doing. There are situations, however, mostly in models that involve rare important events, where a crude MC estimator is practically useless, and for which IS or a similar technique is essential. The next example illustrates this. A more extensive study of IS is done in Section 6.12.

Example 1.44 *Ruin probability of an insurance firm.* A (simplified) insurance company receives *premiums* at constant rate $c > 0$ (i.e., the money arrives continuously at that rate), and receives *claims* (sums of money that they must pay) according to a Poisson process

$\{N(t), t \geq 0\}$ with rate $\lambda > 0$. The claim sizes C_j, $j \geq 1$, are i.i.d. with density f. The *reserve* (amount of money in hand) at time t is then

$$R(t) = R(0) + ct - \sum_{j=1}^{N(t)} C_j,$$

where $R(0)$ is the initial reserve. We want to estimate the *ruin probability*, i.e., the probability μ that $R(t)$ eventually becomes negative during the time interval $[0, \infty)$. Typically (hopefully for the firm and its customers) this probability is very small.

The first time when $R(t)$ becomes negative, if it ever does, must be at the occurrence of a claim. If R_j denotes the reserve just after claim j, then

$$R_j = R_{j-1} + A_j c - C_j$$

for $j \geq 1$, where $R_0 = R(0)$ and A_j is the time between claims $j-1$ and j. The process $\{R_j, j \geq 1\}$ is a *random walk* on the one-dimensional real line and we have $\mu = \mathbb{P}[T < \infty]$ where $T = \inf\{j \geq 1 : R_j < 0\}$.

The standard way of estimating μ would be to generate n independent copies of T, say $T_1, \ldots, T_n$, compute the indicator $X_i = \mathbb{I}[T_i < \infty]$ for each, and take the average $\bar{X}_n$ as an estimator. However, generating values of T by straightforward simulation is problematic, because $T = \infty$ with very high probability, and we cannot be 100% sure that $T = \infty$ for a given sample path unless we simulate the system for an infinite amount of time! And even if we could easily generate values of T, estimating a very small probability μ (e.g., 10^{-9} or less) with small relative error would require an enormous value of n.

IS provides a convenient trick to get around both of these difficulties at the same time: Change the probability distributions of A_j and C_j so that the ruin occurs with probability 1, and multiply the estimator by the appropriate likelihood ratio to recover an unbiased estimator of μ. We will see in Example 6.66 that a suitable change in this case is to replace the density $f(x)$ of C_j by

$$f_\theta(x) = f(x)e^{\theta x}/M_f(\theta)$$

and to increase the rate of the Poisson process to

$$\lambda_\theta = \lambda + \theta c,$$

for some $\theta > 0$, where $M_f(\theta) = \int_{-\infty}^{\infty} f(x)e^{\theta x}\mathrm{d}x$ is the normalizing constant required to make f_θ a probability density, and that a good choice of θ is the largest solution to the equation $M_f(\theta) = (\lambda + \theta c)/\lambda$. We assume that $M_f(\theta) < \infty$ for θ in some neighborhood of 0. Then, under the new probability distributions, it turns out that $T < \infty$ with probability 1, so an unbiased estimator of μ is simply the likelihood ratio that corresponds to the change of probability distributions. Here, the vector of input random variables that are generated for the simulation is $\boldsymbol{Y} = (A_1, C_1, A_2, C_2, \ldots, A_T, C_T)$ and the corresponding likelihood ratio (the IS estimator) is

$$
\begin{aligned}
L(\boldsymbol{Y}) &= L(A_1, C_1, A_2, C_2, \ldots, A_T, C_T) \\
&= \prod_{j=1}^{T} \frac{f(C_j)\lambda e^{-\lambda A_j}}{f_\theta(C_j)\lambda_\theta e^{-\lambda_\theta A_j}} \\
&= \prod_{j=1}^{T} \frac{M_f(\theta) e^{-\theta C_j}\lambda e^{-\lambda A_j}}{(\lambda+\theta c)\lambda_\theta e^{-(\lambda+\theta c A_j}} \\
&= \prod_{j=1}^{T} (\lambda+\theta c)\lambda \exp\left[-\theta C_j - \lambda A_j + (\lambda+\theta c)A_j\right] \\
&= \prod_{j=1}^{T} (\lambda+\theta c)\lambda \exp\left[\theta \sum_{j=1}^{T} (cA_j - C_j)\right] \\
&= e^{\theta(R_T - R_0)}.
\end{aligned}
$$

The random vector $\boldsymbol{Y}$ has random dimension $2T$, but this is not a problem and the IS estimator is unbiased. Since $R_T < 0$, this estimator never takes a value larger than $e^{-\theta R_0}$, so its variance $\mathbb{E}[L^2(\boldsymbol{Y})] - \mu^2$ does not exceed $e^{-2\theta R_0} - \mu^2$.

As a special case, if the claim sizes C_j are exponential with rate parameter β, then their density is $f(x) = \beta e^{-\beta x}$ for $x > 0$, we have $M_f(\theta) = \beta/(\beta - \theta)$, and the modified density is $f_\theta(x) = \beta e^{-(\beta-\theta)x}/M_f(\theta) = (\beta - \theta)e^{-(\beta-\theta)x}$ for $x > 0$. This is an exponential with rate $\beta_\theta = \beta - \theta$. In this case, the equation $M_f(\theta) = (\lambda + \theta c)/\lambda$ becomes $\beta\lambda = (\beta - \theta)(\lambda + \theta c)$, or equivalently $\theta(c\theta + \lambda - \beta c) = 0$, whose only positive solution is $\theta = \beta - \lambda/c$. With this choice of θ, the C_j are exponential with mean $1/(\beta - \theta) = c/\lambda$. This means that if λ was left unchanged, we would have $\mathbb{E}[R_k - R_0] = k\mathbb{E}[A_j c - C_j] = 0$ under IS. But since λ is also increased under IS, $\mathbb{E}[A_j]$ is decreased and we have $\mathbb{E}[A_j c - C_j] < 0$, which implies that $R_k \to -\infty$ as $k \to \infty$, so $T < \infty$ with probability 1.

For a numerical illustration with this special case, let $\lambda = 1$, $\beta = 1/2$, and $R(0) = 200$. Table 1.4 gives the values of θ, λ_θ, β_θ, the estimated value $\hat{\mu}_n$ of the mean μ, and the estimated value S_n^2 of the variance σ^2, from $n = 2^{15} = 32768$ simulation runs with IS, for $c = 3$, 5, and 10. The relative error on these estimators is less than 1% in all three cases.

With straightforward simulation (without the IS), the variance σ^2 is approximately equal to the mean μ (see Example 1.29), so the variance reduction factor provided by IS compared with MC is approximately equal to the ratio of values of $\hat{\mu}_n$ and S_n^2 in the table. For $c = 10$, for example, IS reduces the variance by the (huge) factor $3.6 \times 10^{-36}/2.3 \times 10^{-71} \approx 1.5 \times 10^{35}$. Our variance estimates tell us that to get a relative error below 10%, we need a sample size of approximately $n = 183$ with IS and $n = 2.8 \times 10^{37}$ without IS. In other words, computing a meaningful estimator of μ would be impractical with standard MC (without IS) in this case, because μ is too small. □

1.7 Common Random Numbers for Comparing Systems

Simulation is often employed to compare the performance measures of two or more similar systems, or of slightly different parameter values for the same system. This typically occurs

Table 1.4. Importance sampling distributions, estimated ruin probability, variance estimate with IS, and variance reduction factor (VRF) by using IS, for Example 1.44.

c	θ	λ_θ	β_θ	$\hat{\mu}_n$	S_n^2	VRF
3	0.1667	1.5	0.3333	2.2×10^{-15}	6.3×10^{-31}	3.5×10^{15}
5	0.3000	2.5	0.2000	3.5×10^{-27}	6.9×10^{-54}	5.1×10^{26}
10	0.4000	5.0	0.1000	3.6×10^{-36}	2.3×10^{-71}	1.5×10^{35}

in *optimization* settings (Section 1.15) or for estimating the *sensitivity* of a model to some of its parameters (Section 1.8). The *common random numbers (CRNs)* methodology is a key tool for improving the efficiency in this context. It will be studied more extensively in Section 6.4, but we introduce it here to highlight its importance and give a flavor of how effective a simple (but clever) variance reduction method can be.

Suppose we want to estimate the difference $\mu_2 - \mu_1$ by $X_2 - X_1$, where X_1 and X_2 are estimators such that $\mu_1 = \mathbb{E}[X_1]$ and $\mu_2 = \mathbb{E}[X_2]$. If X_1 and X_2 are the simulated performances of two similar systems, then one can simulate these two systems using the same underlying sequence of uniform random numbers for both systems, and use them at the same place as much as possible. This is a simulation experiment with CRNs. Doing that does not change the individual distributions of X_1 and X_2, which implies that $\mathbb{E}[X_2 - X_1]$ is unchanged, but it is likely to induce a *positive covariance* between them. The variance of the difference $X_2 - X_1$ can be written as

$$\mathrm{Var}[X_2 - X_1] = \mathrm{Var}[X_2] + \mathrm{Var}[X_1] - 2\,\mathrm{Cov}[X_1, X_2].$$

If the two systems are simulated with *independent random numbers (IRNs)*, the covariance term is zero. A positive covariance term, on the other hand, reduces the variance of $X_2 - X_1$.

Example 1.45 In the stochastic activity network of Example 1.4, suppose we would like to examine the impact of changing the mean durations of activities 2 and 4 from 7.0 and 16.5 to 10.0 and 18.5, respectively. It could be because we consider reducing the resources allocated to those activities to save money. Suppose want to estimate $\mu_2 - \mu_1 = \mathbb{E}[X_2 - X_1]$ where the random variable X_1 represents the project duration with the original parameters for the Y_j's, and X_2 is the project duration with the new parameters.

To generate X_1 we simulate the system as usual by generating 13 independent uniform random numbers $U_1, \ldots, U_{13}$, put $Y_j = F_j^{-1}(U_j)$ for each j, and compute the length X_1 of the longest path. To generate X_2 with IRNs, we do the same using 13 independent uniform random numbers $\tilde{U}_1, \ldots, \tilde{U}_{13}$ also independent of the previous ones, compute $\tilde{Y}_j = \tilde{F}_j^{-1}(\tilde{U}_j)$ for each j, and compute the length X_2 of the longest path when the arc lengths are those $\tilde{Y}_j$. In this case, X_1 and X_2 are independent random variables. For the CRN implementation, we simply reuse U_j in place of $\tilde{U}_j$, for each j, to generate X_2. That is, the same random number used to generate Y_j in the first configuration is used again to generate Y_j in the second configuration, for each j. This still gives $\mathbb{E}[X_2] = \mu_2$, but now X_1 and X_2 are correlated. For any of these two methods, we would simulate n independent replicates of $\Delta = X_2 - X_1$, and perhaps use them to compute a confidence interval on $\mu_2 - \mu_1$.

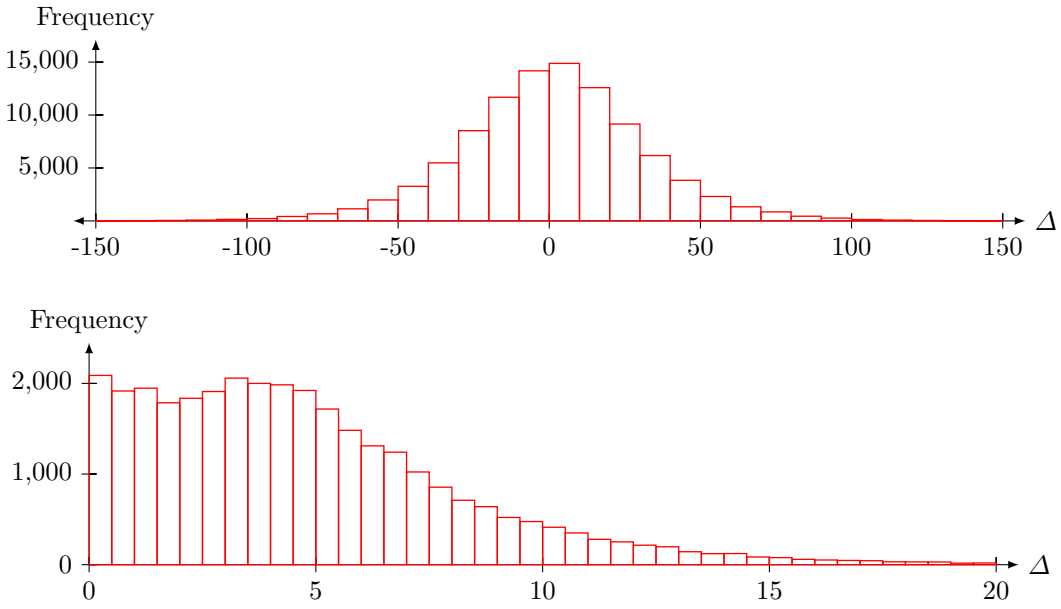


Fig. 1.20. Histograms of the $n = 100,000$ realizations of Δ with IRNs (above) and the 32,120 nonzero realizations of Δ with CRNs (below).

We did this with $n = 100,000$. Figure 1.20 shows histograms of the n realizations of Δ with IRNs (above) and of the nonzero realizations of Δ with CRNs (below). With CRNs, for 67,880 realizations, we had $\Delta = 0$ because the two modified Y_j did not belong to the longest path even after their increase. All other 32,120 realizations of Δ were positive and the histogram shows only these positive realizations (otherwise the rectangle at 0 would be much too high). The fact that Δ is never negative with CRNs is easy to explain: Each Y_j whose mean is changed is an exponential with mean θ_j and is generated by inversion via $Y_j = -\theta_j \ln(1 - U_j)$ where $U_j \sim U(0,1)$. When we increase θ_j and keep the same U_j, Y_j can only increase, because $-\ln(1 - U_j) > 0$. Then the longest path can only increase. With IRNs, on the other hand, we resample all the U_j's, so all the Y_j's are modified and they can either decrease or increase.

With IRNs, the realizations of Δ range from -223.22 to 280.92, with an average of 1.326, and a variance of 967. A 95% confidence interval on $\mu_2 - \mu_1$ is (1.133, 1.519). With CRNs, Δ ranges from 0 to 49.88, with an average of 1.528, a variance of 9.1, and the 95% confidence interval on $\mu_2 - \mu_1$ is (1.510, 1.547). The variance is reduced by a factor of about 106. □

Example 1.46 *A simple inventory model.* This example is taken from L'Ecuyer (2008) and L'Ecuyer (2015). We consider a simple inventory model for a single product, for which the demand on any given day is a Poisson random variables with mean $\lambda > 0$, and the demands on different days are assumed to be independent. On day j, let X_j be the stock level in the morning and D_j the demand during that day. Then on that day there are $\min(D_j, X_j)$ sales, $\max(0, D_j - X_j)$ lost sales, and the stock at the end of the day is $Y_j = \max(0, X_j - D_j)$. We make a revenue c for each sale and pay a storage cost h for each unsold item at the end

of the day. The inventory is controlled using a (s, S) policy, defined as follows: If $Y_j < s$, order $S - Y_j$ items, otherwise do not order. The threshold s and S are control parameters that we have to select. When an order is made in the evening, with probability p it arrives during the night and can be used for the next day, and with probability $1-p$ it never arrives (in which case a new order will have to be made the next evening). If the order arrives, we pay a fixed cost K plus a marginal cost of k per item, and we put $O_j = 1$, otherwise $O_j = 0$. With all these ingredients, we can write the net profit for day j (including the sales, storage cost for the night, and cost of the arriving order for the next morning, if any) as $c \min(D_j, X_j) - hY_j - (K + (S - Y_j)k)O_j$. The stock at the beginning of the first day is $X_0 = S$.

We want to compare several policies (s, S) in terms of expected net profit per day for the first m days, by simulating this system with common random numbers across the policies, with all the other model parameters fixed to specific values. For each policy, we will replicate the simulation n times independently. Figure 1.21 shows the main parts of a Java code that does that. It uses the SSJ library (L'Ecuyer 2023). The method `simulateOneRun` simulates the model for m days with a given policy (s, S) and returns the average profit per day.

We use two streams of uniform random numbers to simulate the model: one named `streamDemand` to generate the demand D_j on successive days and one named `streamOrder` to decide if the order will arrive or not when an order is made. To generate the demands, we construct a Poisson distribution object with mean λ (this construction precomputes a set of tables used for fast inversion) and a generator `genDemand` which samples the D_j's by applying inversion to the random numbers produced by `streamDemand`.

For each replication, we use a different substream from these two streams. The CRN implementation uses exactly the same streams and substreams for all policies (s, S). For each policy, after each m-day simulation run, the two streams are reset to the beginning of their next substreams. After the last run, they are reset to their first substreams, so exactly the same random numbers are re-used for the next policy, for each run.

Why do this and use two different streams? We want to make sure that the same random numbers are used for the same purpose (e.g., each D_j should be the same) when (s, S) is changed, even if it changes the decisions of when and how to order. If we use a single stream for everything, a random number used to generate a demand for a given pair (s, S) could be used to decide if an order has arrived for another pair (s, S). When simulating larger systems, we may need thousands of distinct streams to simulate different parts of the system, to maintain appropriate synchronization of the random numbers across policies. One may think of an inventory system with thousands of different products, for example.

□

For a numerical illustration, we made an experiment with $\lambda = 40$, $c = 2$, $h = 0.1$, $K = 30$, $k = 1$, $p = 0.95$, $m = 50$, $n = 1000$, and a grid of 121 pairs (s, S) defined by taking $s = 22, 23, \ldots, 32$ and $S = 158, 159, \ldots, 168$. We estimated the expected profit per day for each of those 121 policies, first using CRNs as in Figure 1.21, and then with IRNs. For the latter, we just removed the `resetStartStream` and `resetNextSubstream` statements, so different random numbers were used for the different policies. The results are displayed in Figure 1.22. We see that CRNs produce a much smoother sample function than IRNs. This

```
  RandomStream streamDemand = new MRG32k3a();
  RandomStream streamOrder  = new MRG32k3a();
  RandomVariateGenInt genDemand
     = new PoissonGen (streamDemand, new PoissonDist (lambda));

  public double simulateOneRun (int m, int s, int S) {
     // Simulates inventory model for m days, with the (s,S) policy.
     int Xj = S, Yj;          // Stock in morning and in evening.
     double profit = 0.0;     // Cumulated profit.
     for (int j = 0; j < m; j++) {
        Yj = Xj - genDemand.nextInt(); // Subtract demand for the day.
        if (Yj < 0) Yj = 0;            // Lost demand.
        profit += c * (Xj - Yj) - h * Yj;
        if ((Yj < s) && (streamOrder.nextDouble() < p)) {
           // We have a successful order.
           profit -= K + k * (S - Yj);
           Xj = S;
        } else
           Xj = Yj;
     }
     return profit / m;    // Average profit per day.
  }

 public void simulatePoliciesCRN (int n, int m, int nump, int[] s, int[] S {
   Tally statProfits = new Tally();
   for (int j = 0; j < nump; j++) {
     statProfits.init();
     // Perform n runs for Policy j.
     for (int i = 0; i < n; i++) {
       statProfits.add (simulateOneRun (m, s[j], S[j]));
       streamDemand.resetNextSubstream();
       streamOrder.resetNextSubstream();
     }
     System.out.println(s[j] + ", " + S[j] + ", " + statProfits.average());
     streamDemand.resetStartStream();
     streamOrder.resetStartStream();
   }
 }
```

Fig. 1.21. Simulating n replications of the inventory model for m days, for `nump` policies, with CRNs.

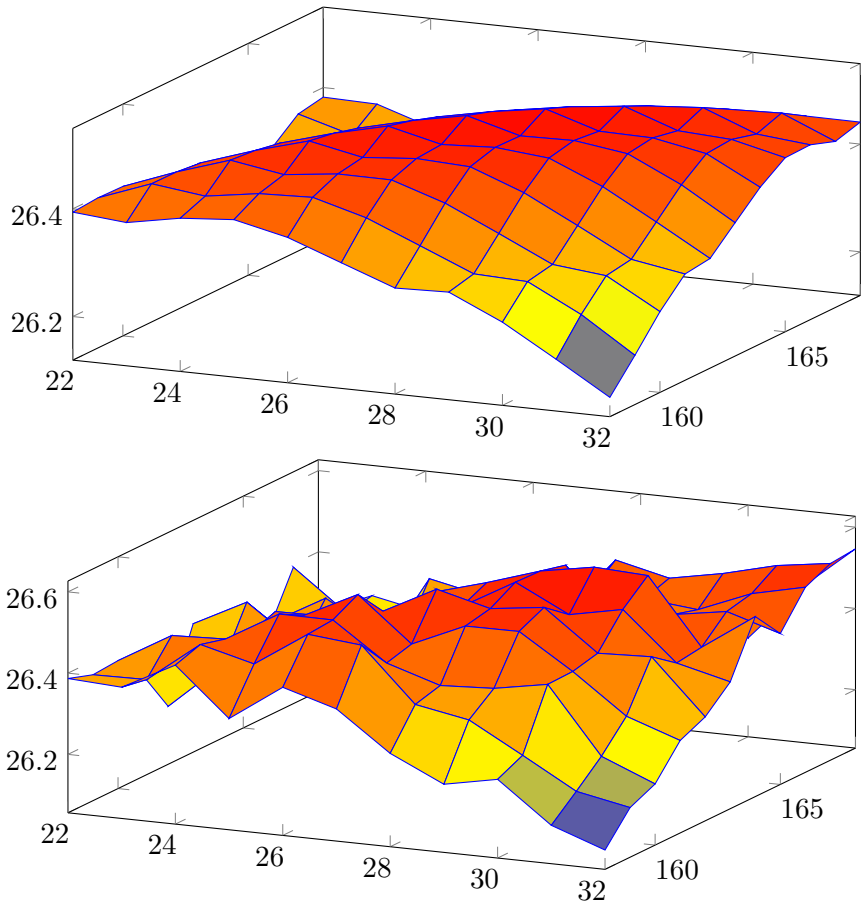


Fig. 1.22. Estimated profit as a function of (s, S) with CRNs (above) and IRNs (below).

sample function with CRNs reaches its maximum at $(s, S) = (28, 164)$, with an average cost of 26.5135.

The previous examples illustrate how CRNs can reduce the variance when estimating the *difference* between two (related) expectations, by inducing a positive covariance between the two corresponding estimators. More examples of this are given in the next subsection. The CRN idea applies more generally to improve efficiency when estimating a *function of several expectations* (for example a ratio between two expectations, etc.). We will return to this in Chapter 6.

1.8 Sensitivity Analysis and Derivative Estimation

In Section 1.7, we saw a simple way to improve efficiency when estimating the change of an expectation when the values of certain model parameters are modified by a fixed amount. In the case of continuous parameters, we are often interested in fact in the *derivative* of the expectation with respect to some of the parameters, i.e., the impact of infinitesimal changes to parameter values.

Consider a model that depends on a real-valued parameter θ, let $\mu(\theta)$ be the expectation of interest (the performance measure) as a function of θ, and suppose we want to estimate the derivative

$$\mu'(\theta) = \frac{\mathrm{d}\mu(\theta)}{\mathrm{d}\theta}$$

at $\theta = \theta_1$. To do this for several parameters (for a vector $\boldsymbol{\theta}$ of parameters), we can consider one parameter at a time. The vector of partial derivatives with respect to the coordinates of $\boldsymbol{\theta}$ is the *gradient* of $\mu(\boldsymbol{\theta})$ with respect to the vector $\boldsymbol{\theta}$. Estimating such a derivative or gradient is useful in the following situations, among others:

(a) We want to estimate the sensitivity to see which parameters are more important and how they affect the performance. It could be because we want to fit a regression model to approximate the performance measure as a function of the parameters, or preferably as a function of a small subset of them (this is called a *metamodel*), or for another reason.

(b) We are uncertain about the true values of certain *parameters in the system* of interest, and we want to estimate how errors in their values would affect the response. For example, it could be the parameters of some probability laws in the model, estimated from data. Such gradient estimators are needed to compute confidence intervals that take into account (simultaneously) the uncertainty coming from simulation noise and the uncertainty in the parameter values. These intervals can be computed via the delta theorem (Section 5.4.1), or bootstrap, or another technique.

(c) The parameters in $\boldsymbol{\theta}$ are *decision parameters* and we want to know the effect of changing their values. For example, it could be the speed of a conveyor in a warehouse, of the strike price of a financial option.

(d) We need a gradient estimator in an optimization algorithm whose aim is to minimize $\mu(\boldsymbol{\theta})$ as a function of $\boldsymbol{\theta}$. Efficient optimization algorithms for continuous parameters often require gradient estimators.

(e) In finance, implementing hedging strategies require the availability of derivative estimates with respect to certain models parameters. These derivatives are known as the *Greeks* (see Example 1.11 and Exercise 1.11, for example).

A simple way of estimating a derivative $\mu'(\theta)$ is via finite differences. We can define a (one-sided) *finite-difference* derivative estimator as follows. We select a small constant $\delta > 0$, simulate the model with $\theta = \theta_1$ to obtain an estimator $X_1 = X(\theta_1)$ of $\mu(\theta_1)$, then at $\theta_2 = \theta_1 + \delta$ to obtain an estimator $X_2 = X(\theta_2)$ of $\mu(\theta_2)$, [10] and estimate the derivative $\mu'(\theta_1)$ by

$$\Delta = (X_2 - X_1)/\delta.$$

This estimator is biased for any fixed $\delta > 0$, but the bias converges to zero when $\delta \to 0$, provided that $\mu'(\theta)$ exists at θ_1. On the other hand,

$$\mathrm{Var}[\Delta] = \frac{\mathrm{Var}[X_1] + \mathrm{Var}[X_2] - 2\mathrm{Cov}[X_1, X_2]}{\delta^2} \approx \frac{2\mathrm{Var}[X_1] - 2\mathrm{Cov}[X_1, X_2]}{\delta^2}$$

if δ is very small and if $\mathrm{Var}[X(\theta)]$ is continuous in θ. If X_1 and X_2 are simulated with IRNs, then $\mathrm{Cov}[X_1, X_2] = 0$ and $\mathrm{Var}[\Delta]$ increases to infinity as $\mathcal{O}(1/\delta^2)$ when $\delta \to 0$, because of the δ^2 in the denominator.

Simulating X_1 and X_2 with CRNs often improves efficiency tremendously, especially when δ is very small, because then $\mathrm{Cov}[X_1, X_2]$ can be large. Note that this scheme requires two separate simulations, one at θ_1 and another one at θ_2. If we want to estimate the derivative with respect to a vector $\boldsymbol{\theta}$ of d parameters, the finite-difference estimator requires $d + 1$ simulations. There are practical situations where d is in the hundreds or even in the thousands.

Under certain conditions, one can compute the limit of $(X_2 - X_1)/\delta$ as $\delta \to 0$ with the underlying uniform random numbers fixed, and use this limit as a unbiased derivative estimator from a single simulation. More specifically, suppose our unbiases estimator of $\mu(\theta)$ can be written as $X(\theta) = f(\theta, \boldsymbol{U})$ where $\boldsymbol{U}$ is the vector of uniform random numbers that drive the simulation, and suppose that

$$f'(\theta, \boldsymbol{U}) = \frac{\partial f(\theta, \boldsymbol{U})}{\partial \theta} = \lim_{\delta \to 0} \frac{f(\theta + \delta, \boldsymbol{U}) - f(\theta, \boldsymbol{U})}{\delta}$$

exists w.p.1 at θ_1. Then we may simply take this *stochastic derivative* (or *sample derivative*) $f'(\theta_1, \boldsymbol{U})$ as an estimator of $\mu'(\theta_1)$.

Does it make sense? Is it an unbiased estimator? The answer is yes under certain conditions, but not always. We have unbiasedness if and only if the middle equality holds in

$$\mathbb{E}[f'(\theta, \boldsymbol{U})] \stackrel{\text{def}}{=} \mathbb{E}[\partial f(\theta, \boldsymbol{U})/\partial\theta] \stackrel{?}{=} \partial \mathbb{E}[f(\theta, \boldsymbol{U})]/\partial\theta \stackrel{\text{def}}{=} \mu'(\theta, \boldsymbol{U}), \tag{1.37}$$

i.e., if we can interchange the expectation with the partial derivative. Section A.2 of the appendix provides conditions under which this interchange is valid. In particular, the *dominated convergence theorem* (Theorem A.2) implies that if there is a $\delta_1 > 0$ and a random variable Y such that

[10] From Pierre: Conflict of notation here: X_1 and X_2 are used earlier with a different meaning.

$$\sup_{\delta\in(0,\delta_1]} \frac{|f(\theta+\delta,\boldsymbol{U}) - f(\theta,\boldsymbol{U})|}{\delta} \le Y$$

and $\mathbb{E}[Y] < \infty$, then the interchange in (1.37) is valid.

If we want to estimate a *gradient*, i.e., a vector of derivatives with respect to several parameters, we can compute the stochastic derivative with respect to each of those parameters from a single simulation run, regardless of how many parameters there are. The vector of those stochastic derivatives is the *stochastic gradient*.

Example 1.47 In the stochastic activity network of Example 1.4, suppose we want to estimate the derivative of $\mathbb{E}[T]$ with respect to each θ_j. We will consider each θ_j separately, assuming that the other parameters are fixed. We can write T as a function of θ_j as $T = f_j(\theta_j, \boldsymbol{U})$, where $\boldsymbol{U} = (U_1, \ldots, U_{13})$ is the vector of 13 independent $U(0,1)$ random variables that drive the simulation,

Here we have $f_j'(\theta_j, \boldsymbol{U}) = Y_j'(\theta_j)$ if link j belongs to the longest path, and $f_j'(\theta_j, \boldsymbol{U}) = 0$ otherwise, because an infinitesimal change in $Y_j = Y_j(\theta_j)$ changes T by the same amount if Y_j contributes to the longest path, and does not affect T otherwise. If j is the index of an exponential random variable with mean θ_j, we have $Y_j = Y_j(\theta_j) = -\theta_j \ln(1-U_j)$ and $Y_j'(\theta_j) = -\ln(1-U_j)$. We also have

$$0 \le \frac{f_j(\theta_j+\delta,\boldsymbol{U}) - f_j(\theta_j,\boldsymbol{U})}{\delta} \le \frac{-\delta\ln(1-U_j)}{\delta} = -\ln(1-U_j) = \tilde{Y}_j,$$

where $\tilde{Y}_j$ is an exponential random variable with mean 1. Thus, the dominated convergence theorem applies and the stochastic derivative provides an unbiased estimator. A similar analysis can be made for the case where Y_j has a normal distribution: We have $Y_j = Y_j(\theta_j) = \theta_j + (\theta_j/4)\Phi^{-1}(U_j)$ and $Y_j'(\theta_j) = 1 + \Phi^{-1}(U_j)/4$. Thus, for this example, the stochastic gradient is an unbiased estimator of the gradient with respect to the vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{13})$.

♣ Numerical illustration: estimate the 13-dimensional gradient, with $n = 100,000$. Compare stochastic gradient with finite differences. □

Example 1.48 In Example 1.47, suppose we want to estimate the derivative of $\mathbb{P}[T > x]$ instead of the derivative of $\mathbb{E}[T]$, and that our basic estimator of $\mathbb{P}[T > x]$ as a function of θ_j is the indicator $f_j(\theta_j, \boldsymbol{U}) = \mathbb{I}[T > x]$, as in Example 1.4. This estimator can only take two values: 0 and 1. Thus, its derivative $f_j'(\theta_j, \boldsymbol{U})$ is either undefined (when the function has a jump exactly at θ_j, which happens with probability 0) or is 0. That is, $f_j'(\theta_j, \boldsymbol{U}) = 0$ w.p.1, and therefore it cannot be an unbiased estimator of $\mu'(\theta_j) = \partial\mathbb{P}[T > x]/\partial\theta_j$. Here we cannot apply the dominated convergence theorem to justify the interchange of derivative and expectation because $[f_j(\theta_j+\delta,\boldsymbol{U}) - f_j(\theta_j,\boldsymbol{U})]/\delta$ can be $1/\delta$ for any $\delta > 0$, which is unbounded when $\delta \to 0$. The problem is that f_j is discontinuous at a random point as a function of θ_j, and the finite difference is unbounded at the discontinuity. The stochastic derivative is a useless estimator in this case.

In Example 6.21, we will introduce a different estimator of $\mathbb{P}[T > x]$, with smaller variance than the indicator $\mathbb{I}[T > x]$, based on conditional Monte Carlo. It will also be continuous everywhere in each θ_j and its sample derivative will provide an unbiased estimator of $\mu'(\theta_j)$. □

For a discrete-event model that involves thousands or millions of events and where the performance measure $f(\theta, \boldsymbol{U})$ depends on θ in a very complicated way (for example, a large queueing or supply chain model), special techniques have been developed to track down all the changes in the sample path caused by an infinitesimal change in θ, in order to compute $f'(\theta, \boldsymbol{U})$. This is known as *infinitesimal perturbation analysis (IPA)* (Fu 2006, Glasserman 1991, L'Ecuyer 1990b). These techniques are often coupled with various ways of changing the definition of $f(\theta, \boldsymbol{U})$ when it is originally discontinuous in θ, to make it smoother and continuous so that IPA provides an unbiased estimator. Derivative estimation is studied further in Chapter 6.

♣ Add exercises: Greeks in financial options

More references: Asmussen and Glynn (2007), Fu (2006), Glasserman (1991), Glasserman (2004), Glynn (1990), L'Ecuyer (1990b), L'Ecuyer (1991), L'Ecuyer (1992), L'Ecuyer (1993).

1.9 Discrete-Event Models and Simulation

1.9.1 Evolution of a discrete-event model

A *discrete-event* model is one whose evolution can be described by a (discrete) sequence of events $e_0, e_1, e_2, \ldots$ which occur at respective times (or epochs) $0 = t_0 \le t_1 \le t_2 \le \cdots$. We denote by $\mathcal{S}_i$ the state of the model at time t_i, immediately after event e_i has occurred but before the occurrence of event e_{i+1} (see Figure 1.23). The possibility of several events occurring "simultaneously" (i.e., $t_i = t_{i+1} = \cdots t_{i+k}$) is not ruled out, but these events must occur in a well-defined order. It is important to understand that the epochs t_i are with respect to the *model clock* and do not correspond to the time on the computer clock, or to the elapsed CPU time at the occurrence of event e_i, when the simulation program is run on a computer. The current time on the model clock during a simulation is called the *simulation time*.

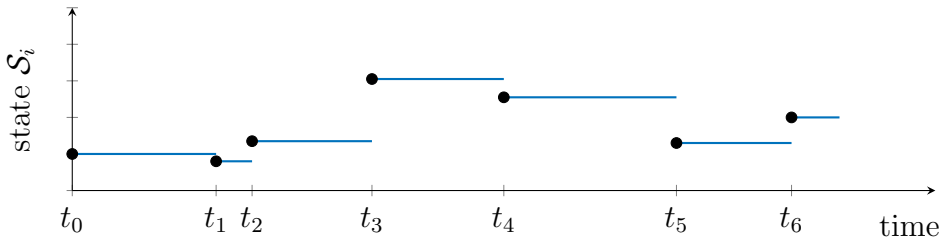


Fig. 1.23. Evolution of a discrete-event model

The event e_0 can be viewed as the one that determines the initial state $\mathcal{S}_0$. In practice, $\mathcal{S}_0$ is often fixed a priori and e_0 is just a dummy event. The sequence of states $\{\mathcal{S}_i, i \ge 0\}$ evolves in a *state space* which is typically multidimensional and uncountable. We assume that $\mathcal{S}_i$ contains enough information for the process $\{(t_i, \mathcal{S}_i), i \ge 0\}$ to be a *Markov chain*. From a practical point of view, this means that $(t_i, \mathcal{S}_i)$ contains (at least) all the information

that must be memorized by the computer program at the simulation time t_i to pursue its evolution. This is not a limiting requirement, because one can always add enough information to the state to make the process Markovian. This is discussed a little further in Section 2.12.1.

1.9.2 Example: a single-server queue

To illustrate how a discrete-event simulation operates, we take a very simple example, a *single-server queue*, also called a $GI/G/1$ *queue* in the terminology of queuing theory (Section A.19). It is defined as follows. Customers arrive randomly to the single server. They are served one by one in *first come, first served (FCFS)* order. Let S_i denote the service time of the ith arriving customer and A_i be the time between the arrivals of customers $i-1$ and i. The S_i's and A_i's are mutually independent random variables, with respective cdfs G and F. At time 0, the system is empty. The first customer arrives at time A_1, leaves at time $A_1 + S_1$, and so on. For the special case where the A_i's are i.i.d. exponential and the S_i's are also i.i.d. exponential, we have an $M/M/1$ queue. Let W_j be the waiting time of customer j.

This simplistic model is hardly to be taken seriously as representative of real-life systems that need to be simulated. Its purpose is to give the flavor of how discrete-event simulation works. For more concreteness, one can view the server as either a clerk, a machine, a computer, or a dock, and the customers as people, parts, jobs, or boats. So, the system could be boats coming to a dock to be loaded with, say, iron ore.

We saw earlier how to simulate the sequence of W_j's with Lindley's recurrence. Here we will simulate the system with discrete events and a simulation clock. This approach applies more generally: it would also work when there are many servers, various types of customers, priority rules, etc.

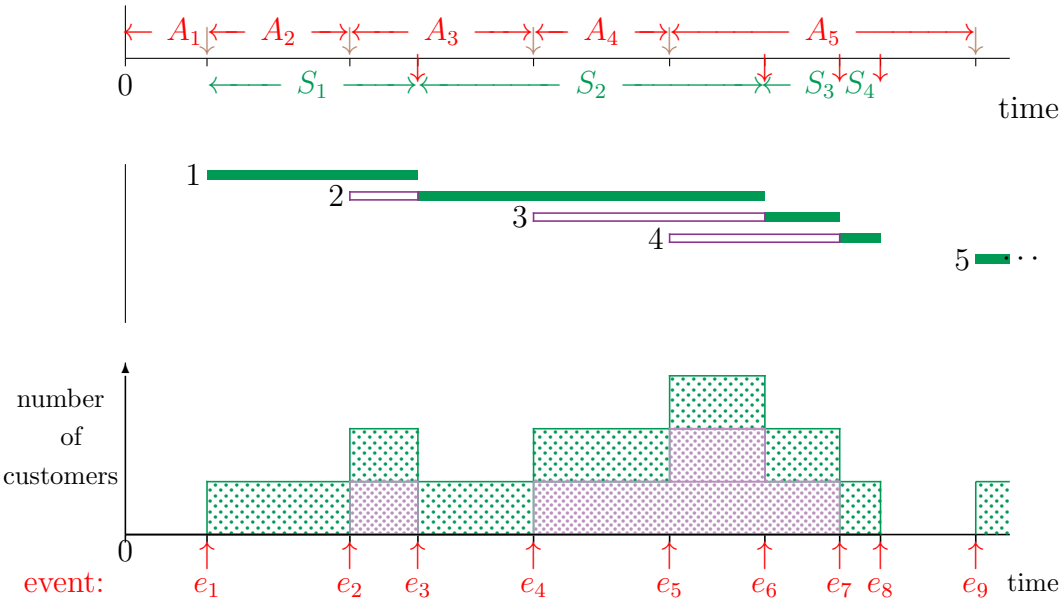


Fig. 1.24. Occurrence of Events for the Single-Server Queue

Figure 1.24 shows how this simple queuing system typically evolves in time, represented by the horizontal axis. In the upper part of the figure, the arrows above the time axis represent the customer arrivals, while those below the axis represent the departures. In the middle part, each horizontal band represents one customer; the green part of the band is the service time and the light part is the waiting time of the customer in the queue. The lower part of the figure shows how the number of customers in the system evolves with time. The height of the fuchsia area represents the number of customers in service (always zero or one) and the height of the light green one is the number of customers waiting in the queue. The time scale is the same for all three parts of the figure.

For the particular sample path realization illustrated in that figure, the second customer arrives before the first one completes its service ($A_2 < S_1$). Then, customers number 3 and 4 arrive while customer 2 is being served, so a queue builds up. Customers 3 and 4 then complete their service before customer 5 arrives, so the system empties out when customer 4 departs. In this model, all things of interest happen when either (a) a customer arrives or (b) a customer leaves the system or (c) the simulation ends. These are the *events*. The arrows at the bottom of Figure 1.24 point to the event epochs $t_1, t_2, \ldots$. Each of these events is either an *arrival* or a *departure*. The two events e_1 and e_9 mark the beginning of new *busy cycles* for the queue: They correspond to a customer arriving in an empty system.

To simulate such a system, one needs to generate the random variates $A_1, A_2, \ldots$ and $S_1, S_2, \ldots$ (not necessarily in that order) and, from that, to retrace the sequence of events e_i together with their epochs t_i. In typical discrete-event simulations, this is implemented using a list of event notices, also called the *event list*. Each *event notice* indicates an event that is already scheduled to occur in the future, and gives its scheduled time of occurrence, with possibly some additional parameters. Event notices appear in the list by increasing order of scheduled occurrence. During the execution of the simulation, event notices are added to the list dynamically (*event scheduling*) and the simulation program repeatedly removes the first event notice from the list, sets the simulation clock to the epoch of the corresponding event, and executes that event, until either a particular event stops the simulation or the event list gets empty. In some cases, event notices may also be removed before ever reaching the head of the event list (*event cancellation*).

Events are typically classified into several *event types*, and each event type corresponds to a piece of code to be executed by the program. The role of this code is to update the state of the system appropriately, given that an event of that type occurs. This may include, in particular, generating other random variables, collecting some statistics, and scheduling other (future) events.

Suppose we want to simulate the single-server queue for T units of simulated time, where T is fixed. We consider three types of events here: *Arrival*, *Departure*, and *EndOfSimulation*. An event of the latter type occurs only once, when time T is reached. The corresponding procedure stops the simulation and may print a report on whatever statistics we are interested in. Now, suppose that each customer is represented by an instance of a special data type called *customer* in our program, which memorizes the arrival time and service time of the customer. The program maintains a list of the customers waiting in the queue (called the *waiting list*), as well as a list of the customers being served (called the *server's list*). The latter list may seem unnecessary because it will never contain more than one customer, but this data structure will also work if the queuing model has more than one server, and in

other more general situations. The *Arrival* and *Departure* events can be implemented by procedures which we outline as follows. Indentation delimits the scope of the **If** and **else** statements. Depending on the programming language, these can be programmed as methods, functions, subroutines, etc.

Event *Arrival*;
  Generate a random variable A from distribution F;
  Schedule a new (the next) arrival to occur in A time units;
  Create a new customer (the one who is arriving now);
  Generate its service time S from distribution G;
  If the server's list is not empty **then**
    Insert this new customer at the end of the waiting list;
  else
    Insert this new customer in the server's list;
    Schedule a departure (for this new customer) to occur in S time units;
  Update the statistics as needed;

Event *Departure*;
  Remove the departing customer from the server's list;
  If the waiting list is not empty **then**
    Remove the first customer from the waiting list;
    Insert it in the server's list;
    Recover its service time S;
    Schedule its departure in S time units;
  Update the statistics as needed;

Event *EndOfSimulation*;
  Stop the simulation executive;

At the beginning of the simulation, one must initialize the lists and the appropriate statistical accumulators, schedule the end-of-simulation event at time T, generate A_1 from distribution F, schedule an arrival at time A_1, and start the *simulation executive*, which will repeatedly remove the first event in the event list and execute it.

The question of which statistics should be collected depends on what we want to estimate. For example, suppose we are interested in (a) the average waiting time in the queue per customer for those customers who have started their service, (b) the average number of customers in the queue (excluding those in service) as a function of time. Then, we need statistical accumulators for the following: (i) number of customers who have started their service, to date, (ii) total waiting time for those customers, (iii) the integral, from time zero to the current simulation time, of the size of the waiting list as a function of time.

Let W_i denote the waiting time in the queue for customer i, $Q(t)$ the number of customers waiting in the queue at time t, $N_c(t)$ the number of customers who started their service during the time interval $[0, t]$, and let $N_c = N_c(T)$, where T is the time horizon. Here, T is deterministic and N_c is a random variable. The statistics in (a) and (b) above are precisely

$$\bar{W}_{N_c} = \frac{1}{N_c} \sum_{i=1}^{N_c} W_i$$

and

$$\bar{Q}_T = \frac{1}{T}\int_0^T Q(t)\mathrm{d}t.$$

Those customers who arrive while the server is idle will have their W_i equal to zero, but are nevertheless counted in the average. In particular, if the system starts empty, one has $W_1 = 0$. Note that the first average is computed over a (random) integer number of customers N_c. The second average is with respect to time and involves the integral of $Q(t)$. Those are two very different types of averages. Since $Q(t)$ is piecewise-constant, the integral for $\bar{Q}_T$ is easy to compute: It is a sum of rectangular areas, and corresponds to the surface of the lightly-shaded green area, from 0 to T, in the bottom part of Figure 1.24.

1.9.3 Event list management

– To do.

1.10 Software for Simulation Programming

[11]

1.10.1 Overview of stochastic simulation software

Programming simulation models to run them on computers requires functions to generate the uniform random numbers and non-uniform random variates from various distributions, manage the event list and the simulation executive, collect the appropriate statistics, compute confidence intervals, produce plots, etc. Rewriting code to do this for every model would be time-consuming and error-prone. Fortunately, specialized software tools have been developed to ease simulation modeling and programming. These tools belong to the following categories:

(a) Extensions/libraries for general-purpose programming languages;

(b) Special-purpose simulation programming languages;

(c) Graphical modeling/programming environments for general simulation;

(d) Specialized modeling environments for specific classes of applications.

Almost every programming language or environment offers simple functions to generate uniform random numbers and random variates from a few distributions, and this can be sufficient to simulate simple static models such as those given in Section 1.2. Functions to generate random numbers were already deemed important and made available in the first digital computers in the 1940's (L'Ecuyer 2017). Functions to compute quasi-random points from lattice rules or digital nets in base 2 are also available in several software libraries.

[11] From Pierre: To be updated. Here we should have a section on "stochastic simulation programming" with a subsection that gives an overview of SSJ.

Statistical data coming out from simulations can be analyzed using R, SAS, Python libraries, and other similar software tools which offer rich collections of statistical and visualization procedures.

Programming large discrete-event simulation models, on the other hand, requires more advanced structures and tools.

Early days: – manufacturing systems, communication systems, queues, ... This gave rise to specialized languages and libraries even from the early days of computer programming.

This section is currently in reconstruction...

When I started teaching simulation in 1983, discrete-event simulation programs were often written in specialized simulation languages. The leading ones were GPSS and SIMSCRIPT, and SIMAN was coming out. I picked SIMSCRIPT 2.5 for my class the first year, mainly because it was free. It was a very wordy language, almost like writing English sentences, the instructions were often interpreted differently than we expected, and the compiler was not very efficient. For the second year, I wrote my own (small) library in Pascal. Then we built a more extensive library in Modula-2 (L'Ecuyer and Giroux 1987, L'Ecuyer 1988), and finally SSJ (L'Ecuyer, Meliani, and Vaucher 2002, L'Ecuyer and Buist 2005), described in Section 1.10.

The special-purpose simulation programming languages had drawbacks and have mostly disappeared. Having to learn a new language only for simulation programming is not convenient, and such a specialized language cannot benefit from the same type of support as the most popular general-purpose languages (e.g., the availability of efficient compilers, libraries, and integrated development environments on several platforms). Ideally, simulation programming should be done in a well-designed and widely used language that offers the primitives required to support nice and clean simulation extensions, and allows fast execution.

Libraries for simulation programming are available for general-purpose languages such as C, C++, Java, Python, etc., usually in the public domain. Tools for simulation are also available for computing environments such as MATLAB, R, SAS, Excel, etc. One can consult the annual *Proceedings of the Winter Simulation Conference* for references; see `https://informs-sim.org/`. Some libraries provide only a small set of tools, others provide more.

Simulation programming environments with graphical interfaces have replaced the specialized languages. They are commercial products that can be purchased from simulation software vendors. Popular ones nowadays include *AnyLogic*, *Arena*, *Automod*, *ExtendSim*, *FlexSim*, *Simio*, and *Simul8*, *Witness*, for example. Certain environments are for discrete-event simulation in general, others are tailored to narrow classes of applications such as manufacturing, scheduling, business process re-engineering, and communication systems (Banks 1998, Chap. 25). Typically, the more specialized tools are easier to use than the general-purpose ones when the application fits, but they also tend to be less flexible. These graphical environments are often advertised as tools to do "simulation without programming". This means that models can be built simply by "point-and-click" operations, with the mouse. More often than not, however, the templates provided by the graphical interface do not cover certain aspects of a model, so one must go down to lower-level programming, either in the base language or in a general-purpose language. See also Law (2014), Chapter 3 and `https://en.wikipedia.org/wiki/List\_of\_discrete\_event\_simulation\_software`.

These four categories do not determine a clean-cut partition: Several products cover more than one category. For example, *Arena*, from Rockwell Software, belongs to (c), but

is built over the simulation language SIMAN and permits one to write part of the code in general-purpose languages such as VisualBasic and C.

♣ Add an outline of the basic facilities in a good simulation library: rng, probdist, randvar, rqmc, event-scheduling, stat collection, etc.

1.10.2 The SSJ library

We do not intend here to teach simulation programming in a specific language. However, to illustrate how simulation models can be programmed, we exhibit small examples of programs written in Java using the SSJ library (L'Ecuyer and Buist 2005, L'Ecuyer 2023). With the given explanations, even those not familiar with Java should be able to understand what goes on in the examples. Some familiarity with object-oriented programming would nevertheless make things easier. In our program examples, the `import` statements (for SSJ and standard Java packages) have been left out to save space; the running Java code for these examples is available with the SSJ distribution.

Give an overview of SSJ, with some examples.

1.11 Example: simulation of a single-server queue

♣ This example could be replaced by a queue with multiple servers.

1.11.1 Discrete-event simulation of the single-server queue

The program given in Figure 1.25 simulates the single-server queue of Section 1.9.2 for a fixed time horizon T, using the event-oriented approach outlined in Section 1.11 and the SSJ library. The simulation is repeated for `numRep` independent replications, and a short report is printed for each replication. The simulation program is implemented as a Java class called `QueueEv`, instantiated by the `main` method via the statement "`new QueueEv()`". This statement calls the *constructor* `QueueEv`, which creates an object `queue` of class `QueueEv`. Here we have an $M/M/1$ queue and the two parameters of the constructor are the arrival rate λ and the service rate μ. Each simulation run (or replication) is performed by the method `simulate`. This method first makes sure that the lists are empty and initializes the statistical collectors (this is needed to clean up the information from previous runs). It then initializes and starts the simulation. When the simulation is over, the `main` method prints a statistical report.

Each event type is implemented as a subclass of the predefined class `Event`, whose method `actions` is invoked whenever this event occurs. These event types are: arrival of a customer (`Arrival`), departure of a customer (`Departure`), and end of the simulation (`EndOfSimulation`). The simulation clock and the event list are managed behind the scenes by the class `Sim` of SSJ. Each event *instance* is inserted into the *event list* with a scheduled time of occurrence by the `schedule` method, and is *executed* when the simulation clock reaches this time.

```
public class QueueEv {
   RandomVariateGen genArr, genServ;
   LinkedList<Customer> waitList = new LinkedList<Customer> ();
   LinkedList<Customer> servList = new LinkedList<Customer> ();
   Tally custWaits     = new Tally ("Waiting times");
   Accumulate totWait  = new Accumulate ("Size of queue");

   class Customer { double arrivTime, servTime; }

   public QueueEv (double lambda, double mu) {
      genArr = new RandomVariateGen (new MRG32k3a(),
                       new ExponentialDist(lambda));
      genServ = new RandomVariateGen (new MRG32k3a(),
                       new ExponentialDist (mu));
   }

   public void simulate (double timeHorizon) {
      Sim.init();
      waitList.clear();  servList.clear();
      custWaits.init();  totWait.init();
      new EndOfSimulation().schedule (timeHorizon);
      new Arrival().schedule (genArr.nextDouble());
      Sim.start();
   }

   class Arrival extends Event {
      public void actions() {
         new Arrival().schedule (genArr.nextDouble()); // Next arrival.
         Customer cust = new Customer();  // Customer that just arrived.
         cust.arrivTime = Sim.time();
         cust.servTime = genServ.nextDouble();
         if (servList.size() > 0) {       // Must join the queue.
            waitList.addLast (cust);
            totWait.update (waitList.size());
         } else {                         // Starts its service.
            custWaits.add (0.0);
            servList.addLast (cust);
            new Departure().schedule (cust.servTime);
         }
      }
   }

   class Departure extends Event {
      public void actions() {
         servList.removeFirst();   // Remove from service.
         if (waitList.size() > 0) {
            // Start service for next one in queue.
            Customer cust = (Customer)waitList.removeFirst();
            totWait.update (waitList.size());
            custWaits.add (Sim.time() - cust.arrivTime);
            servList.addLast (cust);
            new Departure().schedule (cust.servTime);
         }
      }
   }
```

Fig. 1.25. Event-oriented simulation of a single-server queue.

```
   class EndOfSimulation extends Event {
      public void actions() {
         Sim.stop();
      }
   }

  public static void main (String[] args) {
      double lambda = 1.0/10.0; // Arrival rate.
      double mu = 1.0/9.0;      // Service rate.
      double T = 1000.0;        // Time horizon.
      int n = 8;                // Number of simulation replications.
      QueueEv queue = new QueueEv (lambda, mu);
      for (int rep = 0; rep < n; rep++) {
         queue.simulate (T);
         System.out.println (queue.custWaits.report());
         System.out.println (queue.totWait.report());
      }
   }
}
```

Fig. 1.25. Event-oriented simulation of a single-server queue (continuation).

The five statements at the beginning of class `QueueEv` declare two random variate generators, then create two lists and two statistical collectors. When the class `QueueEv` is instantiated by the `main` method, the latter four objects are first created, then the two random variate generators are created by the constructor, each with an attached random number stream and an exponential distribution. We now explain what these objects are.

Objects of the class `MRG32k3a` in SSJ are *random number streams* as introduced at the end of Section 1.3. They return sequences of numbers that can be taken as independent uniform random variables over $(0, 1)$. These sequences of numbers are in fact disjoint segments of the sequence produced by a single generator with a huge period length. We will see in Chapter 3 how this is implemented, and in Section 1.7 why it is useful to have several streams. Each `ExponentialDist` object represents an exponential distribution whose rate is given by its parameter (its mean is the inverse of the rate). These streams and distributions are attached to the random variate generators `genArr` and `genServ` which are used to generate the times between successive arrivals and the service times, respectively. Each call to the method `nextDouble` for one of these generators returns an exponential variate generated by inversion from the attached distribution and using the attached random number stream. Why not just use the same stream to generate both the inter-arrival and service times? For this particular (very simple) simulation there is no special reason, but this is good practice in general and we will see why later on in the book.

The class `LinkedList`, provided by the standard `java.util` package, implements lists of objects (of any type). Two lists are created here: `waitList` is the list of waiting customers and `servList` is the list of customers in service. Methods are provided to manipulate those lists, insert and remove objects in/from them, and so on. For example, `waitList.addLast (cust)` inserts the customer `cust` at the tail of the list of waiting customers, while `waitList.removeFirst` removes and returns the first customer in the queue.

The `Tally` and `Accumulate` objects are statistical collectors, or *probes*. They correspond to the two different flavors of probes typically available in simulation packages. The `Tally`

type (e.g., `custWaits`) is appropriate when the statistical data of interest consist of a sequence of observations $X_1, X_2, \ldots$ of which we might want to compute the sample mean, variance, and so on. One can bring a new observation X_i to a `Tally` probe by calling its method `add`. In the program of Figure 1.25, new observations are added to the probe `custWaits` by calling `custWaits.add(x)` every time a customer starts it service (this is when we know its waiting time), where the observation `x` is the waiting time W_i of that customer (the current time minus its arrival time). At the end of the simulation, the total number of observations W_i given to the probe will be $N_c = N_c(T)$.

The `Accumulate` type of probe (e.g., `totWait`), on the other hand, is used to integrate (or compute the time-average) of the value of a continuous-time stochastic process with piecewise-constant trajectory. Whenever the process value changes in time (this may occur only at event times), the method `update` should be called to give the new value. The system then updates the current value of the integral by adding to it the previous value of the process multiplied by the time elapsed since the last update. In our program, the probe `totWait` is updated every time the size of the waiting list changes. An accumulator hidden in that probe is equal, after each update, to the total waiting time in the queue, for all the customers (i.e., the integral of $Q(t)$), since the beginning of the simulation.

Each customer is an object of the class `Customer`, with two fields: `arrivTime` memorizes this customer's arrival time to the system, and `servTime` memorizes its service time. This object is created, and its fields are initialized, when the customer arrives.

The method `simulate`, invoked by the `main` program, first calls `Sim.init()` to initialize the clock and the event list. It then schedules two events: The end of the simulation (an `EndOfSimulation` event) at time $T =$`timeHorizon` and the arrival of the first customer. The time until this first arrival is an exponential random variable with mean $1/\lambda$, generated using the random stream `genArr`. Finally, calling `Sim.start()` starts the simulation by advancing the clock to the time of the first event in the event list, removing this event from the list, and executing it. This is repeated until either `Sim.stop()` is called or the event list becomes empty. `Sim.time()` returns the current time on the simulation clock.

The method `actions` of the class `Arrival` describes what happens when an arrival occurs. Arrivals are scheduled by a domino effect: The first action of each arrival event is to schedule the next one in A time units, where A is an exponential with mean $1/\lambda$. Then, the newly arrived customer is created, its arrival time is set to the current simulation time, and its service time is generated from the exponential distribution with mean $1/\mu$. If someone is already in service, the customer is inserted in the waiting list (the queue) and the statistical probe `totWait` that keeps track of the size of that list is updated. Otherwise, the customer is inserted in the server's list, its departure is scheduled in S time units where S is the customer's service time, and its waiting time, equal to 0 in this case, is given as a new observation to the statistical probe for the waiting times.

When a `Departure` event occurs, the customer in service is removed from the list and disappears. If someone is waiting, then the first in the queue is removed and inserted in the server's list, and its departure is scheduled. The waiting time of that customer (the current time minus its arrival time) is computed and given as an observation to the probe `custWaits`. The probe `totWaits` that keeps track of the size of the waiting list is also updated.

The event `EndOfSimulation` stops the simulation.

The program assumes that the interarrival and service time distributions are exponential with *means* 10 and 9 (or *rates* 1/10 and 1/9), respectively, that the time horizon is $T = 1000$, and that we make 8 simulation runs. These parameter values are fixed in the `main` program to simplify the code. Note that the units of time are never specified: they can be seconds, minutes, hours, or someting else. It does not matter, as long as we always use the same units. After each simulation run, the `main` method prints statistical reports for the two probes. In a serious simulation experiment, we would also compute confidence intervals from these runs, perhaps build a histogram of the waiting times, etc., but our goal here is just to illustrate how to program the simulation logic.

It is important to understand that to obtain n independent simulation runs, one *cannot* just put $n = 1$ in the program and run it n times on the computer. By doing that, one would obtain exactly the same results (the results for the first run) n times, because the RNGs streams will restart from their same default seeds for each run. By making a loop in a single program, the RNGs keep moving ahead across the runs. Another option would be to change the RNG seeds explicitly between the runs, but this is more cumbersome and not recommended. Keeping the same default seeds (as opposed to using truly random seeds) makes the simulation experiments reproducible, and this has enormous advantages, as we shall see along the book.

```
REPORT on Tally stat. collector: Waiting times
    num. obs.     min         max      average    variance    standard dev.
        97       0.000     113.721     49.554     498.883       22.336

REPORT on Accumulate stat. collector: Size of queue
    from time  to time      min       max       average
      0.00     1000.00     0.000     12.000      4.850
```

Fig. 1.26. Reports from the program `QueueEv` for the first simulation run.

Figure 1.26 shows the output of the program `QueueEv` for the first simulation run (only). By time $T = 1000$, 97 customers have completed their waiting ($N_c = 97$). The peak and time-average queue sizes have been 12 and 4.85, respectively. The waiting times are spread out from 0 to 113.72, with an average of 49.55. The report also gives the empirical variance and standard deviation of the waiting times. This is typical of simulation software which often provides some irrelevant information: here the empirical variance is not really useful because the 97 waiting times are *not* independent observations; they are (strongly) positively correlated. It would therefore be *grossly wrong* to use the average and standard deviation of these 97 waiting times to compute a confidence interval on the expected average waiting time

$$w_T = \mathbb{E}[\bar{W}_{N_c}] \tag{1.38}$$

via the standard approach with the normal distribution. This standard procedure assumes i.i.d.observations, so it would underestimate the true variance and therefore produce and overly optimistic interval; that is, with a much lower coverage than specified (the *coverage* is the probability that the interval includes the exact value of the expectation).

To compute a confidence interval for $w_T = \mathbb{E}[\bar{W}_{N_c}]$, one may replicate the simulation n times with independent random numbers, then use the standard procedure of Section 1.4.3 with these n replicates of $\bar{W}_{N_c}$ as independent observations. To give an idea of how these

Table 1.5. Results for $n = 8$ simulation runs with `QueueEv`, with $T = 1000$.

Run	N_c	$\bar{W}_{N_c}$	$\bar{Q}_T$
1	97	49.55	4.85
2	96	15.64	1.58
3	78	79.26	6.88
4	93	78.28	8.69
5	103	57.22	6.85
6	100	13.15	1.44
7	93	45.61	4.24
8	93	98.04	9.31

observations would look like, Table 1.5 shows the results for $n = 8$ independent simulation runs. We observe large variations of the averages $\bar{W}_{N_c}$ and $\bar{Q}_T$ between runs. For example, they are about 98 and 9.3 for run 8, compared to about 13 and 1.4 for run 6. Interestingly, run 6 had more customers than run 8, 100 vs 93. This means that the larger waiting times for run 8 were not due to more customers than usual, but probably from clustered arrivals at some point during the time period. We see that even if $\bar{W}_{N_c}$ is an average over approximately a hundred customers, it cannot be trusted very much as an estimator of w_T. Moreover, even if they were known with great precision, w_T and q_T would only tell a small part of the story; it is also important to be aware of the high variability of the waiting times and of the averages $\bar{W}_{N_c}$ and $\bar{Q}_T$.

A confidence interval on w_T is also unlikely to be what we want, for another reason. We may be interested in the expected waiting time of an "average" customer, i.e., a customer picked at random among all the customers who arrive to this system over an infinite (or very large) number of "days" of length T. But this expected waiting time *is not* w_T, because in w_T, the customer waits are not all weighted equally. When computing $\bar{W}_{N_c}$, each waiting time is divided by the number N_C of the current run, so the waiting time of a customer who is in a run with a larger N_C than the average will be divided by a larger number and therefore will count less in the global average, and the opposite when N_C is smaller than the average. To estimate the expected waiting time of an average customer in the above sense, we should rather sum the waiting times of all the customers over all runs and divide by the total number of customers over all runs. A confidence interval can be computed as explained in Section 5.4.2. A histogram of all the individual nonzero waiting times would also be informative in this case.

Alternatively, instead of considering an average customer over a large number of runs of fixed length T, one may be interested in an average customer over a single very long simulation run (i.e., when $T \to \infty$). We will examine this in Section 1.11.5.

1.11.2 A process-oriented program

Certain simulation languages offer higher-level constructs than those used in the program of Figure 1.25. This is illustrated by our second implementation of the single-server queue model, in Figure 1.27, based on a paradigm called the *process-oriented* or *process interaction*

```
public class QueueProc {

   Resource server = new Resource (1, "Server");
   RandomVariateGen genArr;
   RandomVariateGen genServ;

   public QueueProc (double lambda, double mu) {
      genArr = new RandomVariateGen (new MRG32k3a(),
                     new ExponentialDist(lambda));
      genServ = new RandomVariateGen (new MRG32k3a(),
                     new ExponentialDist (mu));
   }

   public void simulate (double timeHorizon) {
      SimProcess.init();
      server.setStatCollecting (true);
      new EndOfSimulation().schedule (timeHorizon);
      new Customer().schedule (genArr.nextDouble());
      Sim.start();
   }

   class Customer extends SimProcess {
      public void actions() {
         new Customer().schedule (genArr.nextDouble());
         server.request (1);
         delay (genServ.nextDouble());
         server.release (1);
      }
   }

   class EndOfSimulation extends Event {
      public void actions() {
         Sim.stop();
      }
   }

   public static void main (String[] args) {
      QueueProc queue = new QueueProc (1.0/10.0, 1.0/9.0);
      queue.simulate (1000.0);
      System.out.println (queue.server.report());
   }
}
```

Fig. 1.27. Process-oriented simulation of a single-server queue

approach, initiated in the early sixties by the GPSS and Simula languages (Schriber 1974, Nygaard and Dahl 1978, Birtwistle et al. 1979). The program of Figure 1.27 gives a good sense of how process interaction works. It is taken from an early version of SSJ, as described in L'Ecuyer, Meliani, and Vaucher (2002), in which processes were implemented using "green thread" facilities that are no longer available in Java. For this reason, the processes are not supported in the current version of SSJ, but they could be eventually re-introduced by using a lightweight coroutine library for Java. The SimPy simulation library in Python (SimPy 2020) is one example of a recent simulation library based on process interaction.

In the event-oriented implementation, each customer is a *passive* object that just stores two real numbers and performs no action by itself. In the process-oriented implementation of Figure 1.27, each customer (instance of the class `Customer`) is a process whose behavior is described by its method `actions`. The server is an object of class `Resource`, created when `QueueProc` is instantiated by `main`. A `Resource` can be viewed as a service facility, with an arbitrary number of servers, and a specific service discipline. It is a *passive* object, in the sense that it executes no code. Active resources, when needed, can be implemented as processes. Here, the number of servers is 1 (specified when `server` is created) and the service discipline is FCFS (the default value).

When it starts executing its actions, a customer first schedules the arrival of the next customer, as in the event-oriented case. Processes are scheduled in the same way as events. Here, customers generate each other sequentially as in the event-view implementation. Behind the scenes, the `schedule` statement simply inserts in the event list an event notice which, when executed, will create and activate a new `Customer` instance. Several distinct `Customer` instances can co-exist in the simulation at any given point in time, and be at different steps in their `actions` method. After scheduling the next one, the customer requests the server by invoking `server.request(1)`. The parameter 1 indicates that *one* server is requested. If the server is free, the customer gets it and can continue its execution immediately. Otherwise, the customer is automatically (behind the scenes) placed in the server's queue, is suspended, and resumes its execution only when it obtains the server. When its service can start, the customer invokes `delay` to freeze itself for a duration equal to its service time, generated from the exponential distribution by the generator `genServ`. Invoking `delay(d)` schedules an event (behind the scenes) that will resume the execution of the process in `d` units of time. After this delay has elapsed, the customer releases the server and dies.

We point out that here, the service time is generated only when the customer starts its service, whereas in the event-oriented program (Figure 1.25), it was generated at customer's arrival. This difference in the implementation turns out to have no effect on the results for this particular model, because the customers are served in a FIFO order and because one random number stream is dedicated to the generation of service times. However, it may have considerable impact in other situations, when comparing systems with common random numbers; we will see examples in Chapter 6.

The `simulate` method initializes the simulation, invokes `setStatCollecting` to specify that detailed statistical collection must be performed automatically for the resource `server`, schedules an event `EndOfSimulation` at time $T =$ `timeHorizon`, schedules the first customer's arrival, and starts the simulation. When the simulation is over, the `main` program prints a detailed statistical report on the resource `server`. This illustrates how events and processes can be mixed in a simulation program. Behind the scenes, a single event list takes

care of synchronizing the execution of all the process instances (customers) and events. Processes are created, suspended, reactivated, and so on, by hidden events.

The process-oriented program of Figure 1.27 is more compact and more elegant than its event-oriented counterpart of Figure 1.25. This tends to be true in general: Process-oriented programming often gives less cumbersome and better looking programs. On the other hand, process-oriented implementations usually execute more slowly, because managing the pseudo-concurrent processes (hidden in the simulation package) involves overhead. When the execution time is an important issue, it is often better to stick with an event-view implementation, even if it involves a little more programming. Moreover, using processes instead of events does not always simplify the simulation program.

1.11.3 Streamlining the program using Lindley equations

The single-queue model that we just used to illustrate the notions of events and processes is in fact a special case of the tandem-queue model examined in Example 1.10, so it can be simulated more simply just by using the recurrence equations given there. The waiting times obey the Lindley recurrence (1.7): $W_1 = 0$ and

$$W_i = \max(0,\ W_{i-1} + S_{i-1} - A_i) \tag{1.39}$$

for $i \geq 2$. The program of Figure 1.28 simulates this recurrence and computes the average waiting time of the first 100 customers. The statistics computed here are different than in the two preceding programs, because here we fixed N_c instead of T. The program also illustrates a different way of generating the exponential random variates in SSJ, without creating random variate generator and distribution objects. We create only the random number streams and invoke a static method that computes the inverse cdf of the exponential for each uniform random number.

For this simple model, it is also possible to compute the entire distribution of each W_i numerically, without simulation, via the following recurrence: For $t \geq 0$, $\mathbb{P}[W_1 > t] = 0$ and

$$\mathbb{P}[W_{i+1} > t] \ = \ \mathbb{P}[W_i + S_i - A_{i+1} > t] = \int_0^\infty \int_0^\infty \mathbb{P}[W_i > t - s + a]\mathrm{d}F(a)\mathrm{d}G(s)$$

for $i \geq 1$, while for $t < 0$, $\mathbb{P}[W_i > t] = 1$ for all $i \geq 1$. In the case where the inter-arrival and service times have densities, say f and g respectively, one can write $\mathrm{d}F(a)\mathrm{d}G(s)$ as $f(a)g(s)\,\mathrm{d}a\,\mathrm{d}s$. For $i = 1, 2, \ldots$, an approximation of $\mathbb{P}[W_{i+1} > t]$ as a function of t can be obtained after computing it at several values of t by a numerical integration method, using the previously computed approximation of $\mathbb{P}[W_i > x]$ as a function of x, and using for example a spline approximation. Then, $\mathbb{E}[W_{i+1}]$ can be computed by numerical integration as well. This approach can be more efficient than simulation if high accuracy is desired, but it is also more complicated to implement. Since these types of numerical methods are outside the scope of this book, we won't pursue their study any further.

1.11.4 Reformulating as a MC integration problem

The program of Figure 1.28 actually computes a function f defined over the 198-dimensional unit hypercube. There are indeed 99 turns into the "`for`" loop and two uniform random

```
public class QueueLindley {

   RandomStream streamArr  = new MRG32k3a();
   RandomStream streamServ = new MRG32k3a();

   public double simulate (int numCust, double lambda, double mu) {
      double Wi = 0.0;
      double sumWi = 0.0;
      for (int i = 2; i <= numCust; i++) {
         Wi += ExponentialDist.inverseF (mu, streamServ.nextDouble()) -
               ExponentialDist.inverseF (lambda, streamArr.nextDouble());
         if (Wi < 0.0) Wi = 0.0;
         sumWi += Wi;
      }
      return sumWi / numCust;
   }

   public static void main (String[] args) {
      System.out.println ("Average waiting time: " +
         (new QueueLindley()).simulate (100, 1.0/10.0, 1.0/9.0));
   }
}
```

Fig. 1.28. Programming Lindley's recurrence

variates are generated at each turn. The average waiting time computed by the program is a function f of these 198 uniforms. It is an unbiased estimator of the expected average waiting time that we want to estimate.

If $U_1, U_2, \ldots$ is the sequence of all i.i.d. $U(0,1)$ random variables used to generate the inter-arrival times A_i and service times S_i, then the program computes $S_i = -9\ln(1 - U_{2i-1})$ and $A_{i+1} = -10\ln(1 - U_{2i})$, for $i = 1, 2, \ldots$. Then, $W_1, \ldots, W_{100}$ are computed using $S_1, A_2, S_2, A_3, \ldots, S_{99}, A_{100}$ together with Eq. (1.39). Therefore, $\bar{W}_{100}$ is a function f of $U_1, \ldots, U_{198}$, and $\mathbb{E}[\bar{W}_{100}]$ is the expectation of $f(U_1, \ldots, U_{198})$, which can be written as the integral of $f(u_1, \ldots, u_{198})$ with respect to $u_1, \ldots, u_{198}$ over the 198-dimensional unit hypercube.

Suppose now that we are interested in the expected average waiting time for the N_c customers who start their service before time 1000. The number of uniforms U_i that are required to compute the estimator $\bar{W}_{N_c}$ is a random variable. The dimension s in (1.16) is really infinite because there is no deterministic upper bound on N_c. A priori, we need an infinite sequence of U_i's, although only a finite (but random) number of those U_i's are actually used in the simulation.

1.11.5 Steady-state estimation

Suppose that we want to estimate

$$w = \lim_{T\to\infty} w_T \quad \text{and} \quad q = \lim_{T\to\infty} q_T,$$

the average waiting time and average queue length in the long run, i.e., over an infinite time horizon. Under our exponential assumptions, i.e., in the case of an $M/M/1$ queue, these values can be readily computed via standard analytical formulæ from queuing theory

Table 1.6. Eight runs of `QueueProc` with $T = 10000$.

Run	N_c	$\bar{W}_{N_c}$	$\bar{Q}_T$
1	1033	124.52	12.87
2	981	50.46	4.95
3	983	35.58	3.51
4	959	53.75	5.17
5	1012	78.76	7.97
6	1001	88.28	8.85
7	989	65.82	6.51
8	1027	171.81	17.64

(Section A.19), so simulation is not needed. But playing with a simple example where the exact values are know is instructive because we can see how good (or bad) our simulation results are. For an $M/M/1$ queue with arrival rate λ and service rate μ (i.e., exponential inter-arrival times with mean $1/\lambda$ and exponential service times with mean $1/\mu$), the infinite-horizon (steady-state) average waiting time and average queue length are $w = \rho/((1-\rho)\mu)$ and $q = \lambda w = \rho^2/(1-\rho)$, respectively, where $\rho = \lambda/\mu$ is the *traffic intensity*. In our example, we had $\lambda = 1/10$ and $\mu = 1/9$, so $\rho = 0.9$, $w = 81$, and $q = 8.1$.

The results of our simulations in Table 1.5 are systematically much lower than these theoretical values. What's wrong? The explanation is that since we started the system empty ($W_1 = 0$), the early customers have a smaller expected waiting time than w, so $\bar{W}_{N_c}$ turns out to be a negatively *biased* estimator of w. If we perform n simulation runs, as $n \to \infty$, $\bar{W}_{N_c}$ will converge to a *different* value than w. If $T \to \infty$, the bias goes to zero and the estimator is consistent (in fact, it converges to w with probability 1 when $T \to \infty$ for any fixed n, even for $n = 1$), but the simulation results indicate that $T = 1000$ is not enough here for the bias to be negligible.

Table 1.6 gives the results of 8 new independent runs, performed with a time horizon of $T = 10000$. The sample averages now have lower variance and are getting closer to the theoretical values, but there is still some bias. The horizon T can be increased further, and as $T \to \infty$, both the bias and the variance of $\bar{W}_{N_c}$ converge to zero, and similarly for $\bar{Q}_T$. In fact, the bias and variance turn out to decrease linearly in T in both cases. However, the simulation cost also increases linearly in T. In terms of practical considerations and common sense, one should wonder if steady-state behavior is an appropriate criterion in real life for a model like this, considering the slow decrease of the bias. This aspect will be discussed further in Chapter 2. For this particular $M/M/1$ queue example, the bias and the variance are smaller when the traffic intensity ρ is closer to 0 and larger when ρ is closer to 1 (they increase to infinity as $\rho \to 1$).

One way to reduce the bias is to discard the early customers from the statistics: Fix a constant $T_0 < T$, simulate over the horizon T, but take the average only over the time interval $(T_0, T]$. The estimators $\bar{W}_{N_c}$ and $\bar{Q}_T$ are then replaced by

$$\frac{1}{N_c(T) - N_c(T_0)} \sum_{i=N_c(T_0)+1}^{N_c(T)} W_i \quad \text{and} \quad \frac{1}{T - T_0} \int_{T_0}^{T} Q(t)\mathrm{d}t,$$

respectively. How do we choose T_0? If it is too small, too much bias remains. If it is too large, we lose too much information and, as a result, the variance is increased. So, what should we try to minimize when choosing T_0, and how do we do it in practice? This difficult question will be addressed (but not completely resolved) in Chapter 5. To reduce the variance, one can also perform a larger number of simulation runs and average over these runs, or perhaps change the estimators, e.g., using appropriate variance reduction techniques (see Chapter 6). One should not forget, however, that an estimator with slightly less variance is no improvement if it takes a lot more time to compute, or if it has much more bias. Estimators should be compared in terms of their efficiencies.

1.12 Example: One Day in a Telephone Call Center

Example 1.49 Customer relationships in modern businesses and public services are made via *contact centers*, where agents handle customer orders for products and services, provide support, register complaints, perform direct marketing campaigns, handle emergency calls, etc. These centers rely on telephone and the Internet as primary communication channels (Gans, Koole, and Mandelbaum 2003, Koole 2013).

We consider here a simplified model of a telephone contact center (or *call center*) where agents answer incoming calls. Each day, the center operates for m hours. The number of agents answering calls and the arrival rate of calls vary during the day; we shall assume that they are constant within each hour of operation but depend on the hour. Let n_j be the number of agents in the center during hour j, for $j = 0, \ldots, m-1$. For example, if the center operates from 8 AM to 9 PM, then $m = 13$ and hour j starts at $(j + 8)$ o'clock. All agents are assumed to be identical. When the number of occupied agents at the end of hour j is larger than n_{j+1}, ongoing calls are all completed but new calls are answered only when there are less than n_{j+1} agents busy. After the center closes, ongoing calls are completed and calls already in the queue are answered, but no additional incoming call is taken.

The calls arrive according to a Poisson process with piecewise constant rate, equal to $R_j = B\lambda_j$ during hour j, where the λ_j are constants and B is a random variable having the gamma distribution with parameters (α_0, α_0) (with mean 1 and variance $1/\alpha_0$; see Section 2.8.12). It represents the *busyness factor* of the day; it is more busy than usual when $B > 1$ and less busy when $B < 1$. The Poisson process assumption means that conditional on B, the number of incoming calls during any subinterval $(t_1, t_2]$ of hour j is a Poisson random variable with mean $(t_2 - t_1)R_j$ and that the arrival counts in any disjoint time intervals are independent random variables (see Section 2.13). This arrival process model is motivated and studied in Whitt (1999) and Avramidis, Deslauriers, and L'Ecuyer (2004).

Incoming calls form a FIFO queue for the agents. A call is *lost* (abandons the queue) when its waiting time exceeds its *patience time*. The patience times of calls are assumed to be i.i.d. random variables with the following distribution: with probability p the patience time is 0 (so the person hangs up unless there is an agent available immediately), and with

probability $1-p$ it is exponential with mean $1/\nu$. The service times are i.i.d. gamma random variables with parameters (α, β), with mean α/β and variance α/β^2.

We want to estimate the following quantities in the long run (i.e., over an infinite number of days): (a) w, the average waiting time per call, (b) $g(s)$, the fraction of calls whose waiting time is less than s seconds for a given threshold s, and (c) ℓ, the fraction of calls lost due to abandonment. Suppose we simulate the model for n days. For each day i, let A_i be the number of arrivals, W_i the total waiting time of all calls (including those who have abandoned), $G_i(s)$ the number of calls who waited less than s seconds (including those who abandoned before s seconds), and L_i the number of abandonments. For this model, the expected number of incoming calls in a day is $a = \mathbb{E}[A_i] = \sum_{j=0}^{m-1} \lambda_j$. We can write

$$w \overset{\text{w.p.1}}{=} \lim_{n\to\infty} \frac{W_1 + \cdots + W_n}{A_1 + \cdots + A_n} = \lim_{n\to\infty} \frac{(W_1 + \cdots + W_n)/n}{(A_1 + \cdots + A_n)/n} \overset{\text{w.p.1}}{=} \frac{\mathbb{E}[W_i]}{\mathbb{E}[A_i]} = \frac{\mathbb{E}[W_i]}{a},$$

so that W_i/a is an unbiased estimator of w. Similarly, one can see that $G_i(s)/a$ and L_i/a are unbiased estimators of $g(s)$, and ℓ, respectively. Moreover, if n is large, the n i.i.d. replicates $W_1/a, \ldots, W_n/a$ can be used to compute a confidence interval on w in the standard way, and similarly for $g(s)$, and ℓ.

Figure 1.29 gives an event-oriented simulation program for this call center model. When the `CallCenter` class is instantiated by the `main` method, the random streams, list, and statistical probes are created, and the model parameters are read from a file by the method `readData` (whose uninteresting code is not show here). The `main` program then simulates $n = 1000$ operating days and prints the value of a, as well as 90% confidence intervals on a, w, $g(s)$, and ℓ, based on their estimators $\bar{A}_n$, $\bar{W}_n/a$, $\bar{G}_n(s)/a$, and $\bar{L}_n/a$, assuming that these estimators have approximately the Student distribution. This is justified by the fact that A_i, W_i, $G_i(s)$, and L_i are themselves "averages" over several observations, so we may expect their distribution to be not far from a normal. The n observations of these quantities are collected by four statistical collectors at the end of `simulateOneDay`. These collectors are placed in the array `allTal`. The last instruction of the `main` produces a short report for each of them.

To generate the service times, we use a gamma random variate generator called `genServ`, created in the constructor after the parameters (α, β) of the service time distribution have been read from the data file. For the other random variables in the model, we simply create random streams of i.i.d. uniforms (in the preamble) and apply inversion explicitly to generate the random variates. The latter approach is more convenient, e.g., for patience times because their distribution is not standard, and for the inter-arrival times because their mean changes every period. For the gamma service time distribution, on the other hand, the parameters always remain the same and inversion is rather slow, so we decided to create a generator that uses a special (faster) method.

The method `simulateOneDay` simulates one day of operation. It initializes the simulation clock, event list, and counters, schedules the center's opening and the first arrival, and starts the simulation. When the day is over, it updates the statistical collectors. Note that there are two versions of this method; one that generates the random variate B and the other that takes its value as an input parameter. This is convenient in case one wishes to simulate the center with a fixed value of B.

```
public class CallCenter {

   static final double HOUR = 3600.0;  // Time is in seconds.

   // Data
   // Arrival rates are per hour, service and patience times are in seconds.
   double openingTime;    // Opening time of the center (in hours).
   int numPeriods;        // Number of working periods (hours) in the day.
   int[] numAgents;       // Number of agents for each period.
   double[] lambda;       // Base arrival rate lambda_j for each j.
   double alpha0;         // Parameter of gamma distribution for B.
   double p;              // Probability that patience time is 0.
   double nu;             // Parameter of exponential for patience time.
   double alpha, beta;    // Parameters of gamma service time distribution.
   double s;              // Want stats on waiting times smaller than s.

   // Variables
   double busyness;       // Current value of B.
   double arrRate = 0.0;  // Current arrival rate.
   int nAgents;           // Number of agents in current period.
   int nBusy;             // Number of agents occupied.
   int nArrivals;         // Number of arrivals today.
   int nAbandon;          // Number of abandonments during the day.
   int nGoodQoS;          // Number of waiting times less than s today.
   double nCallsExpected; // Expected number of calls per day.

   Event nextArrival = new Arrival();        // The next Arrival event.
   LinkedList<Call> waitList = new LinkedList<Call> ();

   RandomStream streamB        = new MRG32k3a(); // For B.
   RandomStream streamArr      = new MRG32k3a(); // For arrivals.
   RandomStream streamPatience = new MRG32k3a(); // For patience times.
   GammaGen genServ;   // For service times; created in readData().

   Tally[] allTal = new Tally [4];
   Tally statArrivals = allTal[0] = new Tally ("Daily arrivals");
   Tally statWaits    = allTal[1] = new Tally ("Daily waits");
   Tally statGoodQoS  = allTal[2] = new Tally ("Proportion of waits < s");
   Tally statAbandon  = allTal[3] = new Tally ("Proportion of abandons");
   Tally statWaitsDay = new Tally ("Waiting times within a day");

   public CallCenter (String fileName) throws IOException {
      readData (fileName);
      // genServ can be created only after its parameters are read.
      // The acceptance/rejection method is much faster than inversion.
      genServ = new GammaAcceptanceRejectionGen (new MRG32k3a(), alpha, beta);
   }

   // Reads data and construct arrays.
   public void readData (String fileName) throws IOException {
      ...
   }
```

Fig. 1.29. Event-view simulation of the call center.

```
// A phone call object.
class Call {
   double arrivalTime, serviceTime, patienceTime;

   public Call() {
      serviceTime = genServ.nextDouble(); // Generate service time.
      if (nBusy < nAgents) {           // Start service immediately.
         nBusy++;
         nGoodQoS++;
         statWaitsDay.add (0.0);
         new CallCompletion().schedule (serviceTime);
      } else {                         // Join the queue.
         patienceTime = generPatience();
         arrivalTime = Sim.time();
         waitList.addLast (this);
      }
   }

   public void endWait() {
      double wait = Sim.time() - arrivalTime;
      if (patienceTime < wait) { // Caller has abandoned.
         nAbandon++;
         wait = patienceTime;    // Effective waiting time.
      }
      else {
         nBusy++;
         new CallCompletion().schedule (serviceTime);
      }
      if (wait < s) nGoodQoS++;
      statWaitsDay.add (wait);
   }
}

// Event: A call arrives.
class Arrival extends Event {
   public void actions() {
      nextArrival.schedule
         (ExponentialDist.inverseF (arrRate, streamArr.nextDouble()));
      nArrivals++;
      new Call();               // Call just arrived.
   }
}

// Event: A call is completed.
class CallCompletion extends Event {
   public void actions() { nBusy--;   checkQueue(); }
}
```

Fig. 1.29. Event-view simulation of the call center (continuation).

```
// Event: A new period begins.
class NextPeriod extends Event {
   int j;     // Number of the new period.
   public NextPeriod (int period) { j = period; }
   public void actions() {
      if (j < numPeriods) {
         nAgents = numAgents[j];
         arrRate = busyness * lambda[j] / HOUR;
         if (j == 0)
            nextArrival.schedule (ExponentialDist.inverseF
                                  (arrRate, streamArr.nextDouble()));
         else {
            checkQueue();
            nextArrival.reschedule ((nextArrival.time() - Sim.time())
                                    * lambda[j-1] / lambda[j]);
         }
         new NextPeriod(j+1).schedule (1.0 * HOUR);
      }
      else
         nextArrival.cancel();  // End of the day.
   }
}

// Start answering new calls if agents are free and queue not empty.
public void checkQueue() {
   while ((waitList.size() > 0) && (nBusy < nAgents))
      ((Call)waitList.removeFirst()).endWait();
}

// Generates the patience time for a call.
public double generPatience() {
   double u = streamPatience.nextDouble();
   if (u <= p)
      return 0.0;
   else
      return ExponentialDist.inverseF (nu, (1.0-u) / (1.0-p));
}

public void simulateOneDay (double busyness) {
   Sim.init();        statWaitsDay.init();
   nArrivals = 0;     nAbandon = 0;
   nGoodQoS = 0;      nBusy = 0;
   this.busyness = busyness;

   new NextPeriod(0).schedule (openingTime * HOUR);
   Sim.start();
   // Here the simulation is running...

   statArrivals.add ((double)nArrivals);
   statWaits.add (statWaitsDay.sum() / nCallsExpected);
   statGoodQoS.add ((double)nGoodQoS / nCallsExpected);
   statAbandon.add ((double)nAbandon / nCallsExpected);
}
```

Fig. 1.29. Event-view simulation of the call center (continuation).

```
   public void simulateOneDay () {
      simulateOneDay (GammaDist.inverseF (alpha0, alpha0, 8,
                                          streamB.nextDouble()));
   }

   static public void main (String[] args) throws IOException {
      CallCenter cc = new CallCenter ("CallCenter.dat");
      for (int i=1; i <= 1000; i++)  cc.simulateOneDay();
      System.out.println ("\nNumber of calls expected per day = "
                          + cc.nCallsExpected +"\n");
      for (int i = 0; i < cc.allTal.length; i++) {
         cc.allTal[i].setConfidenceIntervalStudent();
         cc.allTal[i].setConfidenceLevel (0.90);
      }
      System.out.println (Tally.report ("CallCenter:", cc.allTal));
   }
}
```

Fig. 1.29. Event-view simulation of the call center (continuation).

An event `NextPeriod(j)` marks the beginning of each period j. The first of these events (for $j = 0$) is scheduled by `simulateOneDay`; then the following ones schedule each other successively, until the end of the day. This type of event updates the number of agents in the center and the arrival rate for the next period. If the number of agents has just increased and the queue is not empty, some calls in the queue can now be answered. The method `checkQueue` verifies this and starts service for the appropriate number of calls. The time until the next planned arrival is readjusted to take into account the change of arrival rate, as follows. The inter-arrival times are i.i.d. exponential with mean $1/R_{j-1}$ when the arrival rate is fixed at R_{j-1}. But when the arrival rate changes from R_{j-1} to R_j, the residual time until the next arrival should be modified from an exponential with mean $1/R_{j-1}$ (already generated) to an exponential with mean $1/R_j$. Multiplying the residual time by λ_{j-1}/λ_j is an easy way to achieve this (see Section 2.13 for further details on non-stationary Poisson processes). We give the specific name `nextArrival` to the next arrival event to be able to reschedule it to a different time. Note that there is a *single* arrival event which is scheduled over and over again during the entire simulation. This is more efficient than creating a new arrival event for each call, and can be done here because there is never more than one arrival event at a time in the event list. At the end of the day, simply canceling the next arrival makes sure that no more calls will arrive.

Each arrival event first schedules the next one. Then it increments the arrivals counter and creates the new call that just arrived. The call's constructor generates its service time and decides where the incoming call should go. If an agent is available, the call is answered immediately (its waiting time is zero), and an event is scheduled for the completion of the call. Otherwise, the call must join the queue; its patience time is generated by `generPatience` and memorized, together with its arrival time, for future reference.

Upon completion of a call, the number of busy agents is decremented and one must verify if a waiting call can now be answered. The method `checkQueue` verifies that and if the answer is yes, it removes the first call from the queue and activates its `endWait` method. This method first compares the call's waiting time with its patience time, to see if this call is still waiting or has been lost (by abandonment). If the call was lost, we consider its waiting time as being equal to its patience time (i.e., the time that the caller has really waited), for

the statistics. If the call is still there, the number of busy agents is incremented and an event is scheduled for the call completion.

We ran the program with the following input parameters, with the time measured in seconds: $\alpha_0 = 10$, $p = 0.1$, $\nu = 0.001$, $\alpha = 1.0$, $\beta = 0.01$ (so the service time has mean 100 and variance 10,000), and $s = 20$. The center starts empty and operates for 13 one-hour periods. The number of agents and the arrival rate for the first configuration are given in Table 1.7. With these values, the expected number of arrivals per day is $a = \mathbb{E}[A_i] = 1660$. We simulated $n = 1000$ independent days. Table 1.8 shows part of the results; it gives estimates of a, w, $g(s)$, and ℓ, together with 90% confidence intervals.

We see that the number A_i of arrivals in a day varies a lot, mainly because of the random busyness factor B_i. Its variance is about 2.6×10^5, whereas if each B_i was fixed at 1, then A_i would be a Poisson random variable with both mean and variance equal to 1660. The (overall) average waiting time is around 11.8, about 85.3% of the calls are answered within 20 seconds, and we have about 3.4% abandonment. The averages within a day vary a lot from day to day. For example the average waiting time was near 0 seconds on one day and 570.7 seconds on another day. This large variation is due mainly to the variation of B_i. Note also that the proportion estimate $G_i(s)/a$ was 1.155 (larger than 1) on one occasion, which may look strange since a proportion cannot be larger than 1, but $G_i(s)/a$ is not really an empirical proportion. Clearly, the number of arrivals on that particular day was larger than a.

Call center managers usually want to see those types of statistics for each time period as well (e.g., each hour) to better understand when longer waits occur and perhaps adjust the staffing accordingly. Adding those detailed reports in our program wuld be easy. We could also easily produce histograms of the nonzero waiting times, globally and per period.

Table 1.7. Number of agents n_j and arrival rate λ_j (per hour) for 13 one-hour periods in the call center.

j	0	1	2	3	4	5	6	7	8	9	10	11	12
n_j	4	6	8	8	8	7	8	8	6	6	4	4	4
λ_j	100	150	150	180	200	150	150	150	120	100	80	70	60

This model is certainly an oversimplification of actual call centers. It can be embellished and made more realistic by considering refined arrival processes, different types of agents, different types of calls, call assignment strategies, agent-dependent service times, redials and

Table 1.8. Results from $n = 1000$ days of simulation of the call center.

	Min	Max	average	variance	std dev	90% conf. int.
Arrivals A_i	460	4206	1639.5	2.634E5	513.2	(1612.8, 1666.2)
Waits W_i/a	0.000	570.7	11.83	1161.3	34.07	(10.06, 13.61)
GoodQoS $G_i(s)/a$	0.277	1.155	0.853	0.028	0.169	(0.844, 0.862)
Abandon L_i/a	0.000	0.844	0.034	3.7E-3	0.061	(0.031, 0.037)

returns, agents taking breaks for lunch, coffee, or going to the restroom, agents making outbound calls to reach customers (e.g., for marketing purpose or for returning calls), and so on. See Akşin, Armony, and Mehrotra (2007), Chan, Koole, and L'Ecuyer (2014), Ibrahim et al. (2016), Oreshkin, Régnard, and L'Ecuyer (2016).

One could also model the revenue generated by calls and the operating costs for running the center, and use the simulation model to compare alternative operating strategies in terms of the expected net revenue, for example.

A simulator of large, complex, call centers would normally not be written as a single program (in a single class) as in Figure 1.29, but rather as well-organized collection of classes that take care of different aspects of the model or the simulation experiment, and are used as building blocks to construct specific simulation programs. A library of this type, for call center simulation, is described by Buist and L'Ecuyer (2005) and is available at `http://www-etud.iro.umontreal.ca/~buisteri/contactcenters/`.

Call center managers in the real world often go the opposite way and use oversimplified models, simpler than our above model (Gans, Koole, and Mandelbaum 2003, Avramidis and L'Ecuyer 2005). The most popular one is the Erlang-C formula, which holds under the assumptions that there is a single call type, s identical servers, the calls arrive according to a Poisson process with fixed arrival rate, the service times are i.i.d. and exponentially distributed, there are no abandons, and the system is in steady-state. The formula gives the entire cdf of W, i.e., $\mathbb{P}[W \leq x]$ for any $x \geq 0$, where W is the waiting time of a customer selected at random. More details and other formulas are given in Section A.19. This formula is often used to determine the minimal number of agents required to satisfy certain constraints on the quality of service. It is typically applied to each time period (e.g., of 30 or 60 minutes) by assuming that the model is in steady-state during that period, perhaps with a different arrival rate for each period. These models make assumptions that are far from realistic, so they may provide very unreliable answers, and they are limited to a single call type. Simulation models can cover a much wider range of situations and can be much more accurate. □

1.13 Common Random Numbers for the Call Center

Example 1.50 For the call center example of Section 1.12, suppose now that we want to compare two configurations of the system with slightly different numbers of agents. The first configuration is the same as in Section 1.12. For the second configuration, we simply increase the number of agents by 1 in periods 5 and 6.

The program of Figure 1.30 defines an *extension* of class `CallCenter`. It simulates the two configuration for $n = 1000$ days with and without CRNs, and prints a confidence interval on the expected difference in QoS in each case, as well as on the expected QoS for the first configuration. The results are shown in Figure 1.31. Empirically, the variance of the difference is approximately 223 times smaller with CRNs than without. Compare the two confidence intervals: with IRNs, the difference between the two configurations is not significant since the interval contains 0, whereas the interval obtained with CRNs is much more useful.

```
public class CallCenterCRN extends CallCenter {
   Tally statDiffIndep = new Tally ("stats on difference with IRNs");
   Tally statDiffCRN   = new Tally ("stats on difference with CRNs");
   int[] numAgents1, numAgents2;

   public CallCenterCRN (String fileName) throws IOException {
      super (fileName);
      numAgents1 = new int[numPeriods];
      numAgents2 = new int[numPeriods];
      for (int j=0; j < numPeriods; j++)
         numAgents1[j] = numAgents2[j] = numAgents[j];
   }
   // Set the number of agents in each period j to the values in num.
   public void setNumAgents (int[] num) {
      for (int j=0; j < numPeriods; j++) numAgents[j] = num[j];
   }
   public void simulateDiffCRN (int n) {
      double value1, value2;
      statDiffIndep.init();  statDiffCRN.init();
      for (int i=0; i<n; i++) {
         setNumAgents (numAgents1);
         streamB.resetNextSubstream();
         streamArr.resetNextSubstream();
         streamPatience.resetNextSubstream();
         (genServ.getStream()).resetNextSubstream();
         simulateOneDay(); // Simulate first config.
         value1 = (double)nGoodQoS / nCallsExpected;
         setNumAgents (numAgents2);
         streamB.resetStartSubstream();
         streamArr.resetStartSubstream();
         streamPatience.resetStartSubstream();
         (genServ.getStream()).resetStartSubstream();
         simulateOneDay(); // Simulate second config. with CRNs
         value2 = (double)nGoodQoS / nCallsExpected;
         statDiffCRN.add (value2 - value1);
         simulateOneDay(); // Simulate second config. with IRNs
         value2 = (double)nGoodQoS / nCallsExpected;
         statDiffIndep.add (value2 - value1);
      }
   }
   static public void main (String[] args) throws IOException {
      int n = 1000;   // Number of replications.
      CallCenterCRN cc = new CallCenterCRN ("CallCenter.dat");
      cc.numAgents2[5]++;   cc.numAgents2[6]++;
      cc.simulateDiffCRN (n);
      System.out.println (
         cc.statDiffIndep.report (0.9, 5) + cc.statDiffCRN.report (0.9, 5));
      double varianceIndep = cc.statDiffIndep.variance();
      double varianceCRN   = cc.statDiffCRN.variance();
      System.out.println ("Variance ratio:  " +
         PrintfFormat.format (10, 1, 3, varianceIndep / varianceCRN));
   }
}
```

Fig. 1.30. Comparing two configurations of the call center: program

```
REPORT on Tally stat. collector ==> stats on difference with IRNs
    num. obs.      min         max        average     variance    standard dev.
      1000      -0.76325     0.77530      9.873E-3     0.05485       0.23420
  90.0% conf. interval for the mean (Student approx.): (-2.321E-3, 0.02207 )

REPORT on Tally stat. collector ==> stats on difference with CRNs
    num. obs.      min         max        average     variance    standard dev.
      1000      -0.01325     0.10060      9.920E-3     2.452E-4       0.01566
  90.0% conf. interval for the mean (Student approx.): (0.00910, 0.01074 )

Variance ratio:       223.7
```

Fig. 1.31. Comparing two configurations of the call center: results.

The implementation exploits the concept of random number *streams* and *substreams*, described at the end of Section 1.3, and available in SSJ and other modern simulation software. Our simulation program uses four different random number streams, one for each type of random variable in the model. This separation is to make sure that each random number is used for the same purpose in both configurations. This guarantees that no random number can be used to simulate a service time in one configuration and a patience time in the other configuration, for example.

For each simulation run (one day), the four random number streams are reset to a new substream (`ResetNextSubstream`) before simulating one day with the first configuration. After this simulation, the four streams are reset to the beginning of their current substreams (`ResetStartSubstream`), to make sure that the simulation with the second configuration uses exactly the same sequence of uniform random numbers as the first one. Since the parameters of the probability distributions are the same for both configurations, this implies that the value of B for the day and the sequences of customer's arrival times, patience times, and service times will be exactly the same for both configurations. The difference in QoS will be due *only* to the (slightly) different number of agents in the two configurations. In fact, all customers have exactly the same service times in both configurations, including the customers who abandon in one configuration and not in the other one, because the service time is generated when the customer arrives. The patience time, on the other hand, is generated only for the customers who wait, which are not necessarily the same customers in the two configurations.

After each pair of runs with CRNs, the method `simulateDiffCRN` simulates another day with the second configuration, *without* resetting the streams, so that this second run will use random numbers "*independent*" from those of the first run. This is done for comparing the variance with and without CRNs. The variance ratio printed by the program is the variance with independent random numbers across configurations divided by the variance with common random numbers. It corresponds roughly to the *efficiency improvement factor* obtained by using CRNs instead of independent random numbers. Here, to obtain an estimator of a given accuracy, the simulation with CRNs requires a computing time approximately 223 times smaller than without CRNs. We also tried generating the patience time for all customers on their arrival, so each customer had the same patience in both configurations, and the (empirical) variance ratio improved slightly, from 223 to 256. □

1.14 Continuous Simulation

♣ To do: Short introduction to "system dynamics" and to simulation models based on deterministic and stochastic differential equations.

1.15 Simulation-based optimization

The end goal of a majority of simulation projects is to *optimize* decision parameters or operating rules for the system of interest. For stochastic systems, the objective function that we want to optimize is often a mathematical expectation that needs to be estimated by simulation, and there may be constraints on quantities that also need to be estimated by simulation. Here we start with simple examples, followed by an overview of important simulation-based optimization methods, then more examples. Chapter 7 gives a more elaborate coverage.

1.15.1 Introductory examples

Example 1.51 In the SAN of Example 1.4, one may have to pay a cost $c(T)$ that increases as a function of the project duration T, and it may be possible to improve the distribution of the durations of activities at some cost, for example by allocating more resources to these activities. As a simple illustration, suppose that the mean duration θ_j of activity j can be changed but a cost $c_j(\theta_j)$ must be paid, where each c_j is a decreasing function. The total cost is $c(T) + \sum_{j=1}^{13} c_j(\theta_j)$. The vector of decision variables is $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{13}) \in \boldsymbol{\Theta}$ where $\boldsymbol{\Theta} \subseteq [0, \infty)^{13}$ is a set of feasible (allowed) decisions. Since $c(T)$ is random, we cannot really minimize the total cost as given, but we may want to minimize its expectation:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left(\alpha(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{\theta}}[c(T)] + \sum_{j=1}^{13} c_j(\theta_j) \right),$$

where $\mathbb{E}_{\boldsymbol{\theta}}$ is the expectation when the decision variables are fixed to $\boldsymbol{\theta}$.

The set $\boldsymbol{\Theta}$ can be continuous or discrete. If it is a small finite set, say of no more than about 20 possibilities, then one can simulate each one to estimate its cost and pick the best one. If it is discrete with a very large number of possibilities, then the optimization problem can be very hard to solve so one would have to settle with a heuristic. For the case where $\boldsymbol{\Theta}$ is continuous and $\alpha(\boldsymbol{\theta})$ is a unimodal and smooth function, there are effective stochastic optimization algorithms that provably converge to the optimal solution when the number of simulation runs increases to infinity. A prominent one is the iterative *stochastic approximation* (*gradient descent*) algorithm, which at each step estimates the gradient of $\alpha(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ at the current value of $\boldsymbol{\theta}$, and moves $\boldsymbol{\theta}$ by a small amount in the opposite direction of this gradient estimate. Recall that efficient gradient estimation for this particular example was discussed in Example 1.47.

Another possible problem formulation could be that we want to minimize only $\sum_{j=1}^{13} c_j(\theta_j)$ but under the constraint that $\mathbb{P}_{\boldsymbol{\theta}}[T > x] \leq p$ for given values of x and p. For example, one

may impose that $\mathbb{P}_{\boldsymbol{\theta}}[T > 120] \leq 0.02$. This is a difficult problem, because for any choice of $\boldsymbol{\theta}$, $\mathbb{P}_{\boldsymbol{\theta}}[T > 120]$ cannot be computed exactly; it can only be estimated by simulation, with some error. One appropriate solution approach for this case could be *sample average approximation*, to be discussed later. □

Example 1.52 Similar optimization problems occur with the reliability models of Example 1.5. Often, one can increase the reliability of certain components j at a cost, say by paying $c_j(r_j)$ where c_j is an increasing function. We may have a large cost to pay when the system fails and we want to minimize the total expected cost, or we may have a constraint on the probability $1 - r$ of system failure, for example $1 - r \leq 10^{-4}$, and want to minimize $\sum_{j=1}^{m} c_j(r_j)$ under that constraint. In either case, the probability $1-r$ of system failure must be estimated by simulation for each choice of the component reliabilities. □

Example 1.53 In the single-server queue example of Section 1.11, the server may be a machine whose speed can be increased at a cost. Another part of the cost could be a function of the average waiting time per customer. L'Ecuyer and Glynn (1994) consider the simple optimization problem:

$$\min_{\theta \in \Theta} \Big(\alpha(\theta) \stackrel{\text{def}}{=} c(\theta) + w(\theta) \Big) \tag{1.40}$$

where $\Theta = [\theta_{\min}, \theta_{\max}]$ is a range of admissible values for the server speed θ, $c(\theta)$ is the cost of that server speed, and $w(\theta)$ is the average waiting time of customers for this speed. They use this example to show how convergence to the optimum can be proved for the stochastic approximation algorithm, under certain conditions. □

Example 1.54 In machine learning, the parameters of neural networks are *learned* using stochastic optimization methods that typically combine Monte Carlo gradient estimation with gradient-descent algorithms. These methods are variants of the stochastic approximation algorithm. A neural network can in fact be viewed as a complicated function with a large number of real-valued parameters. *Training* the network for a specific application means trying to optimize the values of these parameters for the network to have the best possible success rate in performing its task. This is often achieved by simulation-based iterative methods where, at each step, the gradient of the performance measure of the network with respect to its parameters is estimated, and the parameters are changed slightly in the opposite direction of the gradient. □

Example 1.55 In the call center example of Section 1.12, we may want to optimize the number of agents working in the center during each hour. The objective function can be the expected operating cost per day, under constraints on the mathematical expectations w, $g(s)$, and ℓ. An example of such a constraint is $g(20) \geq 0.80$, i.e., that at least 80% of the calls be answered within 20 seconds in the long run. Constraints on the admissible staffing decisions also come from the fact that the working hours of the individual agents must satisfy certain rules often determined by union agreements. For example, it may be that each agent must have an eight-hour shift, including a one-hour lunch break after three, four, or five hours of work. A *scheduling solution* for the center determines how many agents are available on

a given day, when their shifts start, and when they have their lunch break. Most of the cost of a scheduling solution is typically the sum of costs of all agents, where the cost of an agent may depend on his work schedule and perhaps on other factors. Optimal scheduling is an optimization problem with integer decision variables and stochastic constraints, in the sense that the quantities such as w, $g(s)$, and ℓ that appear in the constraints are unknown. These quantities can be estimated by simulation for any given schedule, together with the value of the objective function, but they depend on the schedule. Moreover, the simulation-based estimators will not offer a 100% guarantee that the constraints are satisfied, or that the schedule with the smallest average cost in the simulation really has the smallest expected operating cost, because these estimators are noisy. Typically, we will only ask for the constraint to be satisfied with a certain probability, in the same spirit as when we compute confidence intervals. For the degree of optimality, our target may be, for example, that with probability at least 95%, the expected cost for the selected schedule does not exceed the expected cost of the (unknown) optimal schedule by more than 2%. Large scheduling problems are already hard to solve in a deterministic framework. Their stochastic versions are even harder. But this is what real-life decision making is about. □

1.15.2 A general formulation of the optimization problem

Simulation-based optimization has many different facets (Banks 1998, Fu 2002, Shapiro 2003, Andradóttir 2006, Shapiro, Dentcheva, and Ruszczyński 2014, Fu and Henderson 2017). The following general problem formulation covers many practical situations:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left(\alpha(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{\theta}}[h(\boldsymbol{\theta},\boldsymbol{Y})]\right) \qquad \text{subject to} \quad \boldsymbol{\beta}(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{g}(\boldsymbol{\theta},\boldsymbol{Y})] \geq \mathbf{0}, \tag{1.41}$$

where $\boldsymbol{\theta}$ is a vector of *decision variables* that must belong to the set $\boldsymbol{\Theta}$, $\boldsymbol{Y}$ is a random vector whose distribution may depend on $\boldsymbol{\theta}$ (hence the notation $\mathbb{E}_{\boldsymbol{\theta}}$), h is a real-valued cost function, and $\boldsymbol{g}$ is a vector of functions used to define *stochastic constraints* in terms of their unknown mathematical expectations $\boldsymbol{\beta}(\boldsymbol{\theta})$. Constraints that are purely deterministic may be included as well (the components of $\boldsymbol{\beta}(\boldsymbol{\theta})$ for these constraints will be known) or can be incorporated into the definition of $\boldsymbol{\Theta}$. Sometimes, there are no stochastic constraints. Sometimes, the objective function $\alpha(\boldsymbol{\theta})$ is deterministic and only the constraints are stochastic. A value of $\boldsymbol{\theta}$ is called a *configuration* of the system (or a solution). The ultimate goal would be to find: (a) an *optimal solution*, which a configuration $\boldsymbol{\theta}^*$ that minimizes the cost function α while satisfying the constraints in (1.41) and (b) the *optimal value* of the objective function, $v^* = \alpha(\boldsymbol{\theta}^*)$. The optimal solution (or configuration) is not necessarily unique. In case there is a *set* $\mathcal{S}^*$ of optimal configurations $\boldsymbol{\theta}^*$, it suffices to find one of them.

An optimization method will return an *estimator* $\hat{\boldsymbol{\theta}}$ of an optimal configuration $\boldsymbol{\theta}^*$, often together with estimators of $\alpha(\hat{\boldsymbol{\theta}})$ and $\beta(\hat{\boldsymbol{\theta}})$. The *error* is sometimes defined as the distance between $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}^*$ (or between $\hat{\boldsymbol{\theta}}$ and $\mathcal{S}^*$ if $\boldsymbol{\theta}^*$ is not unique), but a more appropriate measure is the *optimality gap* $\alpha(\hat{\boldsymbol{\theta}}) - v^*$, which is the expected additional cost for using $\hat{\boldsymbol{\theta}}$ instead of a truly optimal configuration. When there are stochastic constraints, the expectations $\boldsymbol{\beta}(\boldsymbol{\theta})$ are also estimated with some error, so the retained solution may not satisfy all the constraints even if the estimated constraints are satisfied. One must be aware of this possibility and perhaps impose constraints on the probability that this happens.

In a deterministic setting, $\alpha(\boldsymbol{\theta})$ and the constraints can be evaluated exactly for any $\boldsymbol{\theta}$, whereas in the stochastic setting considered here, only noisy observations are available. This makes the optimization more difficult.

Robustness to the modeling error is an important issue that brings additional difficulties in stochastic optimization. The input distributions are usually uncertain and one wishes a solution estimate $\hat{\boldsymbol{\theta}}$ that is *distributionally robust* in the sense that it would perform well uniformly for all distributions that are not too far from the ones selected in the model. There are many ways of formalizing and solving this problem. The usual approach is to define a convex set $\mathcal{D}$ of probability distributions for the model instead of just a single selection, then replace $\alpha(\boldsymbol{\theta})$ by its worst-case (maximum) value over this set $\mathcal{D}$ of distributions and $\boldsymbol{\beta}(\boldsymbol{\theta})$ by its minimum value over the set $\mathcal{D}$. Taking a larger $\mathcal{D}$ improves the robustness of the solution, but degrades the optimal value $\alpha(\boldsymbol{\theta}^*)$ because we take the worst-case over a larger set of distributions, so a compromise must be made when selecting $\mathcal{D}$. One important question is how to measure the distance between distributions when defining $\mathcal{D}$; see Gao and Kleywegt (2022).

Another type of robustness is *robustness to bad realizations*. That is, even if there is no modeling error, considering only the expected cost in the objective function, and also only expectations in the constraints, does not prevent some very bad realizations of $(\boldsymbol{\theta}, \boldsymbol{Y})$ to occur occasionally. Those bad realizations can be very far from satisfying the constraints, for example. One simple way to address this problem is to *add large penalties* to the cost $h(\boldsymbol{\theta}, \boldsymbol{Y})$ for the bad realizations that we want to avoid. A more drastic solution would be to add constraints to ensure that these bad realizations cannot occur, but this may be too demanding and lead to solutions that are not very good on average. *Chance constraints* offer a flexible compromise. They impose a bound on the probability that a bad realization occurs. A chance constraint has the form

$$\mathbb{P}_{\boldsymbol{\theta}}[\boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{Y}) \geq \mathbf{0}] \geq p$$

for some selected probability p, where $\boldsymbol{g}$ can be a vector. The function $\boldsymbol{g}$ would be selected in a way that the inequality $\boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{Y}) \geq \mathbf{0}$ would not hold for the bad realizations that we want to avoid. There may be several of these constraints for the same optimization problem. The probabilities in these chance constraints will have to be estimated by simulation. In fact, these probabilities are also expectations, so these constraints still fit our original problem formulation of (1.41). These probabilities are often very close to 1, in which case they can be difficult to estimate accurately. For example, with a estimate of 0.999 when the true value is 0.997, we would think that the bad event has no more than 0.1% chance or occurring while in reality it has 0.3% chance, which is three times larger. Such situations may benefit from techniques for rare-event simulation. The concept of chance constraint is very closely related to the *scenario approach* in stochastic optimization, in which a set of n scenarios are obtained either directly from real data or by simulation, and a given set of constraints must be satisfied for at least a given proportion of the scenarios (Calafiore and Campi 2005).

1.15.3 Ranking and selection

Suppose that there are no constraints, so the problem simplifies to

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left(\alpha(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathbb{E}_{\boldsymbol{\theta}}[h(\boldsymbol{\theta}, \boldsymbol{Y})]\right), \tag{1.42}$$

and that $\boldsymbol{\Theta}$ is a finite set with small cardinality (e.g., no more than 50 configurations). A straightforward approach in this case would be to make a very large number of simulation runs *for each configuration* $\boldsymbol{\theta}$, i.e., estimate $\alpha(\boldsymbol{\theta})$ for each $\boldsymbol{\theta}$ by the average of n i.i.d. copies of $h(\boldsymbol{\theta}, \boldsymbol{Y})$ for n large enough to make the error negligible, and then select the value of $\boldsymbol{\theta}$ with the smallest average. But efficiency can be improved substantially by eliminating at an early stage the configurations that are quickly found to be non-competitive. *Ranking and selection methods* (Bechhofer, Santner, and Goldsman 1995, Goldsman and Nelson 1998, Kim and Nelson 2006) are statistical tools for comparing a fixed and finite set of alternatives in a stochastic context. Given a value δ of the acceptable optimality gap and a confidence level $1 - \alpha$, the aim of these methods is to determine adaptively how many simulation runs to perform with each configuration, and to return a configuration $\hat{\boldsymbol{\theta}}$ for which

$$\mathbb{P}[\alpha(\hat{\boldsymbol{\theta}}) - \alpha(\boldsymbol{\theta}^*) \leq \delta] \geq 1 - \alpha. \tag{1.43}$$

Procedures are also offered to screen out rapidly the bad configurations (after a small number of runs) when the number of alternatives is large (but still finite).

Returning a (random) $\hat{\boldsymbol{\theta}}$ that satisfies (1.43) is frequently much easier than estimating $\alpha(\boldsymbol{\theta})$ with precision δ for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, and sometimes even easier than estimating $\alpha(\boldsymbol{\theta})$ at a single $\boldsymbol{\theta}$ (Ho et al. 2000, Fu 2002). This goes with the intuition that, for example, if two persons are in front of us and their sizes differ by at least 2 cm, finding which one is tallest is generally easier than telling the size of each within 2 cm of precision.

1.15.4 Infinite or very large feasible region

When $\boldsymbol{\Theta}$ is infinite or too large, it becomes impossible to test all alternatives. Better strategies for searching the space of feasible decisions (or configurations) are needed. The appropriate optimization methods depend on whether $\boldsymbol{\theta}$ is discrete or continuous, and on whether the function α is nicely behaved (e.g., is differentiable and convex or unimodal) or not (e.g., is discontinuous or has millions of local optima). Most methods are adaptations of methods used in the deterministic context. However, their convergence is typically much slower in the stochastic case, because the error is dominated by the stochastic noise.

Stochastic approximation. Suppose again that there are no stochastic constraint, as in (1.42). For the case where $\boldsymbol{\theta}$ is continuous and α is differentiable, the best known and most widely-studied method is the *stochastic approximation (SA)* algorithm (Robbins and Monro 1951, Kushner and Yin 2003), which actually searches for a solution (a root) to the equation $\nabla\alpha(\boldsymbol{\theta}) = \mathbf{0}$, where $\nabla\alpha$ denotes the gradient of α (the vector of partial derivatives). The SA algorithm moves iteratively with small steps in the space $\boldsymbol{\Theta}$ as follows. We start from some initial feasible solution $\boldsymbol{\theta}_0 \in \boldsymbol{\Theta}$. Then at each step n, we compute a simulation-based estimator $\boldsymbol{D}_n$ of $\nabla\alpha(\boldsymbol{\theta}_n)$, the gradient of α at the current solution $\boldsymbol{\theta}_n$, and we move to the next solution which is

$$\boldsymbol{\theta}_{n+1} = \Pi_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_n - a_n \boldsymbol{D}_n),$$

where $\Pi_{\boldsymbol{\Theta}}$ is an operator that projects the vector $\boldsymbol{\theta}$ back into the feasible region $\boldsymbol{\Theta}$ when it goes outside (otherwise it does nothing), and $\{a_n, n \geq 0\}$ is a predefined deterministic sequence of positive real numbers such that $\sum_{n=1}^{\infty} a_n = \infty$ and $\sum_{n=1}^{\infty} a_n^2 < \infty$. Informally, the first of these two conditions is to make sure that the total displacement $\sum_{n=1}^{\infty} a_n \boldsymbol{D}_n$ can be unbounded, so the optimal solution can be reached even if we start far away from it, and the second condition (the finite sum of squares) is for a_n to decrease fast enough for the algorithm to converge. A popular choice that satisfies these conditions is $a_n = a/n$ for some constant $a > 0$. Under certain conditions on α, $\boldsymbol{\Theta}$, and $\boldsymbol{D}_n$, there is a unique optimal configuration $\boldsymbol{\theta}^*$ and one can prove that $\boldsymbol{\theta}_n \to \boldsymbol{\theta}^*$ with probability 1 when $n \to \infty$. This holds for example if α is strictly convex in $\boldsymbol{\Theta}$, the optimum is in the interior of $\boldsymbol{\Theta}$ (not on the boundary), and $\boldsymbol{D}_n$ is an unbiased gradient estimator whose covariance matrix is bounded uniformly over $\boldsymbol{\Theta}$. There are also weaker sets of sufficient conditions for convergence (e.g., "strictly convex" can be relaxed to something weaker).

SA has many variants, depending on how we estimate the gradient, how we select a_n, how we project, and which final solution we retain. Some *adaptive* variants replace a_n by a random variable or a random matrix; they try to improve the convergence by "learning" the shape of the function near the optimum $\boldsymbol{\theta}^*$ and optimizing a_n accordingly. If α is strictly convex and the gradient estimator is unbiased, the optimal a_n (to optimize the asymptotic convergence rate when $n \to \infty$) turns out to be $a_n = \boldsymbol{H}/n$ where $\boldsymbol{H}$ (a matrix) is the inverse of the Hessian matrix of α at $\boldsymbol{\theta}^*$ (Chung 1954). The adaptive methods try to learn the matrix $\boldsymbol{H}$ along the way. When using this optimal sequence, $\sqrt{n}(\boldsymbol{\theta}_n - \boldsymbol{\theta}^*)$ converges to a $\mathsf{N}(\mathbf{0}, \boldsymbol{\Sigma})$ distribution for some covariance matrix $\boldsymbol{\Sigma}$, so we have a CLT (Fabian 1968). L'Ecuyer and Yin (1998) provide convergence rates and CLTs for more general situations in which the gradient estimators $\boldsymbol{D}_n$ may have bias and variance that depend on n (which is frequent), and the computational effort also depends on n. Their resulting CLTs are then expressed in terms of the total computational effort T rather than in terms in n, and the rates are often slower than $\mathcal{O}(T^{-1/2})$.

One important and effective variant of SA estimates $\boldsymbol{\theta}^*$ by the average

$$\tilde{\boldsymbol{\theta}}_{n_0,n} = \frac{1}{n - n_0} \sum_{n=n_0+1}^{n} \boldsymbol{\theta}_n$$

for $n > n_0$, for a fixed $n_0 \geq 0$, instead of by $\boldsymbol{\theta}_n$ only (Ruppert 1988, Polyak and Juditsky 1992, Kushner and Yin 2003). This *averaging approach* permits one to decrease a_n more slowly and is more robust than the basic SA algorithm with respect to a mis-specification of $\boldsymbol{H}$. One can take in particular $a_n = an^{-\gamma}$ for $1/2 < \gamma < 1$. The slower convergence of a_n means that $\boldsymbol{\theta}_n$ keeps moving a lot more than with $a_n = a/n$, so it can get close to the optimum faster, and the resulting increased noise on $\boldsymbol{\theta}_n$ is alleviated by taking the average.

A key ingredient for making the SA algorithm efficient is the availability of a high-quality *gradient estimator*. This has motivated a great deal of research on gradient estimation in recent decades (L'Ecuyer 1990b, L'Ecuyer 1991, Glasserman 1991, Rubinstein and Shapiro 1993, Fu and Hu 1997, Fu 2006). These gradient estimators are useful not only for optimization, but also for estimating the *sensitivity* of the performance measure $\alpha(\boldsymbol{\theta})$ with respect to parameters of the system estimated from data, for example. There are many applications for which all available gradient estimators are biased (e.g., when they must be based on finite

differences) but SA can still converge to the optimum if the bias converges to 0 when $n \to \infty$ (L'Ecuyer 1992, L'Ecuyer and Yin 1998).

Example 1.56 For Example 1.53, L'Ecuyer and Glynn (1994) gave sufficient conditions under which they proved that the SA algorithm converges to the optimum, for a variety of gradient estimation methods, including finite differences, IPA, and LR. L'Ecuyer, Giroux, and Glynn (1994) reported extensive numerical experiments with several derivative estimation methods and choices of parameters in the algorithm, for the same example.

♣ Add numerical illustrations here. □

One limitation of SA is that it can hardly handle stochastic constraints, especially when the optimum lies on one of these constraints. It can also have problems when the optimum is on (or near) the boundary of the set $\boldsymbol{\Theta}$. The method was originally designed for optimization problems with smooth and unimodal objective functions and without constraints. The following method is usually more appropriate when there are stochastic constraints.

Sample-average approximation. The *sample-average approximation (SAA)* approach, sometimes called *sample-path optimization*, provides a class of techniques which, instead of considering only local information at each step as does SA, estimate the objective function α and the constraints $\boldsymbol{\beta}$ in the entire region $\boldsymbol{\Theta}$ all at once (conceptually) by sample averages $\hat{\alpha}_n$ and $\hat{\boldsymbol{\beta}}_n$ which are functions of $\boldsymbol{\theta}$, typically obtained from the outputs of n simulation runs, and then optimize this SAA problem by an appropriate deterministic (linear or nonlinear) optimization algorithm. The SAA problem has the form:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left(\hat{\alpha}_n(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} h(\boldsymbol{\theta}, \boldsymbol{Y}_i)\right) \quad \text{subject to} \quad \hat{\boldsymbol{\beta}}_n(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{g}(\boldsymbol{\theta}, \boldsymbol{Y}_i) \geq \boldsymbol{0}. \tag{1.44}$$

This formulation assumes that $\boldsymbol{Y}$ does not depend on $\boldsymbol{\theta}$. Let $\mathcal{S}_n^*$ denote the set of optimal configurations $\boldsymbol{\theta}_n^*$ for this SAA problem, and v_n^* the corresponding optimal value. The SAA problem cannot always be solved exactly in practice. Sometimes, it is much too large, and we have to settle with a suboptimal solution $\hat{\boldsymbol{\theta}}_n^*$ with value $\hat{v}_n^* < v_n^*$. So there are two sources of error: (1) we are solving the SAA instead of the exact problem and (2) we are sometimes solving the SAA only approximately. The second source of error depends very much on the type of problem and on the solution approach, and we will not discuss it further for now. A large body of theory has been developed for the first source of error, i.e., to study the convergence of $\mathcal{S}_n^*$ to $\mathcal{S}^*$ and of v_n^* to v^*. Several sets of sufficient convergence conditions which depend on the specific form of the problem, can be found in Rubinstein and Shapiro (1993), L'Ecuyer (1993), Robinson (1996), Ruszczyński and Shapiro (2003), Shapiro, Dentcheva, and Ruszczyński (2014), Kim, Pasupathy, and Henderson (2015), for example. There are conditions for almost sure convergence of v_n^* to the optimal value, for the convergence of the distance between $\mathcal{S}_n^*$ and $\mathcal{S}^*$ to zero, results on convergence rates, and central limit theorems. As a simple illustration, if there are no constraints, α is continuous and bounded over $\boldsymbol{\Theta}$, there is a single optimal solution $\boldsymbol{\theta}^*$, and $\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} |\hat{\alpha}_n(\boldsymbol{\theta}) - \alpha(\boldsymbol{\theta})| \stackrel{\text{w.p.1}}{\to} 0$ when $n \to \infty$, then $v_n^* \stackrel{\text{w.p.1}}{\to} v^*$ and $\boldsymbol{\theta}_n^* \stackrel{\text{w.p.1}}{\to} \boldsymbol{\theta}^*$. When $\boldsymbol{\Theta}$ is a finite set, one can also show that with probability 1,

there is a random but finite N_0 such that $\mathcal{S}_n^* = \mathcal{S}^*$ for all $n \geq N_0$. That is, for $n \geq N_0$, any optimal solution of the SAA is an optimal solution of the true problem.

An important question is how we can estimate the functions α and $\boldsymbol{\beta}$ in the entire region $\boldsymbol{\Theta}$ all at once and obtain estimators that are smooth enough functions so that the SAA problem is sufficient well-behaved to be solved effectively by a deterministic optimization algorithm. In real applications, we do not compute these function estimates explicitly for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, but only at the values of $\boldsymbol{\theta}$ required by the selected deterministic optimization algorithm. A key ingredient for the function estimators to be well-behaved is the proper use of common random numbers to simulate the system at different values of $\boldsymbol{\theta}$. That is, we want the SAA function estimates to look like the upper panel of Figure 1.22 and not like the lower panel. Picking the best value of (s, S) from this figure (upper panel) is an example of applying SAA. Most of the SAA theory was developed in a setting where the distribution of $\boldsymbol{Y}$ does not depend on $\boldsymbol{\theta}$ and the SAA estimator is defined by using the same $\boldsymbol{Y}$ for all $\boldsymbol{\theta}$'s, for each of the n realizations. If $\boldsymbol{Y}$ is interpreted as the sequence of underlying uniform random numbers, this corresponds to using common random numbers.

Stochastic constraints, when they are present, are often approximated by sets of linear constraints, in which case the SAA becomes an optimization problem with linear constraints. If the objective function is also linear, then the SAA is a linear programming problem (which can be very large). One special case in which everything is already linear in the first place is a *two-stage linear stochastic programming* problem (Birge and Louveaux 2011), with general form

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} \Big(\alpha(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{c}^{\mathsf{t}} \boldsymbol{x} + \mathbb{E}[Q(\boldsymbol{x}, \omega)] \Big) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \geq \boldsymbol{0}, \tag{1.45}$$

where $\boldsymbol{x}$ is a vector of decision variables that must be fixed in the first stage, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is a set of linear constraints, $Q(\boldsymbol{x}, \omega)$ is the optimal value of the second-stage linear program

$$Q(\boldsymbol{x}, \omega) = \min_{\boldsymbol{y} \in \mathbb{R}^s} \boldsymbol{q}(\omega)^{\mathsf{t}} \boldsymbol{y} \quad \text{subject to} \quad \boldsymbol{B}(\omega)\boldsymbol{x} + \boldsymbol{C}(\omega)\boldsymbol{y} = \boldsymbol{d}(\omega), \quad \boldsymbol{y} \geq \boldsymbol{0}, \tag{1.46}$$

which contains a second set of linear constraints, and ω represents some uncertainty that is revealed only in the second stage. That is, all the quantities that depend on ω can be observed only after $\boldsymbol{x}$ has been selected, but before selecting the values of the second-stage decision variables $\boldsymbol{y}$. Note that $\boldsymbol{x}$ and ω here correspond to $\boldsymbol{\theta}$ and $\boldsymbol{Y}$ in our general formulation (1.41).

A version of SAA for the case where $\boldsymbol{\Theta}$ is discrete was proposed and studied by Kleywegt, Shapiro, and Homem-de Mello (2002). SAA with chance constraints is studied in Ahmed and Shapiro (2008). The *response surface methodology* (Kleijnen 1998, Myers and Montgomery 2002) and *metamodel-based optimization* (Barton and Meckesheimer 2006) are also closely related to SAA.

The optimization problem (1.41) and its corresponding SAA can be very difficult to solve when α is not smooth or has many local optima. Several types of *random search* methods have been designed for these situations (Yakowitz, L'Ecuyer, and Vázquez-Abad 2000, Glover, Kelly, and Laguna 1999, de Mello 2003, Andradóttir 2006, Ólafsson 2006). Some of them come with convergence proofs. Others are metaheuristics (based, e.g., on neighborhood search ideas such as tabu search, or on evolutionary algorithms) adapted from the world of deterministic combinatorial optimization. Often, they offer no convergence proof or error estimate of the form (1.43), but seem to provide "acceptable" solutions in reasonable time for many difficult problems. Several methods currently available in commercial simulation software belong to

this last category. These methods tend to be easier to implement in general form than gradient-based or SAA methods, whose implementation depends on the problem structure.

Example 1.57 Here is an example of a large stochastic optimization problem typically encountered in practice, and for which reliable solutions can only be found by using simulation. The problem and the solution approach, which uses SAA, are taken from Ceşik and L'Ecuyer (2008). Consider a telephone call center receiving K types of calls. There are I types of agents (called *skill groups*). Each agent type can handle a specific subset of the call types. The day is partitioned into P time periods. There are Q admissible types of work schedules (also called *shifts*), each one being defined by the subset of the time periods that are covered by that shift, i.e., during which an agent having that shift will work.

The *cost vector* is $\boldsymbol{c} = (c_{1,1}, \ldots, c_{1,Q}, \ldots, c_{I,1}, \ldots, c_{I,Q})^{\mathsf{t}}$, where $c_{i,q}$ is the cost of an agent of type i having shift q. The vector of *decision variables* is $\boldsymbol{x} = (x_{1,1}, \ldots, x_{1,Q}, \ldots, x_{I,1}, \ldots, x_{I,Q})^{\mathsf{t}}$, where $x_{i,q}$ is the number of agents of type i having shift q. Our problem formulation uses the vector of *auxiliary variables* $\boldsymbol{y} = (y_{1,1}, \ldots, y_{1,P}, \ldots, y_{I,1}, \ldots, y_{I,P})^{\mathsf{t}}$ where $y_{i,p}$ is the number of agents of type i in period p. This *staffing vector* $\boldsymbol{y}$ satisfies $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ where $\boldsymbol{A}$ is a block diagonal matrix with I blocks and the element (p, q) of each block is 1 if shift q covers period p, and 0 otherwise. The *service level* for call type k and period p is defined by

$$g_{k,p}(\boldsymbol{y}) = \frac{\mathbb{E}[G_{k,p}(s_{k,p})]}{\mathbb{E}[A_{k,p}]}$$

for some constant $s_{k,p}$, where $A_{k,p}$ is the number of calls of type k arriving in period p and $G_{k,p}(s_{k,p})$ is the number of those that are answered within $s_{k,p}$ seconds. The aggregate service level over call type k is the expected total number of calls of type k answered within some time limit s_k over the day, divided by the expected total number of calls of type k received over the day. We define similar aggregations for each period p (across the call types) and overall (for all call types and all periods). We denote by $g_p(\boldsymbol{y})$, $g_k(\boldsymbol{y})$ and $g(\boldsymbol{y})$ the aggregate service levels for period p, call type k, and overall, respectively. The corresponding time limits are s_p, s_k, and s, and the corresponding minimal service levels are l_p, l_k and l. The *scheduling problem* can then be formulated as:

$$\begin{aligned}
\text{minimize} \quad & \boldsymbol{c}^{\mathsf{t}}\boldsymbol{x} = \textstyle\sum_{i=1}^{I}\sum_{q=1}^{Q} c_{i,q}x_{i,q} \\
\text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \\
& g_{k,p}(\boldsymbol{y}) \geq l_{k,p} \qquad \text{for all } k, p, \\
& g_p(\boldsymbol{y}) \geq l_p \qquad \text{for all } p, \\
& g_k(\boldsymbol{y}) \geq l_k \qquad \text{for all } k, \\
& g(\boldsymbol{y}) \geq l, \\
& \boldsymbol{x} \geq 0, \text{ and integer.}
\end{aligned}$$

This is a *deterministic* nonlinear optimization problem with integer decision variables. However, to solve this problem, we need a method that provides good estimates or approximations of the functions $g_\bullet$ in the constraints, for any given solution $\boldsymbol{x}$. Simulation seems to be the only viable approach to obtain reliable estimates of these service levels in realistic call centers. The function $g_{k,p}(\boldsymbol{y})$ usually depends on the values of $y_{i,j}$ for all i and $j \leq p$, in a complicated way, and similarly for the other functions $g_\bullet$. The number of agents of each type changes from period to period, the arrival process is generally non-stationary, the

service times may have arbitrary distributions, there could be abandonments, routing rules that decide which agent type handles each call could be complex, etc.

Atlason, Epelman, and Henderson (2004) and Ceşik and L'Ecuyer (2006) have proposed simulation-based heuristic methods to solve special cases of this problem, based on SAA. The idea is to simulate n days of operation of the call center and replace the unknown functions $g_\bullet$ by the sample averages $\hat{g}_{\bullet,n}$ obtained from the simulation, expressed as functions of the staffing vector $\boldsymbol{y}$ when the random numbers in the simulation are fixed. Often, the expectation $\mathbb{E}[A_{k,p}]$ in the denominator of the definition of $g_\bullet$ can be computed exactly, as in Example 1.49, and we use a sample average only for the numerator. Otherwise, we can use a ratio of sample averages, even if it no longer exactly fits the formulation in (1.41). For example, $g_{k,p}(\boldsymbol{y})$ would be replaced in the SAA by

$$\hat{g}_{k,p,n}(\boldsymbol{y}) = \frac{1}{n\mathbb{E}[A_{k,p}]} \sum_{i=1}^{n} G_{k,p,i}(s_{k,p})$$

where $G_{k,p,i}(s_{k,p})$ is the value of $G_{k,p}(s_{k,p})$ observed for the ith simulated day with staffing $\boldsymbol{y}$, and similarly for the other functions. When $\mathbb{E}[A_{k,p}]$ is unknown, it is replaced by the sample average of the values of $A_{k,p}$ observed over the n days. The (sample) problem then becomes deterministic, and it is solved using deterministic integer programming technology combined with some heuristics. In a nutshell, the constraints are relaxed first by ignoring waiting times and abandons and putting constraints that just ensure that the staffing gives enough capacity to handle the calls, in each period. Then the model is simulated with this staffing to evaluate all the SAA averages at the current solution and identify the constraints of the original problem that are violated. Based on that, new constraints are added, the SAA is solved again to obtain a new solution, we simulate again at this new solution, and so on, iteratively, until all the constraints are satisfied. See the paper for more details. To evaluate these sample averages at different values of $\boldsymbol{y}$ for the same random numbers, we resimulate the model for each staffing vector $\boldsymbol{y}$ of interest, with well-synchronized common random numbers. This is expensive in CPU time, but it can at least provide good solutions in reasonable time, and it seems to be the best approach available so far. Avramidis et al. (2010) use a similar approach to jointly optimize both the staffing and the daily work schedules of the agents. □

Example 1.58 *Likelihood estimation and optimization in a mixed logit model.* [12] *Discrete choice models* provide a representation of how individuals make a selection in face of a finite number of alternatives. In a popular form of *multinomial mixed logit* model (see Section 2.11), the utility of alternative j for individual q has the form

$$U_{q,j} = \boldsymbol{\beta}_q^{\mathsf{t}} \boldsymbol{x}_{q,j} + \epsilon_{q,j} = \sum_{\ell=1}^{d} \beta_{q,\ell} x_{q,j,\ell} + \epsilon_{q,j}$$

where $\boldsymbol{\beta}_q = (\beta_{q,1}, \ldots, \beta_{q,d})^{\mathsf{t}}$ is an unobserved random vector of *taste parameters* (or coefficients) for each individual q, $\boldsymbol{x}_{q,j} = (x_{q,j,1}, \ldots, x_{q,j,d})^{\mathsf{t}}$ gives the *observed attributes* for choice

[12] From Pierre: This example is too much of a special case. We should first discuss max likelihood in general.

j and individual q, and the $\epsilon_{q,j}$ are independent random variables that represent unobserved *random noise*. Each component of $\boldsymbol{x}_{q,j}$ may represent an attribute of individual q (such as his range of income, age, habits, etc.), or an attribute of choice j (for example, the price and the number of stopovers in the case of a flight ticket), or a combination of the two. The $\epsilon_{q,j}$ are assumed to have a Gumbel distribution with mean 0 and scale parameter $\beta = 1$ (Section 2.8.9); this implies that the location parameter δ is the Euler constant $\gamma = 0.57721556\ldots$.

It is assumed that if q is an individual selected randomly from the entire population, then its associated (random) vector $\boldsymbol{\beta}_q$ of taste coefficients has a multivariate density $f_{\boldsymbol{\theta}}$ that depends on some parameter (vector) $\boldsymbol{\theta}$. This randomness accounts for the fact that different individuals may have different tastes. A popular choice for $f_{\boldsymbol{\theta}}$ is the (multivariate) normal density. Individual q always selects the alternative j having the largest utility $U_{q,j}$.

Conditional on a given (fixed) $\boldsymbol{\beta}_q$, we have an ordinary *logit model*, and one can show in that case (see Section 2.11) that the individual selects alternative j with probability

$$L_q(j, \boldsymbol{\beta}_q) = \frac{\exp[\boldsymbol{\beta}_q^{\mathrm{t}} \boldsymbol{x}_{q,j}]}{\sum_{a \in \mathcal{A}(q)} \exp[\boldsymbol{\beta}_q^{\mathrm{t}} \boldsymbol{x}_{q,a}]}, \tag{1.47}$$

independently of other individuals, where $\mathcal{A}(q)$ is the set of his alternatives. The unconditional probability that a random individual selects alternative j is then

$$p_q(j, \boldsymbol{\theta}) = \mathbb{E}[L_q(j, \boldsymbol{\beta}_q)] = \int L_q(j, \boldsymbol{\beta}) f_{\boldsymbol{\theta}}(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta}. \tag{1.48}$$

Suppose we have a data set of one observation per individual for m individuals, in which we observe that individual q was given the vector of attributes $\boldsymbol{x}_{q,j}$ for each alternative j and made the choice y_q, for $q = 1, \ldots, m$. We want to estimate $\boldsymbol{\theta}$ from this data. The standard estimator in this case uses the *maximum likelihood method*: We estimate the true $\boldsymbol{\theta}$ by the value that maximizes the joint probability $L(\boldsymbol{\theta})$ (the likelihood) of the observed sample, as a function of $\boldsymbol{\theta}$, or the *logarithm* of this probability, which is equivalent and computationally simpler. That is, we want to maximize the log-likelihood

$$\ln L(\boldsymbol{\theta}) = \ln \prod_{q=1}^{m} p_q(y_q, \boldsymbol{\theta}) = \sum_{q=1}^{m} \ln p_q(y_q, \boldsymbol{\theta}). \tag{1.49}$$

Except for simple special cases, no explicit formula is available for $p_q(j, \boldsymbol{\theta})$, and Monte Carlo is often the most practical way of evaluating the integral in (1.48) for each q. To do that, for any given $\boldsymbol{\theta}$ and each q, we generate n independent realizations of $\boldsymbol{\beta}$ from the density $f_{\boldsymbol{\theta}}$, say $\boldsymbol{\beta}_q^{(1)}(\boldsymbol{\theta}), \ldots, \boldsymbol{\beta}_q^{(n)}(\boldsymbol{\theta})$, and we estimate $\ln L(\boldsymbol{\theta})$ by

$$\ln(\hat{L}(\boldsymbol{\theta})) = \sum_{q=1}^{m} \ln \left(\frac{1}{n} \sum_{i=1}^{n} L_q(y_q, \boldsymbol{\beta}_q^{(i)}(\boldsymbol{\theta})) \right), \tag{1.50}$$

where each $L_q(y_q, \boldsymbol{\beta}_q^{(i)}(\boldsymbol{\theta}))$ can be computed via (1.47).

Then the next task is to maximize $\ln(\hat{L}(\boldsymbol{\theta}))$ in (1.50) as a function of $\boldsymbol{\theta}$. This is generally difficult, because this function is usually non-convex and its evaluation at any given $\boldsymbol{\theta}$ is expensive. See Bastin, Cirillo, and Toint (2006) and Train (2003) for further details. Here,

the vector $\hat{\boldsymbol{\theta}}$ that maximizes (1.50) is an estimator of the maximizer of a function. This is another example of a situation where what we want to estimate is not expressed as a mathematical expectation.

□

1.15.5 Monte Carlo for optimization in sequential decision processes

♣ To do: Discuss dynamic programming combined with simulation. See for example Chang et al. (2007), Dion and L'Ecuyer (2010) and references given there. Connections with reinforcement learning; see Sutton and Barto (2018).

Example 1.59 Pricing a Bermudan-Amerasian option. We return to the Asian option valuation problem of Example 1.11, under the GBM model, with payoff given in (1.10), but with the following modification: At any observation time t_j, for $m_* \le j < d$, where m_* is a fixed positive integer, the holder can either exercise the option and receive the payoff

$$g(\bar{S}_j) = \max\left(0,\, \bar{S}_j - K\right) \tag{1.51}$$

where $\bar{S}_j = (1/j)\sum_{\ell=1}^{j} S(t_\ell)$, or wait for the next observation time t_{j+1}. At time t_d, if the option has not yet been exercised, the holder receives the payoff $\max\left(0,\, \bar{S}_t - K\right)$. The decision to exercise or not at time t_j would depend in general on the pair $(S(t_j),\, \bar{S}_j)$, which can be viewed as the current state of the decision process at step j. A decision policy is a function $\varphi : \{m_*, \ldots, t\} \times [0,\infty)^2 \to \{0, 1\}$, which assign a decision to each possible state, at each step. The holder exercises the option at step j if and only if $\varphi(j,\, S(t_j),\, \bar{S}_j) = 1$. An optimal decision policy is one that maximizes the expected payoff, and this maximal expected payoff is the initial value of the option. For the model considered in Example 1.11 and here, a good approximation of an optimal policy can be computed by dynamic programming (Ben Ameur, Breton, and L'Ecuyer 2002). But in real-life applications, the payoff often depends on the prices of several correlated assets, or on the evolution of more complicated processes. Then, a decision policy must be defined over a high-dimensional state and dynamic programming becomes more difficult to implement (it hits the so-called *curse of dimensionality*). In that case, one possible approach is to estimate an optimal decision policy via simulation at the same time as one estimates the option value. One general class of methods for doing that is *least-squares Monte Carlo* (Glasserman 2004, Chapter 8, and Dion and L'Ecuyer 2010). It approximates the value function (the option price as a function of the state) at each time step by a linear combination of a fixed set of basis functions, where the coefficients are estimated by least-squares regression from a set of (noisy) evaluation points obtained by simulation. There are different variants of this approach; a popular one is that proposed by Longstaff and Schwartz (2001). This is discussed further in Chapter 7. □

Sensitivity analysis and optimization are covered more extensively in Chapter 7 of this book.

1.16 Exercises

Many exercises in this chapter are *training camp* problems, in the sense that they require little or no prior knowledge of simulation. Their aim is to gain insight and intuition for the remainder of the book.

1.1 Suppose that a uniform random number generator is used in some way to shuffle a deck of 52 playing cards, and that the period length of the generator equals its number of states.

(a) What is the number of ways of choosing the first t cards, for $t \leq 52$, taking order into account?

(b) What is the *minimal period length* of the generator, and the minimal number of bits needed to represent its state, to make sure that every possibility can happen for the first t cards? Give expressions that are functions of t.

(c) What is that minimal number of bits for $t = 52$? (Give a numerical value.)

(d) If the period length is $2^{31} - 2$ (many widely-used classical LCGs have that value), what is the maximal value of t for which we can have all the possibilities?

1.2 Suppose we want to generate n i.i.d. random variates $X_1, \ldots, X_n$ with cdf F. Let $X_{(1)}, \ldots, X_{(n)}$ be their values sorted by increasing order. These are the *order statistics*. A simple way of generating these order statistic is to first generate $X_1, \ldots, X_n$ independently, and then sort them. The best sorting algorithms can do that in $\mathcal{O}(n \log n)$ time. The purpose of this exercise is to figure out a way of generating the order statistics in $O(n)$ time instead. The method generates the order statistics of n i.i.d. $U(0,1)$ random variables, $U_{(1)}, \ldots, U_{(n)}$, and applies inversion to the $U_{(i)}$'s to get the $X_{(i)}$'s.

(a) Explain how to generate $U_{(n)}$ directly by inversion from a single uniform. Hint: prove that $(U_{(n)})^n \sim U(0,1)$ by writing its cdf.

(b) After $U_{(n)}, \ldots, U_{(k+1)}$ have been generated, for $1 \leq k < n$, explain how to generate $U_{(k)}$ directly by inversion from a single uniform, and prove that your method works.

Hint: You can show by backward induction on k that the joint density of $(U_{(1)}, \ldots, U_{(k)})$ conditional on $U_{(k+1)} = u$ is *constant* over the simplex defined by $\{0 \leq U_{(1)} \leq \cdots \leq U_{(k)} \leq u\}$, and is the same as the joint density of the order statistics of k i.i.d. uniform random variables over the interval $[0, U_{(k+1)}]$. From this, you can show that $[U_{(k)}/U_{(k+1)}]^k \sim U(0,1)$. To start the induction, show that the vector $(U_{(1)}, \ldots, U_{(n)})$ has density $n!$ over this simplex for $k = n$.

(c) By putting these ingredients together, give an algorithm that generates $X_{(1)}, \ldots, X_{(n)}$ in $\mathcal{O}(n)$ time (and explain why it works).

(d) Implement your algorithm, try it for the case where the X_i are exponential with mean 1, and compare its speed with an algorithm that generates the X_i's independently and sort them using a quicksort algorithm, for $n = 10^2$, 10^4, and 10^6. Discuss your results.

1.3 (a) An array contains n objects that we want to permute randomly. (This can be used, for example, to shuffle a deck of n cards in a computerized gaming machine.) Give an algorithm that does that in $\mathcal{O}(n)$ time, without using another array, and prove that your algorithm returns any of the $n!$ permutations with equal probability.

Hint: suppose that you select the first object in the permutation at random among all n objects, then you select the second one at random among the $n-1$ that remains, then the third among the $n-2$ that remain, and so on. Show by induction on k that after the objects $0, \ldots, k-1$, of the permutation have been selected, each possible ordered choice of k objects among n (there are $n!/(n-k)!$ such choices) has the same chance of having been chosen, for $1 \le k \le n-1$.

(b) Implement your algorithm, in a way that you can stop after selecting the first k objects of the permutation, for any $k \le n$. Use it to generate 60000 permutations with $k = 3$ and $n = 5$, count how many times each of the $n!/(n-k)! = 60$ possibilities appears, and perform a chi-square test (see Section A.15) on these counts to test your implementation.

1.4 Implement the SWB generator of Example 1.16, whose parameters are $(b, r, k) = (2^{31}, 8, 48)$, and real-valued output defined by $u_n = x_{2n}/2^{62} + x_{2n+1}/2^{31}$. Partition the unit cube into $k = 10^6$ subcubes by partitioning each axis into 100 equal intervals. Number these subcubes from 0 to $k-1$ (in any way).

(a) Use the SWB generator defined above to generate three-dimensional points in $[0,1)^3$, defined by $\boldsymbol{u}_i = (u_{3i}, u_{3i+1}, u_{3i+2})$, for $i = 0, \ldots, m-1$, for $m = 10^4$. For each point $\boldsymbol{u}_i$, find the number of the subcube in which it falls, and count the number C of collisions as in Example 1.8. Repeat this 10 times, to obtain 10 "independent" realizations of C, and compare their distribution with the Poisson approximation given in Example 1.8. You can do the latter comparison informally; there is no need to perform a formal statistical test that it is really a Poisson distribution, but you can compute a confidence interval for the mean.

(b) Do the same with the three-dimensional points $\boldsymbol{u}_i = (u_{25i}, u_{25i+20}, u_{25i+24})$ for $i = 0, \ldots, m-1$. Discuss and explain what you observe.

(c) Redo the same experiment, but this time using a better generator, such as `MRG32k3a` in SSJ, for example. Discuss your results.

1.5 (a) Prove that if X_1 and X_2 are two random variables taking their values in $[0, \infty)$, then $\mathbb{E}[\max(X_1, X_2)] \ge \max(\mathbb{E}[X_1], \mathbb{E}[X_2])$ and $\mathbb{E}[\min(X_1, X_2)] \le \min(\mathbb{E}[X_1], \mathbb{E}[X_2])$. Hint: Think of Jensen's inequality.

(b) In Figure 1.3, we see that when we replace each Y_j by its mean, we obtain a value smaller than $\mathbb{E}[T]$, where T is the length of the longest path. Prove that this is always true, for any network. And what happens if T is the length of a shortest path instead? Prove it.

1.6 Show that if U is a $U(0,1)$ random variable, then the random variable $X = [-\ln(1-U)]^{1/\alpha}/\lambda$ has the Weibull distribution with parameters α and λ, assuming that $\alpha > 0$ and $\lambda > 0$.

1.7 Give an algorithm to generate a Gumbel random variable with parameters $\delta \in \mathbb{R}$ and $\beta \ne 0$ (see Section 2.8.9) by inversion.

1.8 We want to generate a vector (Y_1, Y_2) of correlated standard normal random variables, with linear correlation coefficient ρ. Prove that the following instructions do that:

Generate two independent standard normals, Z_1 and Z_2;

Return $Y_1 \leftarrow Z_1$ and $Y_2 \leftarrow \rho Z_1 + \sqrt{1-\rho^2} Z_2$.

You can use the fact that linear combinations of normal random variables are again normal random variables, so it only remains to check the mean, variance, and correlation.

1.9 (Approximate counting by Monte Carlo) Consider a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is a set of nodes (or vertices) and $\mathcal{A}$ a set of edges, where each edge connects two nodes. For a given set of c different colors, a *feasible coloring* of the graph selects one color for each node, in a way that no two neighbors (two nodes connected by an edge) have the same color. We want to compute (or estimate) the total number of feasible colorings, say n_{f}.

(a) The simplest way of doing that (conceivably) is to examine each of the c^k possible colorings, where $k = |\mathcal{N}|$ is the number of nodes, and check if it is feasible. Checking for feasibility is easy: take each node and check if each of its neighbors has a different color. Is this method practical if $c = 10$ and $k = 100$, for example? Why?

(b) A naive implementation of the Monte Carlo approach of Example 1.6 provides the following unbiased estimator of n_{f}. Generate n random colorings, independently and uniformly over all c^k possibilities, and count the number Y of those colorings that are feasible. Let $p = n_{\mathrm{f}}/c^k$. Show that Y is a $\mathsf{Binomial}(n, p)$ random variable and that $\hat{n}_{\mathrm{f}} = c^k Y/n$ is an unbiased estimator of n_{f}.

(c) Suppose that the graph $\mathcal{G}$ is connected (there is a path from each node to any other node). In that case, show that one must have

$$p \le [(c-1)/c]^{k-1}.$$

For $c = 10$ and $k = 100$ (for example), what does this inequality imply for the relative error of the estimator $\hat{n}_{\mathrm{f}}$? Make the connection with Example 1.29. (Note: one should not conclude that the Monte Carlo method is useless for this problem, but that we need to design a better sampling scheme than just uniform sampling over all c^k possibilities.)

1.10 In Example 1.21, explain in detail where the formula for $v(s_0, T)$ comes from.

1.11 In Example 1.11, consider an European call option, with payoff $g(S(T)) = \max[0, S(T) - K]$ at time T, under the Black-Scholes model. Note that the value of the option at time t, conditional on $S(t) = s$, is

$$\mathbb{E}^*[e^{-r(T-t)} g(S(T)) \mid S(t) = s] = v(s, T-t),$$

and this can be computed explicitly by the Black-Scholes formula (1.9). For $t = 0$, this gives $v(s_0, T)$, the initial price of the option. Define $v'(s, T-t) = \partial v(s, T-t)/\partial s$, which is the *delta of the option* at time t (Hull 2006). An explicit expression for this v' can be obtained by taking the derivative in the Black-Scholes formula. Suppose that we start with a sum of $v(s_0, T)$ (the price of the option) at time 0. We could be the financial institution that just sold this option, for instance. Suppose also that at time t, for $0 \le t \le T$, we hold $w(t) = v'(S(t), T-t)$ shares of the asset, and leave the rest of our money, say $y(t)$, in the bank account. Thus, at time 0 we immediately buy $w(0)$ shares, and then the number of shares varies continuously with t. Then, it can be shown that at any time t, the current value $p(t)$ of the portfolio (the shares and the money in bank account), which is $p(t) = w(t)S(t) + y(t)$,

is exactly equal to the expected discounted payoff of the option, $v(S(t), T-t)$. In other words, this continuously-varying portfolio always replicates exactly the payoff of the option, regardless of the sample path of S. In general, a portfolio (or strategy) with this property is said to *hedge* the option (against the risk). The hedging strategy used here is *delta hedging* (Hull 2006).

One might argue that if the option payoff can be replicated in that manner, then there is no need to create that option in the first place. However, hedging exactly in continuous time is not really possible, and even it it was, one may prefer to avoid the trouble of doing it and buy an option instead. On the other hand, the financial institution that sells the option might want to use a *discrete-time approximation* of the exact delta hedging, to protect itself against large losses. This discrete-time hedging does not reproduce the option exactly so there is still some risk and the institution may end up losing money with the contract once in a while, but at least the risk is reduced.

A portfolio that delta hedges the option at times $0 = t_0 < t_1 < \cdots < t_d = T$ will hold $w(t_j)$ asset units during the time interval $[t_j, t_{j+1})$, where $w(t_j) = v'(S(t_j), T - t_j)$, for $j = 0, \ldots, d-1$. If $y(t_j)$ is the sum that remains in the bank account at time t_j, then

$$y(t_{j+1}) = e^{r(t_{j+1}-t_j)} y(t_j) - [w(t_{j+1}) - w(t_j)] S(t_{j+1}),$$

where the first term on the right is the money that was in the bank, with interest, and the second term is the money taken from the bank account to buy new asset units at time t_{j+1}. Note that with discrete-time hedging, this amount may sometimes become negative.

Suppose that time is in years and that $T = 1$, $d = 12$, $t_j = j/d$, $S(0) = K = 100$, $r = 0.03$, and $\sigma = 0.2$. Simulate the process $n = 1000$ times and compute the pair $(g(S(T)), p(T))$ for each replication. Make a scatter plot with these n points. If hedging is effective, these points should be (approximately) aligned on a 45 degree line. Then make a histogram of the n realizations of the observed difference $p(T) - g(S(T))$, which represents the net profit made by the bank, to estimate the distribution of this net profit. Try also $d = 52$ (weekly hedging) and $d = 250$ (daily hedging on working days), and compare. Discuss your results.

1.12 You will experiment with the Monte Carlo estimator (1.50) of the log-likelihood function in Example 1.58. In practice, the attribute vectors $\boldsymbol{x}_{q,j}$ and the choices y_q are given in the dataset and we want to estimate $\boldsymbol{\theta}$ from that. But here, instead of using real data, you are asked to perform a two-stage artificial experiment, as follows (as in Sivakumar, Bhat, and Ökten 2005).

(a) In the first stage, you will assume that the model and $\boldsymbol{\theta}$ are known, and you will generate an artificial data set from that model. Take $d = 5$ and $\mathcal{A}(q) = 4$ alternatives for each individual q. Assume that the coordinates $x_{q,j,\ell}$ of the attribute vectors are independent normal random variables with variance 1, and with mean 1 for alternatives $j = 1, 2$ and mean 0.5 for alternatives $j = 3, 4$. The coordinates $\beta_{q,\ell}$ of $\boldsymbol{\beta}_q$ are also assumed to be independent $\mathsf{N}(1, 1)$ random variables. The $\epsilon_{q,j}$ are independent Gumbel random variables with mean 0 and scale parameter 1 (so their location parameter δ is the Euler constant γ). For each individual q, you generate the relevant random variables from these distributions, and find the alternative y_q having the largest utility for this individual. Repeat this for $m = 2000$ individuals and store all the relevant information in a file.

(b) In the second stage, you take the data set generated in the first stage, assume that $\beta_{q,\ell} \sim \mathsf{N}(\mu, \sigma^2)$ for each (q, ℓ), but pretend that you do not know the parameter vector $\boldsymbol{\theta} = (\mu, \sigma^2)$. Make $n = 1000$ simulation runs to estimate the log-likelihood function Example 1.58 by (1.50), for $\boldsymbol{\theta} = (1, 1)$. Repeat this experiment three times, with independent random numbers. Discuss your results.

(c) Perform the same estimation experiment as in (b) with $\boldsymbol{\theta} = (0.8, 1)$, $(1.2, 1)$, $(1, 0.8)$, $(1, 1.2)$, all with independent random numbers. What can you conclude from this? Do you think it easy in general to recover the original $\boldsymbol{\theta}$ using Monte Carlo and some optimization procedure? What would be the main sources of error in the final result?

(d) Is (1.50) an unbiased estimator of (1.49)? Why?

1.13 These questions apply to Example 1.24.

(a) Start from the definition of S_n^2 in Eq. (1.22) and show that $S_n^2 = Y(1-Y/n)/(n-1)$.

(b) Explain why it is reasonable to assume that Y/n follows approximately the normal distribution when n is large and μ is not too close to 0 or 1. What is the problem if μ is close to 0 or 1?

(c) Suppose you observe $Y = 2504$ with $n = 10000$. Compute a 99% confidence interval for μ.

1.14 Replace $(0, 1)^s$ in (1.16) by D, where D is a bounded domain in $\mathbb{R}^s$, and replace the estimator (1.19) by

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} \mathrm{vol}(D) f(\boldsymbol{u}_i)$$

where the $\boldsymbol{u}_i$ are i.i.d. uniform random variables over D. Show that $\mathbb{E}[\hat{\mu}_n] = \mu$ and that

$$\mathrm{Var}[\hat{\mu}_n] = \frac{1}{n} \left[\mathrm{vol}^2(D) \int_D f^2(\boldsymbol{u}) \mathrm{d}\boldsymbol{u} - \mu^2 \right].$$

1.15 Compare the variances of the two estimators in Example 1.23 by simulating $n = 10^5$ runs for each and comparing the empirical variances. Compare also their computing times.

1.16 This is a variant of Example 1.29. Suppose that for some simulation model, the outcome (or cost) X is K with probability p and 0 with probability $1 - p$. The value of p is unknown. We perform n independent simulation runs and take the average cost $\bar{X}_n$ over these n runs as an unbiased estimator of the expected cost $\mu = pK$.

(a) What is the exact variance of $\bar{X}_n$?

(b) Suppose now that $p = 1/K$, so that $\mu = \mathbb{E}[X] = 1$. Show that the variance of X is in $\mathcal{O}(1/p) = \mathcal{O}(K)$ as $p \to 0$.

(c) The variance of X can be estimated by the sample variance S_n^2. Write S_n^2 and $\mathrm{Var}[S_n^2]$ as functions of K, n, and the binomial random variable $B = (X_1 + \cdots + X_n)/K$.

(d) The expressions found in (c) involve the first four moments of B. These moments are easily found via the moment generating function of B: The rth moment $\mathbb{E}[B^r]$ is the r-th derivative of the moment generating function $M(s) = [pe^s + (1 - p)]^n$ evaluated at $s = 0$. After some calculations, one obtains

$$\begin{aligned}
\mathbb{E}[B] &= np, \\
\mathbb{E}[B^2] &= np(1+(n-1)p), \\
\mathbb{E}[B^3] &= np(1+(n-1)p(3+(n-2)p)), \\
\mathbb{E}[B^4] &= np(1+(n-1)p(7+(n-2)p(6+(n-3)p))).
\end{aligned}$$

Use these ingredients to obtain an explicit expression for $\mathrm{Var}[S_n^2]$ as a function of K and n only, and to show that this variance is $\mathcal{O}(1/(np^3)) = \mathcal{O}(K^3/n)$ as $p \to 0$. Compare its rate of increase with that of $\mathrm{Var}[\bar{X}_n]$. You are allowed to use a software system (such as Mathematica) to do the algebraic simplifications.

1.17 In the previous exercise, the mean is fixed and the variance is a function of p. Here, both the mean and variance are fixed, but the higher moments change with p. Suppose that $\mathbb{P}[X = K] = p$ and $\mathbb{P}[X = -1/K] = 1 - p$, where $0 < p < 1$, $K > 0$, and $K^2 = (1-p)/p$.

(a) Show that $\mathbb{E}[X] = 0$ and $\mathrm{Var}[X] = 1$ regardless of p.

(b) Explain in words the shape of the probability mass function of X and how it changes when p goes from, say, 0.5 to 0.001.

(c) If the average $\bar{X}_n$ of a sample $X_1, \ldots, X_n$ of i.i.d. copies of X is taken as an estimator of $\mu = \mathbb{E}[X]$ in this case, assuming that μ is unknown and that the total expected simulation time is the same for all p, then the *efficiency* $\bar{X}_n$ (with our definition) does not depend on p. Does this really mean that the estimators for the different values of p are equally attractive? To help answer this question, suppose that to estimate $\mathrm{Var}[X]$ we use the empirical variance S_n^2. How does $\mathrm{Var}[S_n^2]$ behave as a function of p and n when n is large?

1.18 Suppose we want to estimate the area of a disk of radius 1 centered at the origin, by the MC method. This is purely academic, of course, because we know that this area is π. The purpose of this exercise is to understand what happens if we try to estimate π via straightforward MC simulation. In fact, we can simplify the problem by estimating the area of the disk lying in the positive quadrant, which is $\mu = \pi/4$, and then multiply by 4. This μ can be written as $\mu = \int_0^1 \sqrt{1-x^2}\mathrm{d}x$.

(a) In this context, what are the estimators $\hat{\mu}_n$ and $\tilde{\mu}_n$ defined in (1.19) and (1.25), in Section 1.4?

(b) Give explicit formulas for their variances.

(c) Assuming that the computing times of $\hat{\mu}_n$ and $\tilde{\mu}_n$ are $n\kappa_1$ and $n\kappa_2$, respectively, give expressions for their efficiencies. Why can we expect that $\kappa_2 < \kappa_1$? Under what conditions on κ_1 and κ_2 is $\hat{\mu}_n$ more efficient than $\tilde{\mu}_n$?

(d) What is the required value of n to obtain π with approximately r decimals of accuracy with the estimator $\hat{\mu}_n$, in the sense that the error on π is less than 10^{-r} with 95% confidence? What is that n for $r = 3$? What about $r = 20$? Is this feasible?

1.19 We now generalize the previous exercise to s dimensions. There is no need to perform simulations for this exercice; everything can be computed numerically. We want to estimate the volume V_s of a s-dimensional unit sphere (i.e., of radius 1, centered at the origin). For that, we estimate the volume p of the intersection of the sphere with the positive orthant, i.e., with $(0,1)^s$, and observe that $V_s = 2^s p$. This is purely an academic exercise, because we

already have a formula for V_s. The goal is to understand the difficulty of estimating p by Monte Carlo when s is large. This type of difficulty occurs very frequently.

(a) Explain what is the hit-and-miss estimator $\tilde{p}_n$ of p in this case, prove that it is unbiased, and give exact formulas for the variance and relative error of $\tilde{p}_n$ and of $\tilde{\mu}_n = 2^s\tilde{p}_n$, the estimator of V_s, as functions of s and n only.

(b) Suppose that we want the relative error to be bounded by a constant as a function of s, say smaller than 0.01 for example. For this, at what speed should we increase n as a function of s? Give a complexity expression of the form $n = \Theta(g(s))$ for some explicit function g of s only. Use the Stirling approximation to derive your expression. What does this expression implies for large values of s? What n would we need for $s = 10$? For $s = 20$?

(c) Compute V_s, p, and the values of the variance and of n times the square relative error of $\tilde{\mu}_n$ for $s = 2$, 5, 10, and 20. (Give numerical values.)

Hint: The volume of a unit sphere is $V_s = \pi^{s/2}/\Gamma(1 + s/2)$, where $\Gamma(\cdot)$ is the *gamma function* which satisfies $\Gamma(1/2) = \sqrt{\pi}$, $\Gamma(1) = 1$, and $\Gamma(s + 1) = s\Gamma(s)$ (see Section 2.8.12). For large s, you can use the Stirling approximation, which gives

$$\Gamma(1 + s/2) \approx \lfloor s/2 \rfloor ! \approx (\pi s)^{1/2}(s/(2e))^{s/2}.$$

1.20 The following type of model is used to represent the transformation process of grains of powder by some chemical reaction. Consider a population of three-dimensional solids, called *grains*, having the same shape but perhaps different sizes. The grain sizes are i.i.d. random variables. At time 0, all grains are *white* (say). Spots of transformation, called *nuclei*, appear on the surface of the grains according to a Poisson process with rate proportional to their surface. The rate is $\lambda(t)S$ at time t for a surface of size S and the birth locations of the nuclei are independent and uniformly distributed over the surface. Each nucleus can be viewed as the intersection of a sphere with the grain. The radius $r(t)$ of the sphere is 0 at birth and grows at time-dependent rate $r'(t)$ (the same for all grains) thereafter. Thus, both the birth and growth rates of the nuclei may depend on time (they can depend on the temperature, pressure, etc.). For each grain, the part of its volume lying inside any of its nuclei becomes *red* (say).

Denote by V and $R(t)$ the total volume of a grain selected at random and the volume that is red at time t, respectively. For n grains, the fraction of volume that is red at time t converges with probability 1 to

$$p(t) = \mathbb{E}[R(t)]/\mathbb{E}[V]$$

as $n \to \infty$, by the strong law of large numbers. The goal is to compute $p(t)$ as a function of t. Except for special cases, this is too hard to do analytically. The following straightforward simulation (or MC) method can be used.

For each value of t of interest, repeat the following for $i = 1, \ldots, n$, independently:

(1) generate V_i, the volume of grain i, from its appropriate distribution;

(2) generate the birth times of nuclei on the surface of that grain during the time interval $[0, t]$, from the appropriate Poisson process;

(3) generate the positions of these nuclei uniformly on the grain surface;

(4) compute the radius of each nucleus at time t;

(5) generate m_i independent points $X_{i,1}, \ldots, X_{i,m_i}$, uniformly distributed in the grain;

(6) compute Y_i, the number of these points that lie in one of the nuclei, and estimate the red volume in grain i at time t by $R_i(t) = V_i Y_i / m_i$.

At time t, a point X lies inside a nucleus born at point W at time s if and only if the Euclidean distance between X and W is less than $r(s,t) = \int_s^t r'(u)\mathrm{d}u$.

When $\mathbb{E}[V]$ can be computed analytically, we can use the following consistent estimator of $p(t)$:

$$\hat{p}_n(t) = \frac{\bar{R}_n(t)}{\mathbb{E}[V]} = \frac{1}{n\mathbb{E}[V]} \sum_{i=1}^{n} R_i(t).$$

(a) Implement this MC algorithm for estimating $p(t)$ in the case of cubic grains with side length C (volume $V = C^3$ and surface $S = 6C^2$), where C is a normal random variable with mean $\mu = 1$ and standard deviation $\sigma = 0.1$. Take $\lambda(t) = 1$ for $t < 1$, $\lambda(t) = 4$ for $t \geq 1$, and $r'(t) = 0.1$ for all t. Compute the estimator $\hat{p}_n(t)$ of $p(t)$, and an estimator of its variance, for $t = 1, 2, 5, 10$, using $n = 1000$ and $m_i = 1$ for all i.

(b) Do you obtain a more efficient estimator by increasing the value of m_i (and keeping it the same for all i)? Do you think m_i should increase with n? Justify and experiment.

(c) What about varying m_i as a function of the grain size? For example, one could take m_i proportional to the volume. Does it make sense? Why? Are there other interesting choices? Discuss and experiment.

(d) How would you apply QMC sampling to this problem?

(e) Can you think of an efficient way of estimating the function $p(t)$ at all values of t simultaneously? (See Exercise 1.39 for a somewhat related question.)

1.21 Consider a two-dimensional point set P_n for which each of the two coordinates takes each value in $\mathbb{Z}_n/n$ exactly once. We saw that to define this point set it suffices to define one permutation of $\mathbb{Z}_n/n$ for each coordinate. We also pointed out that if these two permutations are the same, then P_n is far from being evenly distributed over $[0,1)^2$.

(a) Give two other examples of a pair of permutations for which the two-dimensional uniformity of P_n is bad.

(b) Explain why one can always take the first permutation as the identity, so it suffices to select one permutation (the second one).

(c) In s dimensions, how many permutations do we have to select? Why?

1.22 In one dimension, give a set P_n of n points such that $D^*(P_n) = 1/(2n)$ and $D_e(P_n) = 1/n$. Give an example of a function f for which $V(f) = 1$ and for which the upper bound in the Koksma-Hlawka inequality is attained. (For $s = 1$, $V(f)$ is the total increase of f over the intervals where f is increasing, plus the total decrease of f over the intervals where f is decreasing, over $(0,1)$.)

1.23 Give an example of a function $f : [0,1) \to \mathbb{R}$ for which the crude MC estimator $f(U)$, where $U : U(0,1)$, has finite variance, but for which the Koksma-Hlawka error bound is infinite because $V(f) = \infty$ (see the previous exercise for a definition of $V(f)$).

1.24 Consider a MC integration problem where the function f in (1.16) is quadratic:

$$f(\boldsymbol{u}) = f(u_1, \ldots, u_s) = \sum_{k=1}^{s} \sum_{\ell=1}^{s} q_{k\ell} u_k u_\ell$$

where the $q_{k\ell}$ are constants.

(a) Show that with $\hat{\mu}_n$ defined as in (1.32), $E_n = \hat{\mu}_n - \mu$ can be written as a sum of functions of 2 variables.

(b) Use the Koksma-Hlawka inequality to bound this sum by a sum of expressions involving the total variation of simple functions of 2 variables and discrepancies of 2-dimensional point sets. (You can consult Kuipers and Niederreiter 1974 for the definition of *total variation* in the sense of Hardy and Krause.)

(c) Suppose we would like to construct a *single s-dimensional* point set P_n to be used for QMC integration for this specific class of quadratic functions. The $q_{k\ell}$'s are supposed unknown, but we may assume that they are i.i.d. uniformly distributed over some positive interval. What quantity should we then try to minimize when choosing P_n, to minimize the error bound that you found?

1.25 For $s = 8$, 16, 32, and 64, find the smallest $n_0 > 0$ such that $n^{-1}(\ln n)^s \leq 1/\sqrt{n}$ for all $n \geq n_0$. What are the implications of your results on the practical usefulness of the Koksma-Hlawka error bounds and on the efficiency comparison of MC versus QMC based on these bounds? (Note: Even if you conclude that the *bounds* are useless, this does not necessarily imply that QMC itself is useless.)

1.26 Let P_n be an arbitrary (deterministic) set of n points in $[0,1)^s$. We randomize this point set by applying independent random shifts, as suggested in Section 1.5.3. Let $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_r$ be independent random points uniformly distributed over $(0,1)^s$. For $j = 1, \ldots, r$, define the point set $P_n^{(j)} = (P_n + \boldsymbol{U}_j) \bmod 1$, where the reduction modulo 1 is component by component, for each component of each vector. In other words, each point $\boldsymbol{u}_i \in P_n$ corresponds to a point $\boldsymbol{U}_i^{(j)} = (\boldsymbol{u}_i + \boldsymbol{U}_j) \bmod 1$ in $P_n^{(j)}$. If you have difficulties having a good intuition of what happens, it may be a good idea to draw yourself a picture of what happens in the case where $s = 1$ and $P_n = \{0, 1/n, \ldots, (n-1)/n\}$.

Define

$$X_j = \hat{\mu}_n^{(j)} = \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{U}_i^{(j)})$$

for each j, where $f : (0,1)^s \to [0, \infty)$, and, as usual,

$$\bar{X}_r = \frac{1}{r} \sum_{j=1}^{r} X_j.$$

(a) Prove that each $\boldsymbol{U}_i^{(j)}$ has the uniform distribution over $(0,1)^s$.

(b) For a fixed j, are the $\boldsymbol{U}_i^{(j)}$'s independent? And for a fixed i, are the $\boldsymbol{U}_i^{(j)}$'s independent? If your answer is yes, prove it.

(c) Show that the X_j's are pairwise uncorrelated and unbiased estimators of $\mu = \int_{(0,1)^s} f(\boldsymbol{u}) \mathrm{d}\boldsymbol{u}$, regardless of P_n and f.

(d) Give an unbiased estimator of the variance of $\bar{X}_r$ and prove that it is unbiased. How would you compute a confidence interval for μ?

This randomization scheme can be used to obtain an estimator for which the error can be estimated in the case where P_n is a quasi-random point set. Since all the points are rotated by the same amount, the low-discrepancy of P_n is preserved for each j. As a consequence, it is possible with good choices of P_n to make the variance of X_j decrease at a faster rate than $\mathcal{O}(1/n)$.

(e) Discuss the compromise to be made in the choice of r and n if the product $b = rn$ (the total number of function evaluations, which can be viewed as the computing budget) is fixed. In this context, taking $r = 1$ (and $n = b$) should be the best choice to minimize the error. Why? So, what is the interest of taking $r > 1$ in practice? Hint: You can assume that both P_n and its randomized version are low-discrepancy point sets, and use the Koksma-Hlawka inequality.

1.27 Let P_∞ be a uniformly distributed sequence and consider the function f defined by $f(\boldsymbol{u}) = \mathbb{I}[\boldsymbol{u} \in P_\infty]$. Show that Weyl's theorem cannot hold for such a function, and therefore that this f cannot be Riemann integrable if P_∞ is uniformly distributed. Explain why, as a consequence, "Riemann-integrable" cannot be replaced by "Lebesgue-integrable" in Theorem **??**. [13]

1.28 Perform a simulation experiment similar to that reported in Example 1.41, but with $K = 1$ and $a = 0$, and with $b = 2$, 3, and 4. Discuss your results.

1.29 Show in detail that the last equality in Eq. (6.101) is true.

1.30 (a) Show that in Example 1.38, $\mathrm{Var}[X_{\mathrm{is}}]$ is minimized by taking

$$\lambda_0 = \lambda_0^* = \lambda + 1/y_0 - (\lambda^2 + 1/y_0^2)^{1/2}.$$

(b) If $\lambda = 1$ and $y_0 = 3$, what is λ_0^* and by what factor is the variance reduced with this λ_0^*?

(c) In (b), what is the range of values of λ_0 for which the variance is reduced compared with $\lambda_0 = 1$ (no IS)? If we choose $\lambda_0 \neq \lambda_0^*$, does the variance reduction degrade faster, as a function of $|\lambda_0 - \lambda_0^*|$, when $\lambda_0 < \lambda_0^*$ (too small) or when $\lambda_0 > \lambda_0^*$ (too large)?

1.31 Consider the hashing model in Example **??** (temporarily removed), with the parameters given there. We want to estimate $\mu = \mathbb{P}[C > x]$ for $x = 50$, 100, and 200. For each x, perform $n = 1000$ simulation runs to estimate μ using the proposed method, first with $\lambda_0 = \lambda_{\mathrm{m}}$, and then with $\lambda_0 = \sqrt{2kx}$. For each case, compute the estimator of μ, estimate its variance, and compute a 95% confidence interval on μ. Are these intervals reliable? Discuss your results.

1.32 In the single-server queue (Section 1.11), suppose we want to estimate q, the average queue length in steady-state, by simulation. (This is only for the purpose of the exercise, of course, because we know the exact value of q.) Our (biased) estimator of q is

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i,$$

[13] From Pierre: To be defined!

where $X_1, \ldots, X_n$ are n i.i.d. copies of $\bar{Q}_T$. The expected computing cost of $\bar{X}_n$ is approximately proportional to nT and it can be proved that both the variance and the (absolute) bias of $\bar{Q}_T$ are in $\mathcal{O}(1/T)$. More specifically, we assume that for large enough T, we have $C(\bar{X}_n) \approx \kappa_0 nT$, $\mathrm{Var}[\bar{Q}_T] \approx \kappa_1/T$, and $|\beta| = |\mathbb{E}[\bar{Q}_T] - q| \approx \kappa_2/T$, for some constants κ_0, κ_1, and κ_2.

(a) Give an expression for the efficiency of $\bar{X}_n$.

(b) Suppose that our available computing budget is B. We want to choose n as a function of B (and then take $T = B/(\kappa_0 n)$) to maximize the efficiency. What is the optimal value of n (it must be a strictly positive integer, of course) as a function of B? If n is chosen in this optimal way, what is the asymptotic efficiency as $B \to \infty$? What is the problem with this choice of n if we want to compute a confidence interval on q based on $X_1, \ldots, X_n$?

(c) Suppose now that we take $n = T$ (assuming that B/κ_0 is always a square, to simplify). This is not optimal. What is the asymptotic efficiency when $B \to \infty$ in that case?

(d) Same question for the case where T is fixed (say, $T = 10000$), so that n is proportional to B.

1.33 Suppose that in the programs `QueueEv` and `QueueProc`, in Figures 1.25 and 1.27, we use a single random number stream instead of using separate streams for the inter-arrival times and service times. Will the two programs give different results? Why? Try it.

1.34 Modify the simulation program based on Lindley's recurrence, given in Figure 1.28, to compute both the average waiting time $\bar{W}_{N_c(T)}$ and the average queue length $\bar{Q}_T$ for a fixed time horizon T, without using an event list. Hint: By keeping the sum of inter-arrival times in an accumulator, you can track the current simulation time. To compute $\int_0^T Q(t)\mathrm{d}t$, you need to figure out, for each customer, its waiting time up to time T, and add these up.

1.35 In the call center example of Section 1.12, we estimate w, $g(s)$, and ℓ by $\bar{W}_n/a$, $\bar{G}_n(s)/a$, and $\bar{L}_n/a$, respectively, and we compute confidence intervals via the Student distribution.

(a) When and why is it reasonable to use the Student distribution to compute these confidence intervals? And the normal distribution? You can run experiments and plot histograms to support your recommendations.

(b) Run the program with the parameters given in file `CallCenter.dat` that comes with it, for $n = 1000$. Compare the performance measures obtained with the following values of α_0: 0.5, 1.0, 2.0, and ∞. (In the latter case, B just becomes the constant 1.) Discuss the effect of increasing the variance of B (by decreasing α_0) on the variance of the estimators and on their theoretical averages, as a function of α_0.

(c) In the case where the variance of the single random variable B has a significant impact on the variance of the estimators, we may consider integrating numerically with respect to the density of B, and generating only the other random variables. A simple way of doing this is as follows: If n days are simulated, replace B by $F^{-1}((i-1/2)/n)$ on day i, where F is the cdf of B. All other computations are performed as usual.

The rationale of this is as follows. To simulate the center for one day, the simulation program generates independent $U(0,1)$ random variates $U_0, U_1, U_2, \ldots$ and transform them to obtain the inter-arrival times, service times, etc., and ultimately all the performance measures of interest for that day. Each performance measure for the day (e.g., W_i or $G_i(s)$

or L_i) can thus be written as a $X = f(U_0, U_1, U_2, \dots)$ for some function f. Suppose that the first uniform, U_0, is used to generate B by inversion; that is, $B = F^{-1}(U_0)$. If the n days are simulated independently by standard MC, then the n copies of U_0 are independent $U(0,1)$ random variates and the n copies of X are also independent. Noticing that

$$\begin{aligned}
\mathbb{E}[X] &= \int_0^1 \int_0^1 \int_0^1 \cdots f(u_0, u_1, u_2, \dots) \mathrm{d}u_0 \mathrm{d}u_1 \mathrm{d}u_2 \cdots \\
&= \int_0^1 \int_0^1 \cdots \left(\int_0^1 f(u_0, u_1, u_2, \dots) \mathrm{d}u_0 \right) \mathrm{d}u_1 \mathrm{d}u_2 \cdots \\
&= \int_0^1 \mathbb{E}\left[f(u, U_1, U_2, \dots) \right] \mathrm{d}u \\
&\approx \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left[f((i-1/2)/n), U_1, U_2, \dots) \right]
\end{aligned}$$

when n is large, the idea is to generate one copy of the random variable $f((i-1/2)/n), U_1, U_2, \dots)$ for each i and take the average. This corresponds to integrating numerically with respect to u_0 by a very simple integration rule.

Explain why we cannot compute confidence intervals by assuming that the random variables W_i, $G_i(s)$, and L_i are i.i.d. under this setting, and why $\bar{W}_n/a$, $\bar{G}_n(s)/a$, and $\bar{L}_n/a$, are (slightly) *biased* estimators of w, $g(s)$, and ℓ, respectively, even if the n days are simulated independently after fixing B.

(d) Implement the approach described in (c) and compare empirically the efficiencies of the estimators $\bar{W}_n/a$, $\bar{G}_n(s)/a$, and $\bar{L}_n/a$, as n increases, for this approach and for the original method, when $\alpha_0 = 0.5$. Try several values of n of your choice and discuss. Note that for large values of n, the bias can be neglected.

(e) Explain how we can replace the elementary integration rule in (c) by Simpson's rule given in Eq. (1.29), using $n+1$ simulation runs. Try it and compare your results with those obtained in (d). Is there a significant improvement? Can you explain why? What if we increase n? Note: When you generate $B = F^{-1}(U_0)$, you may have a problem when $U_0 = 1$. Discuss this difficulty and propose a heuristic solution if you can, for each of the three performance measures.

1.36 Consider a queuing system with two servers and two classes (or types) of customers. Inter-arrival times are i.i.d. exponential random variables with mean 1/2 (time units do not matter). Each arrival is a customer of type A with probability 1/2 and of type B with probability 1/2. Equivalently, customers of each class arrive according to a Poisson process with rate 1 and these two processes are independent. Service times for both classes of customers are exponential with mean 0.8, at each of the two servers. The *customers* may represent calls in a call center, packets of information in a communication network, people at an airline registration desk, jobs in a factory, etc.

(a) If one server is dedicated to each class of customers, what is the average waiting time and the average queue length at each queue, over an infinite time horizon, according to standard queueing formulas?

(b) Suppose now that customers of class A are much more important than those of class B, and that server 1 can only serve customers of class A. We want to serve the class

A customers as soon as possible, while still giving reasonable service to those of class B. There is a FIFO queue for each class of customers. This type of queueing network is called a *N-system* (Gans, Koole, and Mandelbaum 2003).

A simple policy would be to give *full priority* to class A customers: Whenever server 2 becomes free, he will serve a customer of class A if there is one in the queue, otherwise he will serve a customer of class B if one is waiting, and remain idle if both queues are empty. Server 1 serves only customers of class A. An arriving customer of class A will choose server 1 if both servers are free. All customers of class B go to server 2. Once a service is started anywhere, it cannot be interrupted (*no preemption* is allowed).

Write a simulation program for this model and do the following experiment with the proposed policy: Perform 30 independent simulation runs over a time horizon $T = 1000$ and compute 95% confidence intervals for the average queue length for each queue, over that time horizon. Repeat with $T = 10000$ and $T = 100000$. Discuss what happens with queue 2 and explain what is the source of the problem. (Note: the average waiting time w at a given queue can be computed by the classical *Little's formula* $q = \lambda w$, where λ is the arrival rate at that queue and q is the average queue length; see Section A.19.)

(c) To try getting around the problem found in (b), we modify the policy as follows: When server 2 becomes available, it will start serving a customer of class A only if there are at least least ℓ of them in the queue, for some fixed integer $\ell > 0$. For $\ell = 1$, we recover the same policy as in (b). What about the average queue lengths in this case? Repeat the same experiment as in (b) to see what happens with this policy, say for $\ell = 3$ and $\ell = 10$. Discuss your results.

(d) A less aggressive policy for Server 2 would be to start serving a class-A customer only if at least ℓ are in the queue *and* no customer of class B is waiting, i.e., to give priority to class B customers. Repeat the same experiment as in (b) with this policy, for $\ell = 1$. Discuss your results.

(e) Suppose we are allowed to choose between any of the above policies, with any value of ℓ, and want to find one that minimizes the expected waiting time of class A customers under the constraint that the expected waiting time of class B customers does not exceed 5, in the long run (i.e., when $T \to \infty$). What policy would you recommend and with what value of ℓ? (For this exercise, it would be appropriate to use a ranking and selection procedure such as those mentioned in Section 1.15.3, but you do not have to do that at this point; just providing a reasonable answer based on long simulations will suffice.)

1.37 Consider the discretely-observed Asian option of Example 1.11 with the payoff given in (1.10), but suppose that instead of estimating the price $v(s_0, T)$, we want to estimate the *derivative* of $v(s_0, T) = v(s_0, r, \sigma, T)$ with respect to σ,

$$v_\sigma(s_0, r, \sigma, T) = \frac{\partial v(s_0, r, \sigma, T)}{\partial \sigma},$$

which is known as the *vega of the option*. Here we added the parameters r and σ, which were implicit in the notation in Example 1.11. One possible estimator of this derivative is the finite difference:

$$X(\delta) = \frac{Y(s_0, r, \sigma + \delta, T) - Y(s_0, r, \sigma, T)}{\delta},$$

where $Y(s_0, r, \sigma, T) = e^{-rT} g(S(t_1), \ldots, S(t_d))$ is the sample discounted payoff and $\delta > 0$ is a small constant. We could simulate n independent realizations of $X(\delta)$, take the average as our estimator of the vega, and compute a confidence interval.

(a) Explain why this is a biased estimator when $\delta > 0$ is fixed. You can assume that $v(s_0, r, \sigma, T)$ is nonlinear in σ and has a nonzero second derivative with respect to σ. You may use its Taylor expansion around σ. What happens to the bias when $n \to \infty$ while δ is fixed? And when $\delta \to 0$?

(b) To simulate $X(\delta)$, we must perform two simulations of the payoff, one at σ and one at $\sigma + \delta$. These two simulations can be performed with independent random numbers (IRN) or with common random numbers (CRNs). Explain the difference. Explain how you would implement the simulations with CRNs in this particular case. What happens with the variance of $X(\delta)$ when $\delta \to 0$, in each case?

(c) Is it possible to obtain an unbiased estimator of the vega by using the stochastic derivative in this case? If yes, explain how, derive the estimator mathematically, give its explicit formula, and prove that it is unbiased by showing that the dominated convergence theorem applies in this specific case.

1.38 (Bratley, Fox, and Schrage 1987, Problem 1.9.18.) This exercise is about what happens when random variables are replaced by their expectations in an optimization problem. Suppose that the (random) operating cost of a system for one day, under policy π, is

$$C(\pi) = a(\pi) + \sum_{i=1}^{k} b_i(\pi)[f_i(\pi) + E_i] + \sum_{i=1}^{k}\sum_{j=1}^{k} c_{ij}[f_i(\pi) + E_i][f_j(\pi) + E_j],$$

where k and the c_{ij} are constants, $a(\cdot)$, $b_i(\cdot)$ and $f_i(\cdot)$ are functions of π, and the E_i are random variables independent of π.

(a) Show that to estimate $\mathbb{E}[C(\pi)]$, replacing the E_i's by their expectations in the expression for $C(\pi)$ introduces bias (in general). Give an expression for the bias and give a necessary and sufficient condition under which such a replacement introduces no bias.

(b) Explain under what circumstances replacing the E_i's by their expectations improves the efficiency of the estimator of $\mathbb{E}[C(\pi)]$ despite introducing bias, and under what circumstances it also improves the mean square error.

(c) Suppose now that we are interested in estimating $\delta = \mathbb{E}[C(\pi_1)] - \mathbb{E}[C(\pi_2)]$ for two different policies π_1 and π_2. Show that δ can be computed directly by replacing the E_i's by their expectations. To find the π that minimizes $\mathbb{E}[C(\pi)]$, can we replace the E_i's by their expectations without introducing bias on the choice of π?

1.39 (Functional estimation and optimization: A special case). (Adapted from Bratley, Fox, and Schrage 1987, Problem 1.9.7.) Consider a graph with N nodes and M edges, the edges connecting M distinct pairs of nodes (so that $M \le N(N-1)/2$). Two nodes are said to be *connected* if we can go from one to the other by following a sequence of edges. The *minimal* number of edges so that all the nodes are connected is $N - 1$. A set of $N - 1$ edges that connect the N nodes is called a *spanning tree*. If the edges have *lengths* (or costs), a *minimal spanning tree* is a spanning tree whose total length (or cost) is minimal. Given $M \ge N - 1$ edges, the following greedy algorithm constructs a minimal spanning tree in worst-case time

$\mathcal{O}(M \log M)$, provided that one exists (see, e.g., Deo 1974, pp. 277-280). The readers who don't know what is a *priority queue* can think of it as a special type of sorted list, which can be updated efficiently on average when the first entry is removed or an arbitrary entry is added. Priority queues and their properties are studied in all good books on data structures or on algorithmic (e.g., Brassard and Bratley 1988).

Algorithm 3 : Minimal spanning tree

```
Sort the edges by increasing length, in a priority queue L;
Start with the N nodes and no selected edge; j ← 0;
repeat
    Remove the shortest edge e from L;
    if e does not form a loop with the other edges already selected then
        add e to the graph (e is selected) and j ← j + 1
    else
        discard e;
until j = N − 1 or L is empty;
if j = N − 1 then
    the selected edges form a minimal spanning tree
else
    there exists no spanning tree
```

(a) Suppose that each of the M edges works with probability p, independently of the other edges. Explain how to estimate $\mu(p)$, the probability that a spanning tree exists (i.e., that the graph is connected by working edges), by the MC method, for a fixed value of p.

(b) Suppose now that we want to minimize $\alpha(p) = C(p) - \mu(p)$ with respect to p, by simulation, where $C(\cdot)$ is a known (deterministic) increasing function. A naive way of doing this is to estimate $\mu(p)$ over a grid of several values of p, by performing n simulations at each value of p considered, independently. A better way is as follows. For each of the n simulation runs, generate i.i.d. $U(0,1)$ random variables $U_1, \ldots, U_M$, one for each edge, and construct a minimal spanning tree by considering U_j as the length of edge j. For each $p \in [0,1]$, define the Bernoulli random variable $X(p) = \mathbb{I}[L \le p]$, where L is the length of the last edge selected by the algorithm in constructing the tree. Show that $\mathbb{E}[X(p)] = \mu(p)$ for each p.

(c) Let the above scheme be repeated n times, independently. Let $X_i(p)$ and L_i be the values of $X(p)$ and L for replication i, and define

$$\bar{X}_n(p) = \frac{1}{n} \sum_{i=1}^{n} X_i(p).$$

Note that $\bar{X}_n(p)$ can be viewed as a function of p. Show that it is a non-decreasing step function of p, with $\bar{X}_n(p) = 0$ for $p < 0$, $\bar{X}_n(1) = 1$, and with a step of size $1/n$ at each L_i.

(d) The function $\hat{\alpha}$ defined by $\hat{\alpha}(p) = C(p) - \bar{X}_n(p)$ is an estimator of the entire function $\alpha(p)$, $0 \le p \le 1$, and the value of p that minimizes $\hat{\alpha}(p)$, say $\hat{p}$, can be taken as an estimator of p^*, the minimizer of $\alpha(p)$ with respect to p. Give an $\mathcal{O}(\cdot)$ expression for the total time to compute the entire function $\bar{X}_n(\cdot)$ in the worst case, including the time to generate the random numbers, to compute the L_i's from them, and to sort the L_i's (assuming that the latter can be done in time $(n \log n)$).

(e) To compute the exact value of $\hat{p}$ from the function $\hat{\alpha}(\cdot)$, how many values of p must be examined in the worst case? Which ones?

1.40 In Section 1.7, estimate the difference in performance for two slightly different distributions for the Y_j in the SAN example, using CRNs. You will perform a similar experiment with the following two systems. For the first system, assume that all the Y_j have an exponential distribution with the same mean θ_j as in the given numerical example, Example 1.4. For the second system, assume that they all have a normal distribution, again with the same mean θ_j, and standard deviation $\theta_j/4$. Let X_1 be the length of the longest path for the first system and X_2 for the second system. We want to estimate $\mathbb{E}[X_2 - X_1]$. This can be done (a) with IRNs; (b) with CRNs using inversion; (c) with CRNs using inversion for the normals, but $Y_j = -\theta_j \ln(U_j)$ instead of $Y_j = -\theta_j \ln(1 - U_j)$ for the exponential random variables. The latter is formally correct because $1 - U_j$ has the same uniform distribution as U_j. Make an experiment in which you compare the variances obtained with these three methods using $n = 10^4$ simulation runs. Explain what you find.

2. Simulation Modeling

This chapter is about modeling for simulation. Various aspects are covered, including the selection of probability laws that determine the model evolution at all levels, abstraction and simplifying assumptions, validation, and different types of performance measures that we might want to estimate by simulation.

The first three sections contain a general discussion of the main issues in modeling. Guidelines and recommendations on the practice of modeling are provided in the first section. Section 2.2 is on validation and verification. In Section 2.3, we discuss the importance of high-quality data and the impact of simplifying assumptions. In the several sections that follow, we compare different ways of specifying the input distributions and review several classes of standard parametric and semi-empirical distributions. Much of this material (especially the list of distributions) is primarily for reference and could be just browsed over quickly. In Section 2.12, we recall some basic definitions and properties of stochastic processes. The following three sections provide a quick coverage of Poisson processes, Brownian motion, Lévy process, some other related processes, and time-series models. Techniques for fitting distributions to data are briefly discussed in Section 2.18. In Section 2.19 we distinguish models where the performance measure is over a finite and an infinite time horizon, and models with discounting.

2.1 Principles of Simulation Modeling

Abstraction and simplification are the essence of modeling. The appropriate level of simplification depends on what we want to use the model for. Building a valid model is often difficult, especially for the complex dynamic stochastic systems that involve humans. Modeling is an art that requires good judgment and common sense, and it is usually an iterative process. We elaborate a little on these ideas. More extensive discussions on the general principles of modeling for simulation can be found, e.g., in Shannon (1998) and Pritsker (1998).

2.1.1 Purpose of the model

A first important question when building a simulation model is: Why are we building it? The *purpose of the model* and the goal of the project should be established before any further development is engaged. This should be done together with the potential users. The appropriate level of detail or aggregation, and the boundaries of the model, strongly depend on this intended purpose. An ideal model includes no unnecessary detail, but just enough to

provide a realistic representation of the important characteristics of the system, given the purpose of the study. Models that are simpler and easy to understand tend to be more useful because they give better insight into the important aspects of the system's behavior.

2.1.2 System's knowledge

Building a good model requires good knowledge of the system and of the model's purpose. This means getting information from people who know about the system from different viewpoints (e.g., managers, operators, clerks, customers, etc.), examining the real system if possible, and collecting pertinent and reliable data. To decide what to include in the model, one must understand the structure of the system and its operating rules well enough to figure out what simplifying assumptions are reasonable and what is likely to have an impact on the results of interest. Constant interaction with the decision makers or future model users, and with the people who know best about the system, should be maintained all along the stages of the model development. This interaction is crucial for maintaining a high level of confidence in the model and increases the chances that the model be *credible*, satisfies their needs, and be *used*. For a successful simulation project, the client must be ready to commit his time and resources for transferring the appropriate knowledge to the modeler and participating in the modeling process.

A lot can be learned from the process of building a model even if that model is never simulated. Understanding the logic and the interactions and organizing the information together in one place can be worthwhile activities by themselves. It is often said that a clearly formulated problem is half-solved.

2.1.3 Iterative process and flexibility

Simulation model building is an *iterative process* in the sense that new information obtained along the way, or from simulation runs, often leads to changing certain aspects of the model (e.g., assumptions, level of detail, operating policies, input distributions, etc.). In this evolutionary process, simulation provides insight into the system, and this new information can be used for improving the model by successive refinements.

Since a simulation model is frequently modified, refined, extended, and adapted, it is important that both the model and the simulation program be flexible and easy to change. Flexibility means, for example, that input probability distributions can be easily changed or replaced by historical data, that one can easily experiment with different decision making or operating policies in the model, etc. Good practices that make the model easier to change and maintain include building the model from small and relatively self-contained logical components with simple interfaces (*modularity*), maintaining a clear documentation, and separating the model definition from the statistical experiment specification (especially in the program). Even the purpose of the model can change, either during the modeling activity, or after the model has been in use for some time.

2.1.4 No foolproof recipe

Modeling remains an art learned by experience. It requires good human judgment. Discrete-event stochastic systems involved in human activities tend to contain non-trivial logic, non stationary behavior with respect to time, stochastic dependencies that are hard to model, complex operating procedures, etc. These systems have no simple laws such as those found in elementary physics and chemistry, for example.

A typical mistake of inexperienced modelers is to incorporate an excessive amount of detail into the model. One should "look at the forest before looking at the trees"; that is, start with a simple model and add details only as needed while consulting with the system experts, managers, and future users. One should also refrain from having too many model parameters that must be estimated from system's data. Time and money constraints, and availability of data, often determine the level of detail that can be practically achieved. For example, a model of a system that does not yet exist and for which no data is available will be much less detailed that a model made to fine-tune a system already in operation. As another example, when modeling the bottling operations of a large brewery, it is probably not necessary to represent the individual bottles in the model. The process can be modeled as a continuous flow (perhaps with interruptions) or by aggregating the bottles into batches and considering these batches as the smallest representable unit. As a side benefit, the simulation program will run faster.

2.1.5 Implementation

The ultimate success criterion for a simulation study is whether something has been *learned* from the model and if this knowledge has been *used*. A model has more chance of being used and trusted if its future users take part in all stages of its development, and feel to a certain extent that this is also *their* model.

2.2 Model Validation and Program Verification

There is a large set of well-developed software engineering techniques for the design, implementation, and *verification* of computer software in general. These techniques do apply to simulation programs. We refer the reader to standard textbooks on software engineering. Detailed discussions of *validation* principles and techniques can be found in Balci (1998), Kleijnen (1995), Sargent (2001), and in Chapter 5 of Law and Kelton (2000).

The validity of a model is not a "black or white" issue. It depends on the purpose of the model and on the desired level of accuracy in the results. If a given model is to be used to answer several different questions, the validity issue must be addressed separately for each of these questions. One should not forget that validation can be expensive. The more realistic we want the model to be, the more we have to pay for modeling, for data collection, and for validation. In practice, a compromise must be made between modeling and validation costs on the one hand, and the costs associated with the consequences of using an insufficiently realistic model on the other hand. Spending a lot of time and money

on modeling and validation is often necessary, but it is worthwhile only if the simulation can provide knowledge worth that price.

The next section will discuss the issues of simplifying assumptions and approximations, and the choice of input distributions. In the remainder of this section, we survey briefly some validation techniques and ideas that are recommended in the simulation literature. Balci (1998) describes a larger collection of verification, validation, and testing techniques.

2.2.1 Informal validation

Does the model behavior look reasonable to the people knowledgeable about the system? This is the first level of model validation. It should be done *during* model building. It thus requires regular interaction with the client and experts. Experience, intuition, and good tools (e.g., for displaying descriptive statistics and performing graphical animation) can help.

In a *Turing test*, one shows system experts or managers two sets of output data, one from the real system and the other from the model, and asks them to tell which is which. If they succeed, they should explain how they were able to distinguish, and this information can be profitable for improving the model.

A *structured walkthrough* of the model is a recommended procedure to be performed by a team of key people including the developer(s), system experts, and client representative(s). The participants examine documents that describe the model, list all the assumptions, and mention the uncertainties that the modelers may still have in their mind. All aspects of the model are studied during the walkthrough. The goal is to detect the modeling errors, misconceptions, inconsistencies, etc. This should preferably be done before too much effort has been engaged into programming, to avoid costly reprogramming in the case where major modeling errors are found.

2.2.2 Statistical testing of hypotheses

In principle, goodness-of-fit test statistics (see Chapter 5) can be used for detecting significant differences between the model and the real-life system of interest. The null hypothesis $\mathcal{H}_0$ for the statistical test in this situation is that the model and the system are indistinguishable. However, using a formal statistical testing procedure which rejects $\mathcal{H}_0$ when the test statistic exceeds a fixed threshold (Section A.15) is not necessarily appropriate, because we know in advance that there is a difference between the model and the system. In most cases, the null hypothesis will eventually be rejected if we take the sample size large enough to make the difference significant. What we really want to test is not that there is no difference, but that the difference is within the *acceptable limits of precision given the purpose of the model*. This must be kept in mind when applying goodness-of-fit tests for validation. Informal examinations of statistical data and graphics often suffices in this context.

Example 2.1 In Example 1.8, for instance, a chi-square test of hypothesis would certainly conclude that there is a significant difference between the three distributions in Table 1.1 or 1.2. The sample size is large enough to clearly see the difference. Nevertheless, the Poisson

approximation would be good enough in many situations, for example to approximate the expected number of collisions. □

Another frequently encountered limitation is the absence of sufficient data from the system to apply a meaningful statistical test that compares the model with the system. For example, we may have observed the real system only for a period of time that corresponds to a single simulation run. Often, the system of interest does not even exist. In this case, if a different but similar system does exist, building and validating a simulation model of the existing system can bring a significant amount of knowledge, which can be profitable for constructing a valid model of the system of interest.

Goodness-of-fit testing is nevertheless useful for validating the choices of input distributions in the model, and for testing certain assumptions such as time-stationarity, independence, equality between two distributions, and so on.

When comparing an existing system and a model for validation, it is a good idea to run the model with the historical input data of the system, and see if the outputs correspond within acceptable limits to the historical output data. Using the historical inputs reduces the noise in the differences between the outputs because both the model and the system are then fed with exactly the same values of the input random variables. This approach is similar to the CRN methodology used for comparing two similar systems (Section 1.7) To test the validity of the models for the input distributions and processes, one can compare the output data of the model fed with the historical input to that when the input model is used.

2.2.3 Sensitivity Analysis and Robustness

Sensitivity analysis identifies which parameters, input variables, and assumptions of the model have a significant effect on the performance measure of interest. This can be achieved by varying the values of these parameters and input variables over some range and observing the corresponding changes in the output (the performance measure's estimator). If the model's response is sensitive to certain input values (e.g., parameters of some input distribution) or to certain assumptions (e.g., independence of certain random variables, stationarity of an arrival process, type of an input distribution), or to the level of detail or aggregation in a given part of the model, then we may want to spend additional effort to make sure that these input values are accurate enough, or that these assumptions are justified, or to keep more detail in the model. For example, it may be important to look at what happens to the average waiting time in a queue if the service time distribution changes slightly, or if the interarrival times become slightly correlated. A model is said to be *robust* with respect to certain parameters or assumptions if changing slightly these parameters or these assumptions has little effect on the responses of interest.

Advanced techniques for sensitivity analysis are discussed in Section 1.8 and in Chapter 7. Among these, the statistical experimental design techniques and the response surface methodology are appropriate when performing sensitivity analysis with respect to several factors at the same time. It is important in certain situations to consider not only the effect of each factor individually, but also the effects due to the *interaction* between the factors.

♣ Perhaps add an example here.

Sensitivity analysis often involves estimating the difference between the performance measures of two (or more) very similar models. In this situation, the estimator of the difference is typically much less noisy when CRNs are employed across the two models rather than simulating them independently (Sections 1.7 and 6.4). This means simulating the two models with the same streams of random numbers, with proper synchronization to make sure that the same random numbers are used for the same purpose for the two models, as much as possible. In the case where the simulation uses historical data instead of generating random variables, this mean feeding the two models with the same stream of historical data, as mentioned previously. Sensitivity analysis with respect to marginal changes in continuous (real-valued) parameters amounts to estimating the derivative or gradient of the performance measure(s) with respect to these parameters (Section 1.8 and in Chapter 7).

A valid model is sometimes valid only over a given region for the values of its parameters. One must make sure that this region covers the range of values of interest for the parameters, i.e., the range in which the model is likely to be used. How fast does the invalidity grows when we leave the validity region? How can we change the model to enlarge or move the validity region? These questions also relate to some form of sensitivity analysis.

Example 2.2 In the call center example of Section 1.12, a model that neglects the abandonments (i.e., in which assume that all customers have infinite patience) might be valid if very few customers abandon (perhaps because long waits are rare), and if we want to estimate a quantity that is not much affected by these rare abandonments. This is often the case in emergency call centers for police, fires, and ambulances, for example. On the other hand, the model could no longer be valid if there is more traffic and the waiting times get longer, or if we want to estimate the abandonment ratio ℓ in the first place even if it is small. Similarly, a model that does not account for variations in the busyness B (i.e., assumes $B \equiv 1$) might be valid in times when B has low variance, but invalid when B varies more.

♣ Add numerical illustrations here. □

Example 2.3 An $M/G/1$ queue is a special case of the $GI/G/1$ single-server queuing model where the service times are i.i.d. with an arbitrary distribution and the interarrival times are i.i.d. exponential, say with parameter λ. A classical result of queuing theory is the *Pollaczek-Khintchine mean value formula* (see Section A.19); it says that in steady-state (or over an infinite time horizon), the average waiting time per customer and the average queue length in an $M/G/1$ queue, w and q, depend only on the mean ν and variance σ^2 of the service time distribution:

$$w = \frac{\lambda(\sigma^2 + \nu^2)}{2(1-\rho)} \quad \text{and} \quad q = \lambda w, \tag{2.1}$$

where $\rho = \lambda\nu$ is the utilization factor. These equations imply that if we pick the wrong service time distribution but match the first two moments (mean and variance) correctly, the *average* queue length and waiting time will still be correct. However, other quantities (e.g., the variance of the waiting times, or the proportion of waiting times that exceed 20 seconds) may be affected. Moreover, for a general $GI/G/1$ queue (i.e., if the arrival process is not a stationary Poisson process), Eq. (2.1) no longer holds.

For exponential service times (i.e., an $M/M/1$ queue) we have $\sigma^2 = \nu^2$ and Eq. (2.1) simplifies to $q = \rho^2/[\lambda(1-\rho)]$. This means that if we wrongly assume exponential service times in an $M/G/1$ model, the error on the average queue length and average waiting time in our model will be large if and only if the coefficient of variation of the service times differs significantly from 1. □

2.2.4 Operational validation

Calibrating a model means adjusting its parameters, level of detail, and sometimes its assumptions, so that its output is close to a given target when it is fed with certain inputs. This target is usually determined by some historical data. In other words, we want the model to reproduce reasonably well what happened in the past. After its calibration, the model must be *validated* with a data set that is *independent* from the data set used for calibration. Assuming that enough data is available, one should use part of the data for developing and calibrating the model, and reserve another (independent) part of the data for validating the model.

After a model has been built and when the corresponding system has been in operation for some time, additional data becomes available. One can use this fresh data every now and then for model validation. This *prospective validation* usually suggests adjustments to the model (updated calibration). Good agreement between the model's prediction and the new data improves confidence in the model, whereas disagreements may indicate how to improve the model.

Validation can be performed separately for individual components or submodels that comprise the model, then for larger groups of components together, hierarchically. It is important to understand that validity of all the components does not necessarily imply validity of the model as a whole, because the acceptable errors in the responses of the submodels may accumulate to unacceptable errors in the output of the complete model, and because there could be errors in modeling the interactions between the components.

Credibility of a model refers to its acceptance and trust by the managers or clients. Credibility is an important ingredient for success, because it determines to a large extent whether the model is going to be used or not. Sometimes, certain details in a model that can be neglected from the validity viewpoint must be incorporated because of credibility issues. This is relevant in particular for simulation models with graphical animation.

2.3 Data, Assumptions, and Robustness

Building a model requires *information* about the system. Building a detailed and reliable model requires not only a lot of data, but *relevant* and *correct* data. This may seem like a trivial statement, but situations abound where either too little data is available, or the data on hand concerns only a related system. Data is needed not only to fit distributions, but also to test (validate) the several assumptions that are made in the model, such as independence, stationarity, etc. One might think that so much data is collected by computerized systems nowadays that lack of relevant data should no longer be a problem. But in fact, it is still often a problem.

2.3.1 Quality of the Data

Too little data could mean:

1. A *too small* sample.
2. *Aggregated data*, i.e., only summary statistics such as the mean, maximum and minimum, median, etc., or only totals over certain periods. For example, one may have the number of failures of a machine each week, but not the failure times. For a telephone call center, one may have the number of incoming call and the average service times for each half-hour period, but not the arrival times and service times of the individual calls.
3. *Low accuracy*. For example, suppose that the time for a machine to perform a certain operation is almost always between 4 and 7 minutes and that we have the operation times only in minutes (an integer).
4. *Subjective information* from interviews of people who are supposed to know the system.

The following are examples of situations where the data comes from a related but different system.

1. *Wrong period*. We have data for up to 2 months ago, but the model is build to predict what will happen next year.
2. *Wrong place*. We only have data on the demand for a product in the Toronto area and we want to build a stochastic model for the demand in the Montreal area.
3. *Censored distribution*. We have data on the sales instead of the demand (no data was kept on the sales lost because of stockouts). As another example, suppose i.i.d. components have been replaced at failure time or at age T, whichever comes first. If we have data on the replacement times, we know all the lifetimes that were less than T, and we know *how many* were more than T but not their values. Similar censoring occurs if we want to estimate the distribution of the time X until abandonment for impatient customers in a queue (e.g., waiting phone calls in a customer service center). In that case, we may be able to measure X for each lost customer, but for a customer whose call was answered after waiting T units of time, we only know that $X > T$. (Here, both X and T are random variables.)

2.3.2 Privacy Issues and Artificial Data

Data availability is often limited because of privacy issues. This occurs in particular for data on specific individuals or on firms (e.g., medical records, census or financial records, etc.). Typically, the information in this type of data must be reduced or modified so that one cannot easily identify a specific individual or firm by making requests to the data base. It has been shown that such privacy cannot be preserved while keeping the data intact; one must incorporate some amount of noise to the data. The concept of *differential privacy* was introduced by cryptologists and turned into a formal framework and methodology to address

this issue (Dwork and Roth 2014, Dwork et al. 2017). The goal is to modify the data set in a way that it retains its important statistical properties while giving no significant information on whether or not a given individual was included in the data set before the modification. Then, queries to the modified data should not compromise the privacy of anyone. It turns out that the required amount of noise (or its variance) depends on the size of the database and is larger when the number of individuals is smaller. Dwork et al. (2017) provide tools to design and combine algorithms that can make a data set differentially private by adding appropriate noise. They consider in particular adding noise from the Laplace distribution. These types of tools are now used by large data collectors such as Google, Microsoft, and the US Census Bureau.

Adding noise to data inevitably changes its distribution and dependence structure. Increasing the noise provides more privacy but reduces the utility of the data set by altering its distribution even more. So in each case, a compromise must be made between better privacy and better social utility.

Instead of just adding noise to the actual data, one may also build a model from which an arbitrary amount of purely artificial data with all the relevant characteristic of the real data set can be generated, while not representing any of the real individuals. Doing this while capturing all the important dependencies is even more challenging than just adding noise to the data. Since the artificial data will be produced via some type of model, this process can in fact be viewed as a modeling step. It is discussed extensively in the literature under the name of *population synthesis* (Müller and Axhausen 2011, Sun and Erath 2015, Le et al. 2016, Sun, Erath, and Cai 2018, Yin et al. 2018, Chapuis, Taillandier, and Drogoul 2022).

2.3.3 Simplifying Assumptions

Various types of approximations are made in modeling input distributions.

1. *Simplified distributions.* Complicated and unknown real-world probability distributions are approximated by simpler and nicer ones, such as the exponential, the normal, and the uniform. Often, random variables are replaced by constants (e.g., their expectations). This is frequent in optimization contexts. If an estimator X is a *linear* function of a random variable Y, replacing Y by its expectation in the model does not change $\mathbb{E}[X]$ and reduces its variance. This is no longer true if the relation is nonlinear, and the difference can be quite significant, as we saw in Example 1.4. Exercise 1.38 gives an exception.

2. *Aggregation.* This happens when several types of objects are considered as a single class. For example, different classes of customers are aggregated into a single class and are considered to have the same stochastic behavior (same service time distribution, etc.). Or people of different sexes, ages, heights, habits, etc., are aggregated into a small number of categories, and individuals from the same category are considered stochastically identical. Another type is time aggregation; e.g., events occurring during the same minute are assumed to have occurred at the same time.

3. *Independence.* Assuming that certain random variables in the model are independent (in the statistical sense) is commonplace, despite the fact that this rarely holds in

the real world. For example, successive service times at a given station in a queuing system are often assumed independent because modeling the dependence appears too complicated, or there is not enough data to do it. This can be an important source of gross error. One of the main reasons for modelers to assume independence is that commercial software products for building discrete-event simulation models provide easy-to-use tools almost exclusively for the case of i.i.d. random variables. A more serious reason is the difficulty of modeling the dependencies properly.

4. *Stationarity.* A stationary model is one whose logic, input distributions, and parameters, do not change with time. Such a model is easier to handle than a non-stationary one. Typically, this is reasonable if the system of interest is simulated over a relatively short fraction of its lifetime. Most systems involving humans (social, economic, entertainment, etc.) are highly non-stationary. Think for example of the arrival processes to a restaurant, a subway station, or an airport. The arrival rates certainly depend on time. Results of simulations based on stationary models in these types of situations (and in most other situations, for that matter) must be interpreted with a healthy dose of skepticism. Stationarity assumptions are usually needed to obtain analytical formulas. This is an important reason for their popularity. For simulation, stationarity is not needed. In fact, output analysis can be *easier* for finite-horizon non-stationary models than for steady-state stationary models.

2.4 Classes of Approaches for Choosing the Probability Laws

The main classes of approaches for modeling the inputs in stochastic models are

A. *Parametric*. Select a parameterized family of *theoretical* distributions and estimate the parameters from the data. For example, one can assume that the call durations (service times) at a call center have the gamma distribution and then estimate the two parameters of that distribution from the available data. Multivariate parametric distributions can model random vectors with dependent components.

B. *Semi-empirical or non-parametric*. Construct a variant of the empirical cdf of the data and use it to generate the random variates in the simulation. By *variant*, we mean that the empirical distribution function is often smoothened in some way, and that an infinite tail is often added so that arbitrarily large observations can be generated. One can also directly construct a smooth estimate of the inverse cdf F^{-1}, to facilitate random variate generation. Another (often better) approach is to construct a *density estimate* from the data (Section 2.9.4), instead of the cdf or its inverse, and use it to generate the input random variates.

C. *Trace-driven.* In its pure form, this approach simply reuses the historical data of the system of interest as input. For example, to compare a new inventory management policy with the one that was in place at the time the data was collected, one replays the history for the demands, but with the new policy, to see the difference. A simulation can be *partially* trace-driven. For example, the arrival process of people at the emergency room of an hospital can be modeled as a Poisson process, and whenever an arrival

occurs, a patient's record is picked randomly from historical files on actual visits. Then, the detailed characteristics of the patients (and the complicated correlations between those characteristics) do not have to be modeled. An important limitation of this approach is that no patient record that differs from those in the historical files will ever be generated. This may be acceptable in certain situations, e.g., if the historical files are huge. Otherwise, to properly address this limitation, one must model the detailed characteristics as random *vectors* with dependent components. This can be hard. A compromise could be to add some noise to the historical records.

Some authors recommend A and argue that there is rarely enough data to adopt C (e.g., Law and Kelton 2000). Others prefer B to A (e.g., Bratley, Fox, and Schrage 1987). There are frequent situations where C (at least partially) may be the most reasonable approach, because of the difficulties of modeling complicated dependencies in a large number of dimensions. Research on the modeling of data by high-dimensional distributions is very active, especially in the fields of computational statistics and machine learning.

Example 2.4 Years ago, the author visited a large aircraft maintenance facility employing nearly 20,000 people. A huge job shop in the facility contained a network of workstations of several kinds. Parts arrived randomly to the shop (sometimes in batches) to be repaired or rebuilt. There were thousands of different types of parts and each part arriving to the shop had a list of operations to be performed on it, on specific machine types. This list and the times of the operations depended not only on the part type but also (strongly) on the damage that the part had, and were highly random. This involved complicated correlations, that were very hard to model, between the durations of the successive operations performed on a given part. The managers of the facility wanted to build a simulation model of the shop to study how they could reduce the sojourn times of the parts in the shop. The average sojourn time of the parts was more than 25 times the average total working time to perform the operations on the parts. Parts were thus spending most of their time waiting or traveling between the workstations across the shop. After unsuccessful attempts to build a valid model that would generate the transitions of the parts between the workstations using probabilities, and the operation times using probability distributions, it was decided to use partial trace-driven simulation, similar to the emergency room example given in Item C above. That is, instead of trying to model the dependencies between the times of successive operations (by different machines) on a given part, records of part histories were drawn at random from the database to feed the simulation. □

The following arguments support the parametric approach A with a theoretical distribution.

1. This approach is appropriate if we have physical reasons to believe that a particular distribution is *natural* and *expected*. For example, the normal distribution if the random variable X of interest is the average of a large number of independent small effects, the Poisson distribution if it counts independent rare events, etc.

2. Parametric distributions with a small number of parameters have the advantage of being a compact way of representing data, without discarding too much information, when they are appropriate.

3. Procedures for generating random variables from standard theoretical distributions are available in most simulation programming environments and just using them means less programming than building quasi-empirical distribution or sampling records from an historical database.

4. The values that can be generated are not limited to the span of the data sample. With theoretical distributions that have infinite tails, it is possible to generate *large deviations*, which is important for certain applications.

The following are arguments against A.

1. It is often very difficult, sometimes impossible, to know the right distribution family. In some cases, none of the available distributions fits the data correctly.

2. When fitting a distribution with few parameters, information is lost or distorted, and this is not always negligible. The estimation of the parameters is not always robust.

3. For certain types of distributions, random variate generation is difficult or slow (see Chapter 4).

When fitting a distribution to data, one can always improve the fit by selecting a more flexible class of distributions (e.g., by adding more parameters or adopting a semi-empirical distribution). In fact, with the same number of parameters as the number of observations in the data, the empirical distribution itself gives a perfect fit to the data! This is an extreme example of *overfitting*, generally defined as an excessive adjustment to the noise in the data set. A yet more extreme form of overfitting is getting the model inputs by replaying history, as in trace-driven simulation. Overfitting is bad because it gives a distribution less likely to reproduce the general behavior of additional data obtained from the (unknown) real-life distribution that we are trying to model. That is, it tends to give more *generalization error*. Two ways of avoiding overfitting are: (a) to be more parsimonious in the number of parameters and (b) to impose more restrictive smoothness conditions on the cdf or on the density.

2.5 Input Modeling With Little or No Data

The idea of selecting input distributions without objective data to support the choice may sound like a joke, but there are frequent situations where a simulation model must be developed without any data from the system. This could be because the system does not yet exist, or for other reasons. Input distributions can then be selected in a number of ways:

- *Human opinion*. Experts on the process of interest (or a similar process) can often provide a reasonable subjective distribution. A popular approach is to ask the experts for the most frequent value m, minimal value a, and maximal value b, and to fit a triangular distribution with parameters (a, b, m) (Section 2.8.2). This approach can be criticized in several ways (Law and Kelton 2000) but it is simple. Other approaches are to fit a normal (ask for the mean and standard deviation, and perhaps truncate the tails), fit a beta or Johnson distribution (Section 2.8.23), or fit a Bézier distribution (Section 2.9.3).

- *Manufacturer specifications*. Vendors normally provide performance specifications for their products. These specifications can sometimes be used to build parts of the model. For example, a machine can fill up so many beer cans per minute, or a communication device can transmit so many bits per second, etc.

- *Physical constraints*. Due to certain physical limitations, there may be upper and lower bounds on the values that a random variable can take.

One can argue that any results obtained from a simulation model built without objective data should be taken with a large grain of salt. However, this is often better than nothing. In this context of high uncertainty, sensitivity analysis with respect to the unreliable distributions (both their types and their parameters) should be given special attention.

2.6 Parametric Distributions

The most extensive coverage of parametric distributions is the four-volume encyclopedia of Johnson and Kotz (1972a), partially revised in Johnson, Kotz, and Balakrishnan (1995). A limited coverage is given here for convenience. A distribution is called *univariate* if it is for an integer or real-valued random variable and *multivariate* if it is for a random vector. Most univariate distributions have from one to four parameters. The more parameters they have, the more flexible they are (in general). But too many parameters is bad because it leads to overfitting.

2.6.1 Types of parameters

Parameters are often classified in three types: location, scale, and shape. A *location* parameter determines where is the origin on the horizontal axis. Changing it slides the distribution (or the axis) horizontally without modifying the scale or the shape of the distribution. A *scale* parameter changes the scale on the horizontal axis (by stretching or contracting the axis) without changing the shape of the distribution. Usually, doubling the scale parameter multiplies all values by two. Sometimes, the scale is specified by a *rate* parameter, which is the multiplicative inverse of the scale parameter. A *shape* parameter has a more profound effect: It changes the shape of the distribution in a non-trivial way. An affine transformation of a random variable X, of the form $Y = aX + b$, changes the scale and location of the distribution but not its shape. In this case, if F_{x} and F_{y} are the cdf's of X and Y, we have $F_{\mathrm{y}}(y) = F_{\mathrm{x}}((y-b)/a)$. If we have a method to generate X, and Y has the same distribution as X except for a change of scale and location, then we can easily generate Y via $Y = aX+b$.

Example 2.5 For the normal distribution, the mean μ is a location parameter and the standard deviation σ is a scale parameter. The shape of the normal distribution is always the same. For the exponential distribution, the mean is a scale parameter. The Weibull distribution has three parameters, one of each type. Some distributions have multiple shape parameters. □

A univariate distribution for a random variable X can be described via standard characteristics such as the *mean* $\mu = \mathbb{E}[X]$ the *variance* $\sigma^2 = \mathbb{E}[(X-\mu)^2]$ (or the *standard deviation* σ), the *coefficient of variation* σ/μ, the *skewness* coefficient ν, and the *kurtosis* coefficient κ (see Section A.5). Certain classes of distributions (e.g., the Johnson family, see Section 2.8.23) have enough flexibility that any combination of values of μ, $\sigma > 0$, ν, and $\kappa > 0$ can be obtained by an appropriate choice of parameters.

Additional properties can often be recovered from the *moment generating function* (mgf) $M_X(\theta) = \mathbb{E}\left[e^{\theta X}\right]$ for $\theta \in \mathbb{R}$, when it exists.

When a continuous random variable X models the lifetime of a system or component, or the time until a given event, it is often convenient to think in terms of the *hazard rate* (or *failure rate*) of X, defined as $r(x) = f(x)/(1-F(x))$ where f and F are the density and cdf of X (Section A.4). If X denotes the time of an arrival, then $r(x)$ represents the arrival rate after x time units have elapsed, if the arrival has not yet occurred. Distributions with *increasing* [*decreasing*] failure rate are used to model lifetimes of components that become more [less] failure-prone with age. The time until a first bug is found in a software component, for example, typically has a decreasing failure rate. Many types of components have *bathtub-shaped* failure rates: high and decreasing for new components (because of potential manufacturing or installation defects), small for a while, then increasing because of deterioration due to aging.

2.6.2 Choosing a distribution and estimating the parameters

With the parametric approach, one must first select an appropriate generic distribution for the situation at hand. This can be based on *a priori* information and physical justifications. When nothing seems to justify a particular distribution, it might be more appropriate to fit a distribution from a flexible family such as the Johnson or Bézier, for example, or use a quasi-empirical distribution or density estimation techniques.

When a distribution is chosen, one must estimate its parameters from the data. Several general classes of approaches for constructing estimators are described in statistical textbooks. The most prominent approach is the maximum likelihood method, described in Section 2.18. Least-square and moment matching methods are more convenient in some situations.

After the parameters have been estimated, one should measure the goodness of fit between the selected parametric distribution and the empirical distribution of the data. This can be done by visual procedures and formal statistical tests of hypotheses (Section 2.18).

In the next two sections, we describe a number of standard discrete and continuous univariate distributions, giving some of their main properties, their physical justification or interpretation when there is one and, in some cases, specific methods to estimate their parameters or to generate random variates. When we give the density or probability mass for a range of values, it is implicitly assumed to be zero outside that range.

The next two sections are intended to be just looked over quickly and then used for reference when needed.

2.7 Some Discrete Distributions

2.7.1 Discrete uniform distribution

A random variable X has the *discrete uniform distribution* over the range $[i, j]$, denoted $X \sim \mathsf{DiscreteUniform}[i, j]$, if

$$\mathbb{P}[X = x] = 1/(j - i + 1) \qquad \text{for } x = i, i + 1, \ldots, j.$$

The mean and variance are $\mathbb{E}[X] = (i + j)/2$ and $\mathrm{Var}[X] = [(j - i + 1)^2 - 1]/12$.

2.7.2 Bernoulli distribution

A random variable X has the *Bernoulli distribution* with parameter p, for $0 < p < 1$, if $X = 1$ with probability p and $X = 0$ with probability $1 - p$. We denote $X \sim \mathsf{Bernoulli}(p)$. The mean and variance of X are $\mathbb{E}[X] = p$ and $\mathrm{Var}[X] = p(1-p)$. Bernoulli random variables often occur as indicators of events. That is, if $X = \mathbb{I}[A]$, the indicator function of an event A, then $X \sim \mathsf{Bernoulli}(p)$ where p is the probability that event A occurs. A $\mathsf{Bernoulli}(p)$ random variable X can be interpreted as the outcome of a random experiment, called a *Bernoulli trial*, in which $X = 1$ is interpreted as a "success" and $X = 0$ as a failure. The Bernoulli distribution is a special case of the binomial distribution (for $n = 1$) and is also related to the negative binomial and the geometric distributions.

2.7.3 Binomial distribution

A random variable X has the *binomial distribution* with parameters (n, p), denoted $X \sim \mathsf{Binomial}(n, p)$, if it can be written as $X = X_1 + \cdots + X_n$, where $X_1, \ldots, X_n$ are i.i.d. $\mathsf{Bernoulli}(p)$ random variables. It counts the number of successes in n independent Bernoulli trials. One must have $0 < p < 1$ and n must be a positive integer. The probability mass function of X is

$$\mathbb{P}[X = x] = \binom{n}{x} p^x (1 - p)^{n-x} \qquad \text{for } x = 0, \ldots, n. \tag{2.2}$$

The mean and variance are $\mathbb{E}[X] = np$ and $\mathrm{Var}[X] = np(1 - p)$. The Bernoulli is a special case of the binomial, with $n = 1$. A sum of independent binomials with the same p is also a binomial (Exercise 2.8).

2.7.4 Geometric distribution

A random variable X has the *geometric distribution* with parameter p, denoted $X \sim \mathsf{Geometric}(p)$, if X can be interpreted as the number of successive failures before the first success in a sequence of independent Bernoulli trials with parameter p, for $0 < p < 1$. That is, $X + 1 = \min\{i : X_i = 1\}$. The probability mass function is $\mathbb{P}[X = x] = p(1 - p)^x$ for $x = 0, 1, \ldots$. The cdf is

$$F(x) = \sum_{y=0}^{\lfloor x \rfloor} \mathbb{P}[X = x] = 1 - (1-p)^{1+\lfloor x \rfloor} \qquad \text{for } x \geq 0.$$

The mean and variance are $\mathbb{E}[X] = (1-p)/p$ and $\mathrm{Var}[X] = (1-p)/p^2$. The geometric distribution is a special case of the negative binomial (with $n = 1$) and is the discrete analog of the exponential distribution. It is the only discrete distribution that has the *memoryless property*: In a sequence of i.i.d. Bernoulli trials, if no success was obtained in the first y trials, then the number of *additional* failures to be observed before the first success is still a geometric random variable, whose distribution does not depend on y. That is, $\mathbb{P}[X = y + x \mid X \geq y] = \mathbb{P}[X = x]$ (Exercise 2.9).

2.7.5 Negative binomial distribution

A random variable X has the *negative binomial distribution* with real-valued parameters (n, p), where $n > 0$ and $0 < p < 1$, if its probability mass function is

$$\mathbb{P}[X = x] = \frac{\Gamma(n+x)}{\Gamma(n)x!} p^n (1-p)^x \qquad \text{for } x = 0, 1, \ldots,$$

where the Γ function is defined in (5.35). We denote $X \sim \mathsf{NegativeBinomial}(n, p)$. The mean and variance are $\mathbb{E}[X] = n(1-p)/p$ and $\mathrm{Var}[X] = n(1-p)/p^2$.

When n is an integer, we have $\Gamma(n+x) = (n+x-1)!$, and X can be interpreted as the number of failures before obtaining n successes in a sequence of independent Bernoulli trials with parameter p. That is, $X + n = \min\{i : \sum_{j=1}^{i} X_j = n\}$. One can also interpret X as a sum of n i.i.d. $\mathsf{Geometric}(p)$ random variables. This distribution is sometimes called the *Pascal distribution* for n integer and the *Polyá distribution* for general n.

2.7.6 Hypergeometric distribution

A random variable X has the *hypergeometric distribution*, with integer-valued parameters (r, b, n) such that $0 < n \leq b$, if

$$\mathbb{P}[X = x] = \frac{\binom{r}{x}\binom{b-r}{n-x}}{\binom{b}{n}} \qquad \text{for } x = \max(0, n+r-b), \ldots, \min(n, r). \tag{2.3}$$

This X represents the number of red balls selected if we draw n balls at random, without replacement, from an urn that contains b balls, r of which are red. We have $\mathbb{E}[X] = np$ and $\mathrm{Var}[X] = np(1-p)(b-n)/(b-1)$, where $p = r/b$.

2.7.7 Poisson distribution

A random variable X has the *Poisson distribution* with real-valued parameter $\lambda > 0$, denoted $X \sim \mathsf{Poisson}(\lambda)$, if

$$\mathbb{P}[X = x] = \frac{\lambda^x e^{-\lambda}}{x!} \qquad \text{for } x = 0, 1, 2, \ldots. \tag{2.4}$$

The mean and the variance are $\mathbb{E}[X] = \mathrm{Var}[X] = \lambda$. Sums of independent Poisson random variables are Poisson: If $X_1, \ldots, X_q$ are independent and $X_i \sim \mathsf{Poisson}(\lambda_i)$, then $X = X_1 + \cdots + X_q \sim \mathsf{Poisson}(\lambda)$ where $\lambda = \lambda_1 + \cdots + \lambda_q$ (Exercise 2.10). Links between the Poisson and exponential distributions are explained in Section 2.13.

The Poisson distribution is very important and natural, because random variables that count rare events, and sums of such random variables when they are independent, have approximately the Poisson distribution. It is well known that a $\mathsf{Binomial}(n, p)$ random variable with small p and large n is approximately a Poisson random variable with parameter $\lambda = np$. We state a more general and more precise result, whose proof can be found in Taylor and Karlin (1998), Section V. [1]

Theorem 2.1 *Let $X_1, \ldots, X_n$ be independent random variables, where $X_j \sim \mathsf{Bernoulli}(p_j)$. Let $X = X_1 + \cdots + X_n$ and $\lambda = p_1 + \cdots + p_n$. Then, for $x = 0, 1, 2, \ldots$, one has*

$$\left| \mathbb{P}[X = x] - \frac{\lambda^x e^{-\lambda}}{x!} \right| \le \sum_{j=1}^{n} p_j^2.$$

This theorem gives an explicit bound on the difference between the probability function (or mass function) of X and that of the Poisson distribution (note that this difference differs from the total variation distance). If the p_j's are all equal to p (say), then $X \sim \mathsf{Binomial}(n, p)$ and the bound becomes $np^2 = \lambda^2/n$, so the probability function of X converges to the Poisson probability function at the worst-case rate $O(1/n)$ if $\lambda = np$ is fixed while $n \to \infty$.

For example, X can be the number of defective items in a large batch, assuming that item j is defective with probability p_j and that the items are independent, or X can be the number of people going to a given store on a given day, or the number of people buying a certain item in the store on that day, assuming that the jth person (in the entire world) goes to that store on that day with probability p_j, independently of the others.

2.7.8 Zipf distribution

The *Zipf distribution* has probability mass function defined by $\mathbb{P}[X = x] = x^{-\alpha}/\zeta(\alpha)$ for $x = 1, 2, \ldots$, where $\alpha > 1$ and the normalization constant $\zeta(\alpha) = \sum_{x=1}^{\infty} x^{-\alpha}$ turns out to be the *Riemann zeta function* evaluated at α. It provides a discrete distribution with a heavy tail. For variate generation, see Hörmann and Derflinger (1996).

2.7.9 Multinomial distribution

The *multinomial distribution* is a d-dimensional generalization of the binomial distribution. A random vector $(X_1, \ldots, X_d)$ has the multinomial distribution with parameters $(n, p_1, \ldots, p_d)$, where n is a positive integer, $p_j > 0$ for each j, and $p_1 + \cdots + p_d = 1$, if

$$\mathbb{P}[X_1 = x_1, \ldots, X_d = x_d] = \frac{n!}{x_1! \cdots x_d!} p_1^{x_1} \cdots p_d^{x_d} \tag{2.5}$$

[1] From Pierre: A bound on the total variation distance might be more useful.

for $x_j \in \{0, \ldots, n\}$ for each j and $x_1 + \cdots + x_d = n$. It represents the result of an experiment comprised of n independent trials, where the outcome of each trial is j with probability p_j for $j = 1, \ldots, d$, and where X_j is the number of times the outcome j was observed. The marginal distribution of each X_j is binomial with parameters (n, p_j).

2.7.10 Negative multinomial distribution

The *negative multinomial distribution* is a d-dimensional generalization of the negative binomial. Its probability mass function is

$$\mathbb{P}[X_1 = x_1, \ldots, X_d = x_d] \;=\; \frac{\Gamma(n + x_1 + \cdots + x_d)}{\Gamma(n)} p_0^n \prod_{j=1}^{d} \frac{p_j^{x_j}}{x_j!} \tag{2.6}$$

for $(x_1, \ldots, x_d) \in \{0, 1, 2, \ldots\}^d$, where $(n, p_1, \ldots, p_d)$ are positive real numbers such that $p_1 + \cdots + p_d < 1$, and $p_0 = 1 - p_1 - \cdots - p_d$.

When n is a positive integer, we can think of a sequence of independent trials, where the outcome of each trial is j with probability p_j, for $j = 0, \ldots, d$. Then, each X_j represents the number of occurrences of outcome j before n occurrences of outcome 0 have been observed.

2.8 Continuous Univariate Distributions

2.8.1 The uniform distribution

A random variable X has the *uniform distribution* over the real interval (a, b), denoted $X \sim U(a, b)$ or $X \sim \mathsf{Uniform}(a, b)$, if its density is $f(x) = 1/(b - a)$ for $a < x < b$. The cdf of X is

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

The mean and variance are $\mathbb{E}[X] = (b+a)/2$ and $\mathrm{Var}[X] = (b-a)^2/12$. The mgf is $M_X(\theta) = (\mathrm{e}^{b\theta} - \mathrm{e}^{a\theta})/((b - a)\theta)$.

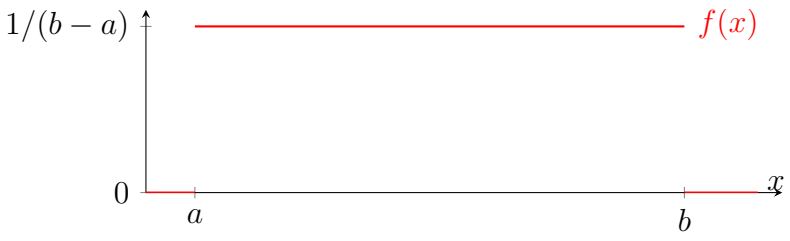


Fig. 2.1. The uniform density

2.8.2 The triangular distribution

A random variable X has the *triangular distribution* with parameters (a, b, m), where $a < m < b$, if its density is (see Figure 2.2)

$$f(x) = \begin{cases} \dfrac{2(x-a)}{(b-a)(m-a)} & \text{if } a \le x \le m, \\ \dfrac{2(b-x)}{(b-a)(b-m)} & \text{if } m \le x \le b. \end{cases}$$

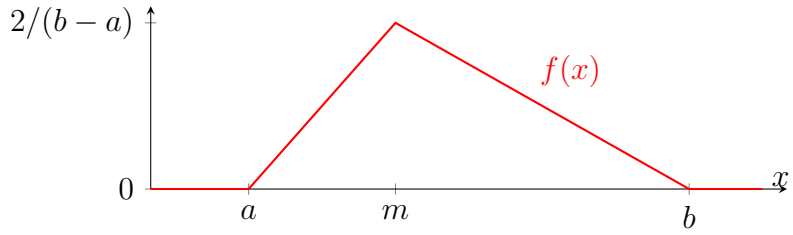


Fig. 2.2. The triangular density

The corresponding cdf is

$$F(x) = \begin{cases} 0 & \text{if } x \le a, \\ \dfrac{(x-a)^2}{(b-a)(m-a)} & \text{if } a \le x \le m, \\ 1 - \dfrac{(b-x)^2}{(b-a)(b-m)} & \text{if } m \le x \le b, \\ 1 & \text{if } x \ge b. \end{cases}$$

The mean and variance are $\mathbb{E}[X] = (a+b+m)/3$ and $\mathrm{Var}[X] = (a^2+b^2+m^2-ab-am-bm)/18$. The special cases where $m = a$ or $m = b$ can also be considered (the density and the cdf simplify in these cases). The triangular distribution is sometimes used as an heuristic approximation when little (or no) data is available. In that case, m, a, and b can be taken as guesses of the most likely, smallest possible, and largest possible values.

2.8.3 The normal distribution

A random variable X has the *normal* (or *Gaussian*) *distribution* with mean μ and variance σ^2, denoted $X \sim N(\mu, \sigma^2)$ or $X \sim \mathsf{N}(\mu, \sigma^2)$, if X has the density

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

for $x \in \mathbb{R}$ (see Figure 2.3). The mgf is $M_X(\theta) = \exp[\mu\,\theta + \sigma^2\theta^2/2]$. If $X \sim N(\mu, \sigma^2)$ and $Y = aX + b$, then $Y \sim N(a\mu + b, a^2\sigma^2)$. In particular, $Y = (X-\mu)/\sigma \sim N(0, 1)$. The $N(0, 1)$

distribution is called the *standard normal*. We denote by $\Phi(\cdot)$ the cdf of the standard normal, i.e.,

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-s^2/2} ds.$$

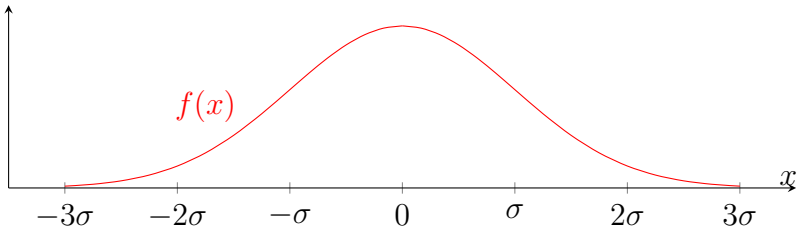


Fig. 2.3. Density of the $\mathsf{N}(0, \sigma^2)$ distribution

The normal distribution is often motivated by various versions of the CLT (Theorem A.15), which states that the standardized average of n i.i.d. random variables $X_1, \ldots, X_n$ converges to a standard normal when $n \to \infty$. There are more general versions of the CLT which allow the X_i's to have *different distributions* and a limited amount of *dependence*, multivariate CLTs, functional CLTs, and so on. What these various CLTs tell us is that we should expect a random variable X to have a nearly normal distribution if it can be expressed as the average (or sum) of a large number of small effects, where no small subset of these effects dominates the others. Some results such as the Berry-Esseen-type bounds (Theorem A.16) provide explicit bounds on the difference between the distribution of the standardized average and the standard normal distribution, in terms of a weighted sum of the third absolute moments of order 3 of the X_i's. They indicate that the convergence to the normal distribution could be very slow if the X_i's have a highly skewed distribution.

2.8.4 The lognormal distribution

Suppose that instead of having a sum of small effects, we have a *product of small effects*, i.e., $Y_n = Z_1 \cdot Z_2 \cdots Z_n$. Then, $X_n = Y_n^{1/n}$ is the *geometric average* of $Z_1, \ldots, Z_n$. Taking the logarithm, we get

$$\ln X_n = \ln Y_n^{1/n} = \frac{\ln Y_n}{n} = \frac{\ln Z_1 + \cdots + \ln Z_n}{n}.$$

If the $\ln Z_j$'s satisfy the conditions of the central limit theorem, we obtain that

$$\frac{\ln X_n - \mathbb{E}[\ln X_n]}{(\mathrm{Var}[\ln X_n])^{1/2}} \;\Rightarrow\; N(0,1) \qquad \text{for } n \to \infty$$

so $\ln X_n$ is approximately normal for large n.

When $\ln X$ has the normal distribution, we say that X has the *lognormal distribution*. If $\ln X \sim N(\mu, \sigma^2)$, then $X \sim \mathsf{Lognormal}(\mu, \sigma^2)$ has density

$$f(x) = \frac{1}{\sigma x\sqrt{2\pi}} \exp\left(\frac{-(\ln x - \mu)^2}{2\sigma^2}\right)$$

for $x > 0$. Its mean and variance are $\mathbb{E}[X] = e^{\mu+\sigma^2/2}$ and $\mathrm{Var}[X] = e^{2\mu+\sigma^2}(e^{\sigma^2} - 1)$. Observe that $\mathbb{E}[\ln X] = \mu < \mu + \sigma^2/2 = \ln \mathbb{E}[X]$. The lognormal distribution appears in the study of geometric Brownian motion and is widely used in economics and financial modeling.

Example 2.6 Someone places an initial amount I_0 in mutual funds and leaves it there for one year. Let us divide the year into n periods of equal length. The (random) growth rate (per year) of the value during period j is R_j. The value after the n periods is thus

$$X_n = I_0(1 + R_1)^{1/n}(1 + R_2)^{1/n} \cdots (1 + R_n)^{1/n}.$$

If the R_j are i.i.d., or more generally if the average of the random variables $\ln(1 + R_1), \ldots, \ln(1 + R_n)$ satisfy a CLT, then

$$\ln X_n = \ln I_0 + \frac{1}{n}\sum_{j=1}^{n} \ln(1 + R_j)$$

is approximately normal when n is large. This means that X_n has approximately the lognormal distribution. □

2.8.5 The inverse Gaussian distribution

The *inverse Gaussian distribution* (or *Wald distribution*) with location parameter $\mu > 0$ and shape parameter $\lambda > 0$ has density

$$f(x) = \left(\frac{\lambda}{2\pi x^3}\right)^{1/2} \exp\left[\frac{-\lambda(x - \mu)^2}{2\mu^2 x}\right] \qquad \text{for } x > 0,$$

and cdf

$$F(x) = \Phi(y_1) + e^{2\lambda/\mu}\Phi(-y_2)$$

where $y_1 = (x/\mu - 1)\sqrt{\lambda/x}$ and $y_2 = (x/\mu + 1)\sqrt{\lambda/x}$. We denote $X \sim \mathsf{InvGaussian}(\mu, \lambda)$ when X has this distribution. It has mean μ, variance μ^3/λ, and skewness $3\sqrt{\mu/\lambda}$. The mgf is $M_X(\theta) = \exp\left[(\lambda/\mu)(1 - (1 - 2\theta\mu^2/\lambda)^{1/2})\right]$. See Chhikara and Folks (1989) and Seshadri (1993) for more on this distribution. The parameters are easily estimated via maximum likelihood (Exercise 2.46). If $\lambda \to \infty$ and $\mu^3/\lambda \to \sigma^2$, then $(X - \mu)/\sigma^2 \Rightarrow N(0, 1)$.

The first passage time at a given positive level, for a BM with positive drift, has an inverse Gaussian distribution (Section 2.14.3).

A random variable $X \sim \mathsf{InvGaussian}(\mu, \lambda)$ can be generated as follows (Michael, Schuchany, and Haas 1976):

Generating $X \sim \mathsf{InvGaussian}(\mu, \lambda)$**;**
generate $Z \sim \mathsf{N}(0, 1)$ and let $Y = Z^2$ (which is $\chi^2(1)$);
let $W = \mu + \frac{\mu}{2\lambda}\left(\mu Y - \sqrt{4\mu\lambda Y + \mu^2 Y^2}\right)$;
let $P = \mu/(\mu + W)$;
generate $U \sim U(0, 1)$;
if $U \le P$ return $X = W$ else return $X = \mu^2/W$.

This requires two random numbers, Z and U. This Z can be generated by inversion from another uniform, or in another way.

To generate X by inversion, one would generate $U \sim \mathsf{Uniform}(0,1)$ and then approximate numerically the root X of $F(X) = U$, as discussed in Section 4.1.2. This is generally slower than the above method.

If $X \sim \mathsf{InvGaussian}(\mu, \lambda)$, then $aX \sim \mathsf{InvGaussian}(a\mu, a\lambda)$. If $X_1, \ldots, X_k$ are independent, $X_i \sim \mathsf{InvGaussian}(\mu_i, \lambda_i)$, and $\mu_i^2/\lambda_i = \mu_0^2/\lambda_0$ for all i, then $X = X_1 + \cdots + X_k \sim \mathsf{InvGaussian}(\mu, \lambda)$ where $\mu = \mu_1 + \cdots + \mu_k$ and $\lambda = \lambda_0(\mu/\mu_0)^2$.

There is also a three-parameter *generalized inverse Gaussian (GIG) distribution*, which includes the inverse Gaussian as a special case and a gamma distribution as a limiting case (Jørgensen 1982).

2.8.6 The normal inverse Gaussian (NIG) distribution

If $X \sim N(\mu + \beta Y, Y)$ where $Y \sim \mathsf{InvGaussian}(\delta, \gamma)$, then X has the *normal inverse Gaussian (NIG) distribution*, denoted $X \sim \mathsf{NIG}(\alpha, \beta, \mu, \delta)$, where $\alpha^2 = \beta^2 + \gamma^2$. This definition provides an easy way to generate X, by first generating Y, then $Z \sim \mathsf{N}(0,1)$, and returning $X = \mu + \beta Y + Z\sqrt{Y}$. The density of X is

$$f(x) = \frac{\alpha\delta e^{\delta\gamma + \beta(x-\mu)} K_1\left(\alpha\sqrt{\delta^2 + (x-\mu)^2}\right)}{\pi\sqrt{\delta^2 + (x-\mu)^2}} \qquad \text{for } x \in \mathbb{R},$$

where K_1 is the modified Bessel function of the second kind, of order 1:

$$K_p(y) = \frac{1}{2}\int_0^\infty z^{p-1} e^{-(z+1/z)y/2} dz.$$

It has mean $\mu + \delta\beta/\gamma$, variance $\delta\alpha^2/\gamma^3$, skewness $3\beta/(\alpha\sqrt{\delta\gamma})$, and kurtosis $3(\alpha^2 + 4\beta^2)/(\delta\alpha^2\gamma)$. Thus, μ is a location parameter, $\delta > 0$ a scale parameter, β a skewness parameter, and $\alpha > 0$ a kurtosis parameter. The mgf is

$$M_X(\theta) = \exp\left[\delta\left(\gamma - \sqrt{\alpha^2 - (\beta+\theta)^2}\right) + \mu\theta\right].$$

If $X_j \sim \mathsf{NIG}(\alpha, \beta, \mu_j, \delta_j)$ for $j = 1, 2$, and if X_1, X_2 are independent, then $X_1 + X_2 \sim \mathsf{NIG}(\alpha, \beta, \mu_1 + \mu_2, \delta_1 + \delta_2)$.

See Barndorff-Nielsen (1997) for further details. The NIG distribution is a special case the a *generalized hyperbolic distribution* (Barndorff-Nielsen, Mikosch, and Resnick 2013).

2.8.7 The exponential distribution

A random variable X has the *exponential distribution* with *rate* parameter λ, denoted $X \sim \mathsf{Exponential}(\lambda)$, if its cdf is

$$F(x) = 1 - e^{-\lambda x} \qquad \text{for } x > 0. \tag{2.7}$$

The density is $f(x) = \lambda e^{-\lambda x}$ for $x > 0$. The mean and variance are $\mathbb{E}[X] = 1/\lambda$ and $\mathrm{Var}[X] = 1/\lambda^2$, so the coefficient of variation is 1. (Beware: In simulation software, the procedures dealing with exponential random variables often have the *mean* $\mu = 1/\lambda$ as a parameter.) The mgf is $M_X(\theta) = (1 - \theta/\lambda)^{-1}$.

The main characterization of the exponential distribution is its memoryless property: If X is exponential, the conditional distribution of $X - t$ given that $X > t$ is the same as the distribution of X. That is,

$$\mathbb{P}[X > t + x \mid X > t] = \frac{\mathbb{P}[X > t + x]}{\mathbb{P}[X > t]} = \frac{e^{-\lambda(t+x)}}{e^{-\lambda t}} = e^{-\lambda x} = \mathbb{P}[X > x]. \tag{2.8}$$

If X represents the lifetime of a component, or a service time, or any other activity duration, the memoryless property means that at any moment, one can forget the duration elapsed so far for the activity and consider the remaining duration as a *new* exponential random variable with mean $1/\lambda$, independent of the past. The exponential is the only continuous distribution having this property (The *geometric* distribution has the corresponding property among discrete distributions). The memoryless property greatly simplifies the mathematical analysis and the simulation of certain systems. It explains the widespread popularity of the exponential distribution.

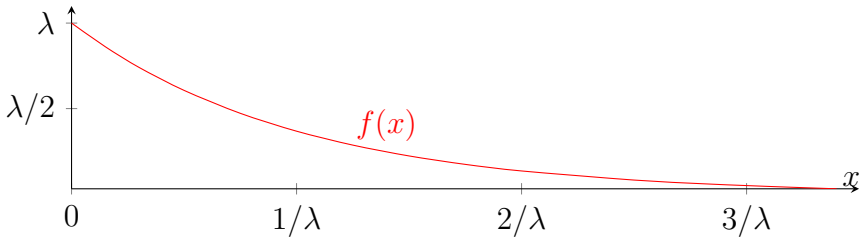


Fig. 2.4. The exponential density with parameter λ

Example 2.7 In the program of Figure 1.29 (for the call center), a `NextPeriod` event occurs at the beginning of each hour j. This event changes the number of agents in the center, and changes the arrival rate from λ_{j-1} to λ_j. If the next arrival was due to occur in X units of time after the beginning of the hour, then the memoryless property implies that X has the exponential distribution with parameter λ_{j-1}. To take into account the change in the arrival rate, we must replace it by an exponential with parameter λ_j. This is easily achieved by multiplying X by λ_{j-1}/λ_j, because if $U \sim U(0,1)$, then $-\ln(1-U)/\lambda$ is exponential with rate λ, and vice-versa. The call to `nextArrival.reschedule` in the program does exactly that. □

The exponential is a special case of the Weibull distribution. It has a *constant failure rate* λ. This means that if a component has an exponential lifetime distribution, the chance of an imminent failure does not depend on its age. If $X_1, \ldots, X_k$ are independent and X_j is exponential with rate λ_j for each j, then one can easily show that $X = \min(X_1, \ldots, X_k)$

is exponential with rate $\lambda = \lambda_1 + \cdots + \lambda_k$ (Exercise 2.14). If the λ_j are equal, the sum $X_1 + \cdots + X_k$ has the gamma distribution (see Section 2.8.12).

Example 2.8 Consider an $M/M/1$ queue, with arrival rate λ and service rate μ. This means that the interarrival times and the service times are independent exponential random variables with rates λ and μ, respectively. If the server is idle, the time to the next event (the next arrival) is an exponential with rate λ. If the server is busy, the time to the next event is the minimum between the time until the next arrival and the time until the next end of service, which are two independent exponentials with rates λ and μ, so it is an exponential with rate $\lambda+\mu$ (or mean $1/(\lambda+\mu)$). This next event is an arrival with probability $p = \lambda/(\lambda+\mu)$, and a service completion with probability $1-p$. This system can be simulated without an event list. The *state* of the model is simply the number of customers in the system. Given the state at any given time, one can generate the time until the next event by generating an exponential. If the state is larger than 0, one also generates a $U \sim U(0,1)$ and decides that the next event is an arrival if $U \leq p$, and a departure otherwise. This scheme can be generalized to queuing networks with exponential interarrivals and service times at all nodes (the state is the number of customers at each node), and to more general continuous-time Markov chains (see Exercise 2.15 and Fox 1993). □

2.8.8 The Weibull distribution

The *Weibull distribution* has density

$$f(x) = \alpha\lambda^\alpha(x-\delta)^{\alpha-1}\exp[-(\lambda(x-\delta))^\alpha] \qquad \text{for } x > \delta,$$

where $\delta \in \mathbb{R}$ is the location parameter (often equal to 0), $\alpha > 0$ is the shape parameter, and $\lambda > 0$ is the rate parameter (the scale is $1/\lambda$). We denote $X \sim \mathsf{Weibull}(\delta, \alpha, \lambda)$ in general and $X \sim \mathsf{Weibull}(\alpha, \lambda)$ when $\delta = 0$. The mean and variance are $\mathbb{E}[X] = \delta + \Gamma(1/\alpha)/(\alpha\lambda)$ and $\mathrm{Var}[X] = [2\Gamma(2/\alpha) - \Gamma^2(1/\alpha)/\alpha]/(\alpha\lambda^2)$. The cdf is

$$F(x) = 1 - \exp[-(\lambda(x-\delta))^\alpha] \qquad \text{for } x > \delta.$$

Figure 2.5 shows the shape of the density for selected values of α. We have $X \sim \mathsf{Weibull}(\alpha, \lambda)$ if and only if $(\lambda X)^\alpha \sim \mathsf{Exponential}(1)$.

This distribution is often used to model lifetimes of deteriorating systems or parts. Let $\delta = 0$. If $\alpha = 1$, this gives the exponential distribution with rate λ, with constant failure rate and coefficient of variation 1. For $\alpha < 1$, the failure rate is decreasing and the coefficient of variation is larger than 1 (more relative variability than the exponential distribution), whereas for $\alpha > 1$ one has the opposite.

The $\mathsf{Weibull}(2, \lambda)$ is also called the *Rayleigh distribution* with parameter $\beta = 1/\lambda$, denoted $\mathsf{Rayleigh}(\beta)$. Its cdf is $F(x) = 1 - \exp[-x^2/\beta]$ for $x > 0$. If the coordinates (X_1, X_2) of a 2-dimensional vector are independent $N(0, \sigma^2)$, its Euclidean length $\sqrt{X_1^2 + X_2^2}$ has the $\mathsf{Rayleigh}(\sigma\sqrt{2})$ distribution.

Why the Weibull distribution? Whereas the average of i.i.d. random variables tends to be normally distributed, the *minimum* of random variables that have a fixed lower bound tends

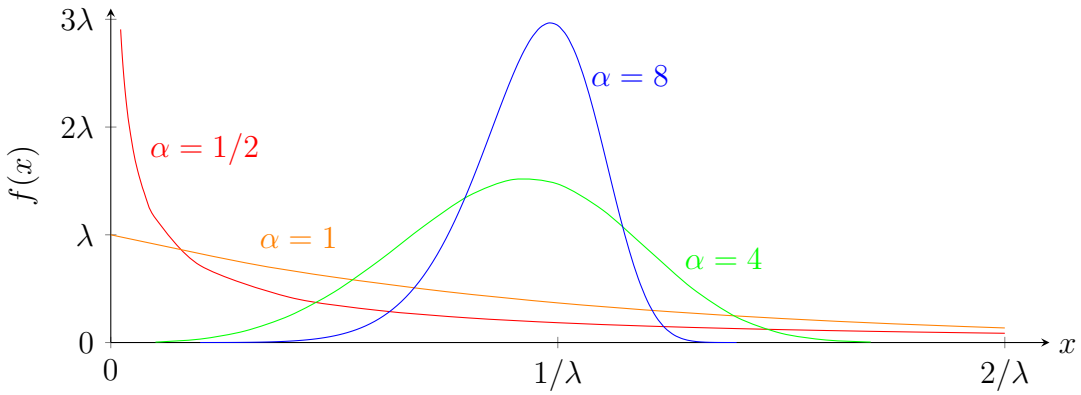


Fig. 2.5. The Weibull density with shape parameter $\alpha = 1/2$, 1, 4, and 8, rate parameter λ, and location parameter $\delta = 0$.

to follow the Weibull distribution. The following theorem, taken from Galambos (1978), page 56, makes this statement precise.

Theorem 2.2 *Let $X_1, \ldots, X_n$ be* i.i.d. *random variables with a distribution function G for which the theoretical minimum is finite, i.e., $\delta = \inf\{x \mid G(x) > 0\} > -\infty$. Let $W_n = \min(X_1, X_2, \ldots, X_n)$. For each integer $n > 0$, define d_n as the solution of $G(\delta + d_n) = 1/n$. Suppose that there is a constant $\alpha > 0$ such that*

$$\lim_{t \to \infty} \frac{G(\delta + 1/(tx))}{G(\delta + 1/t)} = x^{-\alpha}. \tag{2.9}$$

Then

$$\lim_{n \to \infty} \mathbb{P}[W_n - \delta < d_n x] \;=\; \begin{cases} 1 - \exp[-x^\alpha] & \text{if } x > 0, \\ 0 & \text{if } x \le 0. \end{cases}$$

That is, $(W_n - \delta)/d_n$ converges to the Weibull$(\alpha, 1)$ *distribution when $n \to \infty$.*

The condition (2.9) implies that the shape parameter α depends on the form of the distribution G at its starting point δ. Of course, the theorem can also be used to find the distribution of the *maximum* (instead of the minimum) if the distribution G has a theoretical has a maximum; it suffices to reverse the signs.

As an illustration, suppose that a system fails at the occurrence of the first event from a large set of independent potential events whose dates of occurrence have a continuous distribution. Then the theorem justifies a Weibull distribution for the time until the system fails. In a different context, suppose that an optimization (minimization) problem has millions of local minima, and that we perform local optimizations to find "good" solutions from each of n independent random starting points in the solution space. Let W_n be the value of the best solution found by this approach. Under appropriate conditions (to satisfy approximately the assumptions of the theorem), for large n, W_n has approximately the Weibull distribution. Here, δ represents the value of the global minimum.

2.8.9 The Gumbel distribution

Theorem 2.2 does not apply when the cdf G has an infinite left tail, for example if G is the normal distribution. In that case, the asymptotic distribution of the minimum (or of the maximum, if there is an infinite right tail), properly scaled, is Gumbel. See Theorem 2.3. The *Gumbel* (or type I extreme value) *distribution* with location parameter δ and scale parameter $\beta \neq 0$, denoted $\mathsf{Gumbel}(\delta, \beta)$, has density

$$f(x) = (1/|\beta|) \exp\left[-e^{(\delta-x)/\beta} + (\delta - x)/\beta\right]$$

and cdf

$$F(x) = \begin{cases} \exp\left[-e^{(\delta-x)/\beta}\right] & \text{if } \beta > 0, \\ 1 - \exp\left[-e^{(\delta-x)/\beta}\right] & \text{if } \beta < 0. \end{cases}$$

for $x \in \mathbb{R}$. The "scale" is actually determined by $|\beta|$. We have $X \sim \mathsf{Gumbel}(\delta, \beta)$ if and only if $2\delta - X \sim \mathsf{Gumbel}(\delta, -\beta)$, so it is possible to always work with $\beta > 0$. The mean and variance are $\mu = \delta + \gamma\beta$ and $\sigma^2 = (\beta\pi)^2/6$, where $\gamma = 0.5772156649...$ is Euler's constant, and the mgf is $M_X(\theta) = \Gamma(1 - \beta\,\theta)\, e^{\delta\,\theta}$ if $\beta > 0$. The *standard Gumbel* distribution has $\delta = 0$ and $\beta = 1$.

If $Y \sim \mathsf{Weibull}(\alpha, \lambda)$ and $0 \neq c \in \mathbb{R}$, then $X = c \ln Y \sim \mathsf{Gumbel}(\delta, \beta)$ with $\delta = (\ln \lambda)/\alpha$ and $\beta = -1/(c\alpha)$. Thus, in analogy with the lognormal, the Weibull distribution can be interpreted as a *logGumbel* (with negative β).

2.8.10 The Fréchet distribution

A random variable X has the *Fréchet distribution* with location parameter $\delta \in \mathbb{R}$, scale parameter $\beta > 0$, and shape parameter $\alpha \in \mathbb{R}$, if its cdf is given by

$$F(x) = \exp\left[-((x - \delta)/\beta)^{-\alpha}\right]$$

for $x > \delta$. The mean is $\mu = \delta + \beta\Gamma(1 - 1/\alpha)$ for $\alpha > 1$ and the variance is $\beta^2[\Gamma(1 - 2/\alpha) - \Gamma^2(1 - 1/\alpha)]$ for $\alpha > 2$ (otherwise they are infinite).

2.8.11 Generalized extreme value distribution

A random variable X has the *generalized extreme value (GEV) distribution* with location parameter $\delta \in \mathbb{R}$, scale parameter $\beta > 0$, and shape parameter $\xi \in \mathbb{R}$, if its cdf is given by

$$F(x) = \begin{cases} \exp\left[-\left[1 + \xi(x - \delta)/\beta\right]^{-1/\xi}\right] & \text{for } x > \delta - \beta/\xi \text{if} \xi \neq 0, \\ \exp\left[-\exp[-(x - \delta)/\beta]\right] & \text{for all } x \in \mathbb{R} \text{if } \xi = 0. \end{cases}$$

The parameter ξ determines the shape of the tail. When $\xi > 0$, $\alpha = 1/\xi$ is called the *tail index*. The Gumbel, Fréchet, and reverse Weibull ($-X$ where X is Weibull) are all special cases. For $\xi = 0$, we obtain a Gumbel distribution (Section 2.8.9). For $\xi > 0$, we have a Fréchet distribution (Section 2.8.10). For $\xi < 0$, we have a reverse Weibull distribution (Section 2.8.8). These three distributions are important because of the following theorem:

Theorem 2.3 (Fisher-Tippet-Gnedenko theorem, or extreme value theorem.) *If $X_1, X_2, \ldots$ are* i.i.d. *random variables with cdf G, $M_n = \max\{X_1, \ldots, X_n\}$, and if there are sequences of real numbers $\{d_n,\, n \geq 1\}$ and $\{c_n,\, n \geq 1\}$ such that*

$$Y_n = \frac{M_n - d_n}{c_n} \;\Rightarrow\; Y$$

when $n \to \infty$, where Y is not degenerate (a constant), then Y is a random variable with the Gumbel, Fréchet or reverse Weibull distribution.

If G has a finite tail, i.e., $G(x) = 1$ for some $x < \infty$, then $-Y$ has a Weibull distribution. If G has a right tail that decreases exponentially (as for the normal, gamma, or exponential distribution, for example), then Y is Gumbel. If G has a right tail that decreases as a polynomial (as for the Pareto, Student-t, and several mixture distributions), then Y is Fréchet.

2.8.12 The gamma distribution

A random variable X has the *gamma distribution* with shape parameter $\alpha > 0$ and rate parameter $\lambda > 0$, denoted $X \sim \mathsf{Gamma}(\alpha, \lambda)$, if its density is

$$f(x) = \frac{\lambda^\alpha x^{\alpha-1} e^{-\lambda x}}{\Gamma(\alpha)} \qquad \text{for } x > 0,$$

where the gamma function Γ is defined by

$$\Gamma(y) = \int_0^\infty x^{y-1} e^{-x} dx \qquad \text{for } y > 0. \tag{2.10}$$

In particular, one has $\Gamma(1/2) = \sqrt{\pi}$, $\Gamma(1) = 1$, and $\Gamma(y+1) = y\Gamma(y)$ for any $y > 0$. As a consequence, if k is a positive integer, $\Gamma(k+1) = k!$ and $\Gamma(k+1/2) = \sqrt{\pi}(1/2)\cdots(k-1/2)$. The mean and variance are $\mathbb{E}[X] = \alpha/\lambda$ and $\mathrm{Var}[X] = \alpha/\lambda^2$. Figure 2.6 shows the density for selected values of α.

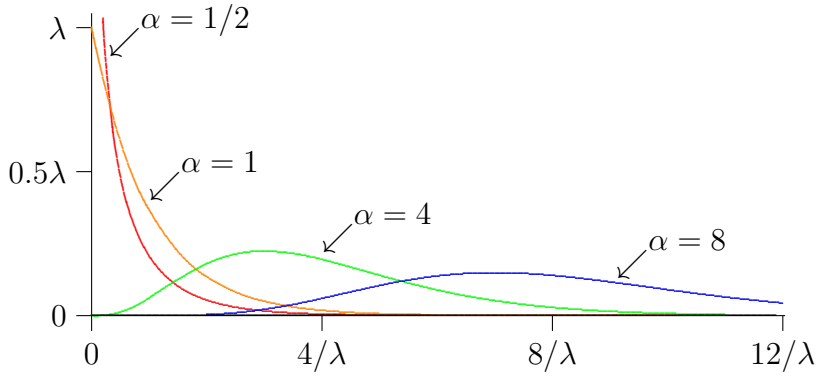


Fig. 2.6. The gamma density with shape parameter $\alpha = 1/2$, 1, 4, and 8

As special cases, the gamma distribution is the same as the exponential distribution when $\alpha = 1$ and the same as the chi-square distribution with $k = 2\alpha$ degrees of freedom when 2α is an integer and $\lambda = 1/2$. If $X \sim \mathsf{Gamma}(\alpha, \lambda)$ then $1/X \sim \mathsf{Pearson\text{-}5}(\alpha, \lambda)$ and vice-versa. If $X_1 \sim \mathsf{Gamma}(\alpha_1, \lambda)$ and $X_2 \sim \mathsf{Gamma}(\alpha_2, \lambda)$ are independent, then $X_1/(X_1 + X_2) \sim \mathsf{Beta}(\alpha_1, \alpha_2)$.

If $X_1, \ldots, X_k$ are independent random variables and $X_i \sim \mathsf{Gamma}(\alpha_i, \lambda)$ for each i, then $X = X_1 + \cdots + X_k \sim \mathsf{Gamma}(\alpha, \lambda)$ where $\alpha = \alpha_1 + \cdots + \alpha_k$.

In particular, if $X_1, \ldots, X_k$ are independent exponentials with the same parameter λ, then $X = X_1 + \cdots + X_k$ is a $\mathsf{Gamma}(k, \lambda)$ random variable. In this case, i.e., when $\alpha = k$ is an integer, the Gamma(k, λ) distribution is also called the *Erlang(k, λ) distribution*.

The gamma cdf does not have a closed-form expression in general, but if $X \sim \mathsf{Erlang}(k, \lambda)$, then $\mathbb{P}[X \le x] = \mathbb{P}[Y \ge k]$ where $Y \sim \mathsf{Poisson}(\lambda x)$ (Exercise 2.12). Therefore, in that case,

$$F(x) = \mathbb{P}[X \le x] = 1 - \sum_{j=0}^{k-1} \frac{e^{-\lambda x}(\lambda x)^j}{j!} \tag{2.11}$$

for $x > 0$. This can be used to compute $F(x)$ when k is small. Conversely, a good numerical approximation of the gamma distribution function can be used to approximate the Poisson distribution function via (2.11).

A good algorithm for generating $\mathsf{Gamma}(\alpha, 1)$ random variates suffices for generating gammas with arbitrary parameters, because λ only changes the scale.

2.8.13 The beta distribution

The *beta distribution* with shape parameters $\alpha > 0$ and $\beta > 0$, over the interval $(0, 1)$, has density

$$f(x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1 - x)^{\beta-1} \qquad \text{for } 0 < x < 1.$$

It is sometimes used to model a random proportion. Its mean and variance are $\mathbb{E}[X] = \alpha/(\alpha + \beta)$ and $\mathrm{Var}[X] = \alpha\beta/[(\alpha + \beta)^2(\alpha + \beta + 1)]$. Some beta densities are illustrated in Figure 2.7 for the symmetric case ($\alpha = \beta$) and Figure 2.8 for the asymmetric case. For the symmetric case, all the density becomes concentrated at 1/2 (asymptotically) when $\alpha \to \infty$ and split half and half between 0 and 1 when $\alpha \to 0$.

One has $X \sim \mathsf{Beta}(\alpha, \beta)$ if and only if $1 - X \sim \mathsf{Beta}(\beta, \alpha)$ if and only if $X/(1 - X) \sim \mathsf{Pearson\text{-}6}(\alpha, \beta, 1)$. The $\mathsf{Beta}(1, 1)$ and $U(0, 1)$ distributions are the same. The $\mathsf{Beta}(1, 2)$ and $\mathsf{Beta}(2, 1)$ are special cases of the triangular distribution. A beta random variable X over $(0, 1)$ can be transformed into a beta random variable Y with the same shape of distribution, but over an arbitrary interval (a, b), via the linear transformation $Y = a + (b - a)X$.

We have the following relationship between the binomial and beta cdf's: If $Y \sim \mathsf{Binomial}(n, p)$, then $\mathbb{P}[Y \ge y] = \mathbb{P}[X \le p]$ where $X \sim \mathsf{Beta}(y, n - y + 1)$. This is sometimes used to approximate the binomial cdf.

2.8.14 The chi-square distribution

The *chi-square distribution* with n degrees of freedom, denoted $\chi^2(k)$, has density

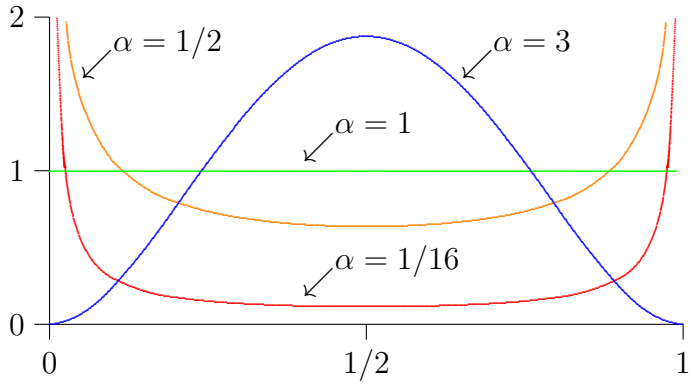


Fig. 2.7. The symmetric beta density with parameters $\alpha = \beta = 1/16, 1/2, 1$, and 3.

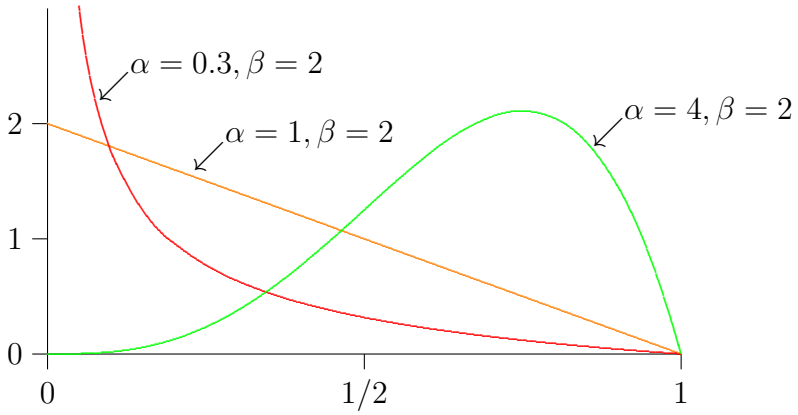


Fig. 2.8. The beta density with parameters $\beta = 2$ and $\alpha = 0.3$, 1, and 4.

$$f(x) = \frac{x^{(k/2)-1} e^{-x/2}}{2^{k/2}\Gamma(k/2)} \qquad \text{for } x > 0.$$

It is illustrated in Figure 2.9. The mean and variance are $\mathbb{E}[X] = k$ and $\mathrm{Var}[X] = 2k$. If $X_1, \ldots, X_k$ are i.i.d. $N(0,1)$, then $X_j^2 \sim \chi^2(1)$ for each j and $X = X_1^2 + \cdots + X_k^2 \sim \chi^2(k)$. As a consequence of this and the central limit theorem, the distribution of $(X - k)/\sqrt{2k}$ converges to the $N(0,1)$ when $k \to \infty$.

The chi-square distribution is widely used in several contexts, including goodness-of-fit and independence tests for discrete distributions, and computing a confidence interval for the variance (Chapter 5). It is a special case of the gamma distribution with $\alpha = k/2$.

2.8.15 The noncentral chi-square

The *noncentral chi-square distribution* generalizes the chi-square. If $X_1, \ldots, X_k$ are independent and $X_j \sim N(\mu_j, \sigma_j^2)$ for each j, then $X = \sum_{j=1}^{k} X_j^2/\sigma_j^2$ has the noncentral chi-square

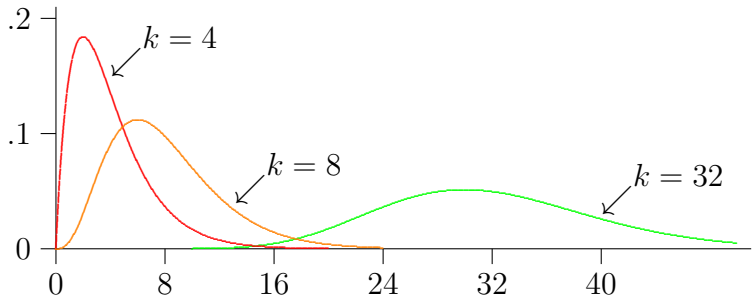


Fig. 2.9. The chi-square density with $k = 4$, 8, 32

distribution with k degrees of freedom and noncentrality parameter $\lambda = \sum_{j=1}^{k} \mu_j^2/\sigma^2$. This distribution is equivalent to a Poisson-weighted mixture of central chi-square distributions: If $N \sim \mathsf{Poisson}(\lambda/2)$ and $X \sim \chi^2(k+2N)$, then X is a noncentral chi-square with parameters (k, λ). The mean and variance are $\mathbb{E}[X] = k + \lambda$ and $\mathrm{Var}[X] = 2(k + 2\lambda)$. With $\lambda = 0$, we recover the usual chi-square distribution.

2.8.16 The Student-t distribution

The *Student-t distribution* with n degrees of freedom, denoted $\mathsf{Student\text{-}t}(n)$, has density

$$f(x) = \frac{\Gamma((n+1)/2)}{\sqrt{n\pi}\,\Gamma(n/2)(1 + x^2/n)^{(n+1)/2}} \qquad \text{for } x \in \mathbb{R}.$$

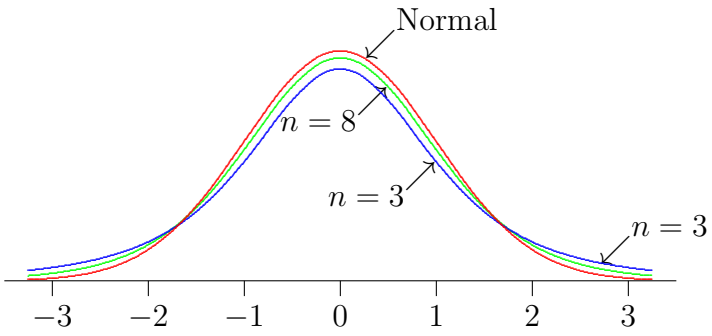


Fig. 2.10. The Student-t density with $n = 3$ and 8, compared with the standard normal density.

This density resembles that of the $N(0, 1)$, but is slightly flatter in the center and thicker in the tails; see Figure 2.10. The Student-t(n) converges to the $N(0, 1)$ distribution when $n \to \infty$. The mean is 0 and the variance is $\mathrm{Var}[X] = n/(n-2)$ for $n \geq 3$. If $Z \sim N(0, 1)$ and $Y \sim \chi^2(n)$ are independent random variables, then $X = Z/\sqrt{Y/n} \sim \mathsf{Student\text{-}t}(n)$.

2.8.17 The F distribution of Fisher

The *F* (or *Fisher*) *distribution* with n_1 and n_2 degrees of freedom, denoted $\mathsf{F}(n_1, n_2)$, has density

$$f(x) = \frac{\Gamma((n_1+n_2)/2)(n_1/n_2)^{n_1/2}x^{(n_1/2)-1}}{\Gamma(n_1/2)\Gamma(n_2/2)(1+n_1x/n_2)^{(n_1+n_2)/2}} \qquad \text{for } x > 0.$$

Its mean and variance are $\mathbb{E}[X] = n_2/(n_2-1)$ for $n_2 \geq 2$ and $\mathrm{Var}[X] = n_2^2(2n_2+2n_1-4)/[n_1(n_2-2)^2(n_2-4)]$ for $n \geq 5$. If $X \sim \mathsf{F}(n_1, n_2)$ then $1/X \sim \mathsf{F}(n_2, n_1)$. If $X_1 \sim \chi^2(n_1)$ and $X_2 \sim \chi^2(n_2)$ are independent random variables, then $X = (X_1/n_1)/(X_2/n_2) \sim \mathsf{F}(n_1, n_2)$. This distribution is often used for analysis of variance (e.g., in linear regression) under normality assumptions.

2.8.18 Pearson distribution of type 5

If $Y \sim \mathsf{Gamma}(\alpha, \lambda)$ then $X = 1/Y$ has the *Pearson type 5* (or *inverted gamma*) *distribution*, denoted $X \sim \mathsf{Pearson\text{-}5}(\alpha, \lambda)$. Its density and cdf are

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

and

$$F(x) = 1 - F_\gamma(1/x)$$

for $x > 0$, where F_γ is the $\mathsf{Gamma}(\alpha, \lambda)$ cdf. The mean and variance are $\mathbb{E}[X] = \lambda/(\alpha-1)$ for $\alpha > 1$, $\mathrm{Var}[X] = \lambda^2/[(\alpha-1)^2(\alpha-2)]$ for $\alpha > 2$, and are infinite otherwise.

Fitting a Pearson type 5 distribution to a set of observations $X_1, \ldots, X_n$ is equivalent to transforming these observations into $Y_1 = 1/X_1, \ldots, Y_n = 1/X_n$ and fitting a gamma distribution to the Y_i's.

2.8.19 Pearson distribution of type 6

If $X_1 \sim \mathsf{Gamma}(\alpha_1, \lambda)$ and $X_2 \sim \mathsf{Gamma}(\alpha_2, 1)$, then $X = X_1/X_2$ has the *Pearson type 6* (or *ratio of gammas*) *distribution* with parameters $(\alpha_1, \alpha_2, \lambda)$, denoted $X \sim \mathsf{Pearson\text{-}6}(\alpha_1, \alpha_2, \lambda)$. Its density and cdf are

$$f(x) = \frac{\Gamma(\alpha_1+\alpha_2)\lambda(\lambda x)^{\alpha_1-1}}{\Gamma(\alpha_1)\Gamma(\alpha_2)(1+\lambda x)^{\alpha_1+\alpha_2}}$$

and

$$F(x) = 1 - F_\beta(\lambda x/(1+\lambda x))$$

for $x > 0$, where F_β is the $\mathsf{Beta}(\alpha_1, \alpha_2)$ cdf. The mean and variance are $\mathbb{E}[X] = \alpha_1/[\lambda(\alpha_2-1)]$ for $\alpha_2 > 1$, $\mathrm{Var}[X] = \alpha_1(\alpha_1+\alpha_2-1)/[\lambda^2(\alpha_2-1)^2(\alpha_2-2)]$ for $\alpha_2 > 2$, and are infinite otherwise.

We have that $Y = X/(1+X) \sim \mathsf{Beta}(\alpha_1, \alpha_2)$ if and only if $X = Y/(1-Y) \sim \mathsf{Pearson\text{-}6}(\alpha_1, \alpha_2, 1)$.

2.8.20 The Pareto distribution

There are several types of *Pareto distributions*; see Arnold (1983) for an extensive coverage. Their common property is that the tail of the density decreases at a slower rate than an exponential. Here, we denote by $\mathsf{Pareto}(\alpha, \beta)$ a distribution with two parameters $\alpha > 0$ and $\beta > 0$, and density

$$f(x) = \alpha\beta^{\alpha}x^{-(\alpha+1)} \qquad \text{for } x > \beta.$$

Figure 2.11 illustrates the case where $\alpha = \beta = 1$ (for which both the mean and variance are infinite). Its cdf is $F(x) = 1-(\beta/x)^{\alpha}$ for $x \geq \beta$. The mean and variance are $\mathbb{E}[X] = \alpha\beta/(\alpha-1)$ for $\alpha > 1$, $\mathrm{Var}[X] = \alpha\beta^2/[(\alpha-2)(\alpha-1)^2]$ for $\alpha > 2$, and are infinite otherwise. This distribution is used, e.g., for modeling insurance claim sizes (Asmussen 2000) and burst lengths in communication networks.

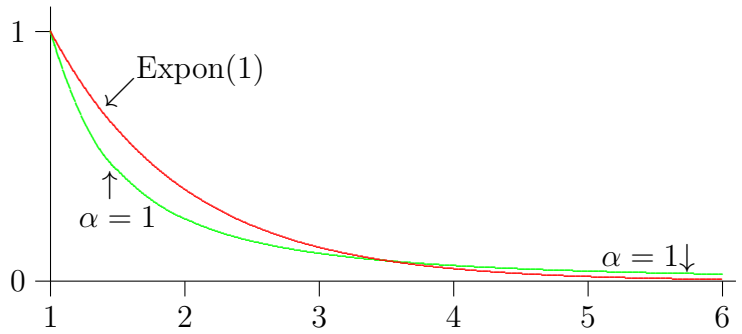


Fig. 2.11. The Pareto density with shape parameters $\alpha = \beta = 1$, compared to the exponential density with $\lambda = 1$, shifted to the right by 1, labeled Expon(1). The tail converges to zero much more slowly for the Pareto than for the exponential.

2.8.21 The logistic distribution

The *logistic distribution* with parameters α and $\lambda > 0$, denoted $\mathsf{Logistic}(\alpha, \lambda)$, has density

$$f(x) = \frac{\lambda e^{-\lambda(x-\alpha)}}{(1 + e^{-\lambda(x-\alpha)})^2}$$

and cdf

$$F(x) = \frac{1}{1 + e^{-\lambda(x-\alpha)}}$$

for $x \in \mathbb{R}$. If $\ln X$ has the logistic distribution, X is said to have the *loglogistic distribution*.

2.8.22 The Cauchy distribution

The *Cauchy distribution* with parameters α and $\beta > 0$, denoted $\mathsf{Cauchy}(\alpha, \beta)$, has density

$$f(x) = \frac{\beta}{\pi[(x-\alpha)^2 + \beta^2]}$$

and cdf

$$F(x) = \frac{1}{2} + \frac{1}{\pi}\arctan((x-\alpha)/\beta).$$

for $x \in \mathbb{R}$. Its mean is undefined and its variance is infinite.

2.8.23 The Johnson Family of Distributions

Johnson (1949) proposed a flexible family of distributions, whose densities can take a rich variety of shapes. The general form of cdf is

$$F(x) = \Phi[\gamma + \delta g((x-\xi)/\lambda)], \qquad \text{for } x \in \mathbb{R}, \tag{2.12}$$

where Φ is the standard normal cdf, γ and $\delta > 0$ are shape parameters, ξ is a location parameter, $\lambda > 0$ is a scale parameter, and g is one of the following four transformations:

$$g(y) = \begin{cases} \ln y & \text{for the } \textit{lognormal family}, \\ \sinh^{-1}(y) = \ln(y + \sqrt{y^2+1}) & \text{for the } \textit{unbounded family}, \\ \ln(y/(1-y)) & \text{for the } \textit{bounded family}, \\ y & \text{for the } \textit{normal family}. \end{cases} \tag{2.13}$$

In each case, $Z = \gamma + \delta g((X-\xi)/\lambda)$ has the $N(0,1)$ distribution, and one can write $X = \xi + \lambda g^{-1}((Z-\gamma)/\delta)$ where

$$g^{-1}(z) = \begin{cases} e^z & \text{for the lognormal family}, \\ (e^z - e^{-z})/2 & \text{for the unbounded family}, \\ 1/(1+e^{-z}) & \text{for the bounded family}, \\ z & \text{for the normal family}. \end{cases} \tag{2.14}$$

We assume, without loss of generality, that $\lambda = 1$ for the normal and lognormal families, and that $\xi = 0$ for the normal family. Members of the normal and lognormal families are the same as the normal and lognormal distributions, except that the lognormal is shifted to the right by ξ.

The density is

$$f(x) = \frac{\delta}{\lambda\sqrt{2\pi}} g'(x-\xi)/\lambda) \exp\left[-(\gamma + \delta g((x-\xi)/\lambda))^2/2\right] \qquad \text{for } x \in H,$$

where g' is the derivative of g and the support H of the distribution is $(-\infty, \infty)$ for the unbounded and normal families, $[\xi, \xi+\lambda]$ for the bounded family, and $[\xi, \infty)$ for the lognormal family. This density function is perfectly smooth: It is infinitely differentiable all over $\mathbb{R}$.

Figures 2.12 and 2.13 show examples of Johnson densities with location parameter ξ fixed to 0 and scale parameter λ fixed to 1. The densities are symmetric about their mean when $\gamma = 0$ and skewed to the left [resp., to the right] when $\gamma < 0$ [resp., $\gamma > 0$]. For fixed γ, the density becomes more sharply peaked when δ increases. For the unbounded family,

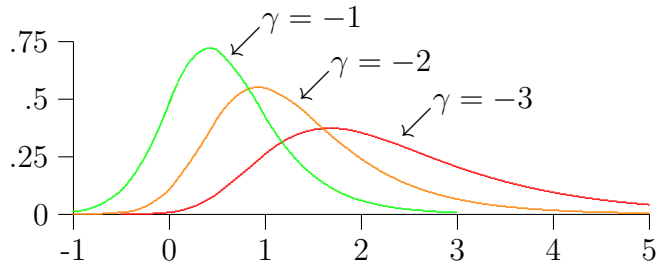


Fig. 2.12. The density for the Johnson unbounded family with parameters $\xi = 0$, $\lambda = 1$, $\delta = 2$ and $\gamma = -1, -2, -3$.

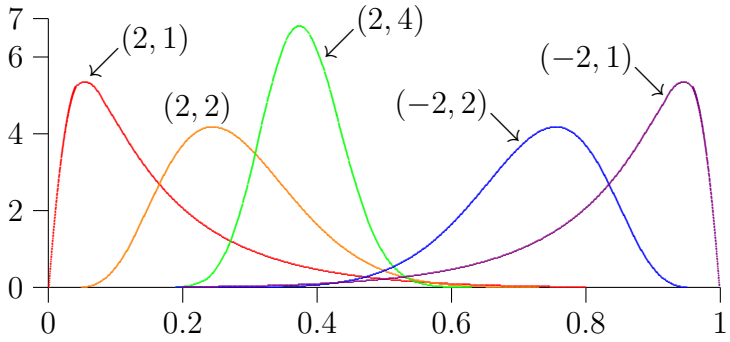


Fig. 2.13. The density for the Johnson bounded family with parameters $\xi = 0, \lambda = 1$ and different pairs (γ, δ).

the density is always unimodal, whereas for the bounded family, it can be either unimodal or bimodal.

DeBrota et al. (1989b) argue that the bounded family should be appropriate for a wide range of practical situations, because it can closely imitate other popular distributions such as the beta, normal, triangular, uniform, etc., and because in the real-world, measurements are always bounded anyway and the extreme values, near the endpoints of the support, are typically unlikely. Swain, Venkatraman, and Wilson (1988) developed a public-domain software package called FITTR1 for estimating the parameters of all types of Johnson distributions based on sample data, via weighted least squares. DeBrota et al. (1989a, 1989b) describe a visual interactive software, called VISIFIT, for fitting bounded Johnson distributions subjectively. The user provides the lower and upper bounds of the distribution, ξ and $\xi + \lambda$, plus two additional quantities such as the mean and standard deviation, or the mode and the width of the central 95% of the distribution, etc. The user may change the shape in a variety of ways until satisfied with the fitting.

2.8.24 Stable distributions

A *stable (or α-stable) distribution* is usually specified by its characteristic function φ_X, because there is no explicit expression for the density or the cdf. It has a location parameter $\mu \in \mathbb{R}$, a scale parameter $\sigma > 0$, a skewness parameter $\beta \in [-1, 1]$, and a "tail-heavyness" parameter $\alpha \in (0, 2]$. We have $X \sim S_\alpha(\sigma, \beta, \mu)$ (we say that X has an α-stable distribution) if

$$\varphi_X(t) = \mathbb{E}[e^{\sqrt{-1}tX}] = \exp\left[-|\sigma t|^\alpha \left(1 - \sqrt{-1}\beta \operatorname{sign}(t)\Psi_\alpha(t)\right) + \sqrt{-1}\mu t\right],$$

where $\Psi_\alpha(t) = (2/\pi)\ln|t|$ if $\alpha = 1$, and $\Psi_\alpha(t) = \tan(\alpha\pi/2)$ otherwise. For $\alpha = 2$, this gives the normal distribution with variance $2\sigma^2$. For $\alpha < 2$, X has infinite variance (and higher moments) and has Pareto-type tails:

$$\lim_{x\to\infty} x^\alpha \mathbb{P}[X > x] = \lim_{x\to\infty} x^\alpha \mathbb{P}[X < -x] = \frac{\Gamma(\alpha)\sin(\alpha\pi/2)}{\pi}(1+\beta)\sigma^\alpha.$$

This convergence is faster for smaller values of α. For $\alpha \le 1$, the mean does not exist.

Stable laws arise as the limiting distributions of centered and properly standardized sums of i.i.d. random variables $X_1, X_2, \ldots$; they are in fact the only distributions with this property. When these random variables have finite variance, this is the usual CLT. Otherwise, if there is a sequence $\{D_n,\, n \ge 2\}$ and a constant $\alpha > 0$ such that $n^{-1/\alpha}(X_1 + \cdots + X_n - D_n) \Rightarrow X$, then X must have an α-stable law. In fact, $X \sim S_\alpha(\sigma, \beta, \mu)$ if and only if one can write $X = n^{-1/\alpha}(X_1 + \cdots + X_n - D_n)$ where the X_j's are i.i.d. with the same distribution as X. We then say that X is *infinitely divisible*. If X_1 and X_2 are independent with the same stable distribution, then any linear combination $X = aX_1 + bX_2 + c$ has a stable law with the same parameters α and β.

Stable distributions thus appear as a natural choice to model a process defined by a sum of a large number of small effects, when these effects have a heavy-tailed (and infinite variance) distribution. Such processes are commonplace in finance, where stable distributions often provide a better fit than the normal or lognormal to stock returns, commodity price returns, and exchange rates, for example. On the other hand, they typically overestimate the tickness of the tails, leading to overestimation of the risk (Weron 2004). *Tempered stable distributions* often provide a better fit; they are modified stable distributions with exponentially decreasing (light) tails (Rosiński 2007).

For further details on stable laws, including parameter estimation, random variate generation, and applications, see Borak, Härdle, and Weron (2005), Samorodnitsky and Taqqu (1994), and Asmussen and Glynn (2007), page 332.

2.8.25 Exponential mixtures and phase-type distributions

The exponential distribution is convenient but not always realistic. Other distributions can often be well approximated by a sum or mixture of (a possibly random number of) exponentials.

We start with a simple case, the Erlang. If $X_1, \ldots, X_k$ are i.i.d. exponential with mean $1/\lambda$, then $X = X_1 + \cdots + X_k$ has the $\mathsf{Erlang}(k, \lambda)$ distribution, a special case of the gamma distribution (Section 2.8.12). The Erlang distribution has less variability than the exponential, its coefficient of variation is $1/\sqrt{k}$.

By summing k exponentials with different rates, $\lambda_1, \ldots, \lambda_k$, one gets more flexibility than with the Erlang. The distribution of the sum is called *hypoexponential* with rates $\lambda_1, \ldots, \lambda_k$. Conceptually, an activity has an hypoexponential duration if it can be decomposed into k shorter activities executed sequentially, and whose durations are independent and exponential. An hypoexponential random variable X with rates $\lambda_1, \ldots, \lambda_k$ has mean $\mathbb{E}[X] = \sum_{j=1}^{k}(1/\lambda_j)$ and variance $\mathrm{Var}[X] = \sum_{j=1}^{k}(1/\lambda_j^2)$. Its coefficient of variation is always less than 1.

To obtain more variability than for the exponential, i.e., a coefficient of variation larger than 1, one can connect exponentials in parallel instead of in series. That is, suppose we generate a random integer J in the set $\{1, \ldots, k\}$, with probabilities $p_j = \mathbb{P}[J = j]$ for $1 \le j \le k$, and then generate X from the exponential distribution with rate λ_J, where $\lambda_1, \ldots, \lambda_k$ are given. Then X has the *hyperexponential distribution* with parameters $p_1, \lambda_1, \ldots, p_k, \lambda_k$. Its mean and variance are $\mathbb{E}[X] = \sum_{j=1}^{k} p_j/\lambda_j$ and $\mathrm{Var}[X] = 2\sum_{j=1}^{k} p_j/\lambda_j^2 - (\mathbb{E}[X])^2$. The coefficient of variation generally increases with k. The hyperexponential is a special case of what is called a *mixture distribution*, where X has the distribution function F_j with probability p_j for $1 \le j \le k$. The CPU time consumed by a computer program has been found to follow an hyperexponential distribution in certain settings.

A very powerful tool for combining exponentials is the concept of phase-type distribution (see, e.g., Wolff 1989, pages 269 and 299, and Asmussen 1987). A random variable X has an exponential *phase-type distribution* if X can be expressed as $X = X_1 + \cdots X_R$, where $X_1, \ldots, X_k$ are independent exponentials with rates $\lambda_1, \ldots, \lambda_k$, and R is a random variable independent of the X_j's and taking its values in $\{1, \ldots, k\}$ as follows: For some real numbers $p_1, \ldots, p_{k-1}$, where $0 < p_j \le 1$, one has

$$\mathbb{P}[R = r] = \begin{cases} 1 - p_1 & \text{if } r = 1, \\ p_1 \cdots p_{r-1}(1 - p_r) & \text{if } 2 \le r \le k-1, \\ p_1 \cdots p_{k-1} & \text{if } r = k. \end{cases}$$

This can be interpreted as follows (see Figure 2.14). To construct X, first generate X_1 and exit with $X = X_1$ with probability $1 - p_1$. With probability p_1, go to phase 2: Generate X_2, exit with $X = X_1 + X_2$ with probability $1 - p_2$ and continue to phase 3 with probability p_2. In phase 3, generate X_3, exit with $X = X_1 + X_2 + X_3$ with probability $1 - p_3$, and so on. In phase k, the probability of exit is 1.

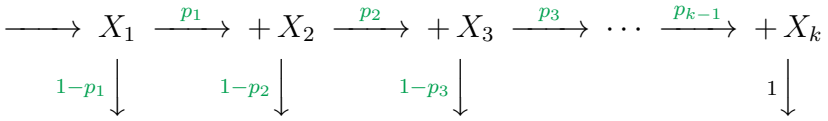


Fig. 2.14. Transition probabilities for a phase-type distribution

The following theorem motivates this class of phase-type distributions. It says that any distribution of a nonnegative random variable can be approximated arbitrarily closely by an exponential phase-type distribution.

Theorem 2.4 *(Wolff 1989, page 270.) If F is a cdf over $[0, \infty)$, then for any $\epsilon_1, \epsilon_2 > 0$, there is an exponential phase-type distribution H and a set $A \subset [0, \infty)$ such that $|H(t) - F(t)| < \epsilon_1$ for all $t \notin A$, and where A is a countable union of small intervals with total length less than ϵ_2.*

The set A in the theorem is needed when F has jumps. It is the union of intervals which are neighborhoods of the points of discontinuity of F. If F is a continuous distribution, then A can be taken as the empty set.

Example 2.9 One can generalize Example 2.8 as follows. A system for which all the events are scheduled with delays that are random variables with exponential or phase-type distributions is in fact a continuous-time Markov chain. Such a system can be simulated without an event list, as follows. In any given state, one can generate the time until the next transition (an exponential) and then determine the next state using the right transition probabilities (which depend on the current state). This type of implementation may run faster than with an event list and this could motivate the use of phase-type distributions in a simulation model (Fox 1993 and Exercise 2.15). If the state space is not too large, one can also avoid simulation altogether and compute the relevant performance measures by using matrix-theoretic methods to compute the steady-state probabilities of the corresponding continuous-time Markov chain. □

Phase-type distributions are studied in several books but are not much used in practice, perhaps because of a lack of support in simulation and distribution-fitting software.

2.8.26 Truncated Distributions

If a distribution has a positive density in an area larger than what we want, we may *truncate* it, i.e., set the density equal to 0 in the areas that we do not want, and rescale in the other areas so that the new density integrates to 1.

As an illustration, suppose the duration of an activity has a distribution close to the normal distribution. However, negative durations must be ruled out because they make no sense. More generally, suppose we want to use a density $f(x)$ but do not want values outside some interval (a, b) (where a or b can be infinite). We can use the truncated density

$$\widetilde{f}(x) = \begin{cases} f(x)/K & \text{for } a < x < b, \\ 0 & \text{elsewhere,} \end{cases} \tag{2.15}$$

where $K = \int_a^b f(u)du$ is the normalization constant required to recover a density. The same technique can also be applied to truncate other subsets of the real line instead of just the tails.

A simple way of generating random variates from the density $\widetilde{f}$ above, if the cdf F corresponding to f is easy to invert, is to generate $U \sim \mathsf{Uniform}(F(a), F(b))$ and return $X = F^{-1}(U)$ (Exercise 2.16).

2.8.27 Shifted Distributions

Several distributions are defined over the interval $[0, \infty)$ (e.g., the exponential, lognormal, gamma, etc.). There are situations where it might be more appropriate to shift the distribution so that it starts from a point $a \neq 0$. If X_0 has a distribution F_0 defined over the interval $[0, \infty)$, then $X = X_0 + a$ has a distribution F over $[a, \infty)$ which is a shifted version of F_0. That is, $F(x) = \mathbb{P}[X \leq x] = \mathbb{P}[X_0 \leq x - a] = F_0(x - a)$.

In Example 1.38, for instance, the optimal IS density turns out to be an exponential with parameter λ, shifted by y_0.

When a is fixed, estimating the parameters of the distribution F from a set of data is no harder for $a \neq 0$ than for $a = 0$. On the other hand, estimating the value of a together with those parameters is much more difficult for certain distributions (e.g., the Weibull).

2.8.28 Mixture Distributions

A continuous random variable X whose density can be decomposed as

$$f(x) = \sum_{j=1}^{\infty} w_j f_j(x), \tag{2.16}$$

where each f_j is a density and the *weights* w_j sum to one, is said to have a *mixture distribution*. Usually, the weights are non-negative and only a finite number of them are nonzero. The hyperexponential distribution discussed in Section 2.8.25 is a special case of a mixture distribution.

Mixture distributions are appropriate for modeling a population where a subset of the population has one distribution, another subset has another distribution, etc. In the call center example, for instance, there could be different types of calls, each type having its own distribution for the call duration. If these distributions differ significantly, a mixture distribution might be appropriate to model the duration of a random call. For example, one type might be the calls having reached a wrong number (usually short calls), a second type might be calls to buy a standard product offered by the company (longer calls), and a third type might be from customers who want to negotiate a non-standard service (long calls with more variable durations).

Densities can also be defined by *uncountable mixtures*, of the form

$$f(x) = \int_{\Theta} f_\theta(x) w(\theta) d\theta, \tag{2.17}$$

where the parameter set Θ is a subregion of the real space, each f_θ is a density over $\mathbb{R}$, and w is a density over Θ. To generate X from the density f, one can first generate Y from the density w over Θ, then generate X from the density f_Y.

Mixtures are defined similarly for discrete distributions; the densities f, f_j, and f_θ are simply replaced by probability mass functions.

2.9 Empirical and Quasi-Empirical Distributions

2.9.1 Variants of the empirical distribution

Let $x_1, \ldots, x_n$ be a sample of n real-valued observations and let $x_{(1)}, \ldots, x_{(n)}$ be the values $x_1, \ldots, x_n$ sorted by increasing order (these are called the *order statistics*). The *empirical distribution* associated with this sample is a discrete distribution that assigns a probability $1/n$ to each of these n observations. The *empirical cdf* of $x_1, \ldots, x_n$ is the cdf of this empirical distribution; it is defined as

$$\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[x_i \le x]. \tag{2.18}$$

This is a step function increasing from 0 to 1, with a step of size $1/n$ at each observation x_i, as shown in Figure 2.15. (A step of size k/n occurs whenever k observations are equal.) Generating data from the empirical distribution is equivalent to selecting i.i.d. observations from the discrete uniform distribution over $\{x_1, \ldots, x_n\}$, i.e., sampling randomly from this set, with replacement.

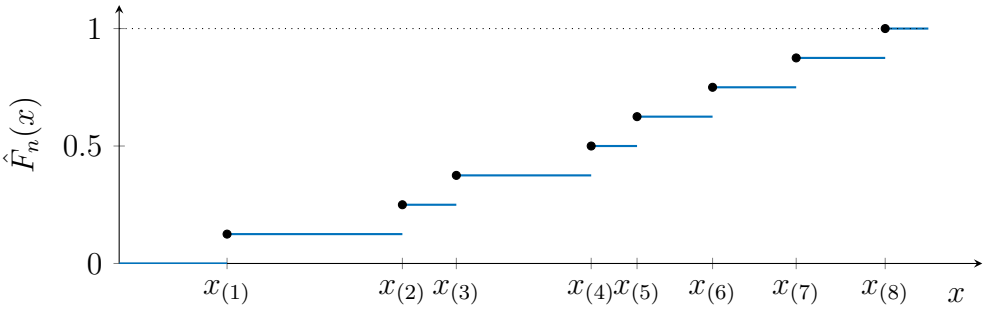


Fig. 2.15. An empirical cdf with $n = 8$

When the x_i are independent realizations of a random variable X with unknown cdf F, this empirical cdf $\hat{F}_n$ (viewed as a random function) is a natural estimator of F. In this case, for any fixed $x \in \mathbb{R}$, $\hat{F}_n(x)$ is the average of n independent Bernoulli random variables with parameter (mean) $p = F(x)$, so from Theorem 1.1 $\hat{F}_n(x) \overset{\text{w.p.1}}{\to} F(x)$ and

$$\frac{\sqrt{n}|\hat{F}_n(x) - F(x)|}{F(x)(1 - F(x))} \;\Rightarrow\; \mathsf{N}(0,1) \text{ when } n \to \infty.$$

This proves that we have pointwise convergence, for any fixed x. The following stronger result shows that the convergence is in fact uniform in x.

For any given fixed cdf F, the $\mathcal{L}_\infty$-distance (or *Kolmogorov-Smirnov distance*) between $\hat{F}_n$ and F is defined as

$$D_n = D_n(F) = \sup_{-\infty < x < \infty} |\hat{F}_n(x) - F(x)|. \tag{2.19}$$

This D_n is also known as the *Kolmogorov-Smirnov test statistic*. When the X_i are generated from cdf F, D_n can be used to measure the estimation error of F by $\hat{F}_n$. A fast algorithm

to compute the cdf $\mathbb{P}[D_n \le x]$ or its complement $\mathbb{P}[D_n > x]$ with good accuracy for any n or x can be found in Simard and L'Ecuyer (2011), with code in C and Java.

The *Glivenko-Cantelli theorem* states that $D_n \overset{\text{w.p.1}}{\to} 0$ when $n \to \infty$. Kolmogorov has proved that for any cdf F (discrete or continuous),

$$\sqrt{n} D_n \;\Rightarrow\; K \overset{\text{def}}{=} \sup_{-\infty \le x \le \infty} |B(F(x))|$$

where B is a standard Brownian bridge. When F is continuous, we have $K = \sup_{0 \le t \le 1} |B(t)|$ and the distribution of this random variable K is known as the Kolmogorov distribution. The theorem implies that D_n converges as $\mathcal{O}(n^{-1/2})$ in probability.

We also have the following large-deviation result. It applies regardless of the form of F (e.g., the distribution can be continuous or discrete). This version, with the leading constant equal to 2, was proved by Massart (1990).

Theorem 2.5 (Dvoretzky-Kiefer-Wolfowitz inequality.) *Let $X_1, \ldots, X_n$ be an independent sample generated from any given cdf F. For any $\epsilon > 0$ and any $n > 0$, one has*

$$\mathbb{P}[\sqrt{n} D_n > \epsilon] \le 2e^{-2\epsilon^2}. \tag{2.20}$$

These results about D_n support the idea of using the empirical distribution $\hat{F}_n$ of observations obtained from the cdf F to generate new observations in a simulation model in place of using F, when F is unknown. However, sampling from $\hat{F}_n$ can only produce numbers that are already in the original sample. To get around this limitation, a first idea could be to replace the empirical distribution by the continuous piecewise-linear variant $\widetilde{F}_n$ defined by

$$\widetilde{F}_n(x) = \begin{cases} 0 & \text{if } x \le x_{(1)}, \\ \dfrac{i-1}{n-1} + \dfrac{x - x_{(i)}}{(n-1)(x_{(i+1)} - x_{(i)})} & \text{if } x_{(i)} \le x \le x_{(i+1)}, \\ 1 & \text{if } x \ge x_{(n)}. \end{cases} \tag{2.21}$$

This is illustrated by the green line in Figure 2.16. With this quasi-empirical distribution $\widetilde{F}_n$, any value in the interval $(x_{(1)}, x_{(n)})$ can be generated. In case we know that the true density really starts at 0, we might want to start the linear approximation at $(0, 0)$, as shown by the red line in Figure 2.16. Then, any value in $(0, x_{(n)})$ can be generated.

To be able to generate values outside these intervals, possibly up to infinity or down to minus infinity, one can add tails to the quasi-empirical distribution. As an illustration, Bratley, Fox, and Schrage (1987) propose the following quasi-empirical distribution which is piecewise linear up to $x_{(n-k)}$, and has an exponential tail on the right, adjusted so that the quasi-empirical distribution has the same mean as the sample (see Exercise 2.17). They *assume* that the density is positive on $(0, \infty)$, with $F(0) = 0$, then they define

$$\check{F}_n(x) = \begin{cases} \dfrac{i}{n} + \dfrac{x - x_{(i)}}{(x_{(i+1)} - x_{(i)})n} & \text{if } x_{(i)} \le x \le x_{(i+1)}, 0 \le i < n-k, \\ 1 - \dfrac{k}{n} \exp\left[-(x - x_{(n-k)})/\theta\right] & \text{if } x > x_{(n-k)}, \end{cases} \tag{2.22}$$

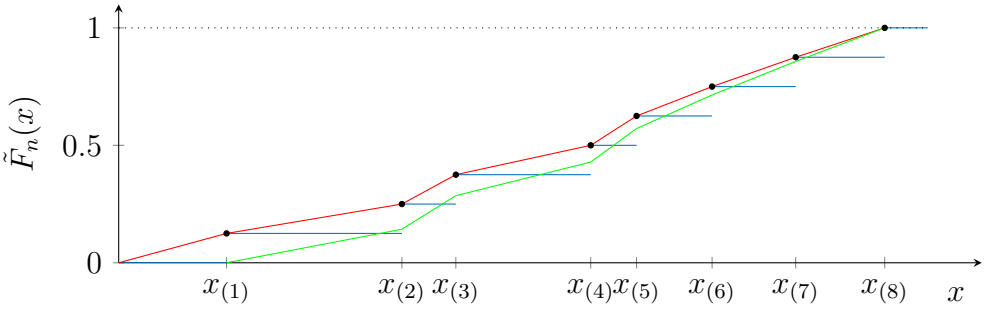


Fig. 2.16. The quasi-empirical cdf $\widetilde{F}_n(x)$ (in green) and a modified version that starts at 0 (in red), for the same $n = 8$ observations as in Figure 2.15. The points $(x_i, \hat{F}_n(x_i)) = (x_i, i/n)$ are shown in black.

where $x_{(0)} = 0$, k is a small positive integer (e.g., $1 \le k \le 5$), and

$$\theta \;=\; \frac{1}{k}\left(\frac{x_{(n-k)}}{2} + \sum_{i=n-k+1}^{n} (x_{(i)} - x_{(n-k)})\right).$$

The assumption that $F(0) = 0$ can be dropped with a minor modification of the distribution (Exercise 2.18). One can also change the exponential tail to a tail that has a different distribution, such as the Weibull or the gamma, for example. Bratley, Fox, and Schrage (1987) justify the exponential tail by their theorem 4.7.1, but their argument is not convincing because it would imply that the tail of any distribution over $[0, \infty)$ could be approximated by an exponential tail.

2.9.2 Approximating the inverse cdf directly

Since inversion is generally the preferred method for generating random variates, a cdf F for which F^{-1} has an explicit and simple expression has an edge over an F for which inversion is hard or impractical. Thus, it appears attractive to model F^{-1} directly by a flexible parameterized family and estimate the parameters from the available data. This was proposed by Hora (1983), who also suggested the general structure

$$F^{-1}(u) = F_0^{-1}(A(u)), \qquad 0 \le u \le 1,$$

where F_0 and A, called the *reference distribution* and the *adjustment filter*, can be parameterized. The function $A(u)$ must be nondecreasing in u, for $0 \le u \le 1$, and must satisfy $A(0) = 0$ and $A(1) = 1$. The purpose of the filter is to improve the fit by providing more flexibility. Hora (1983) proposes a filter of the form

$$A(u) = u^{\alpha_0} \exp\left[\sum_{j=1}^{k} \alpha_j (u^j - 1)/j\right],$$

and gives a method for estimating the parameters $\alpha_0, \ldots, \alpha_k$ via linear regression.

Avramidis and Wilson (1994) point out several problems with Hora's method. For instance, the estimated filter may be decreasing or undefined in some places in $(0, 1)$, in which case the estimated F^{-1} is not a valid inverse distribution, and the estimated inverse may differ substantially from $\hat{F}_n^{-1}$, the inverse of the empirical cdf. They propose a *polynomial filter* of the form $A(u) = \sum_{j=1}^{k} \beta_j u^j$, where $\beta_1 + \cdots + \beta_k = 1$, and where $A(u)$ must be increasing in u for $0 \le u \le 1$. They also propose (ordinary and weighted) least squares estimation procedures, formulated as constrained nonlinear optimization problems, for estimating the parameters β_j for a given value of k. The value of k is chosen by heuristics. According to Avramidis and Wilson (1994), the method seems to perform well on real-world data and a small value of k is usually sufficient for an acceptable fit.

2.9.3 Bézier distributions

Bézier curves were introduced by Pierre Bézier (1970) as design and control tools in the french car industry. They are widely used to model smooth curves and trajectories in font design, robotics, animation, and computer graphics. A two-dimensional *Bézier curve* with $k+1$ *control points* $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_k \in \mathbb{R}^2$ is defined as the locus $\{\boldsymbol{P}(t) = (x(t), y(t)),\ 0 \le t \le 1\}$, where

$$\boldsymbol{P}(t) = \sum_{j=0}^{k} B_{k,j}(t)\boldsymbol{p}_j$$

and the $B_{k,j}(t)$ are the *Bernstein polynomials* defined by

$$B_{k,j}(t) = \binom{k}{j} t^j (1-t)^{k-j} \qquad \text{for } 0 \le t \le 1.$$

The curve is parameterized by t. For each value of t, $\boldsymbol{P}(t)$ is a convex linear combination of the control points, with coefficients $B_{k,j}(t)$ that give more weight to the points with small [resp., large] indices when t is small [resp., large]. The coefficients are non-negative and sum to 1 for each $t \in [0, 1]$. At the extremities of the curve, one has $\boldsymbol{P}(0) = \boldsymbol{p}_0$ and $\boldsymbol{P}(1) = \boldsymbol{p}_k$. In between, the curve does not interpolate the control points, but these points act as magnets attracting the curve from some distance.

Bézier distributions provide a highly flexible class of alternatives to classical distributions (Wagner and Wilson 1996). The idea is to model the cdf F by a Bézier curve, assuming that F has a bounded support. The first and last control points, $\boldsymbol{p}_0 = (a, 0)$ and $\boldsymbol{p}_k = (b, 1)$, determine the boundaries of the distribution. The intermediate control points can then be chosen to obtain the desired shape of distribution, under the constraint that both $x(t)$ and $y(t)$ are increasing in t. The Bézier cdf F_B is defined by putting $F_B(x(t)) = y(t)$ for $0 \le t \le 1$.

Wagner and Wilson (1995) have developed a visual interactive software tool, called PRIME, for fitting Bézier distributions either subjectively or to available data. They also implemented a search procedure for finding the $x(t)$ that corresponds to a given $y(t)$ (without knowing t). This procedure can be used for generating random variates from F_B by inversion: generate $Y = Y(t) \sim U(0, 1)$ and compute $X = X(t) = F_B^{-1}(Y(t))$.

2.9.4 Density estimation from given data

[2] Suppose that our data set $x_1, \ldots, x_n$ comes from a distribution with cdf F having a density f for which we have no parametric form, and that we are really interested in generating new observations from a density $\hat{f}$ close to f. In other words, we want a good estimator for the density f.

Density of the quasi-empirical distribution. A first (naive) idea might be to take the density that corresponds to one of the quasi-empirical distributions defined in Section 2.9.1. The empirical distribution $\hat{F}_n$ is discrete, so it does not have a density. The quasi-empirical versions $\widetilde{F}_n$ and $\check{F}_n$ do have a density if we assume that all the observations are distinct, which happens with probability 1 if these observations come from a continuous distribution. Apart from the exponential tails, these densities are piecewise constant: for $x_{(i)} < x < x_{(i+1)}$, the densities at x are

$$\widetilde{f}_n(x) = \widetilde{F}_n'(x) = \frac{1}{(n-1)(x_{(i+1)} - x_{(i)})} \quad \text{and} \quad \check{f}_n(x) = \check{F}_n'(x) = \frac{1}{n(x_{(i+1)} - x_{(i)})}.$$

These functions take very large values where the observations are close to each other. They give very irregular densities that do not converge to f when $n \to \infty$, as illustrated by the next example.

Example 2.10 Suppose the n observations $x_1, \ldots, x_n$ are generated from the uniform density over the interval $(0, 1)$, i.e., $f(x) = 1$ for $0 < x < 1$. Let $d_n = \min_{1 \le i < n}(x_{(i+1)} - x_{(i)})$, the distance between the nearest pair of observations. It can be shown (see L'Ecuyer, Cordeau, and Simard 2000, Proposition 1, and the references given there) that for large n, the random variable $n(n-1)d_n$ has approximately the exponential distribution with mean 1. The maximum value of the density $\widetilde{f}_n$ is $\max_{0 \le x \le 1} \widetilde{f}_n(x) = 1/[(n-1)d_n]$. The probability that this maximum exceeds any given positive real number y is then

$$P\left[\max_{0 \le x \le 1} \widetilde{f}_n(x) > y\right] = \mathbb{P}[n(n-1)d_n < n/y] \approx 1 - e^{-n/y}.$$

For any fixed y, however large it is, this probability converges to 1 exponentially fast as a function of n. For example, by taking $n = 10y$, the probability that $\widetilde{f}_n(x)$ exceeds y at some point is approximately $1 - e^{-10} \approx 0.99995$. This means that the density $\widetilde{f}_n$ has narrow peaks that grow higher and higher as n increases, so it is an awful estimator of f. □

For the majority of simulation problems, a good approximation of the cdf F should be good enough and we don't really care if the density f is badly approximated. But there are cases for which we should care. For example, if we want to estimate the distribution of the distance between the nearest points when n points are generated from a distribution F, where F itself is estimated from data, then using $\widetilde{F}_n$ to generate the points would give a highly biased answer. See also Exercise 2.22.

[2] From Pierre: Perhaps this should be a separate subsection.

Quality measures for density estimators. Before looking further at specific density estimators, we need to decide how we measure their quality. That is, if $\hat{f}_n$ denotes an estimator of the density f with a sample size n, we need to select a measure of discrepancy (or distance) between $\hat{f}_n$ and f. There are many ways of defining such a measure. In this book, we will restrict ourselves to the most popular one, the *mean integrated square error* (MISE), defined below. We assume that we want to estimate the unknown density f over a fixed finite interval $[a, b]$. We measure the error at any given point $x \in [a, b]$ by the MSE:

$$\begin{aligned} \mathrm{MSE}[\hat{f}_n(x)] &= \mathbb{E}[\hat{f}_n(x) - f(x)]^2 = \mathbb{E}[\hat{f}_n(x) - f(x)]^2 + (\mathbb{E}[\hat{f}_n(x)] - f(x))^2 \\ &= \mathrm{Var}[\hat{f}_n(x)] + \mathrm{Bias}^2[\hat{f}_n(x)]. \end{aligned}$$

Then we integrate this measure with respect to x over $[a, b]$ to obtain the MISE, which decomposes as the sum of the *integrated variance* (IV) and *integrated squared bias* (ISB):

$$\begin{aligned} \mathrm{MISE} &= \mathbb{E}\int_a^b (\hat{f}_n(x) - f(x))^2 \mathrm{d}x \\ &= \int_a^b \mathrm{Var}[\hat{f}_n(x)]\mathrm{d}x + \int_a^b (\mathbb{E}[\hat{f}_n(x)] - f(x))^2 \mathrm{d}x = \mathrm{IV} + \mathrm{ISB}. \end{aligned} \tag{2.23}$$

We also define AMISE, AIV, and AISB as the asympotic values of these measures when $n \to \infty$. For example, $\mathrm{AMISE} = \kappa n^{-\nu}$ means that $\lim_{n\to\infty} n^\nu \mathrm{MISE} = \kappa$. Minimizing the MISE or AMISE involves a bias-variance tradeoff. Other possible error measures could be the Kullback-Leibler distance, the Hellinger distance, or some other L_p distance for $p \neq 2$, but the L_2 distance (the MISE) is simple and more standard (Scott 2015).

In our analysis, for any $g : [a, b] \to \mathbb{R}$, we define the *roughness* of g as

$$R(g) = \int_a^b (g(x))^2 \mathrm{d}x.$$

For any $\phi : \mathbb{R} \to \mathbb{R}$, we use the shorthand notation:

$$\mu_r(\phi) = \int_{-\infty}^{\infty} x^r \phi(x)\mathrm{d}x \quad \text{for } r = 0, 1, 2, \ldots$$

Histograms. A simple and popular way of estimating the density is to make a *histogram*. We can partition the interval into m equal parts of size $h = (b-a)/m$, and count the number n_j of observations that fall in each interval $B_j = [a + (j-1)h,\ a + jh)$, for $j = 1, \ldots, m$. The *histogram density estimator* (HDE) $\hat{f}_{\mathrm{h},n}$ is constant over each interval, and proportional to the number of observations in that interval:

$$\hat{f}_{\mathrm{h},n}(x) = \frac{n_j}{nh} \quad \text{for } x \in B_j, \quad j = 1, \ldots, m.$$

Generating random variates from such a piecewise-constant density is easy. A key question is how do we choose the *bandwidth* (or rectangle width) h.

A very small h gives a noisy histogram, with large variations of $f_{\mathrm{h},n}$ across successive intervals. A larger h gives a smoother histogram, but also introduces bias in the sense that $f_{\mathrm{h},n}$ tends to be smaller than f where f is large (in the peaks) and larger where f is small

(in the valleys). When h is too large, this bias becomes excessive; this is *oversmoothing*. To complicate things further, a given h may oversmooth in some areas and undersmooth in other areas, so a proper choice may often require visual (human) judgment (Scott 2004). The quality could often be improved by using a different h in different areas, but we will not examine this here.

♣ Add an example based on the histogram in Example 1.4, for instance if we use 10 rectangles or if we use 100 rectangles. Perhaps more interesting: an example with a multimodal density.

Under the assumption that f has an absolutely continuous and square-integrable derivative over $[a, b]$, Scott (1979) showed that the AIV and AISB for the HDE and $1/(nh)$ and $h^2R(f')/12$, respectively. The proofs are simple; they exploit the fact that $n_k \sim \mathsf{Binomial}(n, p_k)$ where $p_k = \int_{B_k} f(x)\mathrm{d}x$ to obtain the variance, and use a Taylor expansion of f in each interval B_k to bound the bias. See also Scott (2015), Section 3.2.2. For the AMISE to converge to 0, we must have $h \to 0$ and $nh \to \infty$ simultaneously. The asymptotically optimal h, which minimizes the AMISE, is then $h^* = [6/(nR(f')]^{1/3}$ and it gives AMISE $= (9R(f')/16)^{1/3}n^{-2/3} = \mathcal{O}(n^{-2/3})$. The corresponding number of intervals is $m = (b-a)/h = \mathcal{O}(n^{1/3})$.

For comparison, when estimating the parameters for a *parametric* family of distributions, under the assumption that the true density *really belongs to that family*, the convergence rate is typically $\mathcal{O}(n^{-1})$. In general, parametric methods are more efficient than the nonparametric ones in this sense, but we must know the correct parametric family in the first place!

A simple improvement to the histogram estimator is the (piecewise-linear) *polygonal density estimator* (PDE), also called *frequency polygon*, defined as follows: In the histogram constructed as before, put a point in the middle of the upper bound of each rectangle and interpolate these points by linear segments to define the density. More specifically, the density estimator is defined by joining the points $(a, n_1/(n(b-a)))$, $(a+h/2, n_1/(n(b-a)))$, $(a+3h/2, n_2/(n(b-a)))$, $\ldots$, $(b-h/2, n_m/(n(b-a)))$, $(b, n_m/(n(b-a)))$. Under the assumption that f'' is absolutely continuous and $R(f''') < \infty$, Scott (1985b) has shown that for the PDE, AIV $= 2/(3nh)$ and AISB $= 49h^4R(f'')/2880$. The asympotically optimal h is then $h^* = 2[15/(49nR(f'')]^{1/5}$, which gives AMISE $= (5/12)[49R(f'')/15]^{1/5}n^{-4/5} = \mathcal{O}(n^{-4/5})$. We have a better convergence rate!

Another simple way of improving the HDE is to average several histograms, each one being obtained by shifting slightly all the bin boundaries by the same amount, as proposed by Scott (1985a). The idea is to construct $r \geq 1$ distinct histograms all with bin width h, but with the bin separators shifted to the right by $(\ell - 1)h/r)$ for the ℓth histogram, for $\ell = 1, \ldots, r$, and then average these histograms. More specifically, we select $r \geq 1$, let $\delta = h/r$, and partition $[a, b)$ into the rm subintervals $I_k = [a + (k-1)\delta, a + k\delta)$ of size δ, $k = 1, \ldots, rm$. This partitions B_j into r such subintervals. Let $\tilde{n}_k$ be the number of observations in interval I_k. We put $\tilde{n}_k = 0$ for $k < 0$ and for $k > rm$. Bin j of the ℓth shifted histogram covers the interval $B_{j,\ell} = [a+(j-1)h+(\ell-1)\delta,\ a+jh+(\ell-1)\delta)$ and contains all the observations in this interval. Here we assume that for every shift, the shifted histogram covers all the observations. Otherwise, we must apply some kind of correction. [3]

[3] From Pierre: Details?

When adding the r histograms, the total count in interval I_k becomes

$$T_k = r\tilde{n}_k + \sum_{\ell=1}^{r-1}(r-\ell)(\tilde{n}_{k-\ell} + \tilde{n}_{k+\ell}). \tag{2.24}$$

This sum T_k weights the counts in the small bins with weights that decrease linearly with their distance to bin k. Averaging the r corresponding histogram density estimators gives the *average shifted histogram* (ASH) density estimator, defined by

$$\hat{f}_{\text{ash},n}(x) = \frac{T_k}{nhr^2}. \tag{2.25}$$

This method uses overlapping intervals to construct the histogram. It generally produces a smoother histogram than $\hat{f}_{\text{h},n}$. [4] Scott (1985a) derived Taylor series expansions for the squared bias and variance of this ASH estimator as functions of h and r, and obtained that AIV $= 2(1 + 1/2r^2)/(3nh)$ and AISB $= (h^2/12r^2)R(f') + (h^4/144)(1 - 2/r^2 + 3/5r^4)R(f'')$. When $r \to \infty$, this gives AMISE $= 2/(3nh) + h^2R(f'')/144$, which is minimized with $h^* = [24/(nR(f''))]^{1/5}$, which provides AMISE $= \mathcal{O}(n^{-4/5})$. In comparison with the PDE, all the rates are the same. The only difference is that the constant $1/144$ in the AMISE here is only about 40% of the corresponding constant $49/2880$ for the PSE, so the ASH wins due to this improved constant. One can also apply polygonal interpolation to the ASH estimator, but for large r, this *polygonal ASH* provides no further improvement compared with the ASH alone.

The computing effort required to produce the ASH increases at least linearly in r, so assuming that $r \to \infty$ is unrealistic. But according to Scott (1985a), Scott (2015), there is usually no need to take r larger than 10 or so. Increasing r further has almost no effect on the MISE and is not worth the additional effort. In the limit when $n \to \infty$, the ASH and polygonal ASH become equivalent to a kernel density estimator based on a triangular kernel with bandwidth h.

The T_k defined in Eq. (2.24) can be seen as a weighted sum with weight $w_\ell = r - \ell$ given to observations that are at a (integer) distance ℓ from the target bin k. These weights decrease linearly until they reach 0. Other types of weights can be used as well and some may give smoother histograms. [5]

♣ Add some figures.

Kernel density estimators. Histograms are simple and easy to compute and interpret, and are often preferred for this reason. However, *kernel density estimators* (KDEs) are a more powerful class of methods for density estimation. The general form of a KDE is

$$\hat{f}_n(x) = \frac{1}{nh}\sum_{i=1}^{n} k((x - x_i)/h), \tag{2.26}$$

[4] From Pierre: Exercise: Show this in details. Show also that $T_1, \ldots, T_{rm}$ can be computed in $\mathcal{O}(rm)$ time, recursively as follow. Define $S_{1,k} = \sum_{\ell=0}^{r-1} \tilde{n}_{k-\ell}$ and $S_{2,k} = \sum_{\ell=1}^{r-1} \tilde{n}_{k+\ell}$. One has $T_{k+1} = T_k + S_{2,k} - S_{1,k}$, $S_{1,k+1} = S_{1,k} + \tilde{n}_{k+1} - \tilde{n}_{k-r+1}$, and $S_{2,k+1} = S_{2,k} + \tilde{n}_{k+r} - \tilde{n}_{k+1}$. We start by computing T_1, $S_{1,1}$, and $S_{2,1}$, and the apply the recursion.

[5] From Pierre: More details later, perhaps in an exercise.

where the *kernel* k is a fixed density and h is a positive constant called the *smoothing factor* or *bandwidth*. We assume here that the kernel is a symmetric probability density centered at 0, with variance $\sigma_k^2 = \mu_2(k) < \infty$. With the change of variable $y = (x - x_i)/h$, one can easily see that

$$\frac{1}{h}\int_{-\infty}^{\infty} k((x - x_i)/h)dx = \int_{-\infty}^{\infty} k(y)dy = 1,$$

so $\hat{f}_n(x)$ is a probability density. Here, we assume a symmetric kernel for simplicity, but asymmetric ones can be used as well. Figure 2.17 illustrates four common choices of kernels: uniform, triangular, Epanechnikov, and (standard) normal. The Epanechnikov kernel is a beta distribution with parameters $(2, 2)$ stretched over the interval $[-1, 1]$. Its density is $k(x) = (3/4)(1-x^2)$ for $-1 \le x \le 1$. The normal kernel has infinite support, while the other three have bounded support over $[-1, 1]$. When the density f is suspected to have a heavy tail, one may use a Student-t or logistic kernel instead of the normal.

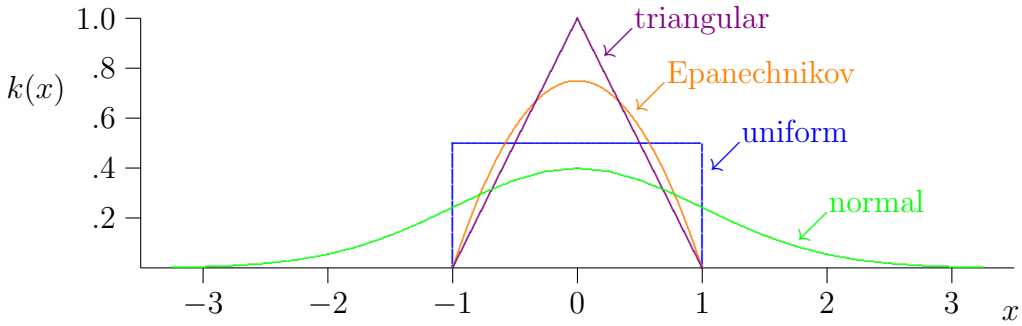


Fig. 2.17. The uniform (blue), triangular (violet), Epanechnikov (orange), and normal (green) kernels.

Generating random variates from the density (2.26) is easy, provided that we can easily generate variates from the density k. It suffices to (1) generate I uniformly over $\{1, \ldots, n\}$, (2) generate D from density k, independently of I, and (3) return $X = x_I + hD$. This technique of redrawing observations randomly from the sample, with replacement, and adding an independent perturbation D, is identical to *smoothed bootstrap* (see Chapter 5 and Shao and Tu 1995). It is *not an inversion method* even if we use inversion in each of the first two steps, because the different copies of the kernel density usually overlap. Inversion for distributions obtained via kernel density estimators is generally hard to implement.

Since the kernel density is used to add independent noise to the empirical distribution, it is clear that the variance of the resulting density (2.26) is always larger than the variance of the empirical distribution of the original data (Exercise 2.21). Silverman (1986) suggests to rescale the empirical distribution so that these two variances match exactly. Let $\bar{x}_n$ and $\widetilde{s}_n^2 = s_n^2(n-1)/n$ be the mean and variance of the empirical distribution of the original observations. The distance between each generated random variate and the sample mean $\bar{x}_n$ is multiplied by the correction factor $1/\sigma_e < 1$, where $1/\sigma_e^2 = 1 - (h\sigma_k/s_n)^2 n/(n-1)$ and σ_k is the standard deviation of the density k. The effect of this transformation is to move all the observations closer to $\bar{x}_n$, to reduce their empirical variance.

There are situations where the random variate X *must* be restricted to a given interval of the real line; e.g., the interval $[0, \infty)$ if we do not want negative values. However, density estimators provided by kernel methods will typically be positive outside that interval (e.g., with a normal kernel, the density is positive everywhere). A simple heuristic to cover that case is to *reflect* the values that fall outside with respect to the boundary. That is, if X must be restricted to some interval $[a, \infty)$ and we happen to generate $X < a$, then we return $2a - X$ instead of X. This reflection method reduces the variance of the estimated density and this can be a problem in cases where a significant fraction of the kernel density lies outside the target interval.

A wide variety of choices for the kernel and of rules for selecting h as a function of the observations $x_1, \ldots, x_n$, have been proposed. No method is universally optimal. Each one performs well for its own favorite class of densities. It also depends on how we decide to measure the discrepancy (or error) between the true density and its estimator f_n. [6] Experimental evidence indicates that the choice of bandwidth h is usually more important than the choice of kernel (Hörmann, Leydold, and Derflinger 2004).

Under our assumptions on k, if f'' is absolutely continuous and $R(f''') < \infty$, then one has (see Scott 2015, Section 6.2) AIV $= \mu_0(k^2)/(nh)$ and AISB $= \sigma_k^2 R(f'')h^4/4$. The AMISE is minimized by taking the bandwidth h equal to

$$h^* = \left[\mu_0(k^2)/(\sigma_k^2 R(f'')n)\right]^{1/5}, \tag{2.27}$$

which gives

$$\text{AMISE} = (5/4)[\mu_0(k^2)\sigma_k]^{4/5} R(f'')^{1/5} n^{-4/5}.$$

The kernel-dependent factor $\mu_0(k^2)\sigma_k$ in this AMISE is minimized by the Epanechnikov kernel, for which $\mu_0(k^2) = 3/5$ and $\sigma_k = 1/5$.

To estimate h^*, we need an estimate of $R(f'')$, which is of course unknown because f is what we want to estimate in the first place. A *plug-in* approach is often successful for estimating $R(f'')$ (Berlinet and Devroye 1994, Jones, Marron, and Sheather 1996, Raykar and Duraiswami 2006, Scott 2015). The idea is to estimate f'' via a KDE and integrate its square over $[a, b]$. To do this, we also need to select a bandwidth h which requires a good estimate of $R(f^{(4)})$, where $f^{(4)}$ is the fourth derivative of f. This $R(f^{(4)})$ can be estimated by integrating a KDE of $f^{(4)}$, and this goes on forever. In practice, one can start with a rough estimate of $R(f^{(r_0+2)})$ for some even $r_0 \geq 0$ and apply the plug-in approach from there. An easy way to do this is to pretend that f is a normal density with mean and variance equals to their sample values in the data, and compute $R(f^{(r_0+2)})$ for this normal density. To estimate $f^{(r)}$ for $r \leq r_0$, one can take the sample derivative of the KDE with a smooth kernel k:

$$\hat{f}_n^{(r)}(x) \approx \frac{1}{nh^{r+1}} \sum_{i=0}^{n-1} k^{(r)}\left(\frac{x - X_i}{h}\right), \tag{2.28}$$

using a bandwidth h which is a good estimate of the asymptotically optimal value

$$h_*^{(r)} = \left(\frac{(2r+1)\mu_0((k^{(r)})^2)}{\sigma_k^2 R(f^{(r+2)})n}\right)^{1/(2r+5)}. \tag{2.29}$$

[6] From Pierre: However, using a normal kernel when the distribution has a finite tail tends to inflate the tail, it seems. Make experiments and give a concrete example of this.

Ben Abdellah et al. (2021) had good success with this approach, with $r_0 = 2$.

Silverman (1986) and Hörmann, Leydold, and Derflinger (2004) propose the simpler alternative (heuristic) formula $h = \alpha_k h_0$, where

$$h_0 = 1.36374 \ \min(s_n, q/1.34) n^{-1/5}, \tag{2.30}$$

s_n and q are the empirical standard deviation and the interquartile range of the n observations, whereas α_k is a constant that depends on the type of kernel k and is defined by

$$\alpha_k = \left(\sigma_k^{-4} \int_{-\infty}^{\infty} k^2(x) dx \right)^{1/5}. \tag{2.31}$$

This formula comes from a heuristic adjustment of Eq. (2.27), in which s_n is deflated to avoid oversmoothing when q is small. Table 2.1, adapted from Hörmann, Leydold, and Derflinger (2004), gives the precomputed values of α_k and σ_k^2 for selected (popular) kernels. The *efficiency* of a kernel k is defined as the ratio of the AMISE of the Epanechnikov kernel (which has optimal efficiency) divided by the AMISE of k.

Table 2.1. Some suggested kernels and the constants involved in the Silverman heuristic.

name	distribution	range	α_k	σ_k^2	efficiency
Epanechnikov	$2\,\mathsf{Beta}(2,2) - 1$	$[-1, 1]$	1.7188	1/5	1.000
triangular	$\mathsf{Triangular}(-1, 1, 0)$	$[-1, 1]$	1.8882	1/6	0.986
uniform	$U(-1, 1)$	$[-1, 1]$	1.3510	1/3	0.930
normal	$\mathsf{N}(0, 1)$	$(-\infty, \infty)$	0.7764	1	0.951
logistic	$\mathsf{Logistic}(0, 1)$	$(-\infty, \infty)$	0.4340	3.2899	0.888
Student-t(3)	$\mathsf{Student\text{-}t}(3)$	$(-\infty, \infty)$	0.4802	3	0.674

Here we have assumed that h must be the same everywhere in $[a, b]$. In general it is possible to improve the KDE locally varying bandwidth $h(x) > 0$ to estimate the density at x. To get an idea of how, note that the optimal $1/h$ is proportional to $nR(f'')$. In a small subinterval $[c, c + \epsilon] \subset [a, b]$ in which f'' is almost constant, the expected number of data points is $n_1 = n \int_c^{c+\epsilon} f(x)\mathrm{d}x$, so $1/h$ in this subinterval should be approximately proportional to $n \int_c^{c+\epsilon} f(x)(f''(x))^2 \mathrm{d}x \approx n_1 (f''(c + \epsilon/2))^2$. That is, the value of $1/h$ used at some point x should be approximately proportional to $f(x)(f''(x))^2$. This makes sense when the interval $[a, b]$ does not contain thin tails or sub-regions in which $\max_x f(x)/\min_x f(x)$ is very large.

♣ To be added elsewhere: MISE reduction by RQMC and other methods.

Summary of convergence rates for histograms and KDEs. We now summarize the convergence rate results for the density estimators discussed so far, when the data comes from independent random samples, under the assumption that h is constant over $[a, b]$. Here, the AMISE depends on both n and h. For all methods, we have

$$\text{AMISE} \ = \ \text{AIV} + \text{AISB} = \frac{C}{nh} + Bh^{\alpha}$$

for some constants C, B, and α. The *asymptotically optimal h* as a function of n is

$$h^* = (C/(B\alpha n))^{1/(\alpha+1)}$$

and it gives

$$\text{AMISE} = \frac{\alpha+1}{\alpha}(C^\alpha B\alpha)^{1/(\alpha+1)} n^{-\alpha/(1+\alpha)}.$$

Table 2.2 summarizes the values obtained for the different methods. To estimate h^*, one can estimate $R(f')$ and $R(f'')$ by using a plugin approach as described earlier.

Table 2.2. Optimal bandwidths and convergence rates for histograms and KDEs. The given AMISE rate is for $h = h^*$.

	C	B	α	h^*	AMISE
HDE	1	$R(f')/12$	2	$\left(\frac{6}{nR(f')}\right)^{1/3}$	$\mathcal{O}(n^{-2/3})$
PDE	2/3	$49R(f'')/2880$	4	$\left(\frac{480}{49nR(f'')}\right)^{1/5}$	$\mathcal{O}(n^{-4/5})$
ASH	2/3	$R(f'')/144$	4	$\left(\frac{24}{nR(f'')}\right)^{1/5}$	$\mathcal{O}(n^{-4/5})$
KDE	$\mu_0(k^2)$	$\sigma_k^2 R(f'')/4$	4	$\left(\frac{\mu_0(k^2)}{n\sigma_k^2 R(f'')}\right)^{1/5}$	$\mathcal{O}(n^{-4/5})$

Density estimation methods for *multivariate distributions* are discussed in Section 2.10.7. See Scott (2015).

2.10 Multivariate Distributions

Independence of random variables is very often *presumed* in simulation models for reasons of simplicity, or lack of information, or because the available software supports only univariate distributions. There are many situations, however, where neglecting the dependencies makes the model grossly invalid.

In Example 2.4, for instance, the durations of the successive operations performed on a given part could be strongly dependent. In a health-care facility model, the relevant characteristics of a patient arriving to the facility should probably be modeled as a vector of dependent random variables. The inter-arrival times to a convenience store near the local train station should probably be modeled as positively correlated, because arrivals would tend to come in bursts (just after train arrivals). This list of examples could go on.

There are two general ways of modeling dependencies: (1) using vectors of random variables with a given multivariate distribution and (2) using time series and other types of stochastic processes. The main challenge is to model the dependence between the vector components or between the successive observations.

We recall from Section A.7 that a random vector $\boldsymbol{X} = (X_1, \ldots, X_d)^\mathsf{t}$ has a d-dimensional *multivariate distribution* with cdf F if for any $\boldsymbol{x} = (x_1, \ldots, x_d)^\mathsf{t} \in \mathbb{R}^d$,

$$F(\boldsymbol{x}) = \mathbb{P}[\boldsymbol{X} \leq \boldsymbol{x}] = \mathbb{P}[X_1 \leq x_1, \ldots, X_d \leq x_d].$$

The jth *marginal* is defined by $F_j(x) = \mathbb{P}[X_j \leq x]$. The random variables $X_1, \ldots, X_d$ are *independent* if and only if $F(x_1, \ldots, x_d) = F_1(x_1) \cdots F_d(x_d)$. Multivariate distributions are normally used to represent vectors of *dependent* random variables. When $d = 2$, we speak of a *bivariate* distribution.

For most distributions, the generalization from univariate to multivariate is not unique, in the sense that there are many ways of defining F so that each F_j is from a specific family or even completely fixed in advance. For example, there is an infinite number of bivariate distributions whose marginals are both exponential with mean 1. This complicates the problem of specifying and using such distributions. If we also consider multivariate distributions whose marginals F_j are from *different families* (which is often useful), there are even more possibilities. When all marginals F_j are from the same family, we use the name of that family to denote the multivariate distribution. For example we speak of a *multivariate exponential* [*normal*, *gamma*, etc.] distribution if all F_j's are exponential [normal, gamma, etc.].

Specific multivariate distributions have been proposed and studied in the literature, sometimes together with methods for generating the random vectors. We look at a few of them here and refer the reader to Johnson and Kotz (1972b), Johnson (1987), Devroye (1986), Joe (1997), Nelsen (1999), Wilson (1997) and Nelson and Yamnitsky (1998) for several others.

2.10.1 Covariance and correlation

The most popular measures of dependence between two random variables X and Y are the *covariance* $\mathrm{Cov}(X, Y)$, and its normalization the linear *coefficient of correlation* $\rho(X, Y)$, defined in Section A.8. The *covariance matrix* $\boldsymbol{\Sigma} = \mathrm{Cov}[\boldsymbol{X}]$ has elements $\sigma_{ij} = \mathrm{Cov}[X_i, X_j]$ and the *correlation matrix* $\boldsymbol{R}$ has elements $\rho(X_i, X_j)$. One can write $\mathrm{Cov}[\boldsymbol{X}] = \mathbb{E}[(\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{X} - \boldsymbol{\mu})^{\mathsf{t}}]$ where $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{X}]$. If $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{b}$ where $\boldsymbol{A}$ is a $d \times d$ matrix and $\boldsymbol{b}$ is a d-dimensional vector, then $\mathbb{E}[\boldsymbol{Y}] = \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}$ and $\mathrm{Cov}[\boldsymbol{Y}] = \mathbb{E}[\boldsymbol{A}(\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{A}(\boldsymbol{X} - \boldsymbol{\mu}))^{\mathsf{t}}] = \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^{\mathsf{t}}$.

Any *valid* (or *consistent*) covariance matrix $\boldsymbol{\Sigma}$, i.e., such that $\boldsymbol{\Sigma} = \mathrm{Cov}(\boldsymbol{X})$ for some random vector $\boldsymbol{X}$, must be *symmetric and nonnegative definite*, because for any vector $\boldsymbol{a} \in \mathbb{R}^d$, $\boldsymbol{a}^{\mathsf{t}}\boldsymbol{\Sigma}\boldsymbol{a} = \mathrm{Var}(\boldsymbol{a}^{\mathsf{t}}\boldsymbol{X}) \geq 0$. A valid correlation matrix must also have all its diagonal elements equal to 1. On the other hand, for given marginal distributions F_j for $\boldsymbol{X}$ and a given matrix that satisfies all these conditions, there may be an infinite number of possibilities for the distribution of $\boldsymbol{X}$, or there may be no possibility at all.

Let X and Y be random variables with finite variance and cdf's F and G, respectively. The range of values that $\rho(X, Y)$ can take depends on F and G. This range is always a subset of $[-1, 1]$ and always contains 0 (because X and Y can be taken as independent). The minimum and maximum values are the *Fréchet bounds* (Fréchet 1957), given in the following theorem, which also provides a concrete way of generating a pair (X, Y) with minimal or maximal correlation. We will rely on this theorem when studying correlation induction methods for variance reduction, in Chapter 6.

Theorem 2.6 *(Hoeffding 1940, Fréchet 1951, Lehmann 1966, Whitt 1976) Let F and G be two cdf's with finite variances. Then, among the pairs of random variables (X, Y) with marginal distributions F and G, the pair*

$$(X, Y) = (F^{-1}(U),\, G^{-1}(U)),$$

where $U \sim U(0,1)$, maximizes the correlation between X and Y, whereas the pair

$$(X, Y) = (F^{-1}(U),\, G^{-1}(1-U))$$

minimizes the correlation between X and Y. Equivalently, the joint distribution with maximal correlation is the one defined by $\mathbb{P}\{X \le x,\, Y \le y\} = \min(F(x),\, G(y))$, and the one with minimal correlation is defined by $\mathbb{P}\{X \le x,\, Y \le y\} = \max(0,\, F(x) + G(y) - 1)$.

Proof. See, for example, Lehmann (1966) or Whitt (1976) for the first part. For the first equivalence in the second part, we have

$$\begin{aligned} P\{X \le x,\, Y \le y\} &= \mathbb{P}[F^{-1}(U) \le x,\, G^{-1}(U) \le y] \\ &= \mathbb{P}[U \le F(x),\, U \le G(y)] \\ &= \mathbb{P}[U \le \min(F(x), G(y)] \\ &= \min(F(x), G(y)). \end{aligned}$$

For the other equivalence,

$$\begin{aligned} P\{X \le x,\, Y \le y\} &= \mathbb{P}[F^{-1}(U) \le x,\, G^{-1}(1-U) \le y] \\ &= \mathbb{P}[U \le F(x),\, 1-U \le G(y)] \\ &= \mathbb{P}[1 - G(y) \le U \le F(x)] \\ &= \max(0,\, F(x) + G(y) - 1). \end{aligned}$$

Example 2.11 Let $U \sim U(0,1)$ and $Z = \Phi^{-1}(U)$, where Φ is the standard normal cdf. Suppose we want $X \sim \mathsf{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathsf{N}(\mu_2, \sigma_2^2)$. The (dependent) random variables X and Y having these distributions and with *maximal correlation* can be written as

$$X = F^{-1}(U) = \mu_1 + \sigma_1 Z \quad \text{and} \quad Y = G^{-1}(U) = \mu_2 + \sigma_2 Z.$$

Their correlation is

$$\rho(X, Y) = \mathbb{E}[(X - \mu_1)(Y - \mu_2)]/(\sigma_1\sigma_2) = \mathbb{E}[\sigma_1 Z \sigma_2 Z]/(\sigma_1\sigma_2) = \mathbb{E}[Z^2] = 1.$$

Likewise, since $\Phi^{-1}(1-U) = -Z$, the random variables with *minimal correlation* are

$$X = F^{-1}(U) = \mu_1 + \sigma_1 Z \quad \text{and} \quad Y = G^{-1}(1-U) = \mu_2 - \sigma_2 Z$$

and their correlation is

$$\rho(X, Y) = \mathbb{E}[(X - \mu_1)(Y - \mu_2)]/(\sigma_1\sigma_2) = \mathbb{E}[\sigma_1 Z(-\sigma_2) Z]/(\sigma_1\sigma_2) = -\mathbb{E}[Z^2] = -1.$$

This means that for a random vector (X, Y) whose marginals are normal with arbitrary given parameters, the correlation $\rho(X, Y)$ can take any value in $[-1, 1]$.

The reader can verify that the same reasoning applies more generally to the case where the distributions of X and Y differ only by location and scale parameters, so $Y - \mathbb{E}[Y]$ can be written as a linear function of $X - \mathbb{E}[X]$ and vice-versa. □

Example 2.12 If $X \sim \mathsf{Lognormal}(0, 1)$ and $Y \sim \mathsf{Lognormal}(0, 4)$, then one can show (Exercise 2.23) that their minimal and maximal possible correlations are -0.090 and 0.666, respectively. □

2.10.2 Other measures of dependence

The Pearson correlation coefficient measures only *linear dependence* between random variables. For example, if $X \sim N(0, 1)$ and $Y = X^2$, then $\mathrm{Cov}[X, Y] = \mathbb{E}[X^3] - \mathbb{E}[X]\mathbb{E}[X^2] = 0$, so $\rho(X, Y) = 0$, yet X and Y are definitely dependent, because Y is determined uniquely by X. More generally, if X has a symmetric density f with respect to the origin and if $Y = \varphi(X)$ where the function φ satisfies $\varphi(-x) = \varphi(x) \neq 0$ for $x \neq 0$, then X and Y are dependent, but it is easily seen that $\rho(X, Y) = 0$ (Exercise 2.24). Independence implies absence of correlation, but the reverse is not true. Thus, the correlation coefficient is not always a good indicator of the level of dependence.

Moreover, $\rho(X, Y)$ depends not only on the level of dependence between X and Y, but also on their marginal distributions. Rescaling nonlinearly the horizontal axis of the marginals changes the value of $\rho(X, Y)$. Finally, $\rho(X, Y)$ exists only if both X and Y have finite variances.

Rank correlation. The *rank correlation* (also called *Spearman's rho* or *fractile correlation*) avoids this last problem: It always exists. It is defined as

$$\rho_{\mathrm{s}}(X, Y) = \rho(F(X), G(Y)). \tag{2.32}$$

In the case of *continuous* marginal distributions F and G, we have

$$\rho_{\mathrm{s}}(X, Y) = 12\,\mathbb{E}[F(X)G(Y)] - 3, \tag{2.33}$$

which can easily be deduced from the fact that both $F(X)$ and $G(Y)$ are $U(0, 1)$, so $\mathbb{E}[F(X)] = \mathbb{E}[G(Y)] = 1/2$ and $\mathrm{Var}[F(X)] = \mathrm{Var}[G(Y)] = 1/12$. This coefficient then measures the dependence independently of the choice of (continuous) marginals, in the sense that if $X = F^{-1}(U_1)$ and $Y = G^{-1}(U_2)$ for some uniforms U_1 and U_2, then $\rho_{\mathrm{s}}(X, Y) = \rho(U_1, U_2)$ is invariant with respect to a change in F or G. It can take any value in the range $[-1, 1]$, regardless of F and G. These properties do not hold if F [or G] is not continuous, because $F(X)$ [or $G(Y)$] is not $U(0, 1)$ in that case.

Kendall's tau. A related and also widely studied measure is *Kendall's tau*, motivated and defined as follows. Suppose we generate two independent pairs (X_1, Y_1) and (X_2, Y_2) with the same distribution as (X, Y), and draw a line that joints the two corresponding points in $\mathbb{R}^2$. We say the the two points are *concordant* if this line has a positive slope and *discordant* it if

has a negative slope. A positive dependence between X and Y should translate into a higher chance of positive slope, i.e., more frequent concordance than discordance. For a negative dependence, this should be the opposite. Kendall's tau is simply defined as the probability of concordance minus the probability of discordance:

$$\tau_{\mathrm{k}}(X,Y) = \mathbb{P}[(X_1 - X_2)(Y_1 - Y_2) > 0] - \mathbb{P}[(X_1 - X_2)(Y_1 - Y_2) < 0]. \tag{2.34}$$

When both F and G are continuous, we have

$$\tau_{\mathrm{k}}(X,Y) = 4\,\mathbb{E}[C(F(X), G(Y))] - 1 \tag{2.35}$$

where $C(u,v) = \mathbb{P}[F(X) \le u, G(Y) \le v]$ does not depend on the marginals F and G.

As for the linear correlation, $\rho_{\mathrm{s}}(X,Y) = 0$ or $\tau_{\mathrm{k}}(X,Y) = 0$ does not imply that X and Y are independent. On the other hand, these two measures always exist and are invariant under nonlinear rescaling of the horizontal axes of the marginal. They are members of a general class of *measures of concordance* of distributions (Scarsini 1984, Nelsen 1999, Embrechts, Lindskog, and McNeil 2003).

Another concept useful in the variance-reduction context is *positive quadrant dependence* (PQD), defined in Section 6.3.2. It ensures non-negative covariance between monotone functions of the random variables.

Other notions and measures of dependence are discussed, e.g., in Devroye (1986), Joe (1997), Nelsen (1999) and Drouet Mori and Kotz (2001). Some of them are zero when and only when the variables are independent. They are generally harder to compute.

2.10.3 The multinormal distribution

A vector $\boldsymbol{X} = (X_1, \ldots, X_d)^{\mathsf{t}}$ has the *multinormal distribution* in d dimensions (or d-dimensional normal distribution), denoted $\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, if its density has the form

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\left(-(\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{t}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})/2\right)$$

for $\boldsymbol{x} = (x_1, \ldots, x_d)^{\mathsf{t}} \in \mathbb{R}^d$, where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_d)^{\mathsf{t}} \in \mathbb{R}^d$ is the mean vector and $\boldsymbol{\Sigma} = \mathrm{Cov}(\boldsymbol{X})$ is the symmetric positive-definite covariance matrix of $\boldsymbol{X}$. One has $\mathbb{E}[X_j] = \mu_j$ and $\mathrm{Cov}(X_j, X_k) = \sigma_{jk}$ where σ_{jk} is the element (j,k) of $\boldsymbol{\Sigma}$.

An equivalent definition is that $\boldsymbol{X}$ is multinormal if and only if every linear combination $X = a_1 X_1 + \cdots + a_d X_d$, where each $a_j \in \mathbb{R}$, has the univariate normal distribution. Note that $\boldsymbol{X}$ can be *multivariate normal* (i.e., have all normal marginal) without being multinormal (Exercise 2.27).

If $\boldsymbol{Z} \sim N(\boldsymbol{0}, \boldsymbol{I})$ where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{Z}$ is said to have the *standard multinormal* distribution. If $\boldsymbol{Z} \sim N(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{Z} + \boldsymbol{\mu}$ where $\boldsymbol{A}$ is a $d \times d$ matrix and $\boldsymbol{\mu}$ is a d-dimensional vector, then $\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}})$. Thus, if we decompose $\boldsymbol{\Sigma}$ as $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, we can easily generate $\boldsymbol{X}$ by generating a vector $\boldsymbol{Z}$ of i.i.d. standard normals and applying the linear transformation

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{Z} + \boldsymbol{\mu}.$$

There are generally many ways of decomposing $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, and some may be more convenient than others depending on the context. The most common choice is to impose that $\boldsymbol{A}$ be lower triangular; the decomposition is then the (unique) Cholesky decomposition.

♣ Add a small example.

For the multinormal, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ specify the distribution uniquely. But this does not apply to other distributions in general. The multinormal distribution is very widely used, largely because it has been extensively studied, is well understood, and gives rise to statistical models that are easy to handle. Tong (1990) provides an extensive coverage.

A vector $\boldsymbol{X} = (X_1, \ldots, X_d)^{\mathsf{t}}$ has the *d-dimensional lognormal* distribution if $\boldsymbol{Y} = (\ln X_1, \ldots, \ln X_d)^{\mathsf{t}}$ is multinormal. To use this distribution, one can just take the log of the X_j's and use the multinormal distribution for the resulting values.

2.10.4 Elliptic Multivariate Distributions

The multinormal is a member of a more general class of distributions defined as follows (Cambanis, Huang, and Simons 1981, Embrechts, Lindskog, and McNeil 2003). For given positive integers $k \le d$, a random vector $\boldsymbol{X}$ has an *elliptic multivariate distribution* in $\mathbb{R}^d$ if it can be written as

$$\boldsymbol{X} = \boldsymbol{\mu} + R\,\boldsymbol{A}\boldsymbol{U} \tag{2.36}$$

where $\boldsymbol{U}$ is a random vector uniformly distributed on the k-dimensional unit hypersphere $\{\boldsymbol{x} \in \mathbb{R}^k : \boldsymbol{x}^{\mathsf{t}}\boldsymbol{x} = 1\}$, $\boldsymbol{A}$ is a $d \times k$ matrix, R is a real-valued random variable with an arbitrary (fixed) distribution and independent of $\boldsymbol{U}$, and $\boldsymbol{\mu}$ is a vector in $\mathbb{R}^d$. The definition (2.36) already indicates one way of generating $\boldsymbol{X}$: Generate a random point $\boldsymbol{U}$ on the unit hypersphere (see Chapter 4, Example 4.12, or Devroye 1986, Section V.4.2), multiply it by the matrix $\boldsymbol{A}$. This generates a point on the surface of an ellipsoid. Then generate R and multiply by it to rescale the ellipsoid randomly. Finally, add $\boldsymbol{\mu}$ to shift the center of the ellipsoid from $\boldsymbol{0}$ to $\boldsymbol{\mu}$. This approach can be convenient if it is easy to generate from the distribution of R. In many cases, however (e.g, for the multinormal and Student-t distributions), a different approach is used for generating $\boldsymbol{X}$.

We have $\mathbb{E}[\boldsymbol{X}] = \boldsymbol{\mu}$ and if $\mathbb{E}[R^2] < \infty$ then

$$\mathrm{Cov}[\boldsymbol{X}] = \mathrm{Cov}[\boldsymbol{\mu} + R\boldsymbol{A}\boldsymbol{U}] = \mathbb{E}[R^2]\boldsymbol{A}\,\mathrm{Cov}[\boldsymbol{U}]\boldsymbol{A}^{\mathsf{t}} = \mathbb{E}[R^2]\boldsymbol{\Sigma}/k$$

where $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, because R is independent of $\boldsymbol{U}$ and $\mathrm{Cov}(\boldsymbol{U}) = \boldsymbol{I}/k$. Without loss of generality, we can always rescale R and $\boldsymbol{A}$ so that $\mathbb{E}[R^2] = k$, and then $\mathrm{Cov}[\boldsymbol{X}] = \boldsymbol{\Sigma}$. The type of an elliptic distribution depends essentially on the choice of distribution for R. If R^2 has a chi-square distribution with k degrees of freedom, then $\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

If $\boldsymbol{X}$ has a density, then it must be of the general form

$$f(\boldsymbol{x}) = |\boldsymbol{\Sigma}|^{-1/2} g((\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{t}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}))$$

for some function $g : \mathbb{R} \to [0, \infty)$. Thus, the contours of the density, where it takes a constant value, are ellipsoids in $\mathbb{R}^d$ (or unions of ellipsoids if g is not strictly decreasing). This implies in particular that the density must be symmetric along each line that passes through the

mean $\boldsymbol{\mu}$. This of course limits the generality of these distributions. Moreover, all univariate marginals must be from the same family, determined by the distribution of R.

In the special case where $\boldsymbol{A} = c\boldsymbol{I}$ for some constant c, the density becomes a function of the Euclidean distance $\|\boldsymbol{x} - \boldsymbol{\mu}\| = Rc$ only, and the distribution is called *radially symmetric*.

Example 2.13 Let $\boldsymbol{X} = \boldsymbol{\mu} + \sqrt{\nu/Y}\boldsymbol{Z}$ where Y has the chi-square distribution with ν degrees of freedom and $\boldsymbol{Z} \sim \mathsf{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ (in d dimensions) is independent of Y. This $\boldsymbol{X}$ has a *d-dimensional Student-t distribution* with parameters ν (the number of degrees of freedom) and $\boldsymbol{\Sigma}$. This formula for $\boldsymbol{X}$ provides an easy way of generating $\boldsymbol{X}$, by generating a chi-square and a multinormal. The distribution of $\boldsymbol{X}$ turns out to be an elliptic distribution for which $\boldsymbol{A} = \boldsymbol{I}$ and $R^2/\nu = \|\boldsymbol{Z}\|^2/Y$, which is a ratio of two independent chi-squares with d and ν degrees of freedom. Thus, R^2/d has a $\mathsf{F}(d, \nu)$ distribution. One has $\mathbb{E}[R^2] = \nu/(\nu - 2)$ and $\mathrm{Cov}[\boldsymbol{X}] = \boldsymbol{\Sigma}\nu/[(\nu - 2)d]$. See Fang, Kotz, and Ng (1990), Kibria and Joarder (2006) for more details. □

2.10.5 Dirichlet distribution

The *Dirichlet distribution* with parameters $(\alpha_1, \ldots, \alpha_d)$, where $\alpha_j > 0$ for each j and $\alpha_0 = \sum_{j=1}^{d} \alpha_j$, has pdf

$$f(x_1, \ldots, x_d) \;=\; \Gamma(\alpha_0) \prod_{j=1}^{d} \frac{x_j^{\alpha_j - 1}}{\Gamma(\alpha_j)} \tag{2.37}$$

for $x_j \geq 0$ for each j and $\sum_{j=1}^{d} x_j = 1$. We write $\boldsymbol{X} = (X_1, \ldots, X_d) \sim \mathsf{Dirichlet}(\alpha_1, \ldots, \alpha_d)$. It is a multivariate generalization of the beta distribution: We have $X_j \sim \mathsf{Beta}(\alpha_j, \alpha_0 - \alpha_j)$ for each j.

The Dirichlet is the conjugate prior of the multinomial, in the sense that if $\boldsymbol{X} = (X_1, \ldots, X_d)$ has a multinomial distribution with fixed parameter n and random parameters $\boldsymbol{p} = (p_1, \ldots, p_d)$ having a Dirichlet distribution with parameters $(\alpha_1, \ldots, \alpha_d)$, then conditional on $\boldsymbol{X}$, the vector $\boldsymbol{p}$ has a Dirichlet distribution with parameters $(\alpha_1 + X_1, \ldots, \alpha_d + X_d)$. This is often used to estimate the unknown parameters $\boldsymbol{p}$ of a multinomial distribution via Bayesian methods, given observations $\boldsymbol{X}$.

2.10.6 Other standard multivariate distributions

Multivariate versions of the Johnson families of distributions, together with methods for estimating the parameters and generating the random vectors, have been developed by Stanfield et al. (1996). However, parameter estimation (beyond the mean and variance) becomes increasingly difficult when the dimension increases.

Several other "standard" types of multivariate distributions are discussed in Johnson and Kotz (1972b), Devroye (1986), Johnson (1987), and Fang, Kotz, and Ng (1987).

2.10.7 Multivariate kernel density estimation

KDE methods (Section 2.9.4) extend to multivariate distributions. The main change is that when estimating a density in the d-dimensional real space, the bandwidth becomes a $d \times d$ matrix of parameters, say $\boldsymbol{B}$, and it rapidly becomes difficult to find a good $\boldsymbol{B}$ when d increases. Thus, this method is practically viable in general only for moderate d.

Silverman (1986) suggests to transform the data so that the empirical covariance matrix becomes the identity, and then take $\boldsymbol{B}$ as a multiple of the identity, with a multinormal kernel. This is equivalent to taking the original data with $\boldsymbol{B}$ equal to a multiple of the empirical covariance matrix $\boldsymbol{S}_n$; e.g., $\boldsymbol{B} = b^2\boldsymbol{S}_n$ for some smoothing factor b. Optimality results for the AMISE in the case where the original density to be estimated is multinormal suggest taking $b = [4/((d+2)n)]^{1/(d+4)}$. As in the unidimensional case, a correction can be made so that the variance of the estimated density matches the sample variance (the observations are moved closer to their sample mean by a factor $c_b = (1+b^2)^{-1/2}$).

To generate random vectors from such a kernel density, one would first compute the sample mean $\bar{\boldsymbol{x}}_n$ and the sample covariance matrix $\boldsymbol{S}_n$, and decompose $\boldsymbol{S}_n$ as $\boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$ (e.g., via Cholesky's decomposition) in a one-time setup. Each random vector can then be generated as follows:

(1) generate I uniformly over $\{1, \ldots, n\}$;
(2) generate a vector $\boldsymbol{Z}$ from the d-dimensional $N(\boldsymbol{0}, \boldsymbol{I})$ distribution;
(3) return $\boldsymbol{X} = \bar{\boldsymbol{x}}_n + c_b(\boldsymbol{x}_I - \bar{\boldsymbol{x}}_n + b\boldsymbol{A}\boldsymbol{Z})$.

This algorithm is given and discussed in Hörmann, Leydold, and Derflinger (2004). These authors also point out that the value of b should be reduced (e.g., by half) when the original distribution departs significantly from the normal (e.g., is highly skewed or multimodal), to avoid oversmoothing. They also point out that $b = 0$ corresponds to straightforward resampling from the empirical distribution whereas $b = \infty$ corresponds to fitting a multinormal distribution to the data. Any b in between represents a compromise between these two extremes.

♣ A more recent discussion on multivariate KDEs can be found in Scott (2015).

2.10.8 Copulas

A very general way of defining multivariate distributions is as follows. If $\boldsymbol{X} = (X_1, \ldots, X_d)$ is a random vector and X_j has the *continuous marginal* cdf F_j for each j, then $\boldsymbol{U} = (U_1, \ldots, U_d) = (F_1(X_1), \ldots, F_d(X_d))$ has a multivariate distribution for which each marginal has the $U(0,1)$ distribution. It turns out that this multivariate distribution of $\boldsymbol{U}$ determines the dependence structure uniquely, independently of the marginals. To generate $\boldsymbol{X}$, it then suffices to generate $\boldsymbol{U}$ from the correct distribution and set $X_j = F_j^{-1}(U_j)$ for each j. But how do we specify the distribution of $\boldsymbol{U}$?

A cdf C whose marginals are $U(0,1)$ is called a *dependence function* or a *copula*. A function $C : [0,1]^d \to [0,1]$ is a d-dimensional copula if and only if it satisfies the following three conditions: (i) if $\boldsymbol{u} \in [0,1]^d$ has at least one coordinate equal to 0, then $C(\boldsymbol{u}) = 0$; (ii) if

$\boldsymbol{u}$ has all its coordinates equal to 1 except for coordinate j which equals u_j, then $C(\boldsymbol{u}) = u_j$; (iii) for any rectangular box B in $[0,1]^d$, C determines the probability of that box (see Exercise 2.29) and that probability must be ≥ 0 for any B. In $d = 2$ dimensions, for example, whenever $a_1 < a_2$ and $b_1 < b_2$, we must have $C(a_2, b_2) - C(a_1, b_2) - C(a_2, b_1) + C(a_1, b_1) \geq 0$.

One can model the dependence between the X_j's by expressing their joint cdf F via a copula C, function of the u_j's, as follows:

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)) = C(u_1, \ldots, u_d) \tag{2.38}$$

where $u_j = F_j(x_j)$. *Sklar's theorem* (Sklar 1959) says that such a copula C always exists, is unique when all the marginals F_j are continuous, and that (conversely) a copula C together with arbitrary marginal distributions F_j define a multivariate cdf F via (2.38). This is called the *standard copula representation* of F. The choice of C has an impact on the joint distribution, but the marginal distributions F_j do not depend on C. This completely decouples the dependence structure from the choice of marginals. The random variables $X_1, \ldots, X_d$ are independent if and only if $C(u_1, \ldots, u_d) = u_1 \cdots u_d$, the product function (also called the *product copula*).

Here, the X_j's may have different types of distributions. For example, X_1 can be normal and X_2 gamma, or X_1 can have a discrete distribution and X_2 a continuous one. When X_j is discrete, $F_j(X_j)$ is no longer $U(0,1)$, but it remains true that X_j has the same distribution as $F_j^{-1}(U_j)$ if $U_j \sim U(0,1)$. So to generate a random vector $(X_1, \ldots, X_d)$ with distribution function F as in (2.38), it suffices to generate $(U_1, \ldots, U_d)$ from the copula C and put $X_j = F_j^{-1}(U_j)$ for each j.

Note that the copula C is often constructed via (2.38) using a *different* class of cdfs than the one from which we want to generate values. If G is an arbitrary d-dimensional cdf with marginals G_j, C can be defined by

$$C(u_1, \ldots, u_d) = G(G_1^{-1}(u_1), \ldots, G_d^{-1}(u_d)). \tag{2.39}$$

This C is called the *copula associated with* G. To generate a vector $(X_1, \ldots, X_d)$ with this copula and marginals F_j, we can generate $(Y_1, \ldots, Y_d)$ from distribution G, set $U_j = G_j(Y_j)$ for each j (if each G_j is continuous) or generate $(U_1, \ldots, U_d)$ from C in any other way, and then put $X_j = F_j^{-1}(U_j)$ for each j. For example, we could use a multinormal distribution G in (2.39) to determine the copula, then use the copula to generate random vectors $(X_1, \ldots, X_d)$ having marginals with a completely different type of cdf F_j, such as the gamma or Weibull (Section 2.10.9).

Example 2.14 The following are four examples of oversimplified copulas that are easy to generate from, and can give either zero or extremal correlation.

(a) If $U_1, \ldots, U_d$ are d *independent* $U(0,1)$ random variables, their joint cdf is the *product copula* C defined by $C(u_1, \ldots, u_d) = u_1 \cdots u_d$.

(b) The strongest possible positive dependence occurs when $U_1 = \cdots = U_d = U$ where $U \sim U(0,1)$, in which case the distribution of $\boldsymbol{U}$ is degenerate and uniform on the diagonal line that joins $(0, \ldots, 0)$ and $(1, \ldots, 1)$. The corresponding copula is defined by $C(u_1, \ldots, u_d) = \mathbb{P}[U \leq u_1, \ldots, U \leq u_d] = \mathbb{P}[U \leq \min(u_1, \ldots, u_d)] = \min(u_1, \ldots, u_d)$.

(c) For $d = 2$, we have a perfect negative correlation between U_1 and U_2 if $U_1 \sim U(0,1)$, and $U_2 = 1 - U_1$. Then (U_1, U_2) is uniformly distributed on the diagonal line that joins $(0,1)$ and $(1,0)$. The corresponding copula is $C(u_1, u_2) = \max(0, u_1 + u_2 - 1)$ (Exercise 2.30).

(d) Let $d = 2$, $U_1 \sim U(0,1)$, $U_2 = U_1$ with probability p, and $U_2 = 1 - U_1$ with probability $1 - p$. Then the distribution of (U_1, U_2) is concentrated on the union of the two diagonal lines mentioned in (b) and (c). The corresponding copula is defined by $C(u_1, u_2) = p \min(u_1, u_2) + (1-p)\max(0, u_1 + u_2 - 1)$ and is called a *Fréchet copula* (Fréchet 1951). By varying p, the correlation $\rho(U_1, U_2)$ can take any value in the interval $[-1, 1]$. For $p = 1/2$, this distribution is uniform over the union of the two diagonals and we have $\rho(U_1, U_2) = 0$, but these two random variables are clearly dependent. Despite having the same (zero) correlation as the product copula, this copula (with $p = 1/2$) is quite different. □

Example 2.15 To illustrate the impact of the difference in Example 2.14(d) when $p = 1/2$, let $q(a) = \mathbb{P}[U_1 U_2 > a]$, where $1/2 < a < 1$. With Fréchet's copula, we have

$$q(a^2) = \mathbb{P}[U_1 > a]/2 = (1-a)/2,$$

whereas with the product copula, when a is very close to 1, we have

$$\begin{aligned} q(a^2) &= \int_0^1 \mathbb{P}[U_1 > a^2/u]du = \int_{a^2}^1 (1 - a^2/u)du \\ &= 1 - a^2(1 - 2\ln a) \\ &= 1 - a^2 + 2a^2\left[(a-1) - (a-1)^2/2 + o((a-1)^2)\right] \\ &= 2(1-a)^2 + o((a-1)^2). \end{aligned}$$

which is much smaller. If $1 - a = 10^{-4}$, for example, the first probability is $10^{-4}/2$ and the second one is 2×10^{-8}. This example shows that for given means, correlations, and marginal distributions, the choice of copula can make a huge difference on the probability of some important rare event, which may correspond for instance with a large loss or a large reward. □

Parts (b) and (c) of Example 2.14 provide copulas that maximize and minimize, respectively, the correlation between U_1 and U_2. The next result gives necessary and sufficient conditions for this maximization or minimization to be transmitted to a pair (X_1, X_2) with arbitrary marginals.

Theorem 2.7 *(See Embrechts, McNeil, and Straumann 2002.) A random vector (X_1, X_2) has the copula $C(u_1, u_2) = \min(u_1, u_2)$, which maximizes the correlation for the given marginals, if and only if it can be written as $(X_1, X_2) = (g_1(U), g_2(U))$ where g_1 and g_2 are both nondecreasing functions. It has the copula $C(u_1, u_2) = \max(0, u_1 + u_2 - 1)$, which minimizes the correlation for the given marginals, if and only if it can be written as $(X_1, X_2) = (g_1(U), g_2(U))$ where g_1 is nondecreasing and g_2 is nonincreasing.*

A pair of random variables X_1 and X_2 that satisfy this theorem are called *comonotonic* in the first case and *countermonotonic* in the second case. When the random variables are discrete, the copula is not unique, but the one given by the theorem is always one of the possibilities. When they are continuous, their rank correlation rho and Kendall's tau are both 1 in the first case (one can take $g_j(u) = F_j^{-1}(u)$, the inverse distribution function of X_j, for each j) and -1 in the second case (take $g_1(u) = F_j^{-1}(u)$ and $g_2(u) = F_2^{-1}(1-u)$).

Example 2.16 Suppose we partition the unit square $[0,1)^2$ into k^2 subsquares of the same size, by partitioning each axis into k intervals of length $1/k$. We then select k subsquares forming a Latin square, i.e., so that each row and each column contains exactly one subsquare. Consider a continuous distribution with density equal to k on the selected squares and 0 elsewhere. This distribution always has $U(0,1)$ marginals, so the corresponding cdf is a copula. This scheme generalizes easily to $d > 2$ dimensions: the unit hypercube $[0,1)^d$ is partitioned into k^d subcubes of equal size, k of these cubes that form a Latin hypercube are selected, and the density is set to k in the selected subcubes and 0 elsewhere. The corresponding cdf is again a copula. Ghosh and Henderson (2001) call it a *chessboard distribution*. Here, the correlations depend on which subcubes are selected. □

Example 2.17 Let G be a bivariate normal cdf where the two marginals are standard normal and have correlation $\rho \in (0,1)$. The correlation matrix $\boldsymbol{\Sigma}$ and its Cholesky decomposition are

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{t}} \quad \text{where} \quad \boldsymbol{L} = \begin{pmatrix} 1 & 0 \\ \rho & (1-\rho^2)^{1/2} \end{pmatrix}.$$

If $\boldsymbol{Z} = (Z_1, Z_2)^{\mathsf{t}}$ is a vector of two independent standard normals, then $\boldsymbol{Y} = (Y_1, Y_2)^{\mathsf{t}} = \boldsymbol{L}\boldsymbol{Z}$ has cdf G. The cdf of $\boldsymbol{U} = (U_1, U_2)^{\mathsf{t}} = (\Phi(Y_1), \Phi(Y_2))^{\mathsf{t}}$ is the copula C associated with G. This copula is a *two-dimensional normal copula*. Note that we have $Y_1 = Z_1$ and $Y_2 = \rho Z_1 + (1-\rho^2)^{1/2} Z_2$. To generate a random vector $\boldsymbol{X} = (X_1, X_2)^{\mathsf{t}}$ with arbitrary marginals F_1 and F_2 and dependence structure specified by C, we just return $X_j = F_j^{-1}(U_j)$ for $j = 1, 2$.

For this normal copula, we have the following simple relationships between the correlation ρ, the rank correlation $r = \rho_{\mathrm{s}}(Y_1, Y_2) = \rho(U_1, U_2)$, and the Kendall's tau $\tau = \tau_{\mathrm{k}}(Y_1, Y_2)$ between Y_1 and Y_2 (Exercise 2.26):

$$\rho = 2\,\sin(r\pi/6) = \sin(\tau\pi/2). \tag{2.40}$$

Note that ρ, r, and τ can reach any value in $(-1, 1)$; when one of these measures goes from -1 to 1, the other two do the same.

We also have $\rho_{\mathrm{s}}(X_1, X_2) = r$ and $\tau_{\mathrm{k}}(X_1, X_2) = \tau$ regardless of what F_1 and F_2 are, provided that they are both continuous. So to specify ρ in practice, we can select r or τ by estimating the rank correlation of Kendall's tau between X_1 and X_2 directly from the data, and compute the corresponding ρ from (2.40). □

Example 2.18 The previous example can be generalized to the case where $\boldsymbol{Y}$ has an elliptic distribution in d dimensions. The associated copula C, obtained by the standard construction (2.39), is called an *elliptic copula*. Two prominent special cases are the normal and Student-t copulas.

In the *normal* (or Gaussian) case, the correlation matrix $\boldsymbol{R}^{(\mathrm{y})}$ (or the rank correlation matrix $\boldsymbol{R}^{(\mathrm{u})}$) of $\boldsymbol{Y}$ determines the copula uniquely; $\boldsymbol{R}^{(\mathrm{y})}$ can be computed from $\boldsymbol{R}^{(\mathrm{u})}$ and vice-versa, via (2.40). To generate a random vector $(U_1, \ldots, U_d)$ from that copula, it suffices to generate $(Y_1, \ldots, Y_d) \sim \mathsf{N}(\boldsymbol{0}, \boldsymbol{R}^{(\mathrm{y})})$ (see Section 2.10.3) and return $U_j = \Phi(Y_j)$ for $j = 1, \ldots, d$. Here, each Y_j is $\mathsf{N}(0,1)$. The special case where $d = 2$ was discussed in the previous example and we will return to the general case in Section 2.10.9.

The *Student-t copula* depends on both the correlation matrix and the number of degrees of freedom, ν. To generate $(U_1, \ldots, U_d)$ from a Student copula with ν degrees of freedom and

correlation matrix $\boldsymbol{R}^{(\mathrm{y})}$, generate $(Y_1, \ldots, Y_d)$ from the d-dimensional Student distribution with parameters $(\nu, \boldsymbol{\Sigma}) = (\nu, \boldsymbol{R}^{(\mathrm{y})})$ (so $Y_j \sim$ Student-t(ν) for each j) and return $U_j = T_\nu(Y_j)$ for $j = 1, \ldots, d$, where T_ν is the Student-t cdf with ν degrees of freedom. □

An *Archimedean copula* is one that can be written as

$$C(u_1, \ldots, u_d) = \begin{cases} \varphi^{-1}(\varphi(u_1) + \cdots + \varphi(u_d)) & \text{if } \varphi(u_1) + \cdots + \varphi(u_d) \le \varphi(0); \\ 0 & \text{otherwise,} \end{cases}$$

where $\varphi : (0, 1] \to \mathbb{R}$ is a d times continuously differentiable function that satisfies $\varphi(u) \to \infty$ when $u \to 0^+$, $\varphi(1) = 0$, $\varphi'(u) < 0$ for all u, and $(-1)^k d^k \varphi^{-1}(x)/dx^k > 0$ for all $x \in [0, \infty)$. This function φ is the *generating function of the copula*. An Archimedean copula is always symmetric and associative (i.e., $C(u_1, u_2) = C(u_2, u_1)$ and $C(u_1, C(u_2, u_3)) = C(C(u_1, u_2), u_3))$), and any of its two-dimensional marginals has a Kendall τ equal to

$$\tau = 1 + 4 \int_0^1 [\varphi(u)/\varphi'(u)] du$$

(Genest and MacKay 1986). Note that for any $d' < d$, all d'-dimensional marginals are identical. This of course limits the flexibility. Examples of generating functions for Archimedean copulas, that depend on a single parameter, are $\varphi(u) = -\ln u$ (the product copula), $\varphi(u) = u^{-\lambda} - 1$ for $\lambda > 0$ (Clayton's copula), $\varphi(u) = (-\ln u)^\lambda$ for $\lambda > 1$ (the Gumbel copula), and $\varphi(u) = -\ln[(e^{-\lambda u} - 1)/(e^{-\lambda} - 1)$ for $\lambda \neq 0$ (Frank's copula). One advantage of these copulas is that they are specified by a single parameter, and there is a one-to-one correspondence between the value of that parameter and the Kendall tau of the two-dimensional marginals. One can estimate τ (e.g., by its empirical value), and it is then easy to find the corresponding parameter value. For example, one has $\lambda = 2\tau/(1-\tau)$ for the Clayton copula, and $\lambda = 1/(1-\tau)$ for the Gumbel copula. On the other hand, having a single parameter is a strong limitation on the flexibility, especially in higher dimensions.

An *extreme-value copula* is a copula that satisfies

$$C(u_1^\alpha, \ldots, u_d^\alpha) = C^\alpha(u_1, \ldots, u_d)$$

for all $\alpha > 0$ and $(u_1, \ldots, u_d) \in [0, 1]^d$. A multivariate extreme value distribution G defines an extreme-value copula via Eq. (2.39). Examples of extreme-value copulas (in two dimensiona) include the product, Gumbel, Gumbel II, Galambos, and Marshall-Olkin, copulas.

Example 2.19 (Marshall-Olkin copula) This class of copulas was introduced by Marshall and Olkin (1967) to model component life in multicomponent systems affected by random shocks. They can also be useful for modeling credit risk (replace components by firms) and insurance losses, for example.

Consider a system of d components with (random) lifetimes $Y_1, \ldots, Y_d$. For each nonempty subset of components $S \subseteq \{1, \ldots, d\}$, there is a shock arriving at time $T(S)$ and which kills all components in S. Each $T(S)$ is an exponential random variable with parameter $\lambda(S)$ and these random variables are all independent. Thus Y_j is the minimum of several independent exponentials,

$$Y_j = \min_{\{S: j \in S\}} T(S), \tag{2.41}$$

so it is also exponential with parameter

$$\lambda_j = \sum_{\{S: j \in S\}} \lambda(S),$$

for $j = 1, \ldots, d$. This λ_j represents the total shock rate for component j. The copula C obtained by the standard construction (2.39) with the joint distribution G of $(Y_1, \ldots, Y_d)$ is a *Marshall-Olkin copula*.

To generate a random vector $(U_1, \ldots, U_d)$ from this copula, we can proceed as follows: Generate $T(S) \sim \mathsf{Exponential}(\lambda(S))$ for each S such that $\lambda(S) > 0$, then for each j compute Y_j via Eq. (2.41) and set $U_j = G_j(Y_j) = 1 - \exp[-\lambda_j Y_j]$, where G_j is the (exponential) cdf of Y_j. Then, a vector $(X_1, \ldots, X_d)$ with arbitrary marginals F_j and whose dependence structure is specified by this copula can be obtained by setting $X_j = F_j^{-1}(U_j)$ for each j.

When d is large, we usually take $\lambda(S) = 0$ for several of the subsets S, so not all $2^d - 1$ possible subsets need to be considered in the algorithm. For example, we may consider only the subsets of cardinality no more than 2, which is generally sufficient for matching a given matrix of rank correlations (but limits the flexibility of the copula on other aspects; e.g., there can only be pairwise correlations).

Let us define

$$\lambda_{i,j} = \sum_{\{S: i \in S,\, j \in S\}} \lambda(S)$$

for all pairs (i, j) of components. In the bivariate case ($d = 2$), the copula can be written as a function of the parameters $\alpha_1 = \lambda_{1,2}/\lambda_1$ and $\alpha_2 = \lambda_{1,2}/\lambda_2$, namely

$$C(u_1, u_2) = \min(u_2 u_1^{1-\alpha_1},\, u_1 u_2^{1-\alpha_2}).$$

This copula has a positive mass on the line $Y_1 = Y_2$; this mass corresponds to the simultaneous failure of both components, which occurs at rate $\lambda_{1,2}$. The Spearman's rho and Kendall's tau between Y_1 and Y_2 are then (see Exercise 2.33)

$$\rho_{\mathrm{s}}(Y_1, Y_2) = \rho(U_1, U_2) = \frac{3\alpha_1\alpha_2}{2\alpha_1 + 2\alpha_2 - \alpha_1\alpha_2}$$

and

$$\tau_{\mathrm{k}}(Y_1, Y_2) = \frac{\alpha_1\alpha_2}{\alpha_1 + \alpha_2 - \alpha_1\alpha_2},$$

respectively. For $d > 2$, each pair (Y_i, Y_j) has a bivariate Marshall-Olkin copula with parameters $\alpha_i = \lambda_{i,j}/\lambda_i$ and $\alpha_j = \lambda_{i,j}/\lambda_j$, so the Spearman's rho and Kendall's tau for each pair can be computed as for the bivariate copula with parameters (α_i, α_j). □

Example 2.20 The two-dimensional *Gumbel copula*, with parameter $\delta \geq 1$, is defined as (Gumbel 1960, Joe 1997, Nelsen 1999):

$$C_\delta(u_1, u_2) = \exp\left[-\left((-\ln u_1)^\delta + (-\ln u_2)^\delta\right)^{1/\delta}\right] \tag{2.42}$$

for $0 \le u_1, u_2 \le 1$. With $\delta = 1$, this gives the product copula (no dependence). The dependence increases with δ and we get perfect correlation in the limit when $\delta \to \infty$. The Kendall's tau is $\tau(U_1, U_2) = (\delta - 1)/\delta$ (Joe 1997, page 146). There is no simple formula for the rank correlation $\rho(U_1, U_2)$ as a function of δ but that function can be estimated by Monte Carlo.

Generating a pair (U_1, U_2) from this copula is not straightforward. One way of doing it is to generate $U_1 \sim U(0,1)$ and then generate U_2 from its conditional distribution given U_1, given by

$$\begin{aligned} C_{2|1}(u_2|u_1) &= \mathbb{P}[U_2 \le u_2 \mid U_1 = u_1] \\ &= \frac{(1 + (\ln u_2 / \ln u_1)^\delta)^{-1+1/\delta}}{u_1} \exp\left[-\left((-\ln u_1)^\delta + (-\ln u_2)^\delta\right)^{1/\delta}\right] \end{aligned}$$

(Joe 1997, page 147). To generate U_2 by inversion from its conditional distribution given U_1, one can generate $U \sim U(0,1)$ independent of U_1 and then find the root U_2 of the equation $C_{2|1}(U_2|U_1) = U$. Note that the conditional distribution is increasing in U_2 and goes from 0 to 1, so the root is unique. Finding this root is equivalent to computing the inverse of the conditional cdf (for which we have no closed-form formula). The root can be found by a simple binary search or by other more refined (and perhaps more efficient) root finding algorithms.

Higher-dimensional generalizations of this copula are defined for example in Joe (1997), Family MM1, pages 163 and 188. There are also other generation methods that work for the multivariate case; see Wu, Valdez, and Sherris (2006). □

Example 2.21 The choice of copula family can have a dramatic impact in certain settings. Embrechts, Lindskog, and McNeil (2003) (Section 7.1) give the following example. An insurance company receives a payoff (from reinsurance against very large losses) if its losses in each of its five lines of business exceed a given threshold, fixed at the 0.99 quantile of the loss distribution. The five-dimensional vector of losses $(X_1, \ldots, X_5)$ is modeled as follows: each X_i has the lognormal distribution with parameters $(\mu, \sigma^2) = (0, 1)$, $\tau_k(X_i, X_j) = 0.5$ for all $i \ne j$, and the dependence structure is further specified by a five-dimensional copula. For this, they compare the normal and (Archimedean) Gumbel copulas, whose parameters are uniquely determined by the values of the Kendall tau. For the Gumbel copula, we have $\tau = (\delta - 1)/\delta$, so $\delta = 1/(1 - \tau) = 2$. For their example, they find that the probability of receiving a payoff is about eight times larger with the Gumbel copula than with the normal, for the same Kendall's tau and the same marginals. This implies for instance that if we use the NORTA method of the next section while the true dependence structure is given by a Gumbel copula, we underestimate the payoff probability by a factor of 8. A similar kind of behavior can be observed when estimating the value-at-risk of a portfolio of several assets whose values change in a dependent way. □

Example 2.22 In an example related to the previous one, Blum, Dias, and Embrechts (2002) fit seven types of two-dimensional copulas to a data set of two-dimensional claims for fire insurance in Denmark. The two coordinates represent two different types of losses, in logarithmic scale. For the marginals F_j, they simply take the empirical cdfs of the data. They use each model to estimate the mean and standard deviation of four different payoff functions that are simplified representations of typical payoff functions in the insurance industry. They

compare these estimates with the corresponding means and standard deviations obtained from the raw data, with the same payoff functions. The Gumbel copula provides the best fit overall and performs much better than the normal copula (for example), which tends to underestimate the standard deviation. Some of the copulas are doing very bad especially if the payoff in nonzero only when both types of losses are large. These types of payoff functions are typical in risk transfer contracts, whose purpose is to protect against large losses in several areas simultaneously. Without this protection, such large losses may trigger bankruptcy of the insurance company. The density in the upper right corner of the joint distribution, where both U_1 and U_2 are large, can be much larger for the Gumbel copula than for the normal with the same rank correlation. Using the wrong copula in this context may lead to a significant underestimation of the value of the contract because it grossly underestimates the risk. See Embrechts, McNeil, and Straumann (2002) for further discussion. □

♣ Add some plots here for the examples.

♣ Other example: Jaoua, L'Ecuyer, and Delorme (2013).

Besides the importance of choosing the correct family, the two previous examples illustrate again a key aspect of modeling with copulas: the distribution F of the vector $\boldsymbol{X}$ that we want to model can be from a different family than the distribution G used to specify the copula via Eq. (2.39). In Example 2.21, each G_j is either from the normal or Gumbel family whereas each F_j is lognormal. In Example 2.22, each F_j is an empirical distribution function from the data for each of the seven different choices of G. As another example, the copula can be from the Marshall-Olkin family of Example 2.19, where the marginals G_j are exponential, and the F_j's could be Weibull for some j's and gamma for others.

We saw in Example 2.22 that certain copulas (such as the Gumbel) provide stronger dependence than others in the upper tails of the marginal distributions, in the sense that they favor realizations where all coordinates are large simultaneously, for the same values of the correlation coefficients. This dependence between extreme values can be measured by the *coefficient of upper tail dependence* λ_{u}, which depends only on C and is defined by

$$\lambda_{\mathrm{u}} = \lim_{u \to 1^-} \mathbb{P}[U_2 > u \mid U_1 > u] = \lim_{u \to 1^-} \frac{1 - 2u + C(u,u)}{1 - u} \tag{2.43}$$

(the limit when u converges to 1 from the left) when this limit exists (Joe 1997, page 33). It measures a stronger level of dependence than correlation. It turns out, for example, that $\lambda_{\mathrm{u}} = 0$ for the normal copula (regardless of the correlation), $\lambda_{\mathrm{u}} > 0$ (decreasing in ν) for the Student-t copula, and $\lambda_{\mathrm{u}} = 2 - 2^{1/\delta} > 0$ for the Gumbel copula with parameter $\delta > 1$ (Exercise 2.36). A larger λ_{u} means (very roughly) that for a given correlation $\rho(U_1, U_2)$, the joint distribution of (U_1, U_2) has more density in the area where both U_1 and U_2 are very large. For applications like those of Example 2.22, modeling this upper tail dependence correctly is crucial.

There is an extremely rich variety of ways of specifying a copula, each one giving rise to a different shape of distribution. The previous examples only show a small subset of the possibilities. Other families of copulas are described and studied in Joe (1997), Nelsen (1999), Drouet Mori and Kotz (2001), for example. The following desirable properties are relevant when selecting a copula family for a simulation model: (a) There should be a relatively

easy and efficient way of generating random vectors $(U_1, \ldots, U_d)$ from that copula; (b) the parameters of the copula should not be too hard to estimate and they should preferably have an interpretation; (c) the range of dependence structure that can be matched should be wide and flexible, or at least flexible enough to match reasonably well the dependence structure found in the data.

To estimate a copula, we can first estimate the marginals F_j, then transform the observations via $(U_1, \ldots, U_d) = (F_1(X_1), \ldots, F_d(X_d))$ (so that they have uniform marginals if the F_j's are continuous), and we find a copula that fits these transformed observations. As d increases, however, we get hit by the curse of dimensionality, in the sense that the difficulty of estimating the copula grows "exponentially" with the dimension. It is generally hard to estimate a copula in more than two or three dimensions, unless we restrict ourselves to certain specific parametric forms.

Sometimes, the target dependence structure may be only *partially specified*, e.g., by specifying only the correlation coefficients (or the rank correlations or Kendall tau's) between the X_j's, or perhaps only between certain pairs of X_j's, and a copula that approximates these correlations is selected from a restricted class. In Section 2.10.9, we discuss one practical approach for specifying the copula indirectly in this way.

Again, it must be stressed that specifying only the marginals and correlations typically leaves a huge amount of freedom for choosing the copula. That is, for given marginals and rank correlations (say), we can often find a large class of *very different* multivariate distributions that match these rank correlations exactly. The choice of a specific distribution within that class may have an important impact on simulation results. This must be kept in mind when reading Section 2.10.9. The NORTA method discussed there is a convenient heuristic for specifying (somewhat arbitrarily) one distribution in that class, using a normal copula. However, the distribution thus specified may turn out to be far from the correct one if the latter has a copula that is far from a normal copula. Hörmann, Leydold, and Derflinger (2004) give other illustrations of this.

2.10.9 The NORTA method

[7] Using normal copulas to model high-dimensional multivariate distribution is natural and attractive, because these copulas are easy to handle and to generate from, and the correlation or rank correlation matrix determines the copula uniquely. In certain situations (e.g., in some queueing systems as in Example 2.3 and for some diffusion approximations), the performance measure of interest depends only or mainly on the mean and covariance matrix (i.e., first two moments) of the distribution and very little on the exact form of the joint distribution beyond these two moments. In these cases, restricting ourselves to normal copulas can be justified because they provide a relatively easy way to approximate a target correlation matrix. Finding the correct type of copula in high-dimensional settings (e.g., 10 to 100 dimensions) is usually much too difficult, so using a normal copula that fits the correlations could be a reasonable compromise between the (unknown) correct dependence model and a naive model that assumes independence of the coordinates.

[7] From Pierre: To be reorganized: this should be placed before the archimedean copulas, and the latter should have more coverage. We ay have a separate subsection for copulas.

Suppose we want a multivariate distribution for $\boldsymbol{X} = (X_1, \ldots, X_d)^\mathsf{t}$ with (a) a given *continuous* marginal cdf F_j for each j and (b) a given rank correlation matrix $\boldsymbol{R}^{(\mathrm{u})}$. For this, it suffices to have a copula C whose correlation matrix is $\boldsymbol{R}^{(\mathrm{u})}$ and to define $X_j = F_j^{-1}(U_j)$ where the random vector $\boldsymbol{U} = (U_1, \ldots, U_d)^\mathsf{t}$ has cdf C. (If some F_j has jumps, then $\boldsymbol{X}$ and $\boldsymbol{U}$ will have different rank correlation matrices, because $F_j(F_j^{-1}(U_j)) \neq U_j$ in that case.) If we decide to use a normal copula, we need to find a valid correlation matrix $\boldsymbol{R}^{(\mathrm{y})}$ for the multinormal distribution, whose corresponding rank correlation matrix equals (or approximates) the target $\boldsymbol{R}^{(\mathrm{u})}$. Once we have done that, we can simply generate a vector $\boldsymbol{Y} = (Y_1, \ldots, Y_d)^\mathsf{t} \sim \mathsf{N}(\boldsymbol{0}, \boldsymbol{R}^{(\mathrm{y})})$ and return $\boldsymbol{U} = (\Phi(Y_1), \ldots, \Phi(Y_d))^\mathsf{t}$, as in Example 2.18. The vector $\boldsymbol{Y}$ has $N(0,1)$ marginals whereas $\boldsymbol{U}$ has $U(0,1)$ marginals. Generating $\boldsymbol{Y}$ is easy: decompose $\boldsymbol{R}^{(\mathrm{y})} = \boldsymbol{L}^{(\mathrm{y})}(\boldsymbol{L}^{(\mathrm{y})})^\mathsf{t}$, generate $\boldsymbol{Z} \sim N(\boldsymbol{0}, \boldsymbol{I})$, i.e., with i.i.d. $N(0,1)$ coordinates, and return $\boldsymbol{Y} = \boldsymbol{L}^{(\mathrm{y})}\boldsymbol{Z}$. The vector $\boldsymbol{U}$ has a correlation matrix $\boldsymbol{R}^{(\mathrm{u})}$ that is easy to compute from $\boldsymbol{R}^{(\mathrm{y})}$ and vice-versa: From Eq. (2.40), we have

$$r_{i,j}^{(\mathrm{y})} = 2\,\sin(r_{i,j}^{(\mathrm{u})}\pi/6) \tag{2.44}$$

where $r_{i,j}^{(\mathrm{u})}$ and $r_{i,j}^{(\mathrm{y})}$ are the elements (i,j) of $\boldsymbol{R}^{(\mathrm{u})}$ and $\boldsymbol{R}^{(\mathrm{y})}$, respectively. Eq. (2.40) also provides a similar relationship between $\boldsymbol{R}^{(\mathrm{y})}$ and the Kendall's tau correlation matrix, so we can specify the latter instead of $\boldsymbol{R}^{(\mathrm{u})}$ and compute the corresponding target $\boldsymbol{R}^{(\mathrm{y})}$ in a similar way. The remainder of this section is written in terms of $\boldsymbol{R}^{(\mathrm{u})}$ but everything adapts to Kendall's tau correlations in a straightforward way.

The use of normal copulas has been proposed by Mardia (1970), Li and Hammond (1975), Iman and Conover (1982), and studied further in more recent articles. Cario and Nelson (1997) coined the appellation *normal to anything (NORTA)*, to indicate that the method transforms a normal vector to a vector having essentially any type of distribution for its marginals. Instead of specifying the rank correlation matrix $\boldsymbol{R}^{(\mathrm{u})}$, some authors specify a target correlation matrix $\boldsymbol{R}^{(\mathrm{x})}$ for the vector $\boldsymbol{X}$. But this complicates the approach, as we will see in a moment, and there seems to be no good reason for wanting to work directly with $\boldsymbol{R}^{(\mathrm{x})}$, as pointed out by Hörmann, Leydold, and Derflinger (2004). In practice, the target $\boldsymbol{R}^{(\mathrm{u})}$ of Kendall's tau correlation matrix would normally be estimated directly from the data.

Again, restricting ourselves to a normal copula severely restrains (indirectly) the form of dependence structure that can be obtained between the U_j's: Only a finite number of parameters (the entries of $\boldsymbol{R}^{(\mathrm{y})}$) can be chosen, whereas the copula C in the general case can be chosen from a much wider class of functions. In other words, only a small subclass of the multivariate distributions can be obtained via the NORTA method. On the other hand, this restriction makes the scheme more manageable.

Can the NORTA method match any valid correlation matrix $\boldsymbol{R}^{(\mathrm{u})}$ for $\boldsymbol{U}$? In the two-dimensional case, the answer is yes, because the Fréchet bounds for the correlation between two normal r.v.'s are ± 1 and they correspond to correlations of ± 1 between the two uniforms, and $r_{1,2}^{(\mathrm{u})}$ is a continuous increasing function of $r_{1,2}^{(\mathrm{y})}$ and vice-versa, as can be seen from Eq. (2.44). In more than two dimensions, however, Ghosh and Henderson (2002) have shown that there are valid rank correlation matrices that cannot be matched by the NORTA method, in the sense that the matrix $\boldsymbol{R}^{(\mathrm{y})}$, whose elements are obtained from those of $\boldsymbol{R}^{(\mathrm{u})}$ via (2.44), is *not* nonnegative definite and therefore is not a valid correlation matrix. The rank correlation matrix is called *NORTA-defective* when this happens and *NORTA-compatible* otherwise. More generally, a correlation matrix $\boldsymbol{R}^{(\mathrm{x})}$ for a random vector $\boldsymbol{X}$ with given marginals

F_j is called *NORTA-defective* [*NORTA-compatible*] *for the given marginals* if the corresponding rank correlation matrix $\boldsymbol{R}^{(\mathrm{u})}$ is NORTA-defective [NORTA-compatible]. Ghosh and Henderson (2002) have shown empirically that a random correlation matrix generated uniformly over the set of all valid correlation matrices for *uniform marginals* F_j is NORTA-defective with a probability that increases rapidly with the dimension d. This probability is near 0.2 for $d = 5$, 0.8 for $d = 10$, and very close to 1 for $d > 15$.

In view of the fact that (2.44) defines a *continuous mapping* between $\boldsymbol{R}^{(\mathrm{u})}$ and $\boldsymbol{R}^{(\mathrm{y})}$, it sounds sensible, when the desired rank correlation matrix $\boldsymbol{R}^{(\mathrm{u})}$ turns out to be NORTA-defective, to seek and use the "nearest" (in some sense) nonnegative definite correlation matrix to $\boldsymbol{R}^{(\mathrm{y})}$. This can be done by defining a measure of distance between matrices and then solving an optimization problem. Both Lurie and Goldberg (1998) and Ghosh and Henderson (2002) suggested and implemented this idea (they were trying to match $\boldsymbol{R}^{(\mathrm{x})}$ instead of $\boldsymbol{R}^{(\mathrm{u})}$, which makes things more complicated, but the general idea is the same). Lurie and Goldberg (1998) define a weighted norm on the space of $d \times d$ matrices $\boldsymbol{A}$ with elements a_{ij} by

$$\|\boldsymbol{A}\|_{\boldsymbol{w}} = \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} a_{ij}^2,$$

where the w_{ij} are arbitrary positive weights. In practice, w_{ij} can be chosen larger for the pairs (i, j) for which we have higher confidence in the estimated value of $r_{i,j}^{(\mathrm{u})}$ for our model, and smaller otherwise. Ghosh and Henderson (2002) use a norm defined by

$$\|\boldsymbol{A}\|_1 = \sum_{i=1}^{d}\sum_{j=1}^{i-1} |a_{ij}| \quad \text{or} \quad \|\boldsymbol{A}\|_\infty = \max_{1\le j<i\le d} |a_{ij}|$$

instead. After selecting a norm $\|\cdot\|$, the next step is to find a valid correlation matrix $\widetilde{\boldsymbol{R}}^{(\mathrm{y})}$ for $\boldsymbol{Y}$, for which the distance $\|\widetilde{\boldsymbol{R}}^{(\mathrm{y})} - \boldsymbol{R}^{(\mathrm{y})}\|$ is minimal, where $\boldsymbol{R}^{(\mathrm{y})}$ is the matrix whose entries are defined by (2.44).

Recall that a matrix $\widetilde{\boldsymbol{R}}^{(\mathrm{y})}$ is a valid correlation matrix for $\boldsymbol{Y}$ if and only if it has unit diagonal elements and it can be written as $\widetilde{\boldsymbol{R}}^{(\mathrm{y})} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{t}}$ where $\boldsymbol{L}$ is a lower triangular matrix. Let $\mathbb{L}$ be the set of lower triangular matrices $\boldsymbol{L}$ such that $\boldsymbol{L}\boldsymbol{L}^{\mathrm{t}}$ has unit diagonal elements. Then the problem of finding the nearest valid correlation matrix can be formulated as:

$$\text{Minimize } \|\boldsymbol{L}\boldsymbol{L}^{\mathrm{t}} - \boldsymbol{R}^{(\mathrm{y})}\| \quad \text{subject to } \boldsymbol{L} \in \mathbb{L}. \tag{2.45}$$

This approximation methodology can be used as well if the target rank correlation matrix $\boldsymbol{R}^{(\mathrm{u})}$ is itself invalid. This may happen if this matrix was estimated from data, for example. Lurie and Goldberg (1998) solve the nonlinear optimization problem (2.45) by a quasi-Newton or Gauss-Newton algorithm (with their weighted quadratic norm), whereas Ghosh and Henderson (2002) use techniques for solving semidefinite programming problems. In their experiments, they find that when $\boldsymbol{R}^{(\mathrm{u})}$ is NORTA-defective, there is almost always a NORTA-compatible rank correlation matrix which is very close.

In the case where we want to specify directly $\boldsymbol{R}^{(\mathrm{x})}$ instead of $\boldsymbol{R}^{(\mathrm{u})}$ (we do not recommend it, but NORTA is usually defined under that framework), we can proceed as follows. Suppose that the X_j's have been standardized so that $\mathbb{E}[X_j] = 0$ and $\mathrm{Var}[X_j] = 1$ (it suffices to

multiply by the desired standard deviation and add the desired mean afterward, for each j). Let $r^{(\mathrm{y})}_{i,j}$ and $r^{(\mathrm{x})}_{i,j}$ denote the elements (i,j) of $\boldsymbol{R}^{(\mathrm{y})}$ and $\boldsymbol{R}^{(\mathrm{x})}$, respectively. By integrating with respect to the bivariate normal density of (Y_i, Y_j), we obtain

$$r^{(\mathrm{x})}_{i,j} = \mathbb{E}[X_i X_j] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_i^{-1}(\Phi(y_i))F_j^{-1}(\Phi(y_j)) \cdot \frac{1}{2\pi\sqrt{1-(r^{(\mathrm{y})}_{i,j})^2}} \exp\left[-\frac{y_i^2+y_j^2-2r^{(\mathrm{y})}_{i,j}y_i y_j}{2(1-(r^{(\mathrm{y})}_{i,j})^2)}\right] dy_i dy_j. \tag{2.46}$$

Given $r^{(\mathrm{x})}_{i,j}$, we must solve this integral equation for $r^{(\mathrm{y})}_{ij}$, whereas we get it directly from Eq. (2.44) when we have $r^{(\mathrm{u})}_{i,j}$ and both F_i and F_j are continuous. This has to be done for all $d(d-1)/2$ pairs (i,j) such that $1 \le i < j \le d$, so this can be very time-consuming, especially if precise values are sought.

Eq. (2.46) actually generalizes (2.44) to $r^{(\mathrm{x})}_{i,j} = \varphi_{i,j}(r^{(\mathrm{y})}_{i,j})$ where each $\varphi_{i,j} : [-1,1] \to [-1,1]$ is a one-dimensional transformation that depends on F_i and F_j. Cario and Nelson (1997) pointed out that for any given pair of marginals (F_i, F_j), $\varphi_{i,j}$ is a nondecreasing and continuous function, $\varphi_{i,j}(0) = 0$, and the range of possible correlations $r^{(\mathrm{x})}_{i,j}$ goes from $\underline{r}_{i,j} = \varphi_{i,j}(-1)$ to $\overline{r}_{i,j} = \varphi_{i,j}(1)$ (see Exercise 2.38). These $\underline{r}_{i,j}$ and $\overline{r}_{i,j}$ are the Fréchet bounds of Theorem 2.6 for the given marginals F_i and F_j. A value of $r^{(\mathrm{y})}_{i,j}$ that solves (2.46) for a given $r^{(\mathrm{x})}_{i,j}$ can be approximated by a root finding procedure that exploits these informations. Solving these equations provides a target correlation matrix $\boldsymbol{R}^{(\mathrm{y})}$. In the case where the solution of (2.46) is unique for each (i,j) (Henderson, Chiera, and Cooke 2000 give conditions under which this is true), it is clear that if $\boldsymbol{R}^{(\mathrm{x})}$ is NORTA-compatible for the given marginals, then solving (2.46) will necessarily yield a valid (nonnegative definite) correlation matrix $\boldsymbol{R}^{(\mathrm{y})}$ for $\boldsymbol{Y}$. So if (2.46) is solved exactly, NORTA will fails if and only if the target matrix $\boldsymbol{R}^{(\mathrm{x})}$ is NORTA-defective for the given marginals. (Note that such a $\boldsymbol{R}^{(\mathrm{x})}$ is not necessarily invalid for these marginals.)

To avoid the high cost of solving the integral equations (2.46), Lurie and Goldberg (1998) first generate a sample of n i.i.d. $N(\boldsymbol{0}, \boldsymbol{I})$ random vectors $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_n$. Then, by an iterative method, they find a matrix $\boldsymbol{L} \in \mathbb{L}$ that approximately minimizes $\|\boldsymbol{L}\boldsymbol{L}^{\mathrm{t}} - \boldsymbol{R}^{(\mathrm{x})}\|_{\boldsymbol{w}}$, where the weights $w_{i,j}$ are all ones and $\boldsymbol{R}^{(\mathrm{x})}$ is the empirical correlation matrix of the n vectors $\boldsymbol{X}_i = (X_{i,1}, \ldots, X_{i,n})^{\mathrm{t}}$, with $X_{i,j} = F_j^{-1}(\Phi(Y_{i,j}))$, and $(Y_{i,1}, \ldots, Y_{i,d})^{\mathrm{t}} = \boldsymbol{Y}_i = \boldsymbol{L}\boldsymbol{Z}_i$. The starting point of their iterative method is the matrix $\boldsymbol{L} \in \mathbb{L}$ such that $\boldsymbol{L}\boldsymbol{L}^{\mathrm{t}} = \boldsymbol{R}^{(\mathrm{x})}$. Their technique is an *approximation* in the sense that they find the best correlation matrix $\boldsymbol{R}$ *only for the particular sample that they have generated* rather than for $\boldsymbol{Z} \sim N(\boldsymbol{0}, \boldsymbol{I})$. Increasing n reduces this approximation error but increases the work.

The NORTA method requires non-negligible work in general, both for its setup and for generating the variates. The amount of work required to generate $\boldsymbol{Y} = \boldsymbol{L}^{(\mathrm{y})}\boldsymbol{Z}$ increases as d^2, the square of the dimension. On the other hand, it is a simple and easy-to-implement heuristic that allows one to capture some of the dependence structure by matching the first two moments of the joint distribution. See Chen and Jeng (1998), Nelson and Yamnitsky (1998), and Ghosh and Henderson (2002) for further discussion.

♣ Add material from Avramidis, Channouf, and L'Ecuyer (2009) for the case of discrete marginals.

2.11 Discrete Choice Models

Situations where an individual makes a choice among a finite number of alternatives are frequent. A natural way to model this is to model the choice by a multinomial random variable: The individual selects alternative j with probability p_j, where j runs over all possible alternatives. For example, the choice could be between different brands when buying a product, between different routes or transportation modes to reach a given destination, or between different airlines, flights, and travel classes when buying a flight ticket between two give cities. The probabilities p_j usually change when the properties of the different alternatives change, and they also depend on the individual.

A standard way of taking this variation into account is to assume that each individual has its own *utility function* that can be expressed as a function of both the properties of any given alternative and the characteristics of the individual, usually with some random terms or parameters in the function. The individual always selects the alternative that gives her/him the largest utility. Thus, all the randomness is captured by the (random) utility function. We already saw an instance of this in Example 1.58.

In a model with additive noise, the utility of alternative j for individual q is

$$U_{q,j} = V_{q,j}(\boldsymbol{\beta}_q, \boldsymbol{x}_{q,j}) + \epsilon_{q,j},$$

where $V_{q,j}$ is a known deterministic function, $\boldsymbol{\beta}_q$ is a vector of parameters for individual q, $\boldsymbol{x}_{q,j}$ is a vector of observed attributes for alternative j and individual q, $\epsilon_{q,j}$ is a random term that reflects the unknown part of the utility, and the $\epsilon_{q,j}$'s are assumed independent across all pairs (q, j), usually with a common distribution. Often, $V_{q,j}(\boldsymbol{\beta}_q, \boldsymbol{x}_{q,j})$ is a linear function of $\boldsymbol{x}_{q,j}$, of the form $V_{q,j}(\boldsymbol{\beta}_q, \boldsymbol{x}_{q,j}) = \boldsymbol{\beta}_q^{\mathsf{t}} \boldsymbol{x}_{q,j}$, in which case the utility obeys essentially a linear regression model. However, the behavior of the model depends very much on the distribution of the residuals $\epsilon_{q,j}$, which is not necessarily normal.

A popular choice discussed earlier in Example 1.58 is to assume that $\epsilon_{q,j}$ is Gumbel with mean 0 and scale parameter 1. Under this assumption, and if we also assume that $\boldsymbol{\beta}_q$ is fixed (not random), one can show that the probability that q chooses alternative j, i.e., that $U_{q,j} > U_{j,a}$ for all $a \neq j$, is given by

$$p_{q,j} = \exp[V_{q,j}(\boldsymbol{\beta}_q, \boldsymbol{x}_{q,j})]/K_q, \tag{2.47}$$

where $K_q = \sum_{a \in \mathcal{A}(q)} \exp[V_{q,a}(\boldsymbol{\beta}_q, \boldsymbol{x}_{q,a})]$ is the appropriate normalizing constant, and $\mathcal{A}(q)$ is the set of all alternatives available to q. In this case, the choice probabilities can be computed rather easily, provided that the number of alternatives is not excessive. In that setting, $\boldsymbol{\beta}_q$ is usually taken as a constant vector $\boldsymbol{\beta}$ that does not depend on q. This is the *multinomial logit model* (Train 2003). It was originally introduced for the case of two alternatives only, say 0 and 1. In that case, if we normalize the utility function so that $V_{q,0} = 0$, we obtain $p_{q,1} = e^{V_{q,1}}/[1 + e^{V_{q,1}}]$, which is equivalent to $V_{q,1} = \ln[p_{q,1}/(1 - p_{q,1})]$. The latter is the *logit function* of $p_{q,1}$ whence the name of the method.

The multinomial logit model has certain limitations and a popular trend is to generalize it to the case when $\boldsymbol{\beta}_q$ is itself a random vector. We then have a mixture of distributions for the choice probabilities. If we still assume a Gumbel distribution for the $\epsilon_{q,j}$, the resulting model is called a *mixed logit model* (Train 2003) that we examined in Example 1.58 and Exercise 1.12.

♣ Multivariate probit model. Parameter estimation. Also probit model for the probability of exceeding a threshold.

2.12 Stochastic processes

2.12.1 Markov processes

Recall that a stochastic process $\{Y_t,\ t \geq 0\}$ is *Markovian* if conditional on the present (t, Y_t), the future $\{Y_s,\ s > t\}$ is independent of the past $\{Y_s,\ s < t\}$. Processes encountered in discrete-event simulation can always be made Markovian simply by incorporating enough information in Y_t (i.e., by enlarging the state space appropriately). A Markovian process whose time index is $I = \{0, 1, 2, \dots\}$ is a *discrete-time Markov chain (DTMC)*. If its time index is $[0, \infty)$ but its state changes only by jumps at random times and is constant between those jumps, we have a *Markovian jump process*, also called a *continuous-time Markov chain (CTMC)*. In this case, the time to the next jump always has the exponential distribution. See Section A.18 in the appendix for more details.

Example 2.23 Consider the $GI/G/1$ queue example (Section 1.11). It follows immediately from the Lindley equation (1.39) that the process $\{W_i,\ i \geq 0\}$ is Markovian, and it is a DTMC. On the other hand, the process $\{Q(t),\ t \geq 0\}$ is not Markovian, unless the interarrival and service time distributions are both exponential. Indeed, if we look only at the number of customers in the queue at time t and forget about the past, we miss important information, such as the elapsed service time for the customer currently being served.

Suppose that the events are the arrivals, ends of service, and end of the simulation at a fixed time T, as in Section 1.11. Define the state $\mathcal{S}_i$ of the model at time t_i, right after event e_i has occurred, as $\mathcal{S}_i = (t_i, Q_i, \zeta_i, \xi_i)$, where Q_i is the number of customers in the queue, ζ_i is the time until the next arrival, and ξ_i is the time until the next departure (put $\xi_i = -1$, say, if the server is idle). Then, $\{\mathcal{S}_i,\ i \geq 0\}$ is a DTMC. □

Example 2.24 In a simulation context, the discrete-time stochastic process that gives the *state* of the simulation program at the successive event times, including the value of the simulation clock and the content of the event list, must be a Markov chain (otherwise the simulation would not be possible). In practice, we are often interested in other stochastic processes related to a given model, some not necessarily Markovian. This causes no problem, as long as the evolution of any such process can be easily deduced from the time-evolution of the state process of the entire simulation model. To be more precise, we introduce the following notation.

Consider the discrete-time process $\{\mathcal{S}_i,\ i = 0, 1, 2, \dots\}$, where $\mathcal{S}_i$ represents the *state of the simulation model* at the time t_i of the ith event e_i, just after the event has occurred. We assume that the state $\mathcal{S}_i$ always includes the clock time t_i. From this discrete-time process, we construct the continuous-time process $\{\mathcal{S}(t),\ t \geq 0\}$ defined by $\mathcal{S}(t) = \mathcal{S}_{N(t)}$, where $N(t) = \sup\{i \mid t_i \leq t\}$ is the number of events that occur in the time interval $(0, t]$. We call $\mathcal{S}(t)$ the state of the model at time t. It is the same as the state of the model at the time of the last event that has occurred during the time interval $(0, t]$. We assume that the process

$\{\mathcal{S}_i,\, i \geq 0\}$ is a Markov chain (a DTMC) and that the *state of the simulation program* at time t_i, just after the occurrence of e_i, always contains $\mathcal{S}_i$ (and usually more information). This does not imply that $\{\mathcal{S}(t),\, t \geq 0\}$ is Markovian; typically, it is not (see Example 2.23). Processes like $\{\mathcal{S}(t),\, t \geq 0\}$ are called *semi-Markov* processes (Ross 1970, Çinlar 1975). The process $\{\mathcal{S}(t),\ t \geq 0\}$ is uniquely determined from $\{\mathcal{S}_i,\ i = 0, 1, \dots\}$, but the opposite is not true. For example, if $t_i = t_{i-1}$ (simultaneous events), $\mathcal{S}_{i-1}$ cannot be deduced from $\{\mathcal{S}(t),\, t \geq 0\}$. So, the DTMC process $\{\mathcal{S}_i,\, i \geq 0\}$ contains the most information. It is the one we usually work with. □

Numerical methods are available for computing steady-state probabilities or average cost/reward for Markov chain models. In principle, they could be applied to compute performance measures for entire simulation models. But from a practical viewpoint, these methods usually break down when the dimension of the state space becomes too large, because they then require excessive computing times and memory. With simulation, in contrast, high-dimensional state spaces are no problem.

Filtrations for stochastic processes and stopping times with respect to filtrations are defined in Section A.16 of the appendix. In short, T is a stopping time with respect to the filtration $\{\mathcal{F}_t,\, t \geq 0\}$ if knowing $\mathcal{F}_T$ always reveals the value of T.

Example 2.25 Consider one day of operation of the call center in Section 1.12 and let $\mathcal{F}_i$ represent all known information at the occurrence of the ith simulation event e_i. If N is the total number of events that occur during the day, then N is a stopping time with respect to $\{\mathcal{F}_i,\, i \geq 0\}$. The last event occurs either when the center closes or when the last call ends, and we have enough information at that time to know that it is the last event. Now, let N' be the event number which corresponds to the *arrival* of the last call of the day. This N' is *not* a stopping time, because at the time of occurrence of that event, we cannot be sure that no additional customer will arrive before the center closes (under the assumption that the state $\mathcal{S}_i$ does not reveal arrival times that occur after time t_i). □

2.12.2 Random walks

A simple, yet very important type of Markovian stochastic process, with applications in hundreds of areas, is a random walk. There are several kinds of random walks; they can evolve over the integers, in the real space, on a graph, or in other types of spaces.

A *random walk* $\{\boldsymbol{S}_j,\, j \geq 0\}$ in $\mathbb{R}^d$ can be defined by

$$\boldsymbol{S}_j = \boldsymbol{S}_{j-1} + \boldsymbol{X}_j,$$

where $\boldsymbol{S}_0 = \backslash_0 \in \mathbb{R}^d$ (a constant vector) and the $\boldsymbol{X}_j$ are i.i.d. random vectors in $\mathbb{R}^d$. To simulate trajectories of continuous-time stochastic processes in the real space $\mathbb{R}^d$, we often approximate them by random walks, as we shall see in the forthcoming subsections.

[8]

[8] From Pierre: Perhaps a subsection on deterministic and random time change around here? Or perhaps before the Lévy process section.

2.13 Poisson Processes

2.13.1 Definition and main properties

A *counting process* is a continuous-time stochastic process $N = \{N(t),\, t \geq 0\}$ taking its values in $\{0, 1, 2, \dots\}$, and with non-decreasing right-continuous trajectories. Usually, $N(0) = 0$ and one can interpret $N(t)$ as the number of events of some sort (often called *arrivals*) that have occurred during the time interval $(0, t]$. A *renewal process* is a counting process for which the times between the successive jumps are i.i.d. random variables.

A counting process with $N(0) = 0$ is a *Poisson process* if it satisfies the following two axioms (Çinlar 1975, page 95):

(a) Arrivals are one by one. That is, the probability that $N(t)$ has a jump of size larger than 1 somewhere in the time interval $[0, \infty)$ is 0.

(b) For any fixed $t_2 > t_1 > 0$, the random variable $N(t_2) - N(t_1)$, the number of jumps in the interval $(t_1, t_2]$, is independent of $\{N(t),\, t \leq t_1\}$, the history of the process up to time t_1.

These axioms give intuition about when we should expect a counting process to be approximately Poisson, namely when events occur randomly in time, independently of each other.

Define $a(t) = \mathbb{E}[N(t)]$. For the remainder of this section, we assume that a is a continuous function of t, differentiable except perhaps at a few isolated points. Let $\lambda(t) = a'(t)$ be the value of the derivative at the points t where it exists. Then one has

$$a(t) = \int_0^t \lambda(s)ds.$$

The functions $\lambda(\cdot)$ and $a(\cdot)$ are called the *rate function* and the *cumulative rate function* of the process, respectively. An interpretation of the *rate* or *intensity* $\lambda(t)$ of the Poisson process at time t is that for a small $\epsilon > 0$, the probability of a jump in the next ϵ units of time is $\lambda(t)\epsilon + o(\epsilon)$ and the probability of two or more jumps is $o(\epsilon)$. That is,

$$\mathbb{P}[N(t+\epsilon) - N(t) = 1] \approx 1 - \mathbb{P}[N(t+\epsilon) - N(t) = 0] \approx \lambda(t)\epsilon.$$

This is similar to the failure rate function of a continuous random variable.

When $\lambda(t) = \lambda$ for all $t \geq 0$, for some constant $\lambda > 0$, the Poisson process is called *stationary*, in the sense that it has a time-stationary rate and its increments have a time-stationary distribution, or *time-homogeneous*. A *standard Poisson process* is a stationary Poisson process with rate $\lambda = 1$.

Proposition 2.8 *(Çinlar 1975, page 97.) If $\{N(t),\, t \geq 0\}$ is a Poisson process, then for any fixed $t_2 > t_1 \geq 0$, $N(t_2) - N(t_1)$ is a Poisson random variable with mean $a(t_2) - a(t_1) = \int_{t_1}^{t_2} \lambda(t)dt$. In the stationary case, the mean is $(t_2 - t_1)\lambda$.*

Define $T_0 = 0$, let $0 < T_1 \leq T_2 \leq T_3 \leq \cdots$ be the jump times of the process $\{N(t),\, t \geq 0\}$, and let $A_j = T_j - T_{j-1}$, $j \geq 1$, be the times between the successive jumps. The following

proposition gives an easy way of generating a stationary Poisson process; it suffices to generate the A_j's, which are i.i.d. exponentials.

Proposition 2.9 *(Taylor and Karlin 1998, page 292.) A counting process* $\{N(t),\, t \geq 0\}$*, with* $N(0) = 0$*, is a stationary Poisson process with rate* λ *if and only if the random variables* $A_1, A_2, \ldots$ *are* i.i.d. *exponential with rate* λ*.*

For a Poisson process $\{N(t),\, t \geq 0\}$, we can generate the number of jumps in any given time interval by generating a Poisson random variable (Proposition 2.8), but that does not tell us the location of these jumps in the interval. The next proposition says that for a *stationary Poisson process*, conditional on the number of jumps, these jumps are uniformly distributed over the interval. One can therefore simulate the Poisson process by first generating the number N of jumps and then generating N i.i.d. uniforms over the time interval where these jumps occur to determine the locations of the jumps. For a proof, see, e.g., Taylor and Karlin (1998), p. 299.

Proposition 2.10 *Given that a stationary Poisson process has* n *jumps in the time interval* $(t_1, t_2]$*, for* $t_2 > t_1 \geq 0$*, the conditional distribution of the times at which these* n *jumps have occurred is the same as the distribution of* n i.i.d. $\mathsf{Uniform}(t_1, t_2)$ *random variables, sorted by increasing order.*

2.13.2 Standardization by nonlinear time change

Any Poisson process $\{N(t),\, t \geq 0\}$ can be transformed into a standard (stationary) one, and vice-versa, by stretching the time scale where the rate needs to be decreased and contracting it where the rate should be increased, as follows. Suppose we are interested in a Poisson process with rate function λ and cumulative rate function a. Let $N_0 = \{N_0(x),\, x \geq 0\}$ be a standard Poisson process (with rate 1) and define the process $N = \{N(t),\, t \geq 0\}$ by setting $N(t) = N_0(a(t))$ for $t \geq 0$. If the jump times of the process N are $T_1, T_2, \ldots$ and those of N_0 are $X_1, X_2, \ldots$, then we have $X_j = a(T_j)$, or equivalently $T_j = a^{-1}(X_j)$.

Proposition 2.11 *The process* $\{N(t),\, t \geq 0\}$ *is a Poisson process with cumulative rate function* a *if and only if* $\{N_0(x),\, x \geq 0\}$ *is a standard Poisson process.*

Proof. Clearly, the axioms (a) and (b) defining the Poisson process are satisfied for the process N if and only if they are satisfied for N_0, because a is a continuous and non-decreasing function. It remains to verify that $\mathbb{E}[N(t)] = a(t)$ for all $t > 0$ if and only if $\mathbb{E}[N_0(x)] = x$ for all $x > 0$. If $\mathbb{E}[N_0(x)] = x$, then $\mathbb{E}[N(t)] = \mathbb{E}[N_0(a(t))] = a(t)$. Conversely, by putting $a(t) = x$ we have $N_0(x) = N(a^{-1}(x))$. Therefore, if $\mathbb{E}[N(t)] = a(t)$, then $\mathbb{E}[N_0(x)] = \mathbb{E}[N(a^{-1}(x))] = a(a^{-1}(x)) = x$.

This transformation is very handy for simulating a Poisson process with an arbitrary rate function, provided that the cumulative rate function a is easy to invert. One generates the jump times X_j of the standardized process N_0 and computes the jump times T_j of the original process N via $T_j = a^{-1}(X_j)$, for $j = 1, 2, 3, \ldots$.

2.13.3 Modeling and estimating non-stationary rates

The rate function of a non-stationary Poisson process is most often modeled as a piecewise-constant function (as in the call center example of Section 1.12), mainly because the process is then easy to simulate. The piecewise-constant rate is also easy to estimate over each interval of stationarity if the boundaries of these intervals are fixed: Just take the average number of arrivals per unit of time over each interval. However, estimating the boundaries when they are unknown is more difficult.

Lee, Wilson, and Crawford (1991) and Kuhl, Wilson, and Johnson (1997) have proposed a model of non-stationary Poisson process whose rate function $\lambda(t)$ has exponential, polynomial, and multiperiodic trigonometric components:

$$\lambda(t) = \exp\left[\sum_{i=0}^{m} \alpha_i t^i + \sum_{k=1}^{p} \gamma_k \sin(\omega_k t + \phi_k)\right], \tag{2.48}$$

where the parameters α_i, γ_k, ω_k, and ϕ_k represent the amplitudes of trends, and the amplitudes, frequencies, and phase shifts of the periodic components, respectively. Kuhl, Wilson, and Johnson (1997, 1998) have also developed procedures and software to estimate the parameters and to simulate the corresponding process.

[9]

Other models for the rate of a Poisson process are discussed in Arkin and Leemis (2000), Leemis (1991), Leemis (2001), Nelson and Yamnitsky (1998), Wilson (1997) and the references given there. One obvious constraint when designing such a model is that $\lambda(t) \geq 0$ for all t, i.e., $a(t)$ must be non-decreasing in t.

[10]

A statistical *goodness-of-fit test* for a stationary Poisson process over a fixed time horizon t can be performed as follows: Test if the empirical distribution of the jump times in $(0, t]$ corresponds to the distribution of $N(t)$ i.i.d. $U(0, t)$ random variables sorted by increasing order. If the number of jumps is fixed, instead of the time t, then test if the spacings are i.i.d. exponential. For a non-stationary process, a simple testing approach is to first standardize the process, and then apply the procedure just described to the standardized process.

2.13.4 Composition and decomposition

Poisson processes can be *superposed* and *decomposed*. If $\{N_1(t),\, t \geq 0\}, \ldots, \{N_k(t),\, t \geq 0\}$ are k independent Poisson processes with respective rates functions $\lambda_1(\cdot), \ldots, \lambda_k(\cdot)$, then $N(t) = N_1(t) + \cdots N_k(t)$ defines a Poisson process with rate function $\lambda(t) = \lambda_1(t) + \cdots + \lambda_k(t)$. Conversely, if $\{N(t),\, t \geq 0\}$ is a Poisson process with rate function $\lambda(\cdot)$, if each arrival is of type j with probability p_j, independently of its time of occurrence and of the past, and if $N_j(t)$ denotes the number of arrivals of type j during $(0, t]$, then $\{N_j(t),\, t \geq 0\}$ is a Poisson process with rate function $\lambda_j(t) = p_j \lambda(t)$, $t \geq 0$.

[9] From Pierre: Spline models.

[10] From Pierre: — Examples: restaurants, banks, etc.

Example 2.26 Suppose that calls arrive to a call center according to a stationary Poisson process with rate λ and that each call is of type j with probability p_j, independently of other calls and arrival times. Then the calls of type j arrive according to a Poisson process with rate $p_j\lambda$. This process can be simulated by generating the times between successive arrivals as i.i.d. exponentials with parameter (rate) $p_j\lambda$. □

2.13.5 Compound Poisson Process

We obtain a *compound Poisson process* if instead of having jump sizes all equal to 1, the size of the jth jump is a random variable Y_j, where the Y_j's are i.i.d. and independent of the T_j's. This can be used to model group arrivals or sales in a store, for example. The Y_j do not have to be integer-valued.

Such processes appear in time-stationary form in the decomposition of Lévy processes. Those time-stationary compound Poisson process are often specified via their *Lévy measure* ν, where for every (measurable) subset $B \subseteq \mathbb{R}$, $\nu(B)$ gives the total jump rate for jumps whose sizes belong to B. In the cases where the total jump rate $\lambda = \nu(\mathbb{R})$ is finite, one can simulate the process $N = \{N(t),\, t \geq 0\}$ that counts all the jumps, and then generate the jump sizes Y_j independently via the measure $\nu(\cdot)/\lambda$. That is, the probability that any given jump has size in B is $\nu(B)/\lambda$. At time t, the compound process takes the value

$$X(t) = \sum_{j=1}^{N(t)} Y_j.$$

In some situations, the total jump rate $\nu(\mathbb{R})$ is infinite, but when this happens, the jump rate usually becomes finite if we consider only the jumps whose absolute sizes are larger than ϵ, for any (small) $\epsilon > 0$. Then, to simulate the process, we can simulate only the jumps whose sizes are in $\mathbb{R} \setminus [-\epsilon, \epsilon]$, whose total jump rate $\nu(\mathbb{R} \setminus [-\epsilon, \epsilon])$ is finite, and approximate the total contribution of the other (very small) jumps by another method. See Asmussen and Glynn (2007) for further details.

As an easy generalization, the Lévy measure ν can also be defined over $\mathbb{R}^d$, in which case the Y_j's and $X(t)$ become d-dimensional vectors.

2.13.6 Cox Processes

If the rate function $\{\lambda(t),\, t \geq 0\}$ is itself a stochastic process, and if for any fixed trajectory of this process, the process $\{N(t),\, t \geq 0\}$ is a Poisson process with rate function λ, then $\{N(t),\, t \geq 0\}$ is called a *doubly stochastic Poisson process*, or *Cox process*. The process $\{N(t),\, t \geq 0\}$ itself is generally *not* a Poisson process in this case; it is a Poisson process only conditionally on $\{\lambda(t),\, t \geq 0\}$. See, e.g., Taylor and Karlin (1998), Sections V.1.4 and VI.7, for examples of such processes.

Example 2.27 Suppose we want to model the arrival process of customers to an outdoor ice cream stand, or the arrival process of phone calls to a taxi central dispatcher, from 8:00 to 22:00 on a week day. A first idea is to use a Poisson process with a time-varying arrival rate

$\lambda(t)$ for $8 \le t \le 22$. However, the arrival rate for a given day should be strongly influenced by external factors such as the weather. One may then consider the following model, which was also adopted for the call center in Section 1.12. Let $\widetilde{\lambda}(t)$, $0 \le t \le 22$, be the arrival rate function for an average day, and let B be a positive random variable with mean 1. On the ith day, the arrival process is Poisson with rate function $\lambda(t) = B_i\widetilde{\lambda}(t)$, for $8 \le t \le 22$, where the B_i are i.i.d. copies of B. The rate process is thus inflated for the entire day when B_i is large and deflated for the entire day when B_i is small. A large B_i would correspond to a hot and sunny day for the ice cream stand, and to a rainy day for the taxis. Avramidis, Deslauriers, and L'Ecuyer (2004) study this model when B has the gamma distribution, as well as alternative models for this type of situation. For a data set coming from a call center, they find that this model is more realistic than a non-stationary Poisson process model with deterministic rate function. □

[11]

2.13.7 Spatial Poisson Process

So far we saw Poisson processes for which arrival events are characterized only by their occurrence times. This can be generalized to Poisson processes in space, where the arrival events have an occurrence location in some region R of the d-dimensional real space $\mathbb{R}^d$. A Poisson process over R is defined by specifying a rate function $\lambda : R \to [0, \infty)$, where $\lambda(\boldsymbol{x})$ represents the arrival rate at $\boldsymbol{x}$, for $\boldsymbol{x} \in R$. The process is *homogeneous* if λ is constant over R. Note that one of the coordinates of $\boldsymbol{x}$ can represent the occurrence time. We recover the "ordinary" Poisson process discussed previously by taking $R = [0, \infty)$ and $\boldsymbol{x} = t$.

For any (measurable) $B \subseteq R$, define

$$\nu(B) = \int_B \lambda(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

and let $N(B)$ be the (random) number of arrival events (or points) in B. By definition, we have a Poisson process with measure ν if and only if the following two conditions are satisfied:

(a) For each $B \subseteq R$, $N(B) \sim \mathsf{Poisson}(\nu(B))$.

(b) If $B_1, \ldots, B_k$ are disjoint subsets of R, then $N(B_1), \ldots, N(B_k)$ are independent random variables.

The main properties seen earlier extend to this more general case. The total number of jumps $N(R)$ is Poisson with mean $\nu(R)$, and conditional on $N(R)$, the positions of these points are independent random variables with density $\lambda(\boldsymbol{x})/\nu(R)$ for all $\boldsymbol{x} \in R$. These two properties provide a general way of simulating realizations of the process: First generate $N(R)$, and then the positions of the points. When the process is homogeneous, these positions have the uniform distribution over R.

[11] From Pierre: Add discussion on other piecewise constant models with random rates in Channouf and L'Ecuyer (2012), Oreshkin, Régnard, and L'Ecuyer (2016). Also the models for arrival bursts in L'Ecuyer, Gustavsson, and Olsson (2018).

In the non-homogeneous case, if the rate function λ is upper bounded by a constant $\bar{\lambda}$, then another way of simulating the process is as follows. Generate a Poisson random variable N_0 with mean $\text{vol}(R)\,\bar{\lambda}$, where $\text{vol}(R) = \int_R \mathrm{d}\boldsymbol{x}$ represents the volume of R. Then repeat the following N_0 times, independently: Generate a point $(\boldsymbol{X}, Y)$ uniformly in $R \times [0, \bar{\lambda})$, and keep the point $\boldsymbol{X}$ if and only if $Y \le \lambda(\boldsymbol{x})$. The points that are kept form the realization of the spatial Poisson process.

♣ Discuss other arrival process models somewhere: Dirichlet, normal copulas, etc.

2.14 Brownian Motion and Gaussian Processes

2.14.1 Brownian motion (BM)

A BM resembles a stationary Poisson process; the difference is that for the BM, the increment over any given time interval has the normal distribution instead of the Poisson distribution, and the trajectory is continuous. We define the BM in the d-dimensional real space $\mathbb{R}^d$. The ordinary univariate BM is the special case where $d = 1$. [12]

Definition 2.1 A d-dimensional BM with *drift* vector $\boldsymbol{\mu}$ and *covariance* matrix $\boldsymbol{\Sigma}$ is a process $\boldsymbol{X} = \{\boldsymbol{X}(t) = (X_1(t), \ldots, X_d(t)) \in \mathbb{R}^d,\ t \ge 0\}$ for which:

(a) $\boldsymbol{X}(0) = 0$;

(b) if $s \ge 0$ and $t > 0$, then $\boldsymbol{X}(s+t) - \boldsymbol{X}(s) \sim N(t\boldsymbol{\mu}, t\boldsymbol{\Sigma})$;

(c) for every set of disjoint time intervals $(t_1, t_2], \cdots, (t_{2k-1}, t_{2k}]$, the increments $\boldsymbol{X}(t_2) - \boldsymbol{X}(t_1), \ldots, \boldsymbol{X}(t_{2k}) - \boldsymbol{X}(t_{2k-1})$ are mutually independent random vectors.

□

When $d = 1$, the single element $\sigma^2 = \sigma_{11}$ of $\boldsymbol{\Sigma}$ is the *variance parameter* or *diffusion coefficient*. When $\boldsymbol{\mu} = \boldsymbol{0}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$ (the identity), we have a (multivariate) *standard BM*; this is a process whose d coordinates are simply independent one-dimensional standard BMs, with zero drift and a diffusion coefficient of 1. A standard BM is also known as a *Wiener process*.

With probability 1, the sample path of a BM is continuous everywhere but is nowhere differentiable, and each coordinate has infinite variation.

For any decomposition of the form $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, we can write

$$\boldsymbol{X}(t) = t\boldsymbol{\mu} + \boldsymbol{A} \cdot \boldsymbol{B}(t) \tag{2.49}$$

where $\boldsymbol{B} = \{\boldsymbol{B}(t),\ t \ge 0\}$ is a d-dimensional standard BM. This provides an easy way to generate a process trajectory at a fixed set of observation times $0 = t_0 < t_1 < \cdots < t_c$ (this is called a *skeleton* of the process trajectory): Generate the increments $\boldsymbol{B}(t_j) - \boldsymbol{B}(t_{j-1}) \sim$

[12] From Pierre: In other places in this book, and also in SSJ, d is the number of observation times and c is the dimension of the state. Here we have the reverse. This should be unified.

$N(\mathbf{0}, (t_j - t_{j-1})\boldsymbol{I})$, for $j = 1, \ldots, c$, and use Eq. (2.49) to compute the $\boldsymbol{X}(t_j)$ from the $\boldsymbol{B}(t_j)$. This method simulates a multivariate random walk. It can also be written as: Generate cd i.i.d. $N(0,1)$ random variables $Z_{1,1}, \ldots, Z_{1,d}, \ldots, Z_{c,1}, \ldots, Z_{c,d}$, and use the recurrence

$$\boldsymbol{X}(t_j) = \boldsymbol{X}(t_{j-1}) + (t_j - t_{j-1})\boldsymbol{\mu} + \sqrt{t_j - t_{j-1}}\boldsymbol{A} \cdot (Z_{j,1}, \ldots, Z_{j,d})^{\mathsf{t}} \tag{2.50}$$

for $j = 1, \ldots, c$.

In general, one has $\mathbb{E}[X_i(t)] = t\mu_i$ and

$$\mathrm{Cov}[X_i(s), X_j(t)] = \min(s,t)\sigma_{i,j}, \tag{2.51}$$

where $\sigma_{i,j}$ is the (i,j)th element of $\boldsymbol{\Sigma}$ (Exercise 2.41). The cd-dimensional vector $\boldsymbol{Y} = (X_1(t_1), \ldots, X_d(t_1), \ldots, X_1(t_c), \ldots, X_d(t_c))^{\mathsf{t}}$ has a multinormal distribution whose mean vector $\boldsymbol{\mu}_{\mathrm{y}}$ and covariance matrix $\boldsymbol{\Sigma}_{\mathrm{y}}$ are readily available from these mean and covariance expressions. It is then possible to decompose $\boldsymbol{\Sigma}_{\mathrm{y}} = \boldsymbol{A}_{\mathrm{y}}\boldsymbol{A}_{\mathrm{y}}^{\mathsf{t}}$ the way we want, and generate $\boldsymbol{Y}$ just like an ordinary multivariate normal vector: $\boldsymbol{Y} = \boldsymbol{\mu}_{\mathrm{y}} + \boldsymbol{A}_{\mathrm{y}}\boldsymbol{Z}$, where $\boldsymbol{Z}$ is a cd-dimensional vector of i.i.d. standard normals.

One possibility for this decomposition is the Cholesky decomposition. Another one is the eigen-decomposition used for principal component analysis. We will return to this in Section 6.11.

♣ Donsker functional CLT (as a motivation for why BM is important).

2.14.2 Time change and rescaling

Starting from a standard BM $\boldsymbol{B}$ over the time interval $[0,1]$, one can obtain an arbitrary BM $\boldsymbol{X}$ with drift $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, over a time interval $[0,T]$, simply by rescaling the time linearly and adding a linear trend:

$$\boldsymbol{X}(t) = \boldsymbol{\mu}t + \boldsymbol{A}\sqrt{T}\boldsymbol{B}(t/T) \qquad \text{for } 0 \le t \le T.$$

By applying a nonlinear time change via a non-decreasing function $a : [0,\infty) \to [0,\infty)$ and defining $\boldsymbol{Y}(t) = \boldsymbol{A}(t)\boldsymbol{B}(a(t))$ for some time-dependent matrix $\boldsymbol{A}(t)$, we obtain a Gaussian process whose covariance matrix at time t is $a(t)\boldsymbol{A}(t)\boldsymbol{A}(t)^{\mathsf{t}}$. As a special case, the nonlinearly rescaled process defined by $\boldsymbol{Y}(t) = t\boldsymbol{B}(1/t)$ is also a standard BM. Reverting the time gives another BM: $\{\boldsymbol{Y}(t) = \boldsymbol{X}(T-t) - \boldsymbol{X}(T),\ 0 \le t \le T\}$ is a BM with drift $-\boldsymbol{\mu}$ and the same covariance matrix $\boldsymbol{\Sigma}$ as $\boldsymbol{X}$.

2.14.3 Maximum and first hitting time for a one-dimensional BM

Consider a one-dimensional BM $\{X(t),\ t \ge 0\}$ with drift μ and variance parameter σ^2. The *maximum process* is defined by $M(t) = \max_{0 \le s \le t} X(s)$, for $t \ge 0$. For $\ell > 0$, let $T_\ell = \min\{t \ge 0 : X(t) = \ell\}$ denote the first *hitting time* of level ℓ. Clearly, $\mathbb{P}[T_\ell \le t] = \mathbb{P}[M(t) \ge \ell]$.

Proposition 2.12 *If $\mu > 0$, then $T_\ell \sim$ InvGaussian$(\ell/\mu, \ell^2/\sigma^2)$. When $\mu = 0$, we have*

$$\mathbb{P}[T_\ell \le t] = \mathbb{P}[M(t) \ge \ell] = 2\,\Phi\left(-\ell/(\sigma\sqrt{t})\right)$$

for all $t > 0$, where Φ is the standard normal cdf. If $X(0) = x_0 \ne 0$, it suffices to replace ℓ by $\ell - x_0$.

2.14.4 Brownian bridge

A d-dimensional *standard Brownian bridge* $\boldsymbol{B}^0 = \{\boldsymbol{B}^0(t),\, 0 \le t \le 1\}$ over the time interval $[0,1]$ is a process whose probability law is the same as that of a standard BM $\boldsymbol{B} = \{\boldsymbol{B}(t),\, t \ge 0\}$ conditional on $\{\boldsymbol{B}(0) = \boldsymbol{B}(1) = \boldsymbol{0}\}$. In other words, the process is tightened by boundary conditions at both ends. More generally, for a BM $\boldsymbol{X} = \{\boldsymbol{X}(t),\, t \ge 0\}$ with drift $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, we obtain a Brownian bridge process $\boldsymbol{X}^0 = \{\boldsymbol{X}^0(t),\, t_1 \le t \le t_2\}$ over the time interval $[t_1, t_2]$ if we condition on $\{\boldsymbol{X}(t_1) = \boldsymbol{x}_1,\, \boldsymbol{X}(t_2) = \boldsymbol{x}_2\}$ for arbitrary vectors $\boldsymbol{x}_1, \boldsymbol{x}_2$ in $\mathbb{R}^d$ and $t_2 > t_1 \ge 0$. We then have:

Proposition 2.13 *For $t_1 < t < t_2$, conditional on $\boldsymbol{X}(t_1) = \boldsymbol{x}_1$ and $\boldsymbol{X}(t_2) = \boldsymbol{x}_2$, the vector $\boldsymbol{X}(t)$ is multinormal with mean and covariance matrix*

$$\begin{aligned} \mathbb{E}[\boldsymbol{X}(t) \mid \boldsymbol{X}(t_1) = \boldsymbol{x}_1, \boldsymbol{X}(t_2) = \boldsymbol{x}_2] &= \boldsymbol{x}_1 + \frac{t - t_1}{t_2 - t_1}(\boldsymbol{x}_2 - \boldsymbol{x}_1), \\ \mathrm{Cov}[\boldsymbol{X}(t) \mid \boldsymbol{X}(t_1) = \boldsymbol{x}_1, \boldsymbol{X}(t_2) = \boldsymbol{x}_2] &= \frac{(t - t_1)(t_2 - t)}{t_2 - t_1}\boldsymbol{\Sigma}. \end{aligned}$$

♣ Perhaps add a figure here for $d = 1$.

This property can be used to generate an approximation of the trajectory of a BM over the time interval $[0, t]$ by successive refinements, as follows. For more generality, here we let $\boldsymbol{X}(0)$ take an arbitrary value. First generate $\boldsymbol{X}(t)$, a multinormal with mean $\boldsymbol{X}(0) + t\boldsymbol{\mu}$ and covariance matrix $t\boldsymbol{\Sigma}$. Then generate $\boldsymbol{X}(t/2)$, whose distribution conditional on $(\boldsymbol{X}(0), \boldsymbol{X}(t))$ is multinormal with mean $(\boldsymbol{X}(0) + \boldsymbol{X}(t))/2$ and covariance matrix $(t/4)\boldsymbol{\Sigma}$. Apply this technique recursively to generate $\boldsymbol{X}(t/4)$ conditional on $(\boldsymbol{X}(0), \boldsymbol{X}(t/2))$, then $\boldsymbol{X}(3t/4)$ conditional on $(\boldsymbol{X}(t/2), \boldsymbol{X}(t))$, then $\boldsymbol{X}(t/8)$ conditional on $(\boldsymbol{X}(0), \boldsymbol{X}(t/4))$, and so on, until the trajectory has been determined with the desired precision. The process $\boldsymbol{X}(\cdot)$ is thus generated exactly, but only at the selected points. The trajectory can be interpolated in between those points. Lévy (1925) used exactly this technology, with a linear interpolation between the evaluation points, to prove the existence of BM and study its properties.

This *Brownian bridge sampling* approach can also be used to simulate the process at a fixed set of observation times $0 = t_0 < t_1 < \cdots < t_c$, which are not necessarily equally spaced. To simplify the notation, we assume here that c is a power of 2, but this can be generalized. First generate $\boldsymbol{X}(t_c)$, a multinormal with mean $t_c\boldsymbol{\mu}$ and covariance $t_c\boldsymbol{\Sigma}$. Then generate $\boldsymbol{X}(t_{c/2})$, whose distribution conditional on $(\boldsymbol{X}(0), \boldsymbol{X}(t))$ is multinormal with mean $\boldsymbol{X}(0) + (\boldsymbol{X}(t_c) - \boldsymbol{X}(0))t_{c/2}/t_c$ and covariance $(t_{c/2}(t_c - t_{c/2})/t_c)\boldsymbol{\Sigma}$. Apply this technique recursively to generate $\boldsymbol{X}(t_{c/4})$ conditional on $(\boldsymbol{X}(0), \boldsymbol{X}(t_{c/2}))$, then $\boldsymbol{X}(t_{3c/4})$ conditional on $(\boldsymbol{X}(t_{c/2}), \boldsymbol{X}(t_c))$, and so on. This sampling scheme was suggested by Moskowitz and Caflisch (1996) (for $d = 1$) as a way to improve the effectiveness of quasi-Monte Carlo integration for a function of these observations by reducing the effective dimension of this function (making the function depend mostly on just the first few coordinates). It requires only the decomposition of $\boldsymbol{\Sigma}$, compared with the general method of Section 2.14.1, which requires a decomposition of the larger matrix $\boldsymbol{\Sigma}_{\mathrm{y}}$ instead.

2.14.5 Approximation via Karhunen-Loève expansion

The *Karhunen-Loève expansion* provides a representation of a stochastic process by a series (an infinite linear combination) of orthogonal sinusoidal functions with random coefficients. In the case of a one-dimensional standard BM over the time interval $[0, T]$, the expansion can be written as

$$B(t) = \sum_{k=1}^{\infty} Z_k \frac{\sqrt{2T}\sin((k-1/2)\pi t/T)}{(k-1/2)\pi} \tag{2.52}$$

for $0 \le t \le T$, where $Z_1, Z_2, Z_3, \ldots$ are i.i.d. $\mathsf{N}(0,1)$. By truncating the sum at a finite number of terms, say c terms, one can obtain a fairly good approximation of the process trajectory over the entire time interval $[0, T]$ by generating $Z_1, \ldots, Z_c$. This contrasts with the generation of a skeleton, where $Z_1, \ldots, Z_c$ are used to generate the process from its exact distribution, but only at a fixed set of observation times $t_1, \ldots, t_c$. When we really need a good approximation of the entire trajectory in continuous-time, the expansion (2.52) can be more accurate than using a skeleton.

♣ Give a numerical illustration, including a comparison with using a skeleton together with linear interpolation. Compare the trajectories and compare the results when this is used to estimate the value of a path-dependent option.

2.14.6 Geometric BM (GBM)

A d-dimensional process $\{\boldsymbol{S}(t) = (S_1(t), \ldots, S_d(t)),\ t \ge 0\}$ is a *multivariate GBM process* with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, denoted GBM$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, if the process $\boldsymbol{X} = \{\boldsymbol{X}(t) = \ln[\boldsymbol{S}(t)/\boldsymbol{S}(0)],\ t \ge 0\}$ is a multivariate BM process with drift $\boldsymbol{\mu} - (\sigma_{1,1}/2, \ldots, \sigma_{d,d}/2)^{\mathsf{t}}$ and covariance matrix $\boldsymbol{\Sigma}$. This means that we can write

$$S_i(t) = S_i(0)\exp[X_i(t)] \tag{2.53}$$

for each coordinate $i = 1, \ldots, d$, so to generate the GBM process, it suffices to generate the corresponding BM process and take the exponential of it coordinatewise.

In the one-dimensional case, the GBM(μ, σ^2) process obeys

$$S(t) = S(0)\exp[X(t)] = S(0)\exp[(\mu - \sigma^2/2)t + \sigma B(t)], \tag{2.54}$$

where $\mu = \mu_1$ and $\sigma^2 = \sigma_{1,1}$ are the *drift* and *volatility* parameters. By applying the formula of Itô in stochastic calculus (Karatzas and Shreve 1998), we find that a GBM(μ, σ^2) process can also be specified via the stochastic differential equation

$$\frac{\mathrm{d}S(t)}{S(t)} = \mu \mathrm{d}t + \sigma \mathrm{d}B(t).$$

For every $t_0 \ge 0$, the ratio $S(t_0 + t)/S(t_0)$ has the lognormal distribution with parameters $((\mu - \sigma^2/2)t, \sigma^2 t)$. This means that the *relative variation* (or percentage of change) of the process over a given time interval does not depend on the initial value. For an ordinary BM, this holds for the *absolute variation* of the process. Whereas the ordinary BM can take

negative values, the GBM cannot. These properties make the GBM more appropriate than the BM in various contexts, e.g., in financial modeling. Example 1.11 gave an illustration of this. [13] Several other examples can be found in Hull (2000) and Glasserman (2004). However, the GBM fails to capture some important characteristics of processes encountered in finance and other areas, and several generalizations have been introduced for this reason, as we will see later.

2.14.7 Gaussian processes

BM can be generalized in many directions. One way is to remove the assumption of stationarity of the increments by allowing the drift vector parameter $\boldsymbol{\mu}$ to depend on the current time t and the covariance matrix $\text{Cov}[\boldsymbol{X}(s), \boldsymbol{X}(t)]$ to depend on s and t. Another way is to allow $\boldsymbol{X}(t)$ to be infinite dimensional. We may also replace the time index $t \geq 0$ by an arbitrary index $t \in \mathcal{I}$ for some general index set $\mathcal{I}$. For example, t may indicate a position in space, or a combination of time and space. This generalization yields a *Gaussian process* $\{\boldsymbol{X}(t),\, t \in \mathcal{I}\}$, which can be specified (uniquely) by specifying its mean function $\{\boldsymbol{\mu}(t),\, t \in \mathcal{I}\}$ and its covariance (matrix) function, $\boldsymbol{\Sigma}(s,t) = \text{Cov}[\boldsymbol{X}(s), \boldsymbol{X}(t)]$ for all $s, t \in \mathcal{I}$. To be admissible, this covariance function must be nonnegative definite. In one dimension, this means that for any integer $c > 0$ and any $t_1, \ldots, t_c \in \mathcal{I}$, the $c \times c$ matrix whose elements (i, j) is $\text{Cov}[X(t_i), X(t_j)]$ must be nonnegative definite.

For the remainder of this subsection, to keep the notation simpler, we assume that $d = 1$ (so we have a one-dimensional process). In this case, the covariance function is nonnegative definite if and only if for any integer $c > 0$ and any $t_1, \ldots, t_c \in \mathcal{I}$, the $c \times c$ matrix whose elements (i, j) is $\text{Cov}[X(t_i), X(t_j)]$ is nonnegative definite. When the mean and covariance functions are known explicitly, the multinormal vector $\boldsymbol{X}_c = (X(t_1), \ldots, X(t_c))^{\mathsf{t}}$ can be generated in a standard manner (as in Section 2.10.3) by decomposing $\boldsymbol{\Sigma}_c$ as $\boldsymbol{\Sigma}_c = \boldsymbol{A}_c \boldsymbol{A}_c^{\mathsf{t}}$ in any way, generating a vector $\boldsymbol{Z}_c$ of c independent standard normals, and putting $\boldsymbol{X}_c = (\mu(t_1), \ldots, \mu(t_c))^{\mathsf{t}} + \boldsymbol{A}_c \boldsymbol{Z}_c$.

Note that if we add new observation points $t_{c+1}, \ldots, t_d$ where $d > c$, and if we use the Cholesky decomposition for $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\Sigma}_d$, then $\boldsymbol{A}_c$ turns out to be a submatrix (first c rows and c columns) of $\boldsymbol{A}_d$, and it suffices to update the decomposition by computing the new $d - c$ rows and $d - c$ columns of $\boldsymbol{A}_d$. Using this, it is possible to add one observation point at a time until we decide to stop, and update the matrix $\boldsymbol{A}_c$ only when needed.

In any case, the decomposition of $\boldsymbol{\Sigma}_c$ generally requires $\Theta(c^3)$ operations, and we also need $\Theta(c^2)$ operations to compute $\boldsymbol{A}_c \boldsymbol{Z}_c$ for each copy of $\boldsymbol{X}_c$ that we want to generate. When c is very large, this may become impractical. Fortunately, in many practical situations, there are faster methods that do not require the explicit decomposition of $\boldsymbol{\Sigma}_c$. This is the case, for example, when we generate an ordinary one-dimensional BM with the random walk approach (Section 2.14.1): there is no explicit decomposition and we only need $\Theta(c)$ time to generate each trajectory. Other simulation methods for Gaussian processes are discussed in Asmussen and Glynn (2007).

[13] From Pierre: Add some discussion of this.

[14] A Gaussian process is *stationary* if $\mu(t)$ and $\text{Cov}[X(t), X(t+s)]$ do not depend on t, for any s. We then denote $r_X(s) = \text{Cov}[X(t), X(t+s)]$. If the function r_X is $2k$ times differentiable, then the trajectory of X is k times differentiable everywhere with probability 1. In one dimension, if

$$\lim_{\delta \to 0} \frac{-\log[r_X(\delta) - r_X(0)]}{\delta} < \infty,$$

[15] then the trajectory of X is continuous with probability 1. The OU process (Section 2.15.2) has this property, for example.

Example 2.28 A one-dimensional BM process is a non-stationary Gaussian process and has $\text{Cov}[X(t), X(t+s)] = t\sigma^2$ for $t < s$. The OU process (Section 2.15.2) and the ARIMA process (Section 2.17.2) are examples of stationary Gaussian processes. □

2.14.8 Fractional BM

♣ To be done.

2.15 Stochastic Differential Equations Driven by BM

A *stochastic differential equation (SDE)* is a differential equation whose definition involves a stochastic process (usually a BM, but not always). The solution of the SDE is also a stochastic process. The SDEs discussed here are driven by a BM. They are by far the most widely used. We first give a definition and provide (approximate) numerical simulation methods for the general case. Then we discuss important special cases where a discrete skeleton can be simulated exactly.

2.15.1 General formulation

We consider a stochastic process $\boldsymbol{X} = \{\boldsymbol{X}(t),\, t \geq 0\}$ in $\mathbb{R}^d$, whose trajectory obeys the SDE

$$\mathrm{d}\boldsymbol{X}(t) = \boldsymbol{\mu}(\boldsymbol{X}(t), t)\mathrm{d}t + \boldsymbol{A}(\boldsymbol{X}(t), t)\mathrm{d}\boldsymbol{B}(t), \tag{2.55}$$

with $\boldsymbol{X}(0) = \boldsymbol{x}_0 \in \mathbb{R}^d$, where $\boldsymbol{B}$ is a standard BM in q dimensions (we may have $q \neq d$), $\boldsymbol{\mu}(\boldsymbol{X}(t), t)$ is a d-dimensional vector which represents the *drift*, $\boldsymbol{A}(\boldsymbol{X}(t), t)$ is a $d \times q$ matrix, and both $\boldsymbol{\mu}$ and $\boldsymbol{A}$ can be functions of the current process state $\boldsymbol{X}(t)$ and current time t. When these two functions do not depend on t, the process is *time-homogeneous*. The term $\mathrm{d}\boldsymbol{X}(t)$ can be interpreted as the (infinitesimal) change in the process state $\boldsymbol{X}(t)$ during the infinitesimal time interval $(t, t+\mathrm{d}t]$. The covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{X}(t), t) = \boldsymbol{A}(\boldsymbol{X}(t), t)\boldsymbol{A}(\boldsymbol{X}(t), t)^{\mathsf{t}}$ is also called the *diffusion coefficient*. Technical conditions under which (2.55) has a solution are given in Asmussen and Glynn (2007) and Kloeden and Platen (1992), for example. Then, an equivalent way of defining this process is via the *Itô integral equation*

[14] From Pierre: Not *time-homogeneous* instead?
[15] From Pierre: Check this.

$$\boldsymbol{X}(t) = \boldsymbol{X}(0) + \int_0^t \boldsymbol{\mu}(\boldsymbol{X}(s), s)\mathrm{d}s + \int_0^t \boldsymbol{A}(\boldsymbol{X}(s), s)\mathrm{d}\boldsymbol{B}(s). \tag{2.56}$$

Example 2.29 If both $\boldsymbol{\mu}$ and $\boldsymbol{A}$ are constant, we recover an ordinary BM:

$$\mathrm{d}\boldsymbol{X}(t) = \boldsymbol{\mu}\,\mathrm{d}t + \boldsymbol{A}\,\mathrm{d}\boldsymbol{B}(t) \quad \text{and} \quad \boldsymbol{X}(t) = \boldsymbol{X}(0) + \boldsymbol{\mu}t + \boldsymbol{A}\,\boldsymbol{B}(t).$$

The GBM in one dimension is obtained via

$$\mathrm{d}X(t) = \mu X(t)\mathrm{d}t + AX(t)\mathrm{d}B(t), \tag{2.57}$$

whose solution is

$$X(t) = X(0)\,\exp\left[(\mu - \sigma^2/2)t + \sigma B(t)\right]. \tag{2.58}$$

□

In some special cases, the exact distribution of $\boldsymbol{X}(t+s)$ conditional on $\boldsymbol{X}(t)$ is known and it is not difficult to generate random vectors from this distribution. Then, one can simulate a discrete-skeleton of the process exactly. This happens in particular for the BM, GBM, and the OU and CIR processes to be defined later. But in other situations, this is not possible and one must discretize the time in small steps to generate approximate sample paths of the SDE.

The simplest and most popular approximation technique is the *Euler method*. It discretizes the time in small steps of length h and just uses Eq. (2.55) iteratively with $\mathrm{d}t = h$. This gives the approximate skeleton $\widetilde{\boldsymbol{X}}$ defined recursively by $\widetilde{\boldsymbol{X}}(0) = \boldsymbol{X}(0)$

$$\widetilde{\boldsymbol{X}}((j+1)h) = \widetilde{\boldsymbol{X}}(jh) + \boldsymbol{\mu}(\widetilde{\boldsymbol{X}}(jh), jh)\,h + \boldsymbol{A}(\widetilde{\boldsymbol{X}}(jh), jh)\,\sqrt{h}\,\boldsymbol{Z}_j, \tag{2.59}$$

for $j = 1, 2, \ldots$, where the $\boldsymbol{Z}_j$ are i.i.d. $\mathsf{N}(\boldsymbol{0}, \boldsymbol{I})$. This is by far the most widely used method.

The *Milstein method* makes an additional correction to (2.59) to account for the fact that $\boldsymbol{A}(\boldsymbol{X}(t), t)$ varies and is not always equal to $\boldsymbol{A}(\boldsymbol{X}(jh), jh)$ for $t \in [jh, jh + h]$ (this is also true for $\boldsymbol{\mu}(\boldsymbol{X}(t), t)$ but the impact is smaller). The method gets complicated and difficult to implement in more than one dimension, so we only give the one-dimensional version. The SDE (2.55) can then be written as

$$\mathrm{d}X(t) = \mu(X(t), t)\,\mathrm{d}t + \sigma(X(t), t)\,\mathrm{d}B(t), \tag{2.60}$$

where σ is the single element of $\boldsymbol{A}$. The correction terms turns out to be $\sigma_x(X(jh), jh)\,\sigma(X(jh), jh)\,(Z_j^2 - 1)h/2$, where $\sigma_x(x, t) = \partial\sigma(x, t)/\partial x$. The approximation then becomes

$$\begin{aligned} \widetilde{X}((j+1)h) \;=\; & \widetilde{X}(jh) + \boldsymbol{\mu}(\widetilde{X}(jh), jh)\,h + \sigma(\widetilde{X}(jh), jh)\,\sqrt{h}\,Z_j \\ & + \sigma_x(\widetilde{X}(jh), jh)\,\sigma(\widetilde{X}(jh), jh)\,(Z_j^2 - 1)h/2. \end{aligned} \tag{2.61}$$

The gain in accuracy depends on how we measure the accuracy of the trajectory. In a majority of applications, there is not much gain. See Asmussen and Glynn (2007), Section X.4, for a discussion.

♣ Exact method of Beskos and Roberts: In Chapter 4.

Next, we examine two important SDE models that admit explicit solutions, so their skeleton can be simulated exactly.

2.15.2 The Ornstein-Uhlenbeck mean-reverting process

A one-dimensional *Ornstein-Uhlenbeck (OU) process* $\{X(t), t \geq 0\}$ can be defined via the stochastic differential equation

$$\mathrm{d}X(t) = \alpha(b - X(t))\,\mathrm{d}t + \sigma\,\mathrm{d}B(t), \tag{2.62}$$

where B is a standard BM, and α, b, and σ are positive constants, called the *mean reversion rate*, the mean, and the volatility, respectively. The process is *mean-reverting* in the sense that it always drifts toward its general mean b, so it tends to return constantly around that level. This process is also known as the Vasicek model for interest rates.

For $0 \leq s < t$, conditional on $X(s) = x$, $X(t)$ is normally distributed with mean $e^{-\alpha(t-s)}x + (1 - e^{-\alpha(t-s)})b$ and variance $(1 - e^{-2\alpha(t-s)})\sigma^2/(2\alpha)$. This provides an easy way to simulate the process at a finite set of observation times, by simulating the increments successively. By taking $t \to \infty$, we see that the limiting distribution of $X(t)$ is normal with mean b and variance $\sigma^2/(2\alpha)$. We also have that for $s < t$, $\mathrm{Cov}(X(s), X(t)) = [e^{-\alpha(t-s)} - e^{-\alpha(t+s)}]\sigma^2/(2\alpha)$.

A *d-dimensional OU process* with mean vector $\boldsymbol{b} = (b_1, \ldots, b_d)^{\mathsf{t}}$, mean reversion rates $\alpha_1, \ldots, \alpha_d$, and covariance matrix $\boldsymbol{\Sigma}$, can be defined via

$$\mathrm{d}\boldsymbol{X}(t) = \boldsymbol{R}(\boldsymbol{b} - \boldsymbol{X}(t))\,\mathrm{d}t + \boldsymbol{A}\,\mathrm{d}\boldsymbol{B}(t), \tag{2.63}$$

where $\boldsymbol{R}$ is a diagonal matrix with diagonal elements $\alpha_1, \ldots, \alpha_d$, $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$, and $\boldsymbol{B}$ is a d-dimensional standard BM. For $0 < s < t$, conditional on $\boldsymbol{X}(s) = \boldsymbol{x} = (x_1, \ldots, x_j)^{\mathsf{t}}$, $\boldsymbol{X}(t)$ is normally distributed with mean $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_d)^{\mathsf{t}}$ and covariance matrix $\boldsymbol{D}\boldsymbol{\Sigma}$, where $\mu_j = e^{-\alpha_j(t-s)}x_j + (1 - e^{-\alpha_j(t-s)})b_j$ and $\boldsymbol{D}$ is a diagonal matrix whose jth diagonal entry is $(1 - e^{-2\alpha_j(t-s)})/(2\alpha_j)$.

2.15.3 Square root process (CIR model)

A one-dimensional *square root diffusion process* of the form

$$dX(t) = \alpha(b - X(t))\,\mathrm{d}t + \sigma\sqrt{X(t)}\,\mathrm{d}B(t), \tag{2.64}$$

where B is a standard BM, and α, b, and σ are positive constants, was proposed by Cox, Ingersoll, and Ross (1985) as a model of the (short) interest rate. It is known as the *CIR model* for interest rates. Note that the only difference with the OU model is that σ is scaled by the factor $\sqrt{X(t)}$. This factor goes to 0 when $X(t)$ approaches 0, and this prevents the process from taking negative values.

It is known that for $0 \leq s < t$, conditional on $X(s) = x$, $X(t)$ has the same distribution as

$$\frac{\sigma^2(1 - e^{-\alpha(t-s)})}{4\alpha} Y,$$

where Y is a noncentral chi-square random variable with $k = 4b\alpha/\sigma^2$ degrees of freedom and non-centrality parameter

$$\lambda = \frac{4\alpha e^{-\alpha(t-s)}x}{\sigma^2(1 - e^{-\alpha(t-s)})}.$$

This provides a way of simulating the process at a finite set of observation times.

2.16 Lévy Processes

2.16.1 Definition and decomposition

The class of Lévy processes contains several of the most widely used types of stochastic processes in applied probability, including the BM and the stationary Poisson process. Other types of Lévy processes recently became popular in financial modeling, for example, because they provide a better fit to financial data than the more traditional (older) models (Schoutens 2003, Cont and Tankov 2004). For simplicity, we do not use boldface notation here, as if the processes were one-dimensional, but much of what we say generalizes easily to multivariate processes.

A *Lévy process* is a continuous-time stochastic process $Y = \{Y(t),\, t \geq 0\}$, with $Y(0) = 0$, and with stationary and independent increments, i.e., for which the increments $X_j = Y(t_{2j}) - Y(t_{2j-1})$ over disjoint time intervals $(t_{2j-1}, t_{2j}]$, $j = 1, 2, \ldots$, are independent random variables and the distribution of X_j depends only on the length of the interval, $t_{2j} - t_{2j-1}$. That is, X_j always has the same distribution as $Y(t_{2j} - t_{2j-1})$. These types of processes have been studied quite extensively (e.g., Bertoin 1996, Sato 1999). The stationarity and independence of the increments imply that they are *infinitely divisible*, which means that for every fixed t, $Y(t)$ can be written as a sum of n i.i.d. random variables for any positive integer n (arbitrarily large). Conversely, every process having this property is a Lévy process.

The BM and the stationary Poisson process are two prominent examples of Lévy processes, where the increments have the normal and the Poisson distribution, respectively. The former has continuous trajectories and the latter is a counting process. It turns out that any Lévy process can be written as the sum of a BM and a jump process with random jump sizes (positive or negative). When the expected number of jumps per unit of time is finite, say λ, then the jump process is a compound Poisson process of rate λ, and we can write

$$Y(t) = \mu t + \sigma B(t) + \sum_{j=1}^{N(t)} D_j \qquad \text{for } t \geq 0,$$

where B is a standard BM, N is a Poisson process of rate λ, and the D_j are i.i.d. random variables, independent of B and N. If we know how to generate D_j, then this process is easy to simulate.

For many Lévy processes, however, the jump process component has an infinite jump rate. The simulation of such processes (which is generally more complicated) is discussed in Asmussen and Glynn (2007). In general, the jump process component can be defined by its Lévy measure ν, where $\nu(B)$ represents the intensity of jumps whose size is in B, for any measurable set $B \subset \mathbb{R}$, at any given time. When ν has a density η with respect to the Lebesgue measure, then $\eta(x)$ is the jump rate for jumps of size x, and $\nu(B) = \int_B \eta(x)\mathrm{d}x$. Often, for a given (small) $\epsilon > 0$, there is a limited number of jumps of size larger than ϵ in any given time interval, but an infinite number of very small jumps, of size smaller than ϵ. To simulate the process in this situation, a common approach is to generate only the large jumps and to approximate the effect of the very small jumps in a different way.

Things are also easy if we know how to generate the increment $Y(t)$ for any given t. Examples of Lévy processes with this property, in addition to the BM and Poisson processes, are given in Section 2.16.4. In that case, to generate the trajectory at the discrete

observation times $0 = t_0 < t_1 < \cdots < t_c$, we can simply generate the independent increments $Y(t_j) - Y(t_{j-1})$ sequentially, for $j = 1, \ldots, c$, and add them. This is the *random walk* (or *sequential*) approach. For a stationary BM, for example, these increments are independent normal random variables whose mean and variance are proportional to $t_j - t_{j-1}$ (see Section 2.14.1).

2.16.2 Lévy bridge generation approach

For certain Lévy processes, we also know explicitly the distribution of $Y(t)$ *conditional* on $\{Y(t_1) = y_1, Y(t_2) = y_2\}$ for arbitrary values of y_1, y_2, and $t_1 < t < t_2$. Then, it is also possible to generate the trajectory of Y by successive refinements, via the following *Lévy bridge* approach, which generalizes the Brownian bridge approach discussed in Section 2.14.4. To keep the notation simple, we assume here that c is a power of 2. We first generate $Y(t_c)$, then we generate $Y(t_{c/2})$ from its conditional distribution given $(Y(t_0), Y(t_c))$, then we apply the same technique recursively to generate $Y(t_{c/4})$ conditional on $(Y(t_0), Y(t_{c/2}))$, then $Y(t_{3c/4})$ conditional on $(Y(t_{c/2}), Y(t_c))$, then $Y(t_{c/8})$ conditional on $(Y(t_0), Y(t_{c/4}))$, and so on, until all c values have been determined. When c is not a power of 2, we need to round up or down the indices to integers, for example replace $c/2$ by $\lfloor c/2 \rfloor$, $c/4$ by $\lfloor c/4 \rfloor$, etc., so the implementation gets a little more complicated, but the method still works. This technique is quite effective to approximate the trajectory of Y up to a certain accuracy, and when the required value of c is not necessarily known in advance. If the accuracy is deemed insufficient after having generated a skeleton with c observation points, it is easy to double the value of c and just continue the conditional generation. This approach also provides a powerful tool to improve the effectiveness of quasi-Monte Carlo methods by reducing the effective dimension of the problem, as we will see in Chapter 6.

2.16.3 Deterministic and random time changes

A convenient way of increasing the flexibility of a given class of Lévy processes is to apply a nonlinear rescaling of the time index, as we saw in Section 2.13.2 for the Poisson process. Such a time change can be specified by a (deterministic) nondecreasing function $a : [0, \infty) \to [0, \infty)$, where $a(t)$ can be interpreted as the reading at time t on a clock with time-varying speed. If the function a is differentiable, then the derivative $a'(t)$ is the clock speed at time t. The function a may also be discontinuous, with positive jumps. Applying this time change to a given process $X = \{X(t), t \geq 0\}$ yields the new process $Y = \{Y(t), t \geq 0\}$ where $Y(t) = X(a(t))$. The *clock function* a acts like a change of variable.

A more powerful device is to replace the deterministic clock function a by a stochastic process $T = \{T(t), t \geq 0\}$ with nondecreasing trajectories, thus making the original process $X = \{X(t), t \geq 0\}$ doubly stochastic. We replace $X(t)$ by $Y(t) = X(T(t))$ for each t, to obtain the new process $Y = \{Y(t), t \geq 0\}$. The process T defines a *random time change* and is called a *subordinator*. This can be used for any process X with index $t \in \mathbb{R}$. If both X and T are Lévy processes, then so is Y.

If X is a one-dimensional BM process, the random time change is equivalent to replacing the constant drift and volatility parameters μ and σ of the BM by stochastic (time-varying)

drift and volatility processes $\{(\mu(t), \sigma(t)),\ t \geq 0\}$. In some cases, we may also keep a deterministic drift component (see Section 2.16.8).

Two notable examples of Lévy processes that can act as subordinators are the gamma process and the inverse Gaussian process. Their use as subordinators for the BM yields the variance gamma and the normal inverse Gaussian processes, defined in Sections 2.16.6 and 2.16.8. BM processes with such a random time change do provide a much better fit to various types of financial data (such as the log prices of stocks and commodities, etc.) than standard BM processes.

2.16.4 Other Lévy Processes

In what follows, we mention other types of Lévy processes $Y = \{Y(t),\ t \geq 0\}$ for which we know how to generate the increment $Y(t)$, so we can generate a skeleton of these processes via the random walk approach. Multivariate versions of these Lévy processes, which evolve in the d-dimensional real space, are discussed in Asmussen and Glynn (2007).

2.16.5 The gamma process

A *gamma process* $\{G(t),\ t \geq 0\}$ with drift parameter $\mu = \alpha/\lambda$ and volatility (or variance) parameter $\nu = \alpha/\lambda^2$ is a Lévy process whose increment over a time interval of length t has a gamma distribution with parameters $(t\alpha, \lambda) = (t\mu^2/\nu, \mu/\nu)$ (i.e., with mean $t\mu$ and variance $t\nu$). This process has nondecreasing trajectories, because gamma random variables cannot take negative values. It is thus admissible as a subordinator.

We can generate a skeleton of this process observed at times $0 = t_0 < t_1 < t_2 < \cdots$ via the random walk approach by generating the successive independent increments $G(t_j) - G(t_{j-1})$, which are gamma with parameters $((t_j - t_{j-1})\alpha, \lambda)$. By default, we take $G(0) = 0$.

Moreover, for any fixed values $t_1 < t < t_2$, the distribution of $(G(t) - G(t_1))/(G(t_2) - G(t_1))$ conditional on $(G(t_1), G(t_2))$ is $\mathsf{Beta}((t-t_1)\alpha,\ (t_2-t)\alpha)$. Thus, a skeleton of the gamma process observed at times $0 < t_1 < \cdots < t_c$ can be also be generated as in Section 2.16.2 by a *gamma bridge* approach, by first generating $G(t_c)$ from the $\mathsf{Gamma}(t_c\alpha, \lambda)$ distribution, then generating $G(t_{c/2})$ conditional on $G(t_c)$ by generating $B \sim \mathsf{Beta}(t_{c/2}\alpha,\ (t_c - t_{c/2})\alpha)$ and putting $G(t_{c/2}) = G(t_c)\, B$, then generating $G(t_{c/4})$ conditional on $G(t_{c/2})$ by generating $B \sim \mathsf{Beta}(t_{c/4}\alpha,\ (t_{c/2} - t_{c/4})\alpha)$ and putting $G(t_{c/4}) = G(t_{c/2})\, B$, then generating $G(t_{3c/4})$ conditional on $(G(t_{c/2}), G(t_{t_c}))$ by generating $B \sim \mathsf{Beta}((t_{3c/4} - t_{c/2})\alpha,\ (t_c - t_{3c/4})\alpha)$ and putting $G(t_{3c/4}) = G(t_{c/2}) + (G(t_c) - G(t_{c/2}))\, B$, and so on. Here we have assumed for simplicity that c is a multiple of 4.

2.16.6 The variance-gamma process

A *variance-gamma (VG) process* $Y = \{Y(t),\ t \geq 0\}$ is a BM subordinated to a random time change that obeys a gamma process. It can be defined as follows (Madan and Milne 1991, Madan, Carr, and Chang 1998, Avramidis, L'Ecuyer, and Tremblay 2003, Avramidis and L'Ecuyer 2006):

$$Y(t) = X(G(t)),$$

where X is a BM with drift parameter $\mu = \theta$ and variance parameter σ^2, G is a gamma process with drift and variance parameters 1 and ν, and X and G are independent. Supported by empirical evidence, Madan, Carr, and Chang (1998) proposed the VG process as a replacement for the BM in the GBM model, to improve its realism for asset price modeling.

To simulate the process at observation times $0 = t_0 < t_1 < \cdots < t_c$, we can generate G and X by either a random walk (sequential) or Lévy bridge approach, as explained earlier. In each case, we can either first generate the entire skeleton of the process G at these observation times, which gives $\tau_1 = G(t_1), \ldots, \tau_j = G(t_j)$, and then generate the skeleton of X at its observation times τ_j, or generate the values in alternation by always obtaining $X(\tau_j)$ immediately after $\tau_j = G(t_j)$. In all cases, we have $G(0) = X(0) = \tau_0 = t_0 = 0$.

For the random walk approach combined with alternating sampling, for instance, for $j = 1, 2, \ldots, c$, we first generate the increment $G(t_j) - G(t_{j-1})$ from the gamma distribution with mean $t_j - t_{j-1}$ and variance $(t_j - t_{j-1})\nu$, then generate $X(\tau_j) - X(\tau_{j-1})$ from the normal distribution with mean $(\tau_j - \tau_{j-1})\mu$ and variance $(\tau_j - \tau_{j-1})\sigma^2$. This requires the generation of c gamma variates and c normal variates, all independent. From these increments, it is trivial to compute the $G(t_j)$ and $Y(t_j)$.

In the bridge and alternating approach, we generate $\tau_c = G(t_c)$, $X(\tau_c)$, $\tau_{c/2} = G(t_{c/2})$, $X(\tau_{c/2})$, $\tau_{c/4} = G(t_{c/4})$, $X(\tau_{c/4})$, $\tau_{3c/4} = G(t_{3c/4})$, $X(\tau_{3c/4})$, $\ldots$, in that order. When generating τ_j for $0 < j < c$, we first look for the largest integer $a < j$ and the smallest integer $b > j$ such that the values at t_a and t_b have already been generated. Then we sample $\tau_j = G(t_j)$ from its beta distribution conditional on $(G(t_a), G(t_b))$, and $X(\tau_j)$ from its (normal) distribution conditional on $(\tau_j, X(\tau_a), X(\tau_b))$. This can be done because for any given values $t_a < t_j < t_b$ and $\tau_a < \tau_j < \tau_b$, we know how to sample from these conditional distributions. This method requires the generation of one gamma variate, $c-1$ beta variates, and c normal variates.

Another set of sampling methods exploit the property that Y can be written as the difference of two gamma processes G^+ and G^-:

$$Y(t) = G^+(t) - G^-(t),$$

where G^+ and G^- are independent gamma processes with parameters (μ^+, ν^+) and (μ^-, ν^-), respectively, with

$$\begin{aligned}
\mu^+ &= (\sqrt{\theta^2 + 2\sigma^2/\nu} + \theta)/2, \\
\mu^- &= (\sqrt{\theta^2 + 2\sigma^2/\nu} - \theta)/2, \\
\nu^+ &= (\mu^+)^2\nu, \\
\nu^- &= (\mu^-)^2\nu,
\end{aligned}$$

and $G^+(0) = G^-(0) = 0$ (Madan, Carr, and Chang 1998, Avramidis and L'Ecuyer 2006). The VG process can then be simulated by simulating G^+ and G^-, either one after the other (first simulate G^+ at all c observation times and then do the same for G^-), or in an alternate way (generate G^+ and then G^- at a given observation time, then do the same at another observation time, and so on). In both cases, the c observation times can be visited either in increasing order (the random walk method) or in a gamma bridge fashion. Avramidis and L'Ecuyer (2006) proposed to combine the alternate scheme with a gamma bridge approach and called it *difference-of-gammas bridge sampling* (DGBS). This scheme

requires two gamma variates and $2c-2$ beta variates. It combines effectively well with RQMC by reducing the effective dimension. It also has the important advantage of providing bounds on the entire (continuous-time) trajectory of the VG process after any given step of the bridge approach. Indeed, since a gamma process is nondecreasing, knowing its value at time t_c and possibly at a few intermediate points provides a lower bound and an upper bound on its trajectory over the interval $[0, t_c]$. These bounds provide bounds on the difference $Y(t) = G^+(t) - G^-(t)$ at any $t \le t_c$.

A *geometric VG process* $S = \{S(t), t \ge 0\}$ is obtained by taking

$$S(t) = S(0) \exp\left[rt + X(G(t)) + \omega t\right],$$

where X is a BM with drift and variance parameters θ and σ, G is a gamma process with mean and variance parameters 1 and ν, X and G are independent, and $\omega = \ln(1 - \theta\nu - \sigma^2\nu/2)/\nu$. In other words, the process Y defined by $Y(t) = X(G(t))$ is a VG process. Geometric VG processes are used model asset prices in finance (Avramidis, L'Ecuyer, and Tremblay 2003, Avramidis and L'Ecuyer 2006, Madan, Carr, and Chang 1998).

2.16.7 The inverse Gaussian process

An *inverse Gaussian (IG) process* $\{I(t), t \ge 0\}$ with parameters (δ, γ), for $\delta > 0$ and $\gamma > 0$, is a Lévy process whose increment over a time interval of length t obeys $I(t) \sim \mathsf{InvGaussian}(t\delta/\gamma, t^2\delta^2)$ (Barndorff-Nielsen, Mikosch, and Resnick 2013, Rydberg 1997). This $I(t)$ has the same distribution as the first hitting time of level t by a BM with drift parameter $\mu = \gamma/\delta$ and variance parameter $\sigma^2 = 1/\delta^2$. It has mean $t\delta/\gamma$ and variance $t\delta/\gamma^3$. The increments are never negative, so this process can be used as a subordinator. We can generate a skeleton of the process observed at times $0 = t_0 < t_1 < \cdots < t_c$ via the random walk approach by putting $I(0) = 0$ and generating the successive independent increments $I(t_j) - I(t_{j-1})$, which are $\mathsf{InvGaussian}((t_j - t_{j-1})\delta/\gamma, (t_j - t_{j-1})^2\delta^2)$.

An *inverse Gaussian bridge* approach can also be used, since for any fixed values $t_1 < t < t_2$, it is known how to sample the ratio $S = (I(t) - I(t_1))/(I(t_2) - I(t))$ from its distribution conditional on $(I(t_1), I(t_2))$ (Ribeiro and Webber 2003). This ratio S has density

$$f_S(x) = (\lambda/2\pi)^{1/2} \frac{1}{1+r} x^{-3/2}(1+x) \exp\left(-\frac{\lambda}{2}\frac{(x-r)^2}{xr^2}\right)$$

where $r = (t_2 - t)/(t - t_1)$ and $\lambda = \delta^2(t_2 - t)^2/(I(t_2) - I(t_1))$. Its density does not depend on γ. One can generate $I(t)$ conditional on $(I(t_1), I(t_2))$ as follows (Ribeiro and Webber 2003):

generate $Q \sim \chi^2(1)$ (one can take $Q = Z^2$ where $Z \sim \mathsf{N}(0,1)$);
let $S_1 = r + \frac{r}{2\lambda}\left(rQ - (4r\lambda Q + r^2Q^2)^{1/2}\right)$;
let $S_2 = r^2/S_1$;
let $P = \frac{r(1+S_1)}{(1+r)(r+S_1)}$;
generate $U \sim U(0,1)$;
if $U < P$ let $S = S_1$, else let $S = S_2$;
return $I(t) = I(t_1) + (I(t_2) - I(t_1))/(1+S)$.

This can be exploited to generate the inverse Gaussian process by a bridge approach as described in Section 2.16.2: generate $I(t_c)$ first, then $Y(t_{c/2})$ conditional on $(I(t_0), I(t_c))$, then $I(t_{c/4})$ conditional on $(I(t_0), I(t_{c/2}))$, etc.

See Wang and Xu (2010), Ye and Chen (2014) for parameter estimation and application to degradation data in reliability settings.

2.16.8 The normal inverse Gaussian process

A *normal inverse Gaussian (NIG) process* $\{Y(t), t \geq 0\}$ is a BM with a random time change that obeys an inverse Gaussian process, plus a deterministic linear trend. The independent increments of this Lévy process have a NIG distribution. See Barndorff-Nielsen (1998), Barndorff-Nielsen, Mikosch, and Resnick (2013), Benth, Groth, and Kettler (2006) for more on this process. For a NIG process with parameters $(\alpha, \beta, \mu, \delta)$, one has $Y(0) = 0$ and $Y(t) \sim \mathsf{NIG}(\alpha, \beta, t\mu, t\delta)$. To generate $Y(t)$, we can generate $I(t) \sim \mathsf{InvGaussian}(t\delta, \gamma)$ and then $Y(t) \sim \mathsf{N}(t\mu + \beta I(t),\, I(t))$, where $\gamma = (\alpha^2 - \beta^2)^{1/2}$. This is equivalent to $Y(t) = t\mu + X(I(t))$ where X is a BM with parameters $(\beta, 1)$. We can generate a skeleton simply by generating the independent NIG increments sequentially in this way.

We can also use a bridge approach to generate the skeleton at observation times $0 = t_0 < t_1 < \cdots < t_c$ by using the bridge to generate both the IG process and the BM, as in Section 2.16.2. One way to do this is to generate first the IG process at all observation times, then the BM at all observation times, conditionally on the realization of the IG process. A second approach is to interleave the bridge generation: first generate $I(t_c) \sim \mathsf{InvGaussian}(t_c\delta, \gamma)$, then $Y(t_c) \sim \mathsf{N}(t_c\mu + \beta I(t_c),\, I(t_c))$, then $I(t_{c/2})$ conditional on $(I(t_0), I(t_c))$, then $Y(t_{c/2})$ conditional on $(Y(t_0), Y(t_c))$, and so on.

A *geometric NIG process* $S = \{S(t),\, t \geq 0\}$ is obtained by taking

$$S(t) = S(0) \exp\left[rt + Y(t) + \omega t\right],$$

where Y is a NIG process with parameters $(\alpha, \beta, \mu, \delta)$, r is a constant that often represents the interest rate, and $\omega = \mu + \delta\gamma - \delta(\alpha^2 - (1+\beta)^2)^{1/2}$.

2.16.9 The stable process

If $t^{1/\alpha} S(t)$ has an α-stable distribution with location parameter $\mu = 0$, i.e., $t^{1/\alpha} S(t) \sim S(1) \sim S_\alpha(\sigma, \beta, 0)$, then $\{S(t),\, t \geq 0\}$ is a *stable process*. For $\alpha < 2$, this process has heavy-tailed increments.

If the increments have a tempered stable distribution instead, then we have a *tempered stable process*. A special case of it is the CGMY process, introduced by Carr et al. (2002).

2.17 Stationary Autocorrelated Stochastic Processes

Service times of successive customers in a queue, the sizes of the demands in an inventory model, etc., are often assumed to be i.i.d. random variables in classical examples of simulation models. This is not always realistic. There is often strong correlation between successive

observations in such input processes. Neglecting this correlation may have a dramatic impact on the results. In queuing systems, for examples, the average waiting times are much larger when the correlations are taken into account. Before discussing a few methods for modeling infinite sequences of dependent random variables by stochastic processes, we fist recall some definitions, which will also be needed in Chapter 5.

2.17.1 Time series and autocorrelation

For a process $\{Y_t,\, t \geq 0\}$, either discrete-time or continuous-time, we define the *mean function* $\mu_t = \mathbb{E}[Y_t]$, the *variance function* $\sigma_t^2 = \mathrm{Var}[Y_t]$, the *autocovariance function* $\mathrm{Cov}[Y_t, Y_s]$, and the *autocorrelation function* $\rho_{t,s} = \mathrm{Cov}[Y_t, Y_s]/\sigma_t\sigma_s$. The process is called *weakly stationary* if μ_t and σ_t^2 do not depend on t and if $\rho_{t,s}$ depends only on the *lag* size $|s-t|$. In this case, we denote μ_t by μ, σ_t^2 by σ^2, and $\rho_{t,t+s}$ by ρ_s, the *autocorrelation of lag s*. Weak stationarity does not imply that the *distribution* of Y_t is independent of t. For this, we need a stronger notion of stationarity. The process $\{Y_t,\, t \geq 0\}$ is called *strictly stationary* if for any $k \geq 1$ and any fixed vector $(t_1, \ldots, t_k)$, the joint distribution of the vector $(Y_{t+t_1}, \ldots, Y_{t+t_k})$ is independent of t. Modeling this joint distribution directly can be very complicated, so it is common practice to restrict attention to a narrow class of models and just fit the mean, variance, and autocorrelation functions. This is in the same spirit as the NORTA method.

A *time series model* is a discrete-time stochastic process $\{Y_n,\, n \geq 0\}$, where the Y_n's are generally dependent. To define such a process, say where Y_n has the distribution function F for all n, we can specify correlations directly on the Y_n's, or induce correlations on the process of underlying uniforms $U_n = F(Y_n)$, assuming that the Y_n's are generated from the U_n's by inversion, just as with copulas for multivariate distributions. There are also hybrid methods where the correlation is induced at some intermediate step. We now give examples.

2.17.2 Autoregressive processes and ARIMA models

A widely used class of time series models is the *autoregressive integrated moving average (ARIMA)* models studied by Box, Jenkins, and Reinsel (1994). The general form of an ARIMA(p, d, q) model is

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(1 - B)^d Y_n = \theta_0 + (1 - \theta_1 B - \cdots - \theta_q B^q)\epsilon_n, \tag{2.65}$$

where B represents the *backshift operator* defined by $B^r Y_n = Y_{n-r}$, the ϕ_j are the autoregressive parameters, the θ_j are the moving average parameters, and $\{\epsilon_n,\, n \geq 0\}$ is an i.i.d. sequence of random disturbances with mean 0 and variance σ^2. For statistical analysis, it is typically assumed that the ϵ_n are normally distributed, which in turns implies that the Y_n have the normal distribution. Thus, this model is a Gaussian process with discrete time index. The parameters must satisfy certain conditions for the model to be stationary (e.g., to avoid that $Y_n \to \infty$).

When $d = q = 0$, the model reduces to

$$(1 - \phi_1 B - \cdots - \phi_p B^p) Y_n = \theta_0 + \epsilon_n, \tag{2.66}$$

i.e.,

$$Y_n = \phi_1 Y_{n-1} + \cdots + \phi_p Y_{n-p} + \theta_0 + \epsilon_n. \tag{2.67}$$

This is an *autoregressive model of order p*, denoted AR(p).

The parameters of ARIMA models are usually estimated by least-squares methods, under the normality assumption. This assumption makes the model quite restrictive, because normally distributed input processes in simulation models are more the exception than the rule. But it certainly facilitates the mathematical analysis.

2.17.3 ARTA and VARTA models

In the same spirit as the NORTA method, Cario and Nelson (1996) introduced a class of processes called *autoregressive to anything (ARTA)*, whose purpose is to match exactly a desired stationary cdf F for Y_n and all autocorrelations up to lag p. The ARTA process $\{Y_n,\, n \geq 0\}$ is simply defined by $Y_n = F^{-1}[\Phi(Z_n)]$ where $\{Z_n,\, n \geq 0\}$ is an AR(p) process whose parameters ϕ_j are selected in just the right way to get the desired autocorrelations. The authors provide procedures to compute those parameters, for given target autocorrelations. Biller and Nelson (2002) propose a methodology for estimating the parameters of an ARTA model from raw data, for the case where F is from the Johnson family of distributions. See also Cario and Nelson (1998), Nelson and Yamnitsky (1998) for further details.

In Deler and Nelson (2001), this methodology is extended to *multivariate time series*, where each Y_n is a d-dimensional vector, yielding *vector autoregressive to anything* (VARTA) processes. Now $\{Z_n,\, n \geq 0\}$ is a d-dimensional vector AR(p) process, denoted VAR(p), whose parameters are specified indirectly by specifying the covariance matrix of Y_n and the autocorrelation matrices up to order p. The authors work out the case where all the marginals of Y_n are Johnson-type distributions.

2.17.4 Other ways of inducing autocorrelation between the underlying uniforms

The *minification* and *maxification* methods of Lewis and McKenzie (1991), and the *transform-expand-sample (TES)* method proposed by Melamed (1991), transform an i.i.d. $U(0,1)$ sequence $\{Z_n,\, n \geq 0\}$ into a sequence of correlated $U(0,1)$ random variables $\{U_n,\, n \geq 0\}$. The autocorrelations of the transformed sequence decrease with the lag, either monotonously or with a damped oscillation. To obtain an autocorrelated time series $\{Y_n,\, n \geq 0\}$ where Y_n has cdf F, one defines $Y_n = F^{-1}(U_n)$. To obtain autocorrelations of alternating signs (e.g., negative correlation between pairs of successive observations), one can replace U_{n-1} by $1 - U_{n-1}$ in the recurrence formulas below.

The minification and maxification methods use a single parameter $c > 1$. The *minification method* defines $U_0 = Z_0$ and

$$U_n = c \cdot \min\{U_{n-1},\ Z_{n-1}/(Z_{n-1} + c - 1)\} \tag{2.68}$$

for $n \geq 1$. The autocorrelations of the U_n's decrease exponentially with the lag: One has $\rho_j = \rho(U_n, U_{n+j}) = c^{-j}$. The *maxification method* defines $U_0 = Z_0$ and

$$U_n = \max\{U_{n-1}^c,\ Z_{n-1}^{c/(c+1)}\} \tag{2.69}$$

for $n \geq 1$. It gives the autocorrelation function $\rho_j = 3/(2c^j + 1)$ for the U_n's.

The *TES method* is motivated by the following simple idea. Given U_{n-1}, instead of generating $U_n \sim U(0,1)$ as in the i.i.d. case, generate U_n uniformly in a small neighborhood of U_{n-1} (for a positive correlation) or in a small neighborhood of $1 - U_{n-1}$ (for a negative correlation). The smaller the neighborhood the larger the (absolute) correlation. The neighborhood is determined by two parameters L and R, where $-0.5 < L < R \leq 0.5$. One defines $U_0 = Z_0$ and

$$U_n = (U_{n-1} + L + (R - L)Z_n) \bmod 1 \tag{2.70}$$

for $n \geq 1$. This recurrence is equivalent to defining

$$U_n = (U_0 + nL + (R - L)(Z_1 + \cdots + Z_n)) \bmod 1.$$

Jagerman and Melamed (1992) give (complicated) expressions for the autocorrelation function of the Y_n's in terms of the Laplace transforms of F^{-1} and of the density function of the Y_n's, and as a function of L and R. With the TES method, the autocorrelations $|\rho_j|$ decrease at a rate slower than exponential as a function of the lag j. The method tends to preserve large correlations much longer than the more traditional methods. It seems appropriate for modeling processes such as the incoming traffic in a communication system dealing with video images, Internet traffic, and the like, where *long range correlations* are commonplace.

Example 2.30 Livny, Melamed, and Tsiolis (1993) give a numerical illustration showing the dramatic effect of autocorrelations in the interarrival times or in the service times or both, on the average waiting time in an $M/M/1$ queue with arrival rate λ and service rate μ. These authors first applied the minification method to induce autocorrelations in the sequence of uniforms used to generate the exponential interarrival times (by inversion), and (separately) to the sequence of uniforms used to generate the exponential service times. In this context, the distribution F is the exponential and one has $Y_n = -\ln(1 - U_n)/\lambda$ if Y_n is an interarrival time and $Y_n = -\ln(1 - U_n)/\mu$ if Y_n is a service time. Then they did the same for the TES method with $L + R = 0$ and compared the results. Table 2.3 gives a small subset of their results, for $\lambda = 0.8$ and $\mu = 1$. The values of $\rho_{a,1}$ and $\rho_{s,1}$ in the table are the autocorrelations of lag 1 on the successive interarrival times and on the successive service times, respectively. The last two columns of the table give estimates of the ratio of the average waiting times with and without the autocorrelation. The TES method gives a much larger increase in the waiting times, due to the larger autocorrelations which make the arrival and service processes more bursty. □

These constructions are simple ways of inducing controlled correlations in a sequence of uniforms, with only one or two parameters. Models with more parameters could provide a larger amount of flexibility in the autocorrelation function, i.e., in how ρ_j behaves as a function of j.

Table 2.3. Ratios of the average waiting times with correlation over that without correlation, for the minification and the TES methods

$\rho_{a,1}$	$\rho_{s,1}$	minification	TES
0.00	0.00	1.0	1.0
0.50	0.00	1.7	6.0
0.00	0.50	1.8	4.6
0.25	0.25	1.5	2.4
0.50	0.50	2.6	10.2
0.85	0.85	10.4	424.6
−0.40	−0.40	0.5	8.9

2.18 Fitting a Distribution

2.18.1 Estimating the Parameters

The maximum likelihood method is arguably the grand favorite among the several approaches for constructing parameter estimators. Suppose an i.i.d. sample $x_1, \ldots, x_n$ of n observations has been obtained from a density $f_\theta(x)$, where θ is an unknown parameter (scalar or vector). The *maximum likelihood estimator (MLE)* $\hat{\theta}_n$ of θ is the value of θ that maximizes the likelihood function

$$L(\theta) = f_\theta(x_1) \cdots f_\theta(x_n).$$

For discrete distributions, one simply replaces the density by the probability mass function.

Example 2.31 Consider the Weibull distribution with location parameter $\delta = 0$. Its density is

$$f(x) = \alpha\lambda^\alpha x^{\alpha-1} e^{-\lambda x^\alpha} \qquad \text{for } x > 0.$$

The parameter θ in this case is the vector (α, λ) and the likelihood function is

$$L(\alpha, \lambda) = \alpha^n \lambda^{n\alpha} (x_1 \cdots x_n)^{\alpha-1} e^{-\lambda^\alpha (x_1^\alpha + \cdots + x_n^\alpha)}.$$

The *logarithm* of the likelihood function reaches its maximum at the same place as the likelihood function itself, so here we will find the maximum of $\ln L(\alpha, \lambda)$ because it is easier to work with than $L(\alpha, \lambda)$ (as is often the case). We have

$$\ln L(\alpha, \lambda) = n \ln \alpha + n\alpha \ln \lambda + (\alpha - 1) \sum_{i=1}^{n} \ln x_i - \lambda^\alpha \sum_{i=1}^{n} x_i^\alpha.$$

Equating the gradient (i.e., the partial derivatives) to zero, we obtain the two equations:

$$\begin{aligned} \frac{\partial \ln L(\alpha, \lambda)}{\partial \alpha} &= \frac{n}{\alpha} + n \ln \lambda + \sum_{i=1}^{n} \ln x_i - \sum_{i=1}^{n} (\lambda x_i)^\alpha \ln(\lambda x_i) = 0, \\ \frac{\partial \ln L(\alpha, \lambda)}{\partial \lambda} &= \frac{\alpha n}{\lambda} - \alpha \lambda^{\alpha-1} \sum_{i=1}^{n} x_i^\alpha = 0. \end{aligned}$$

To solve this system of two nonlinear equations in two unknowns, we can write λ as a function of α using the second equation, which yields the estimator

$$\hat{\lambda}_n = \left(\frac{n}{x_1^\alpha + \cdots + x_n^\alpha} \right)^{1/\alpha},$$

replace λ by this expression in the first equation, then solve the first equation for α numerically by an iterative fixed-point algorithm (Qiao and Tsokos 1994) and define $\hat{\alpha}_n$ as the solution. The MLE estimator of $\theta = (\alpha, \lambda)$ is then $\hat{\theta}_n = (\hat{\alpha}_n, \hat{\lambda}_n)$.

In the special case of the exponential distribution ($\alpha = 1$), one has $\hat{\lambda}_n = 1/\bar{x}_n = n/\sum_{i=1}^n x_i$. □

MLEs enjoy several nice statistical properties that justify their popularity: (1) The MLE is typically unique; (2) MLEs are strongly consistent, i.e., if $\hat{\boldsymbol{\theta}}_n$ is the MLE of $\boldsymbol{\theta}$ (a vector), then $\lim_{n\to\infty} \hat{\boldsymbol{\theta}}_n \overset{\text{w.p.1}}{=} \boldsymbol{\theta}$; (3) MLEs are invariant; i.e., if $\hat{\boldsymbol{\theta}}_n$ is the MLE of $\boldsymbol{\theta}$ and g is some function, then $g(\hat{\boldsymbol{\theta}}_n)$ is the MLE of $\nu = g(\boldsymbol{\theta})$; (4) MLEs are asymptotically normal:

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}) \;\Rightarrow\; N(0, n(\boldsymbol{I}(\boldsymbol{\theta}))^{-1}) \tag{2.71}$$

as $n \to \infty$, where $\boldsymbol{I}(\boldsymbol{\theta})$ is the *Fisher information matrix*, defined as the expectation of the *Hessian* of $-\ln L(\boldsymbol{\theta})$. That is, if $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)^{\mathrm{t}}$, the (i, j)th entry of $\boldsymbol{I}(\boldsymbol{\theta})$ is $-\mathbb{E}[\partial^2 \ln L(\theta_1, \ldots, \theta_d)/\partial\theta_i \partial\theta_j]$. This asymptotic normality provides a way of computing a confidence interval for $\boldsymbol{\theta}$ together with the MLE, using the normal (or multinormal) distribution. For this, an estimate of the matrix $\boldsymbol{I}(\boldsymbol{\theta})$ must be inverted to estimate the covariance matrix of $\hat{\boldsymbol{\theta}}_n$.

Example 2.32 For the exponential distribution, we saw that the MLE of λ is $\hat{\lambda}_n = 1/\bar{x}_n$. In this case it is easy to verify that $\boldsymbol{I}(\lambda) = n/\lambda^2$. Therefore $\sqrt{n}(\hat{\lambda}_n - \lambda) \;\Rightarrow\; N(0, \lambda^2)$, and thus $\sqrt{n}(\hat{\lambda}_n - \lambda)/\hat{\lambda}_n \;\Rightarrow\; N(0, 1)$, when $n \to \infty$. (For this simple example, this result also follows directly from the CLT.) □

There are cases where the MLE is not well defined; e.g., for Weibull, gamma, and lognormal distributions with unknown location parameters. There are also situations where the MLE has significant bias for finite n. Other approaches can then be used, such as the *moment matching*, which consists in selecting the parameter values so that the low-order moments of the fitted distribution match those of the data, the *least-squares method*, widely used in regression, etc. For a solid theoretical coverage of statistical estimation, we refer the reader to Lehmann and Casella (1998).

2.18.2 Assessing goodness of fit

After a distribution has been selected and the parameters have been estimated, one may want to measure or test the quality of fit of the retained distribution to the empirical data. There are two classes of approaches: *visual heuristic procedures* and *formal tests*.

Simple heuristics include comparing (visually) the histogram of the retained distribution with that of the data, comparing *box plots*, etc. These methods and many others are supported by standard statistical software.

Q-Q plots and P-P plots. Comparing the shapes of an empirical cdf with some theoretical cdf by direct visual assessment is usually difficult, especially if the two have approximately the same mean and variance, which often happens when parameters have been estimated. The *quantile-quantile (Q-Q) plots* and *probability-probability (P-P) plots* belong to the general class of *probability plots*, which are graphical techniques designed to facilitate the visual assessment.

Suppose we want to compare an empirical distribution $\hat{F}_n$ with a continuous distribution F. The basic idea of the P-P plot is to plot the values of $\hat{F}_n(x)$ against those of $F(x)$: they should look like a straight line at 45 degrees. More precisely, for each observation $x_{(i)}$, the P-P plot puts a point at $(\delta_i, F(x_{(i)}))$, where $\delta_i = \hat{F}_n(x_{(i)}) - 0.5/n = (i - 0.5)/n$ is the mid-point of the ith jump of $\hat{F}_n$; i.e., the average between $\hat{F}_n(x_{(i)})$ and $\hat{F}_n(x_{(i)}^-) = \lim_{\epsilon \to 0^+} \hat{F}_n(x_{(i)} - \epsilon)$.

For the Q-Q plot, we just apply F^{-1} to both coordinates of the P-P plot points. That is, we plot the points $(F^{-1}(\delta_i), x_{(i)})$, $1 \le i \le n$, which should also form (roughly) a straight line at 45 degrees.

The Q-Q plots tend to amplify differences in the tails of the distributions, whereas P-P plots amplify the differences in the middle.

2.18.3 Testing goodness of fit

Formal *goodness-of-fit tests* consider the null hypothesis $\mathcal{H}_0$: "The data comes from the retained distribution" and look for evidence against this hypothesis. A test is defined by a test statistic Y, function of the data, and whose distribution under $\mathcal{H}_0$ is known at least approximately. The test rejects $\mathcal{H}_0$ if the observed value of Y is deemed too large (say). It is common practice to compute and report the *p-value* of the test, defined as $p = \mathbb{P}[Y \ge y \mid \mathcal{H}_0]$ if y is the value actually taken by Y. When Y has a continuous distribution under $\mathcal{H}_0$, the p-value should be a $U(0,1)$ random variable. A very small p-value suggests that the obtained value of Y is unlikely to have been obtained by chance under $\mathcal{H}_0$ because it is too large, and thus provides evidence against $\mathcal{H}_0$. Deciding how small is too small is a question of subjective judgment. When the p-value is deemed too small, we reject the distribution and seek another one. Standard goodness-of-fit tests include the chi-square, Kolmogorov-Smirnov, Anderson-Darling, etc. (e.g., Durbin 1973, Read and Cressie 1988).

The *chi-square test*, introduced by Karl Pearson in 1900, is typically used for testing the fit of a *discrete distribution* to a particular data set. The possible outcomes (possible values of the random variable whose distribution is tested) are partitioned into a finite number of categories. Suppose there are k categories and that each observation belongs to category i with probability p_i, for $0 \le i < k$, under $\mathcal{H}_0$. If there are n independent observations, the expected number of observations in category i is $e_i = np_i$, and the chi-square test statistic is defined as

$$X^2 = \sum_{i=0}^{k-1} \frac{(O_i - e_i)^2}{e_i} \tag{2.72}$$

where O_i is the actual number of observations falling in category i. Assuming that all e_i's are large enough (a popular rule of thumb asks for $e_i \ge 5$ for each i), X^2 follows approximately the chi-square distribution with $k - 1$ degrees of freedom (e.g., Read and Cressie 1988). If some e_i's are too small, one can simply regroup categories. In fact, the approximation by the

chi-square distribution can also be good when the e_i are small, provided that k is large and the e_i are close to each other. So if x is the value taken by the test statistic X^2, the (right) p-value is given (approximately) by $p = 1 - F(x)$ where F is the cdf of a chi-square random variable with $k - 1$ degrees of freedom. A very small p-value indicates a large discrepancy between the expected and observed counts. There are circumstances where one may also want to reject $\mathcal{H}_0$ when the fit is "too good to be true," i.e., p too close to 1.

A general way of testing the fit of a set of observations $x_1, \ldots, x_n$ to a univariate continuous distribution is to compute a measure of distance between the corresponding cdf F and the empirical cdf $\hat{F}_n$ of the observations (defined in Section 2.9.1) and use the distance as a test statistic. Different measures of distance define different tests. A large distance indicates a poor fit. Let $x_{(1)}, \ldots, x_{(n)}$ be the observations sorted by increasing order and define $u_{(i)} = F(x_{(i)})$ for each i.

The *Kolmogorov-Smirnov (KS) test* uses the $\mathcal{L}_\infty$-distance D_n defined in Eq. (2.19):

$$D_n = \sup_{-\infty<x<\infty} |\hat{F}_n(x) - F(x)|,$$

and which can be computed as follows:

$$\begin{aligned} D_n^+ &= \max_{1\le i\le n} [i/n - u_{(i)}], \\ D_n^- &= \max_{1\le i\le n} [u_{(i)} - (i-1)/n], \\ D_n &= \max\,(D_n^+, D_n^-). \end{aligned}$$

This is the maximal vertical distance between the empirical and theoretical cdfs, over their entire domains. Computing the p-value of the test requires the cdf of D_n under $\mathcal{H}_0$. Approximations of this distribution are given by Darling (1960), Corollary Z, page 356, Stephens (1970), and Marsaglia, Tsang, and Wang (2003). The one-sided distances D_n^+ and D_n^- are also used as test statistics in some situations.

The *Anderson-Darling test* uses the distance

$$A_n^2 = n \int_{-\infty}^{\infty} \frac{(\hat{F}_n(x) - F(x))^2}{F(x)(1 - F(x))} f(x)dx, \tag{2.73}$$

where f is the density of F. It can be computed by the formula

$$A_n^2 = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)(\ln(u_{(i)}) + \ln(1 - u_{(n+1-i)})).$$

(One must be careful about numerical errors when $u_{(1)}$ is too close to 0 or $u_{(n)}$ is too close to 1). This test statistic is designed to be much more sensitive than D_n to detect discrepancies in the tails of the distribution. The denominator of the integrand in (2.73) becomes small in the tails and this gives more weight to the discrepancies in these areas. For approximations of the cdf of A_n^2, see Stephens (1986) and Marsaglia and Marsaglia (2004).

Goodness-of-fit test statistics do not have the same distribution under the null hypothesis if the parameters have been estimated from the same data that are used for the tests, than if the parameters were fixed in advance or estimated from different (independent) data.

One must make sure that the right distribution of the test statistic is used. For a cdf F *fixed in advance*, the distributions of D_n and A_n^2 under $\mathcal{H}_0$, for example, do not depend on F. But if the parameters of F were estimated from the n observations that define $\hat{F}_n$, then the distributions of D_n and A_n^2 depend on the type of distribution of F (normal, exponential, Weibull, gamma, etc.). Recipes to approximate the p-values in these situations are summarized by Law and Kelton (2000).

Moreover, if several distributions are fitted to the data, the one that looks best from visual inspection is retained, and a standard goodness-of-fit test at level α is applied to it, then the *true level* of the test is generally much less than α, because this distribution was retained for its good fit in the first place, so it is less likely to give a bad fit than if it was fixed a priori. The chance of rejecting $\mathcal{H}_0$ with such a procedure can be small even if none of the distributions tried really corresponds to the data, because if many have been tried, it is likely that one will look similar to the data by chance. Distribution-fitting software for simulation often use this type of procedure. Do not trust blindly the p-values that they claim.

For references to tests of independence and stationarity, we refer the reader to Schmeiser (1999) and Law and Kelton (2000).

It is important to realize that applying any of these tests formally is not really appropriate to determine whether or not a given input distribution is adequate, as we already mentioned in Section 2.2.2. In practice we already know that $\mathcal{H}_0$ is never formally true. When it is not rejected, the reason is (most of the time) that the test is not powerful enough to detect the difference. We may say that undetected differences are certainly small differences and are therefore acceptable because they should not influence the end results significantly. But this actually depends on the model and on its intended application. Differences declared significant by some test with a lot of data can also be deemed acceptable in many situations. The truly relevant question is: "What kind of lack-of-fit is acceptable for the application at hand?"

2.19 Performance Measures over Finite and Infinite Time Horizons

A simulation model is called a *finite-horizon model* if we are interested in its evolution only up to the time t_N of the Nth event, where N is in general a random variable, $N < \infty$ with probability 1, and the value of N is known after the Nth event has occurred. That is, N should be a stopping time with respect to the filtration $\{\mathcal{F}_i,\, i \geq 0\}$ generated by $\{\mathcal{S}_i,\, i \geq 0\}$, where $\mathcal{S}_i$ is the *state of the model* immediately after event i has occurred, so $\mathcal{F}_i$ represents the *cumulative information* available thus far (as in Example 2.25). A deterministic time horizon T fits this definition if we schedule an event at time T. We have an *infinite-horizon model* if, for example, we are interested in the total expected discounted cost, or the average cost per unit of time or per customer (or other entity), assuming that the model runs forever. We examine these different kinds of models in this section. What we call the *cost* here can be any real-valued performance measure. It does not have to involve money.

2.19.1 Finite horizon, additive cost function

In a finite-horizon simulation model, we want to estimate an unknown quantity μ by a random variable X that can be computed in finite time from a simulation. This random variable can often be expressed as a sum:

$$X = V_N = \sum_{i=1}^{N} C_i, \tag{2.74}$$

where N is a stopping time with respect to $\{\mathcal{F}_i, i \geq 0\}$, and each C_i is a $\mathcal{F}_i$-measurable random variable which can be interpreted as the *cost* (or reward) incurred at the time t_i of the i-th event e_i. The cost function V_N in (2.74) is *additive*. If $\mathbb{E}[V_N]$ is what we want to estimate, we readily have an unbiased estimator.

In some models, the costs or rewards are accumulated continuously at a state-dependent rate $c(\cdot)$. The total cost until the time t_N of the Nth event is

$$V_N = \int_0^{t_N} c(\mathcal{S}(t))dt, \tag{2.75}$$

where $\mathcal{S}(t)$ is the state of the model at time t. If we assume that $\mathcal{S}(t)$ does not change between events (i.e., that the model is purely "discrete-event"), then (2.75) can be rewritten in the form (2.74) by putting

$$C_i = \int_{t_{i-1}}^{t_i} c(\mathcal{S}(t))dt = (t_i - t_{i-1})c(\mathcal{S}_{i-1})$$

for each i.

Alternatively, one can define C_i as the expected value of $\int_{t_{i-1}}^{t_i} c(\mathcal{S}(t))dt$, conditional on $\mathcal{S}_{i-1}$, as is usually done in dynamic programming contexts (Bertsekas 1995). This gives a *different estimator* than (2.74), but it has the same expectation, is sometimes less expensive to compute, and may have less variance than (2.74) (see Section 6.6).

Sometimes, one may want to split the estimator in two parts, one written as (2.74) and the other written as (2.75). These are just rewritings of the general finite-horizon estimator (2.74).

Quantities of interest are not always naturally expressed as the expectation of an additive function of the form (2.74) or (2.75). This is illustrated by Example 1.12 and the next examples.

Example 2.33 Consider the call center example (Section 1.12) and suppose we want to estimate the expected number of calls whose waiting time exceeds 20 seconds, in a day. We can define $C_i = 1$ if event e_i starts the service for a call having waited more than 20 seconds; $C_i = 0$ otherwise.

If we want to estimate the probability of losing more than five calls on a given day, we can define $N = j$ and $C_N = 1$ if the sixth abandonment of the day occurs at event e_j, and all other C_i's equal 0. If less than six abandonments occur, then e_N is the last event of the day and $C_i = 0$ for all i.

Now, suppose we want to estimate instead the expected total wait in the queue during the day, from the opening time T_0 to the closing time T_1. The waiting time after closing is not counted. This is the expectation of

$$\int_{T_0}^{T_1} Q(t)dt \tag{2.76}$$

where $Q(t)$ is the queue length at time t. The simulation program in Section 1.12 does not compute this integral. Moreover, since $Q(t)$ changes when there is an abandonment, we would need to have an event at every abandonment to be able to write this integral in the form (2.74) and compute it that way during the simulation. In its current form, the program does not have such an event and finds about (earlier) abandonment of a calling customer only when trying to pick up the call from the queue. But if we add an event at each abandonment, then the integral equals (2.74) where $C_i = (t_i - t_{i-1})Q(t_{i-1})$, t_i is the occurrence time of event i, $Q(t_{i-1})$ is the number of calls in the queue at time t_{i-1}, just after event e_{i-1}, and N is the number of the event that closes the center. This integral can be computed in a simulation program by a statistical collector of type `Accumulate`, as in the example of Section 1.11.1.

Things are simpler if we want to compute the total waiting time in the queue for the day, not counting the customers who abandoned (if any), *including* the wait in the queue after the center closes. This total waiting time is simply the sum of waiting times of all calls and it is already computed by the program. This last sum can be written as (2.74) where C_i is the sum of waiting times of all calls that leave the queue at event i. Occasionally, there could be more than one call leaving the queue at a given event. For many events, there would be none and this C_i will be 0.

If we divide the total waiting time of the day by $\max(T, T_1) - T_0$, where T is the epoch when the last call leaves the queue, then we get a *ratio* of two random variables, because T is random. Each of the two terms of this ratio is additive and has the form (2.74) but not the ratio itself, unless we just define C_N as equal to this ratio. Using the expectation of this ratio as a performance measure would be highly questionable, because if $T \gg T_1$ due to the fact that a single call has waited a long time in the queue after the center was closed, the ratio can take a much smaller value than the average queue length during the operation hours. So the expected value of the ratio may differ significantly from the expected average queue length during the operating hours, which is the expectation of (2.76) divided by $T_1 - T_0$. □

Example 2.34 For the $GI/G/1$ queue example, let X be the maximum size of the queue during a deterministic time interval $[0, T]$, and suppose that we are interested in estimating $\mathbb{E}[X]$. This is a non-additive cost function. To fit this into the additive framework (2.74) for which C_i is $\mathcal{F}_i$-measurable, we must define $C_N = X$ and $C_i = 0$ for $i < N$. Likewise, in Examples 1.8 and 1.4, to put the estimator in the form (2.74), we must also define $C_N = X$. For these examples, the costs are not additive. □

2.19.2 Discounted costs

Costs incurred in the future can be multiplied by a *discount factor* smaller than 1 to take into account the fact that they have to be paid only later. Discounting appears in almost

all finance and economic models, for instance. The discount factor is often an exponentially decreasing function of time, equal to $e^{-\rho t}$ at time t. The constant $\rho > 0$ is called the *discount rate*. If a cost C_i is incurred at time t_i, the corresponding discounted cost becomes $\widetilde{C}_i = e^{-\rho t_i} C_i$ and the total discounted cost $V_{\rho,N}$ for the first N events is given by (2.74) with C_i replaced by $\widetilde{C}_i$:

$$V_{\rho,N} = \sum_{i=1}^{N} \widetilde{C}_i = \sum_{i=1}^{N} e^{-\rho t_i} C_i. \tag{2.77}$$

For a cost that accumulates continuously at state-dependent rate $c(\mathcal{S}(t))$, one defines $C_i = \int_{t_{i-1}}^{t_i} e^{-\rho(t-t_i)} c(\mathcal{S}(t)) dt$ and $\widetilde{C}_i = \int_{t_{i-1}}^{t_i} e^{-\rho t} c(\mathcal{S}(t)) dt$.

Example 2.35 In Example 1.11, there is a single payoff C_N at time $t_N = T$, and the discounted payoff is $V_{\rho,N} = \widetilde{C}_N = e^{-\rho t_N} C_N$. □

Example 2.36 This example is taken from L'Ecuyer (1983). Consider a system with M identical components subject to individual failures (e.g., street lights, truck tires, etc.). When a component fails, it must be replaced immediately by a new one. Components can also be replaced preventively at any time. We assume that the component lifetimes are i.i.d. random variables and that the replacements are instantaneous. At each intervention, there is a fixed cost c_1, plus a cost c_2 for each component replaced, plus a cost c_3 if the intervention is forced by a failure.

The replacement decisions are dictated by a *policy* which tells what to do for every possible state of the model. One example of such a policy: Select two arbitrary thresholds $\bar{\theta} > \theta > 0$, and whenever a component fails or its age reaches $\bar{\theta}$, perform an intervention that replaces all components whose age exceeds θ, plus the failed component if any.

If components are not replaced very frequently and the model evolution is simulated over several years, it makes sense to discount the costs, because one dollar paid in several years costs less than one dollar paid today. Suppose the discount rate is ρ per year. Then, the total cost over a time horizon of T years is

$$V_{\rho,N(T)} = \sum_{i=1}^{N(T)} e^{-\rho t_i} [c_1 + \eta_i c_2 + \delta_i c_3] \tag{2.78}$$

where $N(T)$ is the number of events that occur during the time interval $[0, T]$, t_i is the time of the ith intervention (the ith event), η_i is the number of components replaced at that intervention, and $\delta_i = 1$ if this event was forced by a failure, $\delta_i = 0$ otherwise. This expression has the form (2.77), except that N is replaced here by the (random) stopping time $N(T)$. For a given policy, the expected discounted cost over the interval $[0, T]$ can be estimated by simulation. Usually, the end goal is to *optimize the policy*; we will return to this later. □

2.19.3 Ratios and other nonlinear functions of expectations

There are situations where the quantity μ of interest is expressed as a ratio of two mathematical expectations, or as a more general function of several expectations over a finite horizon,

instead of as a single expectation. An important setup where μ is a ratio of two expectations is regenerative simulation (see Chapter 5).

Example 2.37 Consider a random variable X and an event $B \in \mathcal{F}$, and suppose that we are interested in the conditional expectation $\mathbb{E}[X \mid B]$. This conditional expectation can be written as $\mu = \mathbb{E}[X \mid B] = \mathbb{E}[X \cdot \mathbb{I}(B)]/\mathbb{E}[\mathbb{I}(B)]$, where $\mathbb{I}$ is the indicator function, and the problem of estimating μ becomes the problem of estimating each of the two expectations in that ratio. □

Example 2.38 In the call center example, suppose we are interested in estimating the *expected fraction* of calls waiting less than s seconds on a given day. If A is the number of arrivals and $X = G(s)$ the number of those who waited less than s seconds, then we are interested in $\mathbb{E}[X/A]$ (in case $A = 0$, we define $0/0$ as equal to 0) and we readily have an unbiased estimator of that expectation by simulating one day of operation and computing X/A.

On the other hand, suppose we are interested instead in $g(s)$, the (expected) fraction of calls waiting less than s seconds in the long run, assuming that the center operates over an infinite sequence of successive days. It can be shown (see Section 5.12) that this fraction is equal to the ratio $\mathbb{E}[X]/\mathbb{E}[A]$, which *differs* from $\mathbb{E}[X/A]$ and is more difficult to estimate in the sense that no unbiased estimator is readily available, unless we know $\mathbb{E}[A]$ exactly beforehand.

The quantity $\mathbb{E}[X/A]$ is a finite-horizon performance measure, whereas $g(s)$ is a performance measure over an infinite horizon. If one picks a day at random, then choose a random call among those which arrived that day, the probability that this call waits more than 20 seconds is $\mathbb{E}[X/A]$. The calls that happen to be on days when A is small have more chance to be picked than those which are on days where A is large. On the other hand, if one chooses a random call uniformly from the set of all arriving calls over all days, the probability that it waits more than 20 seconds is $g(s)$. Here, all the calls have the same chance of being picked and this measure really corresponds to an average per call in the long run.

Similarly, the average waiting time per call for all arriving calls (over all days) is not the same as the expectation of the average waiting time per call for a given day (see Exercise 2.48). □

In Chapters 5 and 7, we will see other examples where ratios (or more general functions) of expectations are involved. See also Glynn and Heidelberger (1990) and Glynn, L'Ecuyer, and Adès (1991). In those situations, an unbiased estimator for the quantity of interest is rarely available, even if an unbiased estimator is available for each individual expectation, and estimation is more difficult than for a single expectation. However, asymptotically unbiased estimators are generally available.

2.19.4 Infinite-horizon, long-term average

Here, we consider models that run forever. We might want to estimate a *steady-state average*, i.e., a long-term time-average, assuming that such an average exists uniquely. A common case is the time-average

$$\bar{v} \stackrel{\text{def}}{=} \lim_{t\to\infty} \frac{\mathbb{E}[V_{N(t)}]}{t}. \tag{2.79}$$

Under certain ergodicity conditions (Meyn and Tweedie 1993; see also Theorem A.18), one also has

$$\bar{v} \stackrel{\text{w.p.1}}{=} \lim_{t\to\infty} \frac{V_{N(t)}}{t}. \tag{2.80}$$

Typically, in contrast to $\mathbb{E}[V_{N(t)}]$, $\bar{v}$ does not depend on the initial state $\mathcal{S}_0$ but is usually harder to estimate. Assuming that $t_N \to \infty$ as $N \to \infty$, one also has

$$\bar{v} = \lim_{N\to\infty} \frac{\mathbb{E}[V_N]}{t_N} \stackrel{\text{w.p.1}}{=} \lim_{N\to\infty} \frac{V_N}{t_N}. \tag{2.81}$$

Alternatively, instead of being interested in a time-average, one may be interested in a limit of the form:

$$\widetilde{v} = \lim_{t\to\infty} \frac{\mathbb{E}[V_{N(t)}]}{\mathbb{E}[N_c(t)]} \stackrel{\text{w.p.1}}{=} \lim_{t\to\infty} \frac{V_{N(t)}}{N_c(t)}, \tag{2.82}$$

where $N_c(t)$ is the number of events of a certain particular type that have occurred by time t. This expression gives the average cost per event, in the long run, for the events considered.

Example 2.39 In the $GI/G/1$ queue example (Section 1.11), let $V_{N(T)} = \int_0^T Q(t)dt$, so that $V_{N(t)}/t$ represents the average size of the queue over the time interval $[0, t]$, and $\bar{v} = q$, the average queue length over an infinite time horizon. To illustrate (2.82), let $N_c(t)$ be the number of customers who have started their service by time t, and let C_i be the waiting time in the queue for the customer who begins its service at time t_i if event e_i starts a service, $C_i = 0$ otherwise. Then, $V_{N(t)}$ is the total waiting time in the queue for all the customers who have started their service by time t and $\widetilde{v}$ represents the average waiting time *per customer* in the long run. □

The $\bar{v}$ and $\widetilde{v}$ in (2.79)–(2.82) are often referred to as *steady-state averages*. Under certain conditions, they are indeed averages with respect to some steady-state or limiting distribution. Here, "limiting distribution" may have different meanings, as we explain in Chapter 5.

2.19.5 Total discounted cost

When discounting is present, it makes sense to speak of the *total discounted cost over an infinite horizon*. (Without discounting, the total cost over an infinite horizon is typically infinite.) With a discount rate $\rho > 0$, this cost is a random variable defined as:

$$V_\rho^\infty = \lim_{t\to\infty} V_{\rho,N(t)} = \lim_{t\to\infty} \sum_{i=1}^{N(t)} e^{-\rho t_i} C_i = \sum_{i=1}^{\infty} \widetilde{C}_i, \tag{2.83}$$

assuming that $\lim_{t\to\infty} N(t) = \infty$. Note that the distribution of V_ρ^∞ depends on the initial state $\mathcal{S}_0$. We denote

$$v_\rho^\infty = \lim_{t\to\infty} \mathbb{E}[V_{\rho,N(t)}] \tag{2.84}$$

When simulating to estimate v_ρ^∞, we must stop at some finite t, and this gives a biased estimator. For a fixed total computing budget, there is a compromise to be made between increasing t to reduce the bias, and decreasing t to increase the number of replications and thus reduce the variance. We will return to this in Chapter 5.

Example 2.40 In the component replacement problem (Example 2.36), we may be interested in estimating the total expected discounted cost over an infinite time horizon. This expectation is given by v_ρ^∞. □

2.19.6 Finite vs infinite horizon

When is an infinite horizon more appropriate than a finite horizon, and vice-versa? And when should we discount the costs or rewards? Of course, this depends on the situation and it is a matter of judgment. One may argue that real life is always over a finite horizon: In the long run, our world will eventually end anyway. Infinite-horizon models are always approximations for systems that really evolve over a finite horizon. Moreover, simulation models can only be run over a finite period of time, so infinite-horizon models must in turn be approximated (somewhat paradoxically) by finite-horizon simulations. More energy is often spent on the question of whether the (long) finite-horizon simulation approximates the infinite-horizon model well enough than on the (perhaps more relevant) question of how realistic the infinite-horizon model is.

A steady-state infinite-horizon model might be appropriate when:

(a) The system can be assumed time-stationary, i.e., its laws of evolution and underlying probability distributions do not change with time, at least for the time period of interest.

(b) The steady-state distribution is approached relatively quickly for any reasonable initial state.

(c) We are interested in the performance of the system over a long period of time, in comparison with the time required to approach the steady-state distribution.

Steady-state models are also frequently used when we want to compare the output of a simulation (sub)model with values provided by analytical formulas. This is sometimes done at validation and verification stages.

Example 2.41 Steady-state analysis over an infinite horizon might be appropriate for a communication switch where packets of information arrive at a rate so fast that the effect of the initial state essentially disappears after a few seconds on the simulation clock. □

Example 2.42 In telephone call centers, incoming traffic is a random process with an arrival rate that depends on the time of the day, day of the week, and other conditions. Call center managers often divide the time into half-hour periods and assume a constant arrival rate over each such period, as we did in Section 1.12. To be able to use analytic formulas, they also frequently assume that over any given half-hour period, the system behaves as

if it was in steady-state, and they analyze the system by computing infinite-horizon long-term averages. This may give reasonable rough-cut approximations if calls are short and arrive at a frequent rate. If both the interarrival times and service times are assumed to be i.i.d. exponential, for instance, and if the number of agents (the servers) is fixed, then the proportion $g(s)$ of calls that must wait more than s seconds in the queue, over an infinite horizon, can be computed via one of the Erlang formulas (Gross and Harris 1998, Gans, Koole, and Mandelbaum 2003) or by certain approximations based on an analysis under limiting conditions, such as the Halfin-Whitt approximation mentioned in Section A.19. This is the main reason why steady-state models are used. Simulation, on the other hand, can handle more realistic models which are never really in steady-state. □

Example 2.43 Suppose a major reorganization of a production plant is to take effect on Monday in two weeks. We want to simulate the new plant to evaluate what is likely to happen. In that context, the current state of the plant (inventories, demands, machine states, etc.) should be taken as initial state. A steady-state infinite-horizon model is appropriate here only if the effect of this initial state disappears quickly (e.g., the turnaround in the plant is very fast), and if stationarity can be assumed over the time period of interest. One should start collecting statistics to estimate the steady-state average only after the reorganization is in effect. If we are mainly interested in the first few days after reorganization, a finite-horizon model must be used. □

A model with discounting (with finite or infinite horizon) is a simple way to give (exponentially) decreasing importance to the future, so that things that happen far away in time have little influence on the result. This makes sense for models that involve costs over a long period of time (several months or years). It is the standard way of incorporating the effect of the interest rate on the accumulated costs and is used routinely in finance and economics.

If a simulation model is used periodically to predict future costs or performance, a *rolling horizon* is another popular way of removing the effect of what happens too far away in the future, where the validity of the model may become questionable. This technique is often used in optimization, when the decisions regarding the operation of a system have to be periodically re-optimized. The idea is to use a finite horizon T, starting from the current state, each time the simulation (or optimization) is performed. The horizon T thus moves ahead each time the system is re-simulated or re-optimized. For the production plant discussed above, for example, the system can be simulated every week, starting from its current (physical) state, with a time horizon of 52 weeks.

2.20 Exercises

2.1 Identify what could be a reasonable distribution for the random variable X, and explain why, if X represents

(a) The number of raisins in a raisin bread.

(b) The time until the first failure of a component in a large computer network.

(c) The average weight of 10 adults chosen at random in the population of Montreal.

(d) The number of ambulance calls in a given city on a given day.

2.2 (Hawthorne effect.) You need data to estimate the distribution of times taken by workers to perform a certain job. If the workers are aware of your observation and data collection, discuss how this can bias your results.

2.3 Consider a $GI/G/1$ single-server queue with distributions F and G for the interarrival times and service times.

(a) Suppose the interarrival times are exponential with parameter $\lambda = 1$. What is w, the average waiting time per customer in steady-state, for each of the following service time distributions: (1) constant, always equal to 1/2; (2) uniform over the two values 0 and 1; (3) exponential with parameter $\mu = 2$; (4) gamma with parameters (1/10, 1/5)?

(b) What if the interarrival times are constant instead, always equal to 1? Run a simulation experiment to estimate w for the four service time distributions and compare your results with the values found in (a). For this, you can simulate $10^3 + 10^6$ customers and start collecting the statistics only after the first 10^3 customers have started their service. You can repeat this $n = 5$ times, independently, to obtain independent observations for computing a confidence interval.

(c) For interarrival times that are exponential with parameter 1, and the four service time distributions as in (a), run a simulation experiment as in (b) to estimate the fraction of customers whose waiting time exceeds 2, in steady-state. Discuss your results.

2.4 In the call center example of Section 1.12, suppose we want to estimate the expected number of *abandonments* in a day and study the impact of certain approximations or simplifications on that expectation. Perform the following sensitivity analysis, with the parameters given in Section 1.7 for the first configuration (these parameters values should be in the data file that comes with the program). In each case, give an estimate of the relative bias (or change in the expected value of the estimator) caused by the simplification, and comment on the importance of that additional error.

(a) If we model the service time distribution as an exponential instead of a gamma distribution, where the exponential has the same mean α/β as the gamma, does this significantly affect (i.e., by more than 10%) the expected number of abandonments in a day?

(b) Same question if we replace the service time by its expectation in the model, i.e., if we use a constant service time always equal to α/β.

(c) And what if the non-stationary Poisson process is replaced by a stationary Poisson process whose arrival rate λ equals the average expected arrival rate over the day, i.e., $\lambda = (\lambda_1 + \cdots + \lambda_m)/m$?

(d) And if we combine the two simplifications in items (b) and (c)?

2.5 Customer abandonment in a queueing system can have a large impact on the waiting times because they help reducing the queue size just when we need it: when the waiting times are long. In a queueing system with high utilization factor, an abandonment ratio of just a few percent can make a large difference on the fraction of customers whose waiting times exceed a given threshold.

For the call center example of Section 1.12, our estimation of $g(20)/a$ in Section 1.7, with the data given there, was approximately 0.86 ± 0.01 with 90% confidence. The patience time was assumed to be 0 with probability $p = 0.1$ and exponential with rate $\nu = 0.001$ with probability $1 - p$.

Now we want to compare our estimate of $g(s)/a$ with this patience time distribution to that obtained with the two following alternatives: (i) $p = \nu = 0$ (no abandonment) and (ii) $p = 0.2$ and $\nu = 0.04$ (faster abandonment). Simulate the center for 1000 days with the three alternatives, with well-synchronized common random numbers, to evaluate the impact of the patience time distribution on $g(s)/a$ and on $\ell = \mathbb{E}[L_i]/a$. Discuss your results.

2.6 Sensitivity to claim size distribution in ruin probability: See Asmussen (2000), page 86. [16]

2.7 If $b > 0$ is a real number and T is a (continuous) uniformly distributed random variable over the interval $[0, b)$, show that $X = (T + t) \bmod b$ is also uniformly distributed over $[0, b)$ for any fixed $t \geq 0$.

2.8 Show that if $X_1, \ldots, X_q$ are independent random variables where $X_i \sim \mathsf{Binomial}(n_i, p)$, then $X = X_1 + \cdots X_q \sim \mathsf{Binomial}(n, p)$ where $n = n_1 + \cdots + n_q$.

2.9 Show that if $X \sim \mathsf{Geometric}(p)$, then $\mathbb{P}[X = y + x \mid X \geq y] = \mathbb{P}[X = x]$. This is called the *memoryless property* of the geometric distribution.

2.10 Show that if $X_1, \ldots, X_q$ are independent and $X_i \sim \mathsf{Poisson}(\lambda_i)$ for each i, then $X = X_1 + \cdots + X_q \sim \mathsf{Poisson}(\lambda)$ where $\lambda = \lambda_1 + \cdots + \lambda_q$.

2.11 Show that if X is a binomial random variable with parameters (N, p) where N is a Poisson random variable with parameter λ, then $X \sim \mathsf{Poisson}(\lambda p)$.

2.12 Show that if $X \sim \mathsf{Erlang}(k, \lambda)$ and $Y \sim \mathsf{Poisson}(\lambda x)$, then $\mathbb{P}[X \leq x] = \mathbb{P}[Y \geq k]$. Hint: You can use the fact that the times between the successive jumps of a stationary Poisson process are i.i.d. exponentials. This equivalence gives the expression (2.11) for the Erlang cdf.

2.13 Give two examples of distributions F for which Theorem 2.2 holds with $\delta = 0$ and $\alpha = 1$. What is the limiting Weibull distribution of W_n/d_n then?

2.14 Show that if $X_1, \ldots, X_k$ are independent exponential random variables with rates $\lambda_1, \ldots, \lambda_k$, respectively, then $X = \min(X_1, \ldots, X_k)$ is exponential with rate $\lambda = \lambda_1 + \cdots + \lambda_k$. Hint: Write the reliability function $\bar{F}(x) = \mathbb{P}[X > x]$.

2.15 Consider an $M/G/1/K$ queue with arrival rate λ, exponential phase-type service-time distribution with k phases, and finite capacity K (any arrival occurring when there is already

[16] From Pierre: Details... Check first for expressions for the moment generating functions and if change of measure is easy to get ...

K customers in the system is simply dismissed, so the number of customers in the system never exceeds K). For the phase-type distribution, let the duration of phase j be exponential with rate parameter μ_j and let $1 - p_j$ be the probability of exit after phase j.

(a) Explain how to simulate the sequence of events and their times of occurrence in such a system without an event list. How would you define the state of the system to have a finite state space? Hint: Knowing how many customers are in the queue is not enough.

(b) Implement it and run the simulation with $\lambda = 1$, $k = 5$, $\mu_j = 5j$, $p_j = 0.9$, $K = 10$, and a time horizon $T = 10^5$. Compute the number of customers lost due to overflow and the average queue length.

2.16 Let F be the cdf of some random variable Y and let $-\infty < a < b < \infty$.

(a) Prove that if $X = F^{-1}(U)$ where $U \sim \mathsf{Uniform}(F(a), F(b))$, then X has the same distribution as Y, but truncated to the interval $(a, b]$, i.e., X has the distribution of Y conditional on $Y \in (a, b]$. This provides an efficient way to generate X from this conditional distribution when F and F^{-1} are easy to evaluate. Hint: To show that, it suffices to show that the cdf of X is the same as the cdf of Y conditional on $Y \in (a, b]$.

(b) If F is continuous, truncating to $(a, b]$ is the same as truncating to the interval $[a, b]$, but not if F has a jump at a. Why? How would you generate X from distribution F truncated to $[a, b]$ in the latter case?

2.17 Show that the mean of the quasi-empirical distribution $\check{F}_n$ defined in (2.22) equals the sample mean $(1/n)\sum_{i=1}^{n} x_i$. Derive an expression for the variance of this quasi-empirical distribution.

2.18 Construct a modification of the quasi-empirical distribution (2.22) which starts at $x_{(1)}$ instead of $x_{(0)} = 0$. Its definition will be the same as $\widetilde{F}$ in (2.21) for $x \le x_{(n-k)}$, with an exponential tail for $x > x_{(n-k)}$, and the cdf must be continuous. Give a formula for how to choose the parameter of the exponential tail so that the quasi-empirical distribution and the sample have the same means.

2.19 (Bratley, Fox, and Schrage 1987, Problem 4.6.1.) Suppose we have a known theoretical distribution G over $[0, \infty)$ that we don't really like (e.g., because it is hard to generate random variables from G), and we want to approximate it by a piecewise linear distribution F, with an exponential tail, of the same form as in (2.22):

$$F(t) = \begin{cases} \dfrac{i}{n} + \dfrac{t - x_i}{(x_{i+1} - x_i)n} & \text{if } x_i \le t \le x_{i+1} \text{ and } 0 \le i \le n - k - 1, \\ 1 - \dfrac{k}{n} e^{-(t - x_{n-k})/\theta} & \text{if } t > x_{n-k}, \end{cases}$$

where $n > k > 0$ are integer parameters, while $0 = x_0 \le x_1 \le \cdots \le x_{n-k}$ and $\theta > 0$ are real numbers. We define $x_i = G^{-1}(i/n)$ for $i \le n - k$ and then try to define θ so that the mean of the quasi-empirical distribution F equals the mean of G.

(a) Give an expression for θ.

(b) The piecewise linear interpolation can shift the mean to the left or to the right. If it is shifted to the right, it may be impossible to find a $\theta > 0$ such that F and G have the

same mean. Give an example of a G for which this is impossible, for $n = 100$ and $k = 1$. Show that for any G and n, this is always possible by choosing k large enough (e.g., $k = n$). What is the smallest k that works in your example?

2.20 Suppose we want to estimate the expected value of the distance between the nearest pairs of observations when m i.i.d. observations are generated from a given cdf F with density f. That is, estimate $\mu = \mathbb{E}[D_m]$ where $D_m = \min_{1 \le i \le m-1}(X_{(i+1)} - X_{(i)})$. One can perform a simulation experiment that generates the X_j's, computes D_m, and repeats this independently n times to estimate μ.

(a) Do this with $m = 100$ and $n = 100$, assuming that F is the standard normal cdf. Compute a 95% confidence interval on μ.

(b) Now, generate 100 i.i.d. observations $X_1, \ldots, X_n$ from the standard normal distribution, compute the quasi-empirical distribution function $\widetilde{F}_n$ for these observations, and then pretend that you don't know the true cdf F of these observations. Repeat the same experiment as in (a), but using this time $\widetilde{F}_n$ instead of F. Compare and discuss the results, in the light of Example 2.10.

2.21 Suppose that a kernel density estimator is fitted to a data set of size n with sample mean $\bar{x}_n$ and sample variance s_n^2.

(a) Show that the distribution with density given in Eq. (2.26) always has variance larger than $\widetilde{s}_n^2 = s_n^2(n-1)/n$, regardless of k and h. Give an expression for that variance. Hint: The proof stands in two lines.

(b) Prove that if each observation x_i is replaced by $\widetilde{x}_i = \bar{x}_n + (x_i - \bar{x}_n)/\sigma_e$ in Eq. (2.26), where $1/\sigma_e^2 = 1 - (h\sigma_k/\widetilde{s}_n)^2$ and σ_k is the standard deviation of the density k, then the kernel density (2.26) has exactly the same mean and variance as the sample.

(c) If k is the normal kernel, $n = 50$, $\widetilde{s}_n^2 = 25$, and $h = 1.36374\alpha_k s_n n^{-1/5}$, what is the variance for the density (2.26) *without the correction*, and what is the value of σ_e?

2.22 Consider a system with a resource shared by two types of customers. (For example, the resource could be a link in a communication network.) For each customer type, the times (in minutes) between the successive demands for the resource are i.i.d. random variables having the $\mathsf{Gamma}(5000, 50)$ distribution (the same for both customer types). There are thus two independent arrival streams of demands. If the resource is free when a customer asks for it, the customer gets it and holds it for 1 minute. If it is already taken, this demand is lost and a cost has to be paid. We want to estimate the expected fraction p of demands that are lost, in the long run.

(a) Get a rough estimate of p by simulating the system for 10^8 minutes, and repeating this simulation three times.

(b) Suppose now that the distribution of the time between demands is unknown and replaced in the model by a quasi-empirical distribution estimated from some data. Generate 100 independent observations from the $\mathsf{Gamma}(5000, 50)$ distribution (this will act as data collected from the system) and construct the corresponding distribution function $\widetilde{F}_n$ as in Eq. (2.21). Then repeat the simulation as in (a), but by generating the times between demands from distribution $\widetilde{F}_n$ instead of from the original gamma distribution. Discuss your results.

(c) Repeat the same experiment as in (b), but using a kernel density estimator with normal kernel instead of $\widetilde{F}_n$.

(d) Can you figure out the *exact* value of p by mathematical arguments (without any simulation)?

2.23 Let (X_1, X_2) be a random vector. What is the range of possible values of $r_{12} = \rho(X_1, X_2)$ in the following cases?

(a) Both X_1 and X_2 are uniform over $(0, 1)$.

(b) $X_1 \sim N(0, 1)$ and $X_2 \sim N(0, 4)$.

(c) $X_1 \sim U(0, 1)$ and $X_2 \sim \mathsf{Exponential}(1)$.

(d) X_1 and X_2 are both lognormal, with parameters $(\mu_1, \sigma_1^2) = (0, 1)$ and $(\mu_2, \sigma_2^2) = (0, \sigma^2)$, respectively. Give the range of values as a function of σ^2, then evaluate the bounds for $\sigma^2 = 4$ and $\sigma^2 = 100$. What happens to $\rho(X_1, X_2)$ and to $\rho_{\mathrm{s}}(X_1, X_2)$ when $\sigma^2 \to \infty$?

2.24 Prove that if (X_1, X_2) is a random vector such that X_1 has a symmetric density f with respect to the origin and $X_2 = \varphi(X_1)$ where φ is a function that satisfies $\varphi(-x) = \varphi(x)$, then $\rho(X_1, X_2) = 0$. Hint: show that $\int_{-\infty}^{0} x\varphi(x)f(x)dx = -\int_{0}^{\infty} x\varphi(x)f(x)dx$.

2.25 Let $\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in d dimensions. Partition the (column) vector $\boldsymbol{X}$ as $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2)^{\mathrm{t}}$, where $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ have d_1 and d_2 dimensions, respectively. Let

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix} \text{ and } \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}$$

be the corresponding partitions of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Suppose that you have generated $\boldsymbol{X}_1$ from the $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$ distribution and you now want to generate $\boldsymbol{X}_2$ with the correct distribution *conditional* on the value of $\boldsymbol{X}_1$. What is the exact distribution of $\boldsymbol{X}_2$ conditional on $\boldsymbol{X}_1 = \boldsymbol{x}_1$?

2.26 (a) Show that if X and Y are normally distributed, then

$$\rho(X, Y) = 2\sin(\rho_{\mathrm{s}}(X, Y)\pi/6) = \sin(\tau_{\mathrm{k}}(X, Y)\pi/2).$$

(b) Show that the absolute difference $|\rho(X, Y) - \rho_{\mathrm{s}}(X, Y)|$ reaches its maximum at $\rho_{\mathrm{s}}(X, Y) \approx \pm 0.576$, and that the value of this maximum difference is approximately 0.018. (This means that in the case of the normal distribution, using $\rho(X, Y)$ rather than $\rho_{\mathrm{s}}(X, Y)$, or vice-versa, does not make much of a difference.)

2.27 Give an example of a random vector $\boldsymbol{X}$ whose marginals are all standard normal but where $\boldsymbol{X}$ is not multinormal. Hint: you may consider $\boldsymbol{X} = (X_1, X_2) = (\Phi^{-1}(U_1), \Phi^{-1}(U_2))$ where (U_1, U_2) is a bivariate uniform vector generated from the two-dimensional copula considered in Example 2.16. Prove that it satisfies the conditions.

2.28 Show that if $\boldsymbol{X}$ has an elliptic distribution and $\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{X} + \boldsymbol{b}$ for some (constant) matrix $\boldsymbol{B}$ and vector $\boldsymbol{b}$, then $\boldsymbol{Y}$ also has an elliptic distribution.

2.29 Let $F : \mathbb{R}^d \to [0, 1]$ be a d-dimensional cdf and let B be a rectangular box in $\mathbb{R}^d$, with opposite corners $\boldsymbol{a} = (a_1, \ldots, a_d)$ and $\boldsymbol{b} = (b_1, \ldots, b_d)$, where $a_j < b_j$ for all j. The cdf F

determines the probability of box B, say $\mathbb{P}[B]$. Show that $\mathbb{P}[B]$ can be written as a sum of 2^d terms of the form $\pm F(\boldsymbol{x})$ where $\boldsymbol{x} = (x_1, \ldots, x_d)$ is at one corner of the box for each term, and the sign is positive if $x_j = a_j$ for an even number of coordinates and negative otherwise. Therefore, for F to be a cdf, this sum must be non-negative for any B.

2.30 For each of the following functions C, find if it is a copula or not and prove it. (Hint: Condition (iii) in the definition of a copula can often be verified by finding the corresponding density and checking that it is never negative.) For those that are copulas, describe the corresponding multivariate distribution over the unit hypercube and explain how to generate a random vector from that copula.

(a) $C(u_1, \ldots, u_d) = u_1 \cdots u_d$ (the product).

(b) $C(u_1, \ldots, u_d) = \min(u_1, \ldots, u_d)$.

(c) $C(u_1, u_2) = \max(u_1 + u_2 - 1)$, for $d = 2$.

(d) $C(u_1, \ldots, u_d) = \max(u_1 + \cdots + u_d - d + 1)$, for $d > 2$.

(e) $C(u_1, u_2) = (u_1^{-\delta} + u_2^{-\delta} - 1)^{-1/\delta}$ for $\delta \geq 0$. Hint: for random variate generation, you can find and use the conditional distribution of X_2 given X_1.

2.31 Let $(X_1, \ldots, X_d)$ be a continuous random vector with copula C and let $g_j : \mathbb{R} \to \mathbb{R}$ be a strictly increasing function for $j = 1, \ldots, d$. Show that in this case, the random vector $(g_1(X_1), \ldots, g_d(X_d))$ also has copula C.

2.32 Consider a d-dimensional cdf F with associated copula C, with joint densities f and c, respectively, and where each marginal F_j has density f_j. Show that the density f can be written as

$$\begin{aligned} f(x_1, \ldots, x_d) &= c(F_1(x_1), \ldots, F_d(x_d)) f_1(x_1) \cdots f_d(x_d) \\ &= c(u_1, \ldots, u_d) f_1(F_1^{-1}(u_1)) \cdots f_d(F_d^{-1}(u_d)) \end{aligned}$$

for $(u_1, \ldots, u_d) \in (0, 1)^d$. Hint: Consider differentiating (2.38).

2.33 Prove the expressions for the Spearman's rho and Kendall's tau given in Example 2.19 for $d = 2$.

2.34 (Adapted from Hörmann, Leydold, and Derflinger 2004).

Suppose we want a pair (U_1, U_2) of $U(0, 1)$ random variables with given rank correlation ρ_s.

(a) Explain how to achieve this (i) with the NORTA method and (ii) with a Fréchet copula, as defined in Example 2.14. In each case, give a formula for the parameter value that provides the desired rank correlation.

(b) For $\rho_s = 0.9$, estimate $\mathbb{P}[U_1 + U_2 < 0.1]$ by simulating 10^6 independent pairs (U_1, U_2) from the corresponding distribution, for each of these two copulas.

(c) Do the same with $\rho_s = 0.8$ and $\mathbb{P}[U_1 - U_2 < 0.05]$.

2.35 Consider the two-dimensional Gumbel copula with parameter $\delta \geq 1$, defined in Eq. (2.42).

(a) Find the values of δ (if they exist) for which the Kendall tau $\tau_{\mathrm{k}}(X, Y)$ that corresponds to this copula takes the four values 0, 0.5, 0.8, and 1.0.

(b) Suppose we want to generate vectors (X, Y) with $\tau_{\mathrm{k}}(X, Y) = 0.8$ and marginals that are both exponential with mean 1. Explain in detail how to do this with (a) a normal copula (the NORTA method) and (b) a Gumbel copula. Write a program that generates (X, Y) for each case.

♣ Try also a Student copula. Can take $d > 2$.

(c) Estimate $\mathbb{P}[X > 3, Y > 3]$ by simulation for each of the two copulas mentioned in (b). Use a sample size n large enough to obtain an estimator with less than 5% relative error in each of the two cases.

2.36 Show that the coefficient of upper tail dependence λ_{u} is $\lambda_{\mathrm{u}} = 0$ for a normal copula and $\lambda_{\mathrm{u}} = 2 - 2^{1/\delta} \geq 0$ for a Gumbel copula with parameter $\delta \geq 1$.

2.37 Show that if $\boldsymbol{R}^{(\mathrm{y})}$ is a valid correlation matrix for a multinormal vector, then for any marginal distributions $F_1, \ldots, F_d$, the matrix $\boldsymbol{R}^{(\mathrm{x})}$ with elements defined by (2.46) is a valid correlation matrix for a random vector $\boldsymbol{X}$ with marginal distributions $F_1, \ldots, F_d$.

2.38 Show that for any given pair of marginals (F_i, F_j) with mean 0 and variance 1, φ_{ij} defined in Section 2.10.9 has the following properties:

(a) Its minimum and maximum are $\underline{r}_{ij} = \varphi_{ij}(-1)$ and $\overline{r}_{ij} = \varphi_{ij}(1)$, respectively, and $\varphi_{ij}(0) = 0$;

(b) It is a nondecreasing and infinitely differentiable function. (Cario and Nelson (1997) prove this under additional conditions, but J. R. Wilson later found an alternative proof without these conditions.)

2.39 Suppose that arrivals occur according to a Poisson process, and that each arrival is of type j with probability p_j, for $1 \leq j \leq J$, independently of the others. How can you simulate efficiently the process consisting of only the arrivals of type 1?

2.40 Consider a nonstationary Poisson process with rate function $\lambda(t)$, $t \geq 0$. If an arrival occurs at time s, show that it is *incorrect* to generate the time to the next arrival by generating an exponential with mean $1/\lambda(s)$. (Give a specific counterexample.)

2.41 Show that for a d-dimensional BM with covariance matrix $\boldsymbol{\Sigma}$ with elements $\sigma_{i,j}$, for $s, t \geq 0$, one has $\mathrm{Cov}[X_i(s), X_j(t)] = \min(s, t)\sigma_{i,j}$.

2.42 Show that for a GBM(μ, σ^2) process, if $\mu - \sigma^2/2 < o < \mu$, then when $t \to \infty$, we have $S(t) \overset{\text{w.p.1}}{\to} \infty$ but $\mathbb{E}[S(t)] \to \infty$. Why is this possible? Hint: You can use the fact that a standard BM satisfies $B(t)/t \overset{\text{w.p.1}}{\to} 0$.

2.43 Write a simulation program that implements the Monte Carlo method described in Example 1.11 to estimate the value of an Asian option on a single asset whose underlying value follows a GBM. The payoff is given by Eq. (1.10). Consider an option with the following

parameters: $\sigma = 0.3$, $r = 0.05$, $K = 55$, $S(0) = 50$, $T = 1$, $d = 64$, and $\zeta_j = j/64$ for $j = 1, \ldots, 64$.

(a) Use your program to estimate the value of the option by a standard Monte Carlo method, with $n = 10^6$ replications. Compute a 95% confidence interval for this value.

(b) Do the same thing, but by using the Brownian bridge methodology described in Section 2.14.4 to generate the trajectory of the underlying BM. Compare your results.

2.44 Show that for the minification, maxification, and TES methods, defined in Eqs. (2.68), (2.69), and (2.70), U_n is $U(0,1)$ for each n.

2.45 You want to estimate the vector of parameters (μ, σ^2) of a lognormal distribution from an i.i.d. sample of size n, $x_1, \ldots, x_n$.

(a) What is the maximum likelihood estimator for this vector?

(b) What is the moment matching estimator (that matches the first two moments)? Compare these two estimators and discuss.

2.46 Suppose that $X_1, \ldots, X_n$ are an independent sample from the $\mathsf{InvGaussian}(\mu, \lambda)$ distribution, with unknown parameters (μ, λ).

(a) Write the corresponding log-likelihood function and show that the maximum likelihood estimator for (μ, λ) is $(\hat{\mu}_n, \hat{\lambda}_n)$ where $\hat{\mu}_n = \bar{X}_n$ and $n/\hat{\lambda}_n = \sum_{i=1}^{n}(1/X_i - 1/\bar{X}_n)$.

(b) Show that $\hat{\mu}_n$ and $\hat{\lambda}_n$ are independent, that $\hat{\mu}_n \sim \mathsf{InvGaussian}(\mu, n\lambda)$, and $n\lambda/\hat{\lambda}_n \sim \chi^2(n-1)$. Explain how this can be used to compute confidence intervals on μ and λ.

2.47 Consider a single-server queue where the events are the arrivals, ends of service, and end of the simulation at a fixed time T, as in Section 1.11 and Example 2.23. You are asked to express this model by a stochastic recursion of the form [17] . How would you define ω_i, $\mathcal{Q}$, and $\mathcal{T}$, if $\mathcal{S}_i$ is defined as in Example 2.23? In the case of an $M/M/1$ queue, how can $\mathcal{S}_i$ be simplified? What would be ω_i, $\mathcal{Q}$, and $\mathcal{T}$ in that case?

2.48 In Example 2.38, estimate the relative difference $(g(s) - \mathbb{E}[X/A])/g(s)$ for $s = 20$, by simulating $n = 1000$ (independent) days. *Do not* perform separate simulations to estimate the two terms $g(s)$ and $\mathbb{E}[X/A]$ in the difference; i.e., simulate n days, not $2n$ days. Repeat this experiment 10 times, independently, and compute a confidence interval on the difference. What is your conclusion? What type of change in the distribution of A should make the difference larger, if $\mathbb{E}[X]$ and $\mathbb{E}[A]$ remain constant?

2.49 Consider the total discounted cost V_ρ^∞ over an infinite-horizon, defined in (2.83).

(a) Show that in general, v_ρ^∞ can be infinite even if the C_i's are all positive and bounded by a finite constant K.

(b) Show that if $|C_i| \le K$ for all i and $\mathbb{E}[N(t)] \le tL$ for all $t \ge 0$, for some finite constants K an L, then $|v_\rho^\infty| < \infty$. Give an explicit upper bound.

[17] From Pierre: Not yet defined...

(c) If the costs are accumulated at a state-dependent *rate* $c(\mathcal{S}(\cdot))$ (as a function of time) and if the function $|c(\cdot)|$ is bounded by a constant K, show that $|v_\rho^\infty| < \infty$ and give an explicit upper bound.

3. Uniform Random Number Generation

[1] This chapter elaborates on the concept of uniform random number generator (RNG) introduced in Section 1.3. For simulation, the RNG should reproduce the statistical properties on which the analysis of simulation models and results is based. That is, it should provide a good approximation of the correct distribution for the output random variables of interest, so that the user finds no significant difference with the expected behavior of the stochastic model even when the sample size is large. For other types of applications such as cryptology and gambling machines, where an opponent might be actively trying to crack the generator, the requirements are stronger and include some form of "unpredictability": there should be no practical way of predicting the forthcoming numbers better than at random (L'Ecuyer and Proulx 1989, Lagarias 1993, Luby 1996). Most RNGs discussed here are based on linear recurrences and do not satisfy the latter requirement. However, they are faster, easier to implement, easier to analyze, and easier to split into substreams, than the cryptographic generators. The good ones are also reliable enough for almost all practically relevant simulation problems.

The remainder of this chapter is organized as follows. In the next section, we give a definition and the main requirements of a uniform RNG. Generators based on linear recurrences modulo a large integer m, their lattice structure and quality criteria, and their implementation, are covered in Section 3.2. In Section 3.3, we have a similar discussion for RNGs based on linear recurrences modulo 2. Nonlinear RNGs are briefly presented in Section 3.4. In Section 3.5, we discuss empirical statistical testing of RNGs and give some examples. Section 3.6 contains a few pointers to recommended RNGs and software. Other basic references include Knuth (1998), L'Ecuyer (1994b), Niederreiter (1992), and Tezuka (1995).

3.1 Major Issues, Definitions, and Requirements

3.1.1 Why not just use a physical device?

Non-experts in simulation often ask this question. High-quality random numbers *can* indeed be produced by physical mechanisms such as gamma ray counters, fast oscillators sampled at low and slightly random frequencies, amplifiers of thermal noise in semiconductors, photon counters, photon trajectory detectors, and so on (Stefanov et al. 2000, Suematsu et al. 2007). Some of these devices sample a signal at successive epochs; each sampling returns 0 if the signal is below a given threshold, and 1 if it is above the threshold. Some devices return the

[1] From Pierre: This chapter needs a lot of refreshing!

parity of a counter. A key issue when constructing such a physical RNG is that a "random" or "chaotic" output does not suffice; the output values must also approximate realizations of *independent and uniformly distributed* random variables. If the device generates a stream of bits, for example, then each bit should be 0 or 1 with equal probability and should be independent of all the other bits.

Almost all physical devices produce sequences of bits that are slightly correlated and often slightly biased, but the bias and correlation can be reduced to a level that is practically undetectable by statistical tests, by combining the bits in a clever way. For example, a simple technique proposed by von Neumann to eliminate the bias, when there is no correlation, places the successive bits in non-overlapping pairs, discards all the pairs 00 and 11, and replaces the pairs 01 and 10 by 1 and 0, respectively. Generalizations of this technique can eliminate both the bias and correlation (Blum 1986, Chor and Goldreich 1988). Simpler methods such as xoring (adding modulo 2) the bits by blocks of 2 or more, or xoring several bit streams from different sources, are often used in practice. Several reliable devices to generate random bits and number, using these techniques, are available on the market. These types of devices are needed for applications such as cryptography, for example, where some amount of real randomness (or entropy) is essential to provide the required unpredictability and security.

For simulation, physical devices have several *disadvantages* compared to a good algorithmic RNG: (a) they are more cumbersome to install and run; (b) they are more costly; (c) they are slower; (d) they cannot reproduce exactly the same sequence twice. Item (d) is important, for example, for program verification, debugging, and comparison of similar systems with common random numbers to reduce the variance (see Section 6.4). Of course, the sequence can be made reproducible by storing all output values in memory, but this is inconvenient and not very efficient. Physical RNGs are nevertheless useful for selecting the seed of an algorithmic RNG, more particularly for applications in cryptology and for gaming machines, where frequent reseeding of the RNG with an external source of entropy (or randomness) is important. In a poker or keno gambling machine, for example, it is common to have an algorithmic RNG running at full speed in the background, providing an output stream from which a few numbers are taken when needed (e.g., when the player hits some key on the machine's keyboard). The RNG can also be reseeded every few minutes or seconds by modifying its state with random bits coming from a physical source of entropy.

3.1.2 Generators Based on a Deterministic Recurrence

An *algorithmic RNG* can be defined by a mathematical structure $(\mathcal{S}, Q, f, \mathcal{U}, g)$ where $\mathcal{S}$ is a finite set of *states* (the *state space*), Q is a probability distribution on $\mathcal{S}$ used to select the *initial state* (or *seed*) s_0, $f : \mathcal{S} \to \mathcal{S}$ is the *transition function*, $\mathcal{U}$ is the *output space*, and $g : \mathcal{S} \to \mathcal{U}$ is the *output function*. In what follows, unless stated otherwise, we assume that $\mathcal{U} = (0, 1)$. The initial state s_0 is s with probability $Q(s)$ for each $s \in \mathcal{S}$. Then the state evolves according to the deterministic recurrence $s_n = f(s_{n-1})$, for $n \geq 1$, and the *output* at step n is $u_n = g(s_n) \in \mathcal{U}$. The output values $u_0, u_1, u_2, \ldots$ are the so-called *random numbers* produced by the RNG.

Frequently, the initial state s_0 is just fixed arbitrarily in $\mathcal{S}$ instead of being selected at random, because this is simpler. For example, a software that provides an RNG whose state

is a pair of 32-bit integers may use $s_0 = (12345, 12345)$ as *default seed*. In that case, every program that uses this generator will receive exactly the same sequence of random numbers, unless the seed (or state) is changed explicitly by the program.

Because $\mathcal{S}$ is finite, there are necessarily finite integers $l \geq 0$ and $j > 0$ such that $s_{l+j} = s_l$. Then, for all $n \geq l$, one has $s_{n+j} = s_n$ and $u_{n+j} = u_n$. That is, the state and output sequences are eventually periodic. The smallest positive j for which this happens is called the *period* of the RNG, and is denoted by ρ. When $l = 0$, the sequence is said to be *purely periodic*. Otherwise, it has a *transient part* of length l. Obviously, ρ cannot exceed $|\mathcal{S}|$, the cardinality of $\mathcal{S}$. If the state has a k-bit representation on the computer, then $\rho \leq 2^k$. Good RNGs are designed so that their period ρ is not far from that upper bound. In general, it could happen that ρ depends on the seed s_0, but for well-designed RNGs, the period is the same for all admissible seeds.

In practical implementations, it is important that the output be *strictly* between 0 and 1, because $F^{-1}(U)$ is often infinite when U is 0 or 1. Good implementations take care of that. For the mathematical analysis of the uniformity RNGs, on the other hand, we often assume that the output space is $[0, 1)$ (i.e., 0 is admissible), because this simplifies the analysis considerably without making much difference in the mathematical structure of the generator.

3.1.3 Quality Criteria

As argued in Section 1.3, a *long period* and *high uniformity* of the sets Ψ_s and Ψ_I are important requirements for a good RNG. These properties must be guaranteed by mathematical proofs. For an arbitrary set of non-negative integers $I = \{i_1, \ldots, i_s\}$, Ψ_I is formally defined by

$$\Psi_I = \{((u_{i_1}, \ldots, u_{i_s}) = (g(s_{i_1}), \ldots, g(s_{i_s})) : s_0 \in \mathcal{S}\}$$

and Ψ_s is the corresponding set when $I = \{0, \ldots, s-1\}$. This is the set of all vectors of s successive output values that can be produced by the generator. (Here s is used to denote the dimension and s_i is the state at step i; one should pay a little attention not to confuse them.)

The RNG must also be *efficient* (run fast and use a small amount of memory), *repeatable* (able to reproduce exactly the same sequence as many times as we want), and *portable* (work the same way in different software/hardware environments). The availability of efficient *jump-ahead* methods that can quickly compute $s_{n+\nu}$ given s_n, for any large ν and any n, permits one to partition the RNG sequence into long disjoint *streams* and *substreams* of random numbers, to create an arbitrary number of *virtual generators* from a single RNG. These virtual generators can be used on parallel processors or to support different sources of randomness in a large simulation model, for example (see Sections 1.3 and 1.7).

To show that a very long period does not suffice, consider an RNG with state space $\mathcal{S} = \{0, \ldots, 2^{1000} - 1\}$, transition function $s_{n+1} = f(s_n) = (s_n + 1) \bmod 2^{1000}$, and output $u_n = g(s_n) = s_n / 2^{1000}$. This RNG has period 2^{1000} and is easy to implement efficiently, but is far from imitating randomness.

If we select the seed s_0 at random, uniformly over $\mathcal{S}$, e.g., by using a physical device, then the vector $\boldsymbol{u}_{0,s} = (u_0, \ldots, u_{0+s-1})$ has the uniform distribution over Ψ_s. More generally,

if $I = \{i_1, \ldots, i_s\}$, then $\boldsymbol{u}_I = (u_{i_1}, \ldots, u_{i_s})$ is uniformly distributed over Ψ_I. This uniform distribution over Ψ_I acts as an approximation of the uniform distribution over $[0,1)^s$. For the approximation to be good, Ψ_I must be uniformly spread over this unit hypercube.

The design of good-quality RNGs must therefore involve practical ways of measuring the uniformity of the sets Ψ_I of interest, even when they have huge cardinalities. In fact, a large state space $\mathcal{S}$ is necessary to obtain a long period, but an even more important reason for having a huge number of states is to make sure that Ψ_I can be large enough to provide a good uniform coverage of the unit hypercube, at least for the sets I deemed important.

Measures of uniformity of the sets Ψ_I are equivalent to goodness-of-fit test statistics for the multivariate uniform distribution. The choice of a specific measure typically depends on the mathematical structure of the RNG to be studied, because we must be able to compute it quickly even when the cardinality of $\mathcal{S}$ is very large. This obviously excludes any method that requires explicit generation of the sequence over its entire period. The selected measure is usually computed for each set I in a predefined class $\mathcal{J}$. These values are then weighted or normalized by factors that depend on I, and the worst-case (or average) over $\mathcal{J}$ is adopted as a *figure of merit* for ranking RNGs. The choice of $\mathcal{J}$ and of the weights are arbitrary. Typically, $\mathcal{J}$ would contain sets I such that s and $i_s - i_1$ are small. Examples will be given in forthcoming sections.

Algorithmic RNGs discussed in this chapter are purely periodic. For a purely periodic sequence, the transition function f has an inverse, so one can go backwards in the sequence and define the states $s_{-1}, s_{-2}, \ldots$ and the output values $u_{-1}, u_{-2}, \ldots$ in the obvious way. The s_n's and u_n's are then defined for all integer indices n, even the negative ones, and the sets Ψ_s and Ψ_I have the following useful property:

Proposition 3.1 *If the RNG is purely periodic, for any set of integer indices $I = \{i_1, \ldots, i_s\}$, the set Ψ_I can be defined as*

$$\Psi_I = \{(u_{i_1+\tau}, \ldots, u_{i_s+\tau}) : s_\nu \in \mathcal{S}\} \tag{3.1}$$

for any fixed integers τ and ν. That is, this set does not depend on τ and ν.

Proof. To any given state $s_\nu \in \mathcal{S}$, if s_ν is in a cycle of length ρ, there corresponds to s_ν a doubly infinite sequence of states $\ldots, s_{-1}, s_0, s_1, s_2, \ldots$. This sequence is periodic with period ρ. Let $s_* = s_{\nu-\tau}$ in this sequence. The output vector $(u_{i_1+\tau}, \ldots, u_{i_s+\tau})$ that corresponds to this sequence is the same as the vector $(u_{i_1}, \ldots, u_{i_s})$ obtained when $s_0 = s_*$. But when s_ν goes through all the states of the cycle, $s_{\nu-\tau}$ visits exactly the same set of states (in a different order). This holds for every cycle of the recurrence. Therefore, the set Ψ_I defined above is the same regardless of τ and ν. Since Ψ_s is a special case of Ψ_I, the result also applies to Ψ_s.

Example 3.1 Consider the tiny generator defined by the recurrence $x_n = 3x_{n-1} \bmod 7$ and $u_n = x_n/7$, where $\mathcal{S} = \mathbb{Z}_7 = \{0, \ldots, 6\}$. With $x_0 = 1$, the output sequence is 1/7, 3/7, 2/7, 6/7, 4/7, 5/7, 1/7, 3/7, etc.

Let $I = \{0, 2\}$, $\tau = 0$, and $\nu = 0$ in (3.1). Then when $s_\nu = s_0$ goes from 0 to 6 in this order, the vector $(u_{i_1+\tau}, u_{i_2+\tau}) = (u_0, u_2)$ takes the values $(0,0)$, $(1/7, 2/7)$, $(2/7, 4/7)$, $(3/7, 6/7)$, $(4/7, 1/7)$, $(5/7, 3/7)$, $(6/7, 5/7)$. These seven vectors form the set Ψ_I in this case.

If we take $\tau = 3$ and $\nu = 1$ instead, when s_1 goes from 0 to 6, the vector $(u_{i_1+\tau}, u_{i_2+\tau}) = (u_3, u_5)$ takes the values $(0,0)$, $(2/7, 4/7)$, $(4/7, 1/7)$, $(6/7, 5/7)$, $(1/7, 2/7)$, $(3/7, 6/7)$, $(5/7, 3/7)$. This is the same set Ψ_I enumerated in a different order. Any other integer values of τ and ν also give the same set. □

3.1.4 Statistical Testing

After an RNG has been designed, based presumably on a sound mathematical analysis of its properties, it can be implemented in a computer program and then submitted to batteries of *empirical statistical tests*. These tests try to detect empirical evidence against the null hypothesis $\mathcal{H}_0$: "the u_n are realizations of i.i.d. $U(0,1)$ random variables." A test can be defined by any function that maps a sequence $u_0, u_1, \ldots$ in $(0,1)$ to a real number Y, and for which a good approximation is available for the distribution of the random variable Y under $\mathcal{H}_0$. For the test to be practical, Y must depend on only a finite and relatively small (but perhaps random) number of u_n's. Passing many tests may improve one's confidence in the RNG, but never guarantees that the RNG is foolproof for all kinds of simulations.

Building a RNG that *passes all statistical tests* is an impossible dream. Consider, for example, the class of all tests that examine the first (most significant) b bits of n successive output values, $u_0, \ldots, u_{n-1}$, and return a binary value $Y \in \{0,1\}$. There are 2^{nb} possibilities for those n bits. Select $\alpha \in (0,1)$ so that $\alpha 2^{nb}$ is an integer and let $\mathcal{T}_{n,b,\alpha}$ be the tests in this class that return $Y = 1$ for *exactly* $\alpha 2^{nb}$ of the 2^{nb} possible output sequences. We may say that the sequence *fails* the test when $Y = 1$. The number of tests in $\mathcal{T}_{n,b,\alpha}$ is equal to the number of ways of choosing $\alpha 2^{nb}$ distinct objects among 2^{nb}. The chosen objects are the sequences that fail the test. Now, for any given output sequence, the number of such tests that return 1 for this particular sequence is equal to the number of ways of choosing the other $\alpha 2^{nb} - 1$ sequences that also fail the test. This is the number of ways of choosing $\alpha 2^{nb} - 1$ distinct objects among $2^{nb} - 1$. In other words, as pointed out by Leeb (1995), every output sequence fails exactly the same number of tests! Viewed from a different angle, this result is just a disguise of the well-known fact that under $\mathcal{H}_0$, each of the 2^{nb} possible sequences has the same probability of occurring, so one may argue that none should be considered more random than any other (Knuth 1998).

For statistical testing to be meaningful, all tests should not be considered on equal footing. Which ones are more important? Any answer is certainly tainted with its share of arbitrariness. For large values of n, the number of tests is huge and all but a tiny fraction are too complicated even to be implemented. So we may say that *bad* RNGs are those that fail simple tests, whereas *good* RNGs fail only complicated tests that are very difficult to find and run. This common-sense compromise has been generally adopted in one way or another.

Experience shows that RNGs with very long periods, good structure of their set Ψ_I, and based on recurrences that are not too simplistic, pass most reasonable statistical tests, whereas RNGs with short periods or bad structures are usually easy to crack by standard tests. For sensitive applications, it is a good idea (when this is possible) to apply additional specialized statistical tests designed in close relationship with the random variable of interest (e.g., based on a *simplification* of the stochastic model being simulated, and for which the theoretical distribution can be computed). Specific statistical tests are examined in Section 3.5.

3.2 Linear Recurrences Modulo m

3.2.1 The Multiple Recursive Generator

A large class of RNGs are based on the linear recurrence

$$x_n \;=\; (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \bmod m, \tag{3.2}$$

where $k \geq 1$ is the *order*, $m \geq 2$ is the *modulus*, the *coefficients* $a_1, \ldots, a_k$ are in the finite ring $\mathbb{Z}_m = \{0, \ldots, m-1\}$ in which all operations are performed with reduction modulo m, and $a_k \neq 0$. The latter condition ensures that the sequence is always purely periodic, i.e., has no transient state (Lidl and Niederreiter 1986, Theorem 6.11). The *state* at step n is $s_n = \boldsymbol{x}_n = (x_{n-k+1}, \ldots, x_n)^\mathsf{t}$ and the state space $\mathbb{Z}_m^k$ has cardinality m^k. A sequence that satisfies Eq. (3.2) is a *linear recurring sequence* (LRS) with *characteristic polynomial*

$$P(z) = z^k - a_1 z^{k-1} - \cdots - a_k = -\sum_{j=0}^{k} a_j z^{k-j}, \tag{3.3}$$

where $a_0 = -1$. The recurrence (3.2) can also be written in matrix form as

$$\boldsymbol{x}_n = \boldsymbol{A}\boldsymbol{x}_{n-1} \bmod m = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ a_k & a_{k-1} & \cdots & a_1 \end{pmatrix} \boldsymbol{x}_{n-1} \bmod m, \tag{3.4}$$

and $P(z)$ is the characteristic polynomial of the matrix $\boldsymbol{A}$.

A *multiple recursive generator* (MRG) [2] uses (3.2) with a large value of m and defines the output as

$$u_n \;=\; x_n/m. \tag{3.5}$$

The output space is then the finite set $\mathcal{U} = \mathbb{Z}_m/m = \{0, 1/m, \ldots, (m-1)/m\}$, which can be a "good approximation" of the real interval $[0, 1]$ only if m is very large. For $k = 1$, this is the classical *linear congruential generator* (LCG), whose recurrence is defined by

$$x_n \;=\; a x_{n-1} \bmod m. \tag{3.6}$$

In practice, the output function is modified slightly to make sure that u_n never takes the value 0 or 1 (e.g., one may define $u_n = (x_n + 1)/(m + 1)$, or $u_n = x_n/(m + 1)$ if $x_n > 0$ and $u_n = m/(m + 1)$ otherwise) but to simplify the theoretical analysis, we will follow the common convention of assuming that $u_n = x_n/m$ (in which case u_n *does* take the value 0 occasionally). The modification does not change significantly the structure of the point sets Ψ_I.

[2] From Pierre: This is the standard name, but I think *linear congruential generator of order k* would be a better name.

3.2.2 Alternative representations

To study LRSs and MRGs, it is convenient to introduce alternative equivalent representations of the recurrence (3.2). We define one-to-one mappings between the following three spaces; either of them can be interpreted as the state space of the generator:

(i) the space $\mathbb{Z}_m^k$ of k-dimensional *vectors* with coordinates in $\mathbb{Z}_m$ (the state $\boldsymbol{x}_n$ defined earlier and the recurrence (3.4) are in this representation).

(ii) the space $\mathbb{Z}_m[z]/(P)$ of *polynomials* of degree less than k with coefficients in $\mathbb{Z}_m$, which can be interpreted as the space of polynomials reduced modulo $P(z)$, and

(iii) the space $\mathcal{L}(P)$ of *formal Laurent series* of the form $\tilde{s}(z) = \sum_{j=1}^{\infty} c_j z^{-j}$, where the coefficients c_j are in $\mathbb{Z}_m$ and obey the recurrence (3.2), i.e., $c_j = (a_1 c_{j-1} + \cdots + a_k c_{j-k}) \bmod m$ for all $j > k$.

The series $\tilde{s}(z)$ should be viewed as just a way of representing the infinite sequence $\{c_n,\, n \geq 1\}$. These three spaces of cardinality m^k offer three different representations of the generator's state. Each representation has its advantages, depending on what we want to do. For example, the polynomial representation turns out to be convenient for jumping ahead in the sequence and for checking maximal period conditions.

The mappings are defined as follows. To the *vector* representation $\boldsymbol{x}_n = (x_{n-k+1}, \ldots, x_n)^{\mathsf{t}}$ of the state, we associate the *formal series*

$$\tilde{s}_n(z) = \sum_{j=1}^{\infty} x_{n-k+j} z^{-j} \tag{3.7}$$

where $x_{n+1}, x_{n+2}, \ldots$ are uniquely determined by the recurrence (3.2). This series is the *generating function* of the sequence $\{x_{n-k+j},\, j \geq 1\}$. We also associate to $\boldsymbol{x}_n$ the *polynomial*

$$p_n(z) = P(z)\tilde{s}_n(z) \bmod m, \tag{3.8}$$

where the product is meant as a product of formal series (of which polynomials are a special case) and the operations on the coefficients are performed in $\mathbb{Z}_m$. The next proposition states that these mappings define bijections between $\mathbb{Z}_m^k$, $\mathcal{L}(P)$, and $\mathbb{Z}_m[z]/(P)$. It also shows that $p_n(z)$ is effectively a polynomial, and gives an easy way to compute the coefficients of $p_n(z)$ given $\boldsymbol{x}_n$ and vice-versa. For the proof, see Exercise 3.2.

Proposition 3.2 *The mappings $\boldsymbol{x}_n \to \tilde{s}_n(z) \to p_n(z)$ defined above are one-to-one mappings between $\mathbb{Z}_m^k$, $\mathcal{L}(P)$, and $\mathbb{Z}_m[z]/(P)$. Moreover, the corresponding mapping $\boldsymbol{x}_n \to p_n(z)$ satisfies*

$$p_n(z) = \sum_{j=1}^{k} c_{n,j} z^{k-j} \tag{3.9}$$

where

$$\begin{pmatrix} c_{n,1} \\ c_{n,2} \\ \vdots \\ c_{n,k} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ -a_1 & 1 & \ldots & 0 \\ \vdots & & \ddots & \vdots \\ -a_{k-1} & \ldots & -a_1 & 1 \end{pmatrix} \begin{pmatrix} x_{n-k+1} \\ x_{n-k+2} \\ \vdots \\ x_n \end{pmatrix} \bmod m. \tag{3.10}$$

By multiplying the generating function $\tilde{s}_{n-1}(z)$ by z, we obtain

$$z\tilde{s}_{n-1}(z) = z\sum_{j=1}^{\infty} x_{n-k-1+j}z^{-j} = \sum_{j=0}^{\infty} x_{n-k+j}z^{-j} = x_{n-k} + \tilde{s}_n(z).$$

This can be written as

$$\tilde{s}_n(z) = z\tilde{s}_{n-1}(z) \bmod 1, \tag{3.11}$$

where "mod 1" means that we remove the polynomial part and keep only the terms with negative powers of z. Multiplying this equation by $P(z)$ gives

$$p_n(z) = zp_{n-1}(z) \bmod P(z), \tag{3.12}$$

where "mod $P(z)$" means that $p_n(z)$ is the remainder of the division of the polynomial $zp_{n-1}(z)$ by the polynomial $P(z)$, with the operations on the coefficients performed in $\mathbb{Z}_m$. The recurrences (3.11) and (3.12) show that an MRG can be seen as an LCG in a space of formal series, with multiplier z and modulus 1, or in a space of polynomials, with multiplier z and modulus $P(z)$. This last representation is convenient for studying the periodicity of the LRS in (3.2).

Example 3.2 Consider a small MRG with $m = 101$, $k = 3$, and $(a_1, a_2, a_3) = (28, 49, 8)$, so $P(z) = z^3 - 28z^2 - 49z - 8$. Suppose the generator starts in state $\boldsymbol{x}_0 = (12, 12, 12)^\mathsf{t}$. From the recurrence, we can compute $\boldsymbol{x}_1 = (12, 12, 10)^\mathsf{t}$, $\boldsymbol{x}_2 = (12, 10, 55)^\mathsf{t}$, and so on. Using (3.10), we find $p_0(z) = 12z^2 + 80z + 98$, $p_1(z) = 12z^2 + 80z + 96$, $p_2(z) = 12z^2 + 78z + 96$, and so on. One can easily verify (3.12) with these polynomials. □

3.2.3 Jumping Ahead

To jump ahead directly from $\boldsymbol{x}_n$ to $\boldsymbol{x}_{n+\nu}$, for an arbitrary integer ν, it suffices to exploit the relationship

$$\boldsymbol{x}_{n+\nu} = \boldsymbol{A}^\nu \boldsymbol{x}_n \bmod m = (\boldsymbol{A}^\nu \bmod m)\boldsymbol{x}_n \bmod m,$$

which follows readily from Eq. (3.4). If this is to be done several times for the same ν, the matrix $\boldsymbol{A}^\nu \bmod m$ can be precomputed once for all. For a large ν, this can be done in $O(\log_2 \nu)$ matrix multiplications via a standard square-and-multiply (or divide-and-conquer) exponentiation algorithm (Brassard and Bratley 1988, Knuth 1998, Section 4.6.3):

$$\boldsymbol{A}^\nu \bmod m = \begin{cases} (\boldsymbol{A}^{\nu/2} \bmod m)(\boldsymbol{A}^{\nu/2} \bmod m) \bmod m & textif\nu \text{ is even;} \\ \boldsymbol{A}(\boldsymbol{A}^{\nu-1} \bmod m) \bmod m & \text{if } \nu \text{ is odd.} \end{cases}$$

Jumping ahead is also easy in the polynomial representation. It follows from (3.12) that

$$p_{n+\nu}(z) = z^\nu p_n(z) \bmod P(z) = (z^\nu \bmod P(z))p_n(z) \bmod P(z),$$

where the polynomial $z^\nu \bmod P(z)$ can be precomputed once for all with the same type of square-and-multiply algorithm as above. So a second way to jump ahead from $\boldsymbol{x}_n$ to $\boldsymbol{x}_{n+\nu}$ is to transform the state to the polynomial representation via (3.10), jump ahead in that

representation by computing $p_{n+\nu} = (z^\nu \bmod P(z))p_n(z) \bmod P(z)$ for the appropriate ν, then transform back to the vector representation $\boldsymbol{x}_{n+\nu}$.

♣ See also Haramoto et al. (2008) for potentially more efficient methods.

Example 3.3 Let $k = 3$, $m = 2^{32} - 209 = 4294967087$, $a_1 = 0$, $a_2 = 1403580$, and $a_3 = -810728$. (This negative value of a_3 is equivalent to $a_3 + m$, because all operations are performed modulo m.) This gives the matrix

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -810728 & 1403580 & 0 \end{pmatrix}.$$

Let $\nu = 2^{127}$. By squaring $\boldsymbol{A}$ 127 times modulo m, we obtain

$$\boldsymbol{A}^\nu \bmod m = \begin{pmatrix} 2427906178 & 3580155704 & 949770784 \\ 226153695 & 1230515664 & 3580155704 \\ 1988835001 & 986791581 & 1230515664 \end{pmatrix}.$$

So to jump ahead by 2^{127} steps, it suffices to multiply the current state $\boldsymbol{x}_n$ by this matrix, modulo m. Doing this by calling the generator 2^{127} times would take much longer than anyone's lifetime! This particular example, with the same step size, is used in the package of L'Ecuyer et al. (2002). □

3.2.4 Period

To study the periodicity, we will use the polynomial representation. Note that the zero state $\boldsymbol{x}_n = \boldsymbol{0}$ corresponds to the zero polynomial $p_n(z) = 0$. This state must be avoided, because it is absorbing: if $\boldsymbol{x}_n = \boldsymbol{0}$ then $\boldsymbol{x}_{n+i} = \boldsymbol{0}$ and $p_{n+i}(z) = 0$ for all $i \geq 0$. After removing this state, there remains $m^k - 1$ nonzero states, so the period of (3.2) cannot exceed $m^k - 1$. Moreover, the LRS (3.2) has full period $m^k - 1$ if and only if the recurrence (3.12) does, i.e., if it visits all $m^k - 1$ nonzero polynomials in $\mathbb{Z}_m[z]/(P)$ in a single cycle. If we start this recurrence with $p_0(z) = 1$, we have

$$p_n(z) = z^n \bmod P(z),$$

which means that the maximal period is reached if and only if the smallest n such that $z^n \bmod P(z) = 1$ is $n = m^k - 1$. A polynomial $P(z)$ that satisfies this condition is called a *primitive polynomial*. The following is well-known in number theory (Lidl and Niederreiter 1986):

Proposition 3.3 *For $P(z)$ to be a primitive polynomial, it is* necessary *that m be a prime number and $a_k \neq 0$. If $k > 1$ it is also* necessary *that $a_j \neq 0$ for at least one j such that $0 < j < k$.*

When m is prime, $\mathbb{Z}_m$ turns out to be a *finite field*, usually denoted by $\mathbb{F}_m$ (this means that each nonzero element $a \in \mathbb{F}_m$ has a multiplicative inverse $a^* \in \mathbb{F}_m$, which satisfies $a^* a \bmod m = 1$).

Obviously, verifying the primitivity of a polynomial $P(z)$ by computing all powers of z modulo $P(z)$ is impractical. The following two propositions provide equivalent but more convenient conditions. The proof of Proposition 3.4 is left as an exercise.

Proposition 3.4 *If* $z^n \bmod P(z) = 1$ *for* $n = m^k - 1$, *and if* $z^n \bmod P(z) \neq 1$ *for* $n = (m^k-1)/q$ *for all prime numbers* $q > 1$ *that divide* m^k-1, *then* $P(z)$ *is a primitive polynomial.*

This proposition implies that it suffices to compute $z^{(m^k-1)/q} \bmod P(z)$ for $q = 1$ and for the prime factors q of $m^k - 1$. Finding those prime factors is not always easy. Observe, however, that $m^k - 1 = (m - 1)(m^{k-1} + \cdots + m + 1)$ and that $m - 1$ is even, so we can decompose

$$m^k - 1 = 2hr$$

where $h = (m-1)/2$ and $r = (m^k - 1)/(m - 1)$ are integers. The prime factors of $m^k - 1$ are the prime factors of h and r, together with 2. For $k > 2$, r is generally the hardest number to factor. A convenient way to avoid this factoring problem is to select m and k so that h and r are both prime numbers. This is easier to verify because checking the primality of large numbers is much easier than factoring. Then, the only values of q that need to be considered are 1, 2, h, and r.

The next proposition refines the previous one by giving a set of necessary and sufficient conditions that involve smaller powers of z than the numbers $(m^k - 1)/q$ in Proposition 3.4. These are the conditions that are used in practice.

Proposition 3.5 (Alanen and Knuth 1964) *The following three conditions are necessary and sufficient for* $P(z)$ *to be a primitive polynomial modulo* m*:*

(i) $((-1)^{k+1}a_k)^{(m-1)/q} \bmod m \neq 1$ *for each prime factor* q *of* $m - 1$*;*

(ii) $z^r \bmod (P(z), m) = (-1)^{k+1}a_k \bmod m$*;*

(iii) $(z^{r/q} \bmod (P(z), m))$ *has positive degree for each prime factor* q *of* r, *for* $1 < q < r$.

In the special case where m is prime and $k = 1$, one has $r = 1$ and the conditions of the two previous propositions reduce to:

$$a_1^{(m-1)/q} \bmod m \neq 1 \text{ for each prime factor } q \text{ of } m - 1 \tag{3.13}$$

(see Exercise 3.3). An integer $a_1 \in \mathbb{Z}_m$ satisfying this condition is called a *primitive element modulo* m.

♣ Give an efficient algorithm that specifies in what order to do the exponentiations to verify these conditions.

Example 3.4 Let $m = 101$ and $k = 1$. We have $m - 1 = 2^2 \times 5^2$, so the prime factors of $m - 1$ are $q = 2$ and $q = 5$. Then the values of $(m - 1)/q$ to be considered in (3.13) are 50 and 20. For $a_1 = 12$, for example, we get $a_1^{50} \bmod 101 = 100 \neq 1$ and $a_1^{20} \bmod 101 = 95 \neq 1$, so $a_1 = 12$ is a primitive element modulo 101 and this proves that the LCG with $m = 101$ and $a = 12$ has maximal period $\rho = 100$. □

Example 3.5 Let $m = 101$, $k = 3$, and $P(z) = z^3 - 28z^2 - 49z - 8$, as in Example 3.2. Here, $r = (101^3 - 1)/100 = 10303$ is a prime number. We can verify the conditions of Proposition 3.5 as follows. For (i), we have $a_k = 8$ and the relevant values of $(m-1)/q$ are again 50 and 20. We find that $8^{50} \bmod 101 = 100 \neq 1$ and $8^{20} \bmod 101 = 87 \neq 1$, so (i) is verified. For Condition (ii), we find $z^{10303} \bmod (P(z), 101) = 8 = a_k$. Condition (iii) is automatically verified because r is prime. Therefore, $P(z)$ is primitive and the MRG based on the recurrence $x_n = (28x_{n-1} + 49x_{n-2} + 8x_{n-3}) \bmod 101$ has period $101^3 - 1 = 1030300$. □

To search for primitive polynomials of order $k > 1$, it is convenient to first find a value of a_k satisfying (i) in Proposition 3.5, and then perform a search for good values of the remaining coefficients $(a_1, \ldots, a_{k-1})$. Typically, we want to impose conditions on the polynomial coefficients a_j to be able to implement the MRG efficiently (see Section 3.2.11). The search is then restricted to the set of coefficients that satisfy these conditions. When the number of possibilities is too large for an exhaustive examination, we can simply perform a random search. In terms of efficiency, we would prefer to have only a small number of nonzero coefficients a_j and set the other ones to zero, to save multiplications. In view of Proposition 3.3, for $k > 1$, the smallest number of nonzero coefficients for a primitive polynomial is two and this gives the simplified recurrence:

$$x_n = (a_r x_{n-r} + a_k x_{n-k}) \bmod m. \tag{3.14}$$

The primitive polynomials are scattered pretty much randomly over the monic polynomials of degree k in $\mathbb{F}_m[z]$ (i.e., the polynomials of the form (3.3)) for any given m and k. The next proposition implies that the proportion of monic polynomials of degree k that are primitive modulo a prime m is

$$\frac{1}{k} \prod_{j=1}^{J} \frac{p_j - 1}{p_j}$$

where $p_1, \ldots, p_J$ are the distinct prime factors of $m^k - 1$. The proposition's statement uses the *Euler function* ϕ, defined over the positive integers by

$$\phi(n) = n \prod_{j=1}^{J} \frac{p_j - 1}{p_j} = \prod_{j=1}^{J} p_j^{e_j - 1}(p_j - 1) \tag{3.15}$$

where $n = p_1^{e_1} \cdots p_J^{e_J}$ is the prime factorization of n.

Proposition 3.6 (Pellet, 1870) *For a prime number m, the number of monic polynomials of degree k that are primitive modulo m is $\phi(m^k - 1)/k$.*

Example 3.6 Let $m = 2^{31} - 1$ and $k = 1$. Then $m^k - 1 = m - 1 = 2^{31} - 2 = 2 \times 3^2 \times 7 \times 11 \times 31 \times 151 \times 331$ and the *proportion* of polynomials of degree 1 that are primitive modulo m is $\phi(m^k - 1)/(m-1) = \phi(m-1)/(m-1) = (1/2)(2/3)(6/7)(10/11)(30/31)(150/151)(330/331) \approx 0.248943$.

If $m = 2^{31} - 1$ and $k = 2$, then $2h = 2^{31} - 2$, $r = m + 1 = 2^{31}$, and $m^k - 1 = (m+1)(m-1) = 2^{31}(m-1)$, so the p_j's are the same as for $k = 1$. Thus, the proportion of

polynomials that are primitive is $\phi(m^2-1)/(2(m^2-1)) = \phi(m-1)/(2(m-1)) \approx 0.124471$. □

Prime h and r. As we said earlier, choosing m and k such that both m, h, and r are prime avoids the problem of factorizing h and r. Additional benefits are that the primitivity condition (iii) in Proposition 3.5 is always trivially satisfied, there are only 2 values of q (2 and h) to check in condition (i), and the primitive polynomials are more abundant than when there are more factors. There are indeed $(h-1)(r-1)/k$ primitive polynomials when h and r are distinct primes and $m > 2$, so their fraction is $(h-1)(r-1)/(2hrk) \approx 1/(2k)$ for large m. A selection of pairs (m, k) for which m, h, and r are prime, and m is near some large power of 2, can be found in L'Ecuyer (1999a).

If $m = 2$ and $r = 2^k - 1$ is prime (such a r is called a *Mersenne prime*), the fraction is $(2^k-2)/(k(2^k-1)) \approx 1/k$ for large enough k.

Example 3.7 Suppose we take $k = 5$, we want m to be a prime number slightly smaller than 2^{63}, and we also want h and r to be prime. An exhaustive computer search tells us that the largest m that satisfies these conditions is $m = 2^{63} - 19581$. With these values, if we select a polynomial at random uniformly in the set of all m^k monic polynomials of degree 5 with coefficients in $\mathbb{Z}_m$, the probability that this polynomial is primitive is approximately $1/(2k) = 1/10$. Primitive polynomials are thus easy to find by random search in this case. Any such primitive polynomial provides an MRG with period $m^k - 1 \approx 2^{315}$.

As another example, suppose we want $m < 2^{31}$. The largest prime m smaller than 2^{31} is $m = 2^{31} - 1$. For this reason, this m has been used extensively in the past as a modulus for LCGs. Factorizations of $m^k - 1$ for $k \leq 6$ can be found in L'Ecuyer, Blouin, and Couture (1993). However, h and r are not prime for these pairs (m, k).

For $k = 3$, for example, the largest prime integer $m < 2^{31}$ for which both h and r are prime is $m = 2^{31} - 21069$. With this pair (m, k), a random polynomial of degree k with coefficients in $\mathbb{Z}_m$ is primitive with probability near 1/3. □

3.2.5 What if m is a Power of Two?

It is tempting to take the modulus m equal to a power of 2, because computing $ax \bmod m$ is then much easier: compute the product ax and chop off the high-order bits. On a 32-bit computer, for instance, if $m = 2^{32}$ it suffices to make sure that overflow-checking is turned off, compute the product ax using unsigned integers, and the overflow will take care automatically of the modulo m operation.

However, there is a large price to pay in terms of a much shorter period and a poor quality of the least significant bits. If $m = 2^e$, the period of the recurrence (3.2) cannot exceed 2^{e-2} for $k = 1$ and $e \geq 4$, and $(2^k - 1)2^{e-1}$ for $k > 1$ (Knuth 1998, Eichenauer-Herrmann, Grothe, and Lehn 1989). Moreover, for $k = 1$, the period of the ith least significant bit of x_n cannot exceed $\max(1, 2^{i-2})$ (Bratley, Fox, and Schrage 1987), and if a full cycle is split into 2^d equal segments, all segments are identical except for their d most significant bits (De Matteis and Pagnutti 1988). For $k > 1$, the period of the ith least significant bit cannot exceed $(2^k - 1)2^{i-1}$. Thus, the behavior of the low-order bits is much too regular. In conclusion, LCGs and MRGs with power-of-2 moduli should not be used.

Example 3.8 If $k = 7$ and $m = 2^{31}-1$ (a prime), the maximal period is $(2^{31}-1)^7-1 \approx 2^{217}$. On the other hand, for $m = 2^{31}$ and the same value of k, we have $\rho \leq (2^7-1)2^{31-1} < 2^{37}$, which is more than 2^{180} times shorter. Moreover, the least significant bit has period at most $2^7 - 1 = 127$, the second least significant bit has period at most $2(2^7 - 1) = 254$, and so on. □

Example 3.9 The values of $x_0, \ldots, x_7$ obtained from the recurrence $x_n = 10205x_{n-1} \bmod 2^{15}$ with $x_0 = 12345$ are (in base 10 and in base 2):

$$\begin{aligned}
x_0 &= 12345 = 011000000111001_2 \\
x_1 &= 20533 = 101000000110101_2 \\
x_2 &= 20673 = 101000011000001_2 \\
x_3 &= 7581 = 001110110011101_2 \\
x_4 &= 31625 = 111101110001001_2 \\
x_5 &= 1093 = 000010001000101_2 \\
x_6 &= 12945 = 011001010010001_2 \\
x_7 &= 15917 = 011111000101101_2.
\end{aligned}$$

We see that the last two bits never change, the third least significant bit has period 2, the fourth least significant bit has period 4, and so on. □

To improve the period when m is not a prime number (e.g., if it is a power of 2), one can think of adding a constant c to the right side of the recurrence (3.2) before the reduction modulo m. One can show (L'Ecuyer 1990a, page 87) that a linear recurrence of order k with such a constant term is equivalent to some linear recurrence of order $k + 1$ with no constant term. As a result, an upper bound on the period of such a recurrence with $m = 2^e$ is $(2^{k+1} - 1)2^{e-1}$, which is still much smaller than m^k for large e and k.

For $k = 1$ and $c > 0$, the largest possible period is $m = 2^e$ (the size of the state space) and it is achieved if and only if c is odd and $a \bmod 4 = 1$ (Knuth 1998, Page 17).

LCGs with power-of-2 moduli were popular in the past. Some examples are given in Table 3.1. RANDU was the generator offered on the IBM System/360 computers. Its set Ψ_3 (in three dimensions) lies in exactly 15 equidistant planes (see Law and Kelton 2000, page 427, for an illustration). The BSD ANSI C generator is the one provided in the standard ANSI C library. All these RNGs should be avoided.

3.2.6 Linear Recurrences With Carry

We just discarded the idea of using a power-of-two modulus with the linear recurrence (3.2), but this type of modulus may become interesting again if we change the recurrence slightly. To obtain a large period even when m is a power of two, Marsaglia and Zaman (1991) introduced the idea of adding a *carry* to the linear recurrence (3.2). Their proposal was studied and generalized by Tezuka, L'Ecuyer, and Couture (1993), Couture and L'Ecuyer

Table 3.1. Some (old) popular LCGs with power-of-2 moduli

m	a	c	Reference
2^{24}	1140671485	12820163	in early VisualBasic from Microsoft
2^{31}	65539	0	RANDU
2^{31}	134775813	1	in early Turbo Pascal
2^{31}	1103515245	12345	`rand()` in BSD ANSI C
2^{32}	69069	1	on VAX/VMS systems
2^{32}	2147001325	715136305	in the BCLP language
2^{35}	5^{15}	7261067085	Knuth (1998)
2^{48}	68909602460261	0	Fishman (1990)
2^{48}	25214903917	11	Unix's `rand48()`
2^{48}	44485709377909	0	on CRAY system
2^{59}	13^{13}	0	in NAG Fortran/C library

(1994), Couture and L'Ecuyer (1997), and Goresky and Klapper (2003), to get a class of generators called *multiply-with-carry* (MWC), defined by

$$x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1}) d \bmod b, \tag{3.16}$$

$$c_n = \lfloor (a_0 x_n + a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1})/b \rfloor, \tag{3.17}$$

$$u_n = \sum_{\ell=1}^{\infty} x_{n+\ell-1} b^{-\ell}, \tag{3.18}$$

in which b is a positive integer, $a_0, \ldots, a_k$ are arbitrary integers such that a_0 is relatively prime to b, and d is the multiplicative inverse of $-a_0$ modulo b, i.e., $(-a_0 d) \bmod b = 1$. Eq. (3.16) is equivalent to $(a_0 x_n + a_1 x_{n-1} + \cdots + a_k x_{n-k} + c_{n-1}) \bmod b = 0$, and the carry c_n represents the number of times b has to be subtracted to perform the latter "mod b" operation. The state at step n is $\boldsymbol{x}_n = (x_{n-k+1}, \ldots, x_n, c_n)^{\mathsf{t}}$. In practice, b is usually taken as a power of 2 and the sum in (3.18) is truncated to a few terms (it could be a single term if b is large), but the theoretical analysis is much easier for the infinite sum.

Define $m = \sum_{\ell=0}^{k} a_\ell b^\ell$ and let a be the inverse of b in arithmetic modulo m, assuming for now that $m > 0$. A major result proved in Tezuka, L'Ecuyer, and Couture (1993), Couture and L'Ecuyer (1997), and Goresky and Klapper (2003) is that if the initial states agree, the output sequence $\{u_n,\ n \geq 0\}$ is exactly the same as that produced by the LCG with modulus m and multiplier a. Therefore, the MWC can be seen as a clever way of implementing an LCG with very large modulus.

In the original proposals of Marsaglia and Zaman (1991), called *add-with-carry* and *subtract-with-borrow*, one has $-a_0 = \pm a_r = \pm a_k = 1$ for some $r < k$ and the other coefficients a_j are zero. We will see in Section 3.2.10 that all nonzero vectors of the form (u_{n-k}, u_{n-r}, u_n) (or equivalently, $(u_n, u_{n-r+k}, u_{n+k})$) produced by the LCG associated with these generators lie in at most three planes in the three-dimensional unit cube. We saw a concrete illustration of this type of generator in Example 1.16.

In the version studied by Couture and L'Ecuyer (1997), it was assumed that $-a_0 = d = 1$. Then, the period cannot exceed $(m-1)/2$ if b is a power of two. A concrete implementation was given in that paper. Goresky and Klapper (2003) pointed out that the maximal period

of $\rho = m-1$ can be achieved by allowing a more general a_0. They provide specific parameters that give a maximal period for b ranging from 2^{21} to 2^{35} and ρ up to approximately 2^{2521}.

3.2.7 The Lattice Structure

We now examine the structure of the point sets Ψ_I produced by MRGs and discuss how we can measure their uniformity. We will see where the regular structure observed in Figure 1.8 comes from and show that all point sets Ψ_I produced by MRGs have this type of structure.

♣ Perhaps add a small example here.

We use $\boldsymbol{e}_i$ to denote the ith unit vector, with a 1 in position i and 0's elsewhere. The dimension of this vector will be obvious from the context; it is k in this paragraph and s in the next paragraph, for example. Denote by $x_{i,0}, x_{i,1}, x_{i,2}, \ldots$ the values of $x_0, x_1, x_2, \ldots$ produced by the recurrence (3.2) when $\boldsymbol{x}_{k-1} = (x_0, \ldots, x_{k-1})^\mathsf{t} = \boldsymbol{e}_i$, and let $\boldsymbol{y}_i = (x_{i,0}, x_{i,1}, \ldots, x_{i,s-1})$ for any given $s \geq k$. We have

$$\begin{aligned} \boldsymbol{y}_1 &= (1, 0, \ldots, 0, x_{1,k}, \ldots, x_{1,s-1})^\mathsf{t} \\ \vdots &\quad\ \vdots \\ \boldsymbol{y}_k &= (0, 0, \ldots, 1, x_{k,k}, \ldots, x_{k,s-1})^\mathsf{t}. \end{aligned}$$

An arbitrary state $\boldsymbol{x}_{k-1} = (z_1, \ldots, z_k)^\mathsf{t} \in \mathbb{Z}_m^k$ can be written as the linear combination $\boldsymbol{x}_{k-1} = z_1\boldsymbol{e}_1 + \cdots + z_k\boldsymbol{e}_k$. Then the vector of successive values $\boldsymbol{y} = (x_0, x_1, \ldots, x_{s-1})^\mathsf{t}$ produced by the linear recurrence when $\boldsymbol{x}_{k-1}$ is the initial state is given by the linear combination $\boldsymbol{y} = z_1\boldsymbol{y}_1 + \cdots + z_k\boldsymbol{y}_k$, reduced modulo m. That is,

$$\begin{aligned} x_k &= (z_1 x_{1,k} + \cdots + z_k x_{k,k}) \bmod m, \\ x_{k+1} &= (z_1 x_{1,k+1} + \cdots + z_k x_{k,k+1}) \bmod m, \end{aligned}$$

and so on. Reciprocally, any linear combination $z_1\boldsymbol{y}_1 + \cdots + z_k\boldsymbol{y}_k$ reduced modulo m is the vector of values $\boldsymbol{y} = (x_0, x_1, \ldots, x_{s-1})^\mathsf{t}$ obtained from the initial state $\boldsymbol{x}_{k-1} = (z_1, \ldots, z_k)^\mathsf{t} \in \mathbb{Z}_m^k$.

Reduction of the jth coordinate modulo m is achieved by subtracting an integer multiple of $m\boldsymbol{e}_j$, the s-dimensional vector with a m in position j and 0's elsewhere. When computing $\boldsymbol{y} = z_1\boldsymbol{y}_1 + \cdots + z_k\boldsymbol{y}_k$, this reduction can be needed only for $j > k$, because $z_i \in \mathbb{Z}_m$ for each i. This means that for $s \geq k$, a vector $\boldsymbol{y} = (x_0, x_1, \ldots, x_{s-1})^\mathsf{t}$ in $\mathbb{Z}_m$ is produced by the recurrence (3.2) if and only if it can be written as

$$\boldsymbol{y} = z_1\boldsymbol{y}_1 + \cdots + z_s\boldsymbol{y}_s$$

where $\boldsymbol{y}_i = m\boldsymbol{e}_i$ for $i > k$ and $z_1, \ldots, z_s$ are arbitrary integers.

If we divide everything by m and apply Proposition 3.1, we find that a s-dimensional vector $(u_0, \ldots, u_{s-1}) \in [0,1)^s$ is in Ψ_s if and only if it can be written as a linear combination, with integer coefficients, of the vectors

$$\begin{aligned}
\boldsymbol{v}_1 &= (1, 0, \ldots, 0, x_{1,k}, \ldots, x_{1,s-1})^{\mathsf{t}}/m \\
&\vdots \qquad\quad \vdots \\
\boldsymbol{v}_k &= (0, 0, \ldots, 1, x_{k,k}, \ldots, x_{k,s-1})^{\mathsf{t}}/m \\
\boldsymbol{v}_{k+1} &= (0, 0, \ldots, 0, 1, \ldots, 0)^{\mathsf{t}} \\
&\vdots \qquad\quad \vdots \\
\boldsymbol{v}_s &= (0, 0, \ldots, 0, 0, \ldots, 1)^{\mathsf{t}}.
\end{aligned}$$

Let L_s be the s-dimensional vector space

$$L_s = \left\{ \boldsymbol{v} = \sum_{i=1}^{s} z_i \boldsymbol{v}_i \;\mid\; z_i \in \mathbb{Z} \right\}, \tag{3.19}$$

which is the set of all vectors that can be written as integer linear combinations of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$. This L_s is a discrete set that contains an infinite number of points in the s-dimensional real space. A point of this set belongs to Ψ_s if and only if all its coordinates are reduced modulo 1, i.e., are in the interval $[0, 1)$. We have just shown the following:

Proposition 3.7 *For the MRG, for each $s \geq 1$, we have $\Psi_s = L_s \cap [0, 1)^s$.*

Example 3.10 In Figure 1.8(a), we have $k = 1$, $m = 101$, $a_1 = 12$, and $s = 2$. Then, $\boldsymbol{y}_1 = (1, 12)^{\mathsf{t}}$, $\boldsymbol{y}_2 = (0, 101)^{\mathsf{t}}$, $\boldsymbol{v}_1 = (1/101, 12/101)^{\mathsf{t}}$, and $\boldsymbol{v}_2 = (0, 1)^{\mathsf{t}}$. These two vectors are displayed in Figure 1.8(a). The set Ψ_2 is comprised of the 101 vectors $\boldsymbol{v} = (u_0, u_1)^{\mathsf{t}}$ of the form $\boldsymbol{v} = z_1\boldsymbol{v}_1 + z_2\boldsymbol{v}_2$ where $z_1 \in \mathbb{Z}_{101}$ and $z_2 = -\lfloor 12 z_1/101 \rfloor$. In other words, we take the vectors $\mathbf{0}, \boldsymbol{v}_1, 2\boldsymbol{v}_1, 3\boldsymbol{v}_1, \ldots$ and subtract $\boldsymbol{v}_2$ as many times as needed when the second coordinate exceeds 1. As can be seen from the figure, the set Ψ_2 contains the vectors $\boldsymbol{v} = i\,\boldsymbol{v}_1$ for $i = 0, \ldots, 8$, $\boldsymbol{v} = i\,\boldsymbol{v}_1 - \boldsymbol{v}_2$ for $i = 9, \ldots, 16$, $\boldsymbol{v} = i\,\boldsymbol{v}_1 - 2\,\boldsymbol{v}_2$ for $i = 17, \ldots, 25$, and so on, ending with $\boldsymbol{v} = i\,\boldsymbol{v}_1 - 11\,\boldsymbol{v}_2$ for $i = 93, \ldots, 100$. □

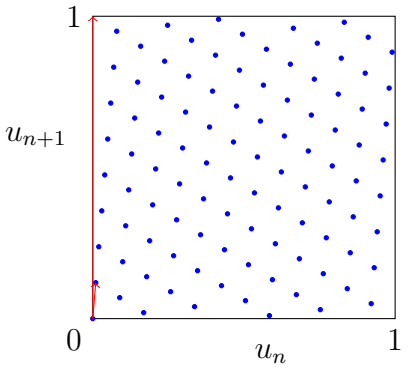


Fig. 3.1. The lattice structure of Ψ_2 for the LCG with $m = 101$ and $a = 12$.

Example 3.11 In Figure 1.8(b), where $a_1 = 51$, we have $\boldsymbol{y}_1 = (1, 51)^\mathsf{t}$, $\boldsymbol{y}_2 = (0, 101)^\mathsf{t}$, $\boldsymbol{v}_1 = (1/101, 51/101)^\mathsf{t}$, and $\boldsymbol{v}_2 = (0, 1)^\mathsf{t}$, and $2\boldsymbol{v}_1 \bmod 1 = 2\boldsymbol{v}_1 - \boldsymbol{v}_2 = (2/101,\ 1/101)^\mathsf{t}$, which represents the nonzero point closest to the origin in Figure 1.8(b). All integral multiples of this vector $\boldsymbol{v}^* = (2/101,\ 1/101)^\mathsf{t}$ belong to the lattice; they include all points on the lower line of Figure 1.8(b). Given that $\boldsymbol{v}^*$ is very short, this line must necessarily contain a large number of points (it actually contains 51 points out of 101). Likewise, starting from the leftmost point of the second line and adding any integral multiple of $\boldsymbol{v}^*$, we obtain a lattice point, so this second line must also contain a large number of points. This implies in turn that all points must stand on a very small number of lines, which is bad from the uniformity viewpoint. This observation applies generally and also in higher dimensions: whenever the lattice L_s contains a very short vector, all the lattice points must lie on a small number of parallel hyperplanes in $[0, 1)^s$ (which are lines when $s = 2$) and the uniformity cannot be good. □

Example 3.12 Let $k = 2$ and $s = 4$, for arbitrary values of m, a_1 and a_2. In that case, for $(x_{1,0}, x_{1,1}) = (1, 0)$ we have

$$\begin{aligned} x_{1,2} &= (a_1 x_{1,1} + a_2 x_{1,0}) \bmod m = a_2 \qquad \text{and} \\ x_{1,3} &= (a_1 x_{1,2} + a_2 x_{1,1}) \bmod m = a_1 a_2 \bmod m. \end{aligned}$$

For $(x_{2,0}, x_{2,1}) = (0, 1)$ we have

$$\begin{aligned} x_{2,2} &= (a_1 x_{2,1} + a_2 x_{2,0}) \bmod m = a_1 \qquad \text{and} \\ x_{2,3} &= (a_1 x_{2,2} + a_2 x_{2,1}) \bmod m = (a_1^2 + a_2) \bmod m. \end{aligned}$$

The vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_4$ are then

$$\begin{aligned} \boldsymbol{v}_1 &= (1, 0, a_2, a_1 a_2 \bmod m)^\mathsf{t}/m \\ \boldsymbol{v}_2 &= (0, 1, a_1, (a_1^2 + a_2) \bmod m)^\mathsf{t}/m \\ \boldsymbol{v}_3 &= \boldsymbol{e}_3 \\ \boldsymbol{v}_4 &= \boldsymbol{e}_4. \end{aligned}$$

□

For $s \le k$, L_s contains all vectors whose coordinates are multiples of $1/m$ and there are m^s such vectors that belong to $[0, 1)^s$. For $s > k$, L_s contains a fraction m^{k-s} of those vectors and m^k distinct ones belong to $[0, 1)^s$. It is easily seen that L_s contains all corners of the unit hypercube $[0, 1]^s$, and therefore all vectors with integer coordinates. Adding an integer vector $\boldsymbol{z} \in \mathbb{Z}^s$ to all the points of Ψ_s gives the set of points of L_s that belong to the unit hypercube $[\boldsymbol{z}, \boldsymbol{z} + \boldsymbol{1})$ having lower left corner at $\boldsymbol{z}$. That is, for each integer vector $\boldsymbol{z}$, $L_s \cap [\boldsymbol{z}, \boldsymbol{z} + \boldsymbol{1})$ is just a shifted copy of Ψ_s that contains m^k points, so L_s is obtained simply by tiling an infinite number of copies of Ψ_s, in all directions.

A discrete vector space of the form (3.19) with vectors $\boldsymbol{v}_i \in \mathbb{R}^s$ is a *lattice* in $\mathbb{R}^s$ (Conway and Sloane 1999). The matrix $\boldsymbol{V}$ whose *rows* are $\boldsymbol{v}_1^\mathsf{t}, \ldots, \boldsymbol{v}_s^\mathsf{t}$ is a *generator matrix* of L_s. The lattice can also be written as $L_s = \{\boldsymbol{V}\boldsymbol{z} : \boldsymbol{z} \in \mathbb{Z}^s\}$. When $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$ are linearly independent, they form a *basis* of the lattice. Then the average number of lattice points per unit of volume,

called the *density* of the lattice, is equal to $1/\det(\boldsymbol{V})$ where $\det(\boldsymbol{V})$ is the determinant of $\boldsymbol{V}$. In our case, the generator matrix is

$$\boldsymbol{V} = \begin{pmatrix} 1/m & 0 & \cdots & 0 & x_{1,k}/m & \cdots & x_{1,s-1}/m \\ \vdots & & \vdots & & & \vdots & \\ 0 & 0 & \cdots & 1/m & x_{k,k}/m & \cdots & x_{k,s-1}/m \\ 0 & 0 & \cdots & 0 & 1 & \cdots & 0 \\ & & & & & & \\ \vdots & & \vdots & & & \vdots & \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Its rows are clearly linearly independent and its determinant is the product of the diagonal elements, m^{-k}. We have $1/\det(\boldsymbol{V}) = m^k$, which corresponds to the *density* of the lattice (the number of points per unit of volume).

This lattice structure implies that the points of Ψ_s are distributed according to a very regular pattern. For example, each point of L_s has a nearest neighbor at the same distance and in the same direction as any other point, and there are families of equidistant parallel hyperplanes that cover all the points, as shown in Figures 1.8 and 1.8 for $k = 1$ and $s = 2$, where the points belong to equidistant parallel lines. In terms of uniformity, we clearly prefer Figure 1.8(a) to Figure 1.8(b). In Figure 1.8(b), each point is too close to its nearest neighbor and the points are covered by lines that are too far away from one another, whereas in Figure 1.8(a), the points are far from one another and the distance between the lines is smaller.

In view of this regular structure, natural ways of measuring the uniformity for this type of point set Ψ_s include the following (among others):

(a) the (Euclidean) distance from a point to its nearest neighbor, which is also the distance from the origin to its closest point, or equivalently the length of the shortest nonzero vector in L_s;

(b) the distance between the two farthest hyperplanes that bound an area that contains no point from L_s;

(c) the minimal number of equidistant parallel hyperplanes that can cover all the points of Ψ_s.

For given values of m and k, we want the quantities in (a) and (c) to be as large as possible, and the distance in (b) to be as small as possible.

Example 3.13 In Figure 1.8(b), a point of L_2 closest to the origin is $(2/101, 1/101)$, at distance $\sqrt{5}/101$, and all the points of Ψ_2 are in only 2 parallel lines. The distance between these two lines is $1/\sqrt{5} \approx 0.4472$. In Figure 1.8(a), a closest point to the origin is $(-8, 5)$, at distance $\sqrt{89}/101$, and we need 13 parallel hyperplanes to cover all the points. The distance between the lines is much smaller. □

3.2.8 Figures of Merit

Distance between the points. We now examine in more detail the uniformity measures (a) to (c) introduced above. We start with (a), the (Euclidean) length of the shortest nonzero vector in the lattice L_s. Let d_s denote this length.

Any vector $\boldsymbol{v}$ in L_s can be written as $\boldsymbol{v} = \sum_{i=1}^{s} z_i \boldsymbol{v}_i$ for some integers z_i. Its square (Euclidean) length is the quadratic form

$$\|\boldsymbol{v}\|^2 = \sum_{i=1}^{s}\sum_{j=1}^{s} z_i \boldsymbol{v}_i^{\mathrm{t}} \boldsymbol{v}_j z_j = \boldsymbol{z}^{\mathrm{t}} \boldsymbol{V}^{\mathrm{t}} \boldsymbol{V} \boldsymbol{z}$$

where $\boldsymbol{z} = (z_1, \ldots, z_s)^{\mathrm{t}}$. Finding the shortest nonzero vector in L_s can thus be formulated as an integer programming optimization problem with a quadratic objective function:

$$\begin{array}{ll} \text{Minimize} & \|\boldsymbol{v}\|^2 = \displaystyle\sum_{i=1}^{s}\sum_{j=1}^{s} z_i \boldsymbol{v}_i^{\mathrm{t}} \boldsymbol{v}_j z_j \\ \text{subject to} & z_1, \ldots, z_s \ \text{ all integers and not all zero.} \end{array}$$

The optimal value is d_s^2, the squared length of the shortest nonzero vector. Dieter (1975) gave an algorithm to solve the problem, based essentially on branch-and-bound methodology. An improved version is detailed by Knuth (1998). A more efficient algorithm that exploits tighter bounds for the branch-and-bound was proposed by Fincke and Pohst (1985) (see also Tezuka 1995, L'Ecuyer and Couture 1997).

How large can be the distance between the points? Suppose we put a solid ball of radius ρ centered at each point of L_s. The largest ρ we can use before the balls overlap is precisely $d_s/2$. Since there are $n = m^k$ lattice points per unit of volume, the fraction of the space occupied by these spheres is $\Delta_s = nV_s\rho^s$ where V_s is the volume of a s-dimensional sphere with radius 1. This problem of packing the space as densely as possible by a lattice of identical spheres has been studied extensively in mathematics (Conway and Sloane 1999). Let Δ_s^* be the maximal value of Δ_s for a general lattice in s dimensions. We then have

$$d_s = 2\rho \le 2(\Delta_s^*/(nV_s))^{1/s}.$$

This gives the upper bound

$$d_s \le d_s^*(n) \stackrel{\text{def}}{=} \gamma_s^{1/2} n^{-1/s} \tag{3.20}$$

where $\gamma_s = 4(\Delta_s^*/V_s)^{2/s}$ is a constant that depends only on s.

These γ_s's are known as the *Hermite constants* (Conway and Sloane 1999, Knuth 1998). (Beware: some authors use γ_s to denote our $\gamma_s^{1/2}$.) They are known exactly only for $s \le 8$. For $s > 8$, however, upper and lower bounds on these constants are available. In particular, Rogers' bound on the density Δ_s of a sphere packing (Conway and Sloane 1999, page 19) yields

$$\gamma_s \le \bar{\gamma}_s \stackrel{\text{def}}{=} 2^{1+2R(s)/s},$$

[3] where the value of $R(s)$ is given in Conway and Sloane (1999), page 15, for $s \le 24$ and can be approximated by

[3] From Pierre: Should be $\log_2(\Delta_s^*/V_s) \le R(s) - s/2$. Check this.

$$R(s) \approx \frac{1}{2}s \lg\left(\frac{s}{4\pi e}\right) + \frac{3}{2}\log_2(s) - \log_2(e/\sqrt{\pi}) + \frac{5.25}{s+2.5},$$

with $O(1/s)$ error and four decimal digits of precision for $s \geq 25$. To obtain a lower bound on γ_s, we can use the largest density Δ_s known so far. The best known values of Δ_s/V_s are given in Conway and Sloane (1999), page xix (in the preface), for $s \leq 48$. The corresponding values of $4(\Delta_s/V_s)^{2/s}$ can be taken as lower bounds $\underline{\gamma}_s$ on γ_s. For all $s \leq 8$, the lower bound is attained by one or more known lattice(s), so $\underline{\gamma}_s = \gamma_s$.

We emphasize that the bound (3.20) is not necessarily attainable by a lattice that can be obtained from the point set Ψ_s of an MRG. These bounds are for *general* lattices with basis vectors $\boldsymbol{v}_i \in \mathbb{R}^s$.

♣ Give an example that compares the bound (3.20) with the best attainable value for an MRG of a given size, found by exhaustive search.

♣ Should give graph showing the bounds as a function of s...

3.2.8.1 The dual lattice and distance between hyperplanes. The *dual lattice* to L_s is defined as

$$L_s^* = \{\boldsymbol{h} \in \mathbb{R}^s : \boldsymbol{h}^t\boldsymbol{v} \in \mathbb{Z} \text{ for all } \boldsymbol{v} \in L_s\}. \tag{3.21}$$

This is the set of vectors $\boldsymbol{h}$ whose scalar product with any vector of L_s is an integer. In our case, the generating matrix $\boldsymbol{V}$ is upper-triangular with positive values on the diagonal, so it clearly has an inverse, say $\boldsymbol{W} = \boldsymbol{V}^{-1}$, whose determinant is $\det(\boldsymbol{W}) = 1/\det(\boldsymbol{V}) = m^k = |\Psi_s|$. It can be verified (Exercise 3.6) that the *columns* of $\boldsymbol{W}$ are $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_s$, where

$$\begin{aligned} \boldsymbol{w}_i &= m\boldsymbol{e}_i \quad \text{for } 1 \leq i \leq k, \quad \text{and} \\ \boldsymbol{w}_i &= \boldsymbol{e}_i - (x_{1,i-1}, \ldots, x_{k,i-1}, 0, \ldots, 0)^t \quad \text{for } k < i \leq s. \end{aligned}$$

These vectors $\boldsymbol{w}_i$ form a basis of the dual lattice L_s^*. That is, we can write

$$L_s^* = \left\{\boldsymbol{h} = \sum_{i=1}^{s} z_i\boldsymbol{w}_i \;\mid\; z_i \in \mathbb{Z}\right\}. \tag{3.22}$$

Since all coordinates of these vectors are integers, we can conclude that all vectors of the dual lattice have integer coordinates, i.e., $L_s^* \subset \mathbb{Z}^s$. This dual lattice has density m^{-k}: it contains only integer vectors, and one integer vector out of m^k.

For each vector $\boldsymbol{h} = (h_1, \ldots, h_s) \in L_s^*$ and each integer z, the set $\{\boldsymbol{v} \in \mathbb{R}^s : \boldsymbol{h}^t\boldsymbol{v} = z\}$ is an hyperplane orthogonal to $\boldsymbol{h}$. When z runs through all integers, we obtain a family $\mathcal{H}(\boldsymbol{h})$ of equidistant parallel hyperplanes which cover all points $\boldsymbol{v}$ of L_s, because $\boldsymbol{h}^t\boldsymbol{v} \in \mathbb{Z}$ for each $\boldsymbol{v} \in L_s$. The distance between two successive hyperplanes of $\mathcal{H}(\boldsymbol{h})$ is the distance from the hyperplane with $z = 1$ to that with $z = 0$, i.e., to the origin, since the latter contain the origin. But the vector $\boldsymbol{v}$ closest to the origin, i.e., with minimal square length $\|\boldsymbol{v}\|_2^2 = v_1^2 + \cdots + v_s^2$, under the constraint that $\boldsymbol{h}^t\boldsymbol{v} = 1$, is easily characterized using a Lagrange multiplier λ, as follows. By differentiating $\|\boldsymbol{v}\|_2^2 + \lambda(1 - \boldsymbol{h}^t\boldsymbol{v})$ with respect to v_i and equaling the derivative to zero, we find that $v_i = (\lambda/2)h_i$, so v_i must be proportional to h_i for each i. The scaling constant $\lambda/2$ can then be determined from the constraint. We have

$$1 = \boldsymbol{h}^t\boldsymbol{v} = (\lambda/2)(h_1^2 + \cdots + h_s^2)$$

which gives $\lambda/2 = 1/\|\boldsymbol{h}\|_2^2$ and then $v_i = h_i/\|\boldsymbol{h}\|_2^2$. This vector $\boldsymbol{v}$ has length $\|\boldsymbol{v}\|_2 = \|\boldsymbol{h}\|_2/\|\boldsymbol{h}\|_2^2 = 1/\|\boldsymbol{h}\|_2$ and this length is the distance between the successive parallel hyperplanes of the family $\mathcal{H}(\boldsymbol{h})$.

If ℓ_s is the Euclidean length of a shortest non-zero vector $\boldsymbol{h}$ in L_s^*, then there is a family of hyperplanes at distance $1/\ell_s$ apart that cover all the points of L_s, and no family of hyperplanes can cover all the points if they are spaced more than $1/\ell_s$ apart. A small ℓ_s means that there are thick slices of empty space between the hyperplanes and we want to avoid that. A large ℓ_s means a better (more uniform) coverage of the unit hypercube by the point set Ψ_s. The problem of computing ℓ_s is the same as that of computing d_s: just replace L_s by L_s^* and use the same algorithm. Computing ℓ_s is often called the *spectral test* for historical reasons. It was first proposed by Coveyou and MacPherson (1967) for LCGs. These authors introduced ℓ_s via Fourier analysis and called it the *minimal wave number*. Knuth (1969) details this viewpoint. The lattice structure of LCGs was also pointed out by Marsaglia (1968).

Example 3.14 For the LCG illustrated in Figure 1.8(a), we have $s = 2$, $k = 1$, $m = 101$, and $a_1 = 12$. The vectors $\boldsymbol{w}_1 = m\boldsymbol{e}_1 = (101, 0)^\mathsf{t}$ and $\boldsymbol{w}_2 = \boldsymbol{e}_2 - (12, 0)^\mathsf{t} = (-12, 1)^\mathsf{t}$ form a basis of the dual lattice L_2^*. Any vector in this dual lattice can be written as $\boldsymbol{h} = z_1\boldsymbol{w}_1 + z_2\boldsymbol{w}_2 = (101z_1 - 12z_2,\ z_2)^\mathsf{t}$ and its square length is $\|\boldsymbol{h}\|_2^2 = (101z_1 - 12z_2)^2 + z_2^2$. To find the shortest nonzero vector in L_2^* we must minimize this expression with respect to $(z_1, z_2)^\mathsf{t} \in \mathbb{Z}^2 \setminus \{\mathbf{0}\}$. This is an integer programming problem, whose optimal solution is $(z_1, z_2) = (1, 8)$. The corresponding shortest vector is $\boldsymbol{h} = \boldsymbol{w}_1 + 8\boldsymbol{w}_2 = (5, 8)^\mathsf{t}$ and has length $\ell_2 = \sqrt{5^2 + 8^2} = \sqrt{89}$, so all points of L_s here are contained in parallel lines that are at distance $1/\sqrt{89} \approx 0.1060$ apart. □

Example 3.15 In Figure 1.8(b), we have $a_1 = 51$ instead, so $\boldsymbol{w}_1 = (101, 0)^\mathsf{t}$ and $\boldsymbol{w}_2 = (-51, 1)^\mathsf{t}$. The vector $\boldsymbol{h} = \boldsymbol{w}_1 + 2\boldsymbol{w}_2 = (-1, 2)^\mathsf{t}$ has length $\ell_2 = \sqrt{5}$ and is the shortest vector. So here all the points are contained in parallel lines that are at distance $1/\sqrt{5} \approx 0.4472$ apart. □

Since L_s has density $n = m^k$, the dual lattice L_s^* has density m^{-k} and the upper bound on ℓ_s that corresponds to (3.20) becomes

$$\ell_s^*(m^k) \stackrel{\text{def}}{=} d_s^*(m^{-k}) = \gamma_s^{1/2} m^{k/s}. \tag{3.23}$$

Number of hyperplanes to cover the points. The smallest number of parallel hyperplanes that cover all the points of Ψ_s can be found by computing a shortest nonzero vector in L_s^* using the $\mathcal{L}_1$-norm instead of the Euclidean norm. Recall that the $\mathcal{L}_1$-norm of a vector $\boldsymbol{h} = (h_1, \ldots, h_s) \in \mathbb{R}^s$ is $\|\boldsymbol{h}\|_1 = |h_1| + \cdots + |h_s|$.

Dieter (1975) pointed out that for any nonzero $\boldsymbol{h} \in L_s^*$, the number of hyperplanes in $\mathcal{H}(\boldsymbol{h})$ that intersect the *open unit hypercube* $(0, 1)^s$ (i.e., excluding the point $\mathbf{0}$) is $n_s = \|\boldsymbol{h}\|_1 - 1$. A vector $\boldsymbol{h}^*$ that minimizes this $\mathcal{L}_1$ norm (and the value of n_s) can be found by solving the optimization problem:

$$\begin{aligned} \text{Minimize} \quad & \|\boldsymbol{h}\|_1 - 1 \\ \text{subject to} \quad & \boldsymbol{h} = z_1\boldsymbol{h}_1 + \cdots + z_s\boldsymbol{h}_s \neq \mathbf{0} \text{ and each } z_i \in \mathbb{Z}. \end{aligned}$$

This problem can be solved by a branch-and-bound procedure, as suggested by Dieter (1975). The algorithm of Dieter (1975) works fine for s up to about 10 but becomes very slow for larger s (the time is exponential in s). If

$$n_s = \|\boldsymbol{h}^*\|_1 - 1$$

denotes the minimal number of hyperplanes thus found, we can conclude that all the nonzero points of Ψ_s are covered by exactly n_s hyperplanes from the family $\mathcal{H}(\boldsymbol{h}^*)$, and they cannot be covered by fewer hyperplanes.

Knuth (1998), Exercise 3.3.4-16, shows that if we include the point $\mathbf{0}$, then we need $\|\boldsymbol{h}\|_1 - \delta(\boldsymbol{h})$ hyperplanes to cover all the points, where $\delta(\boldsymbol{h}) = 1$ if $\boldsymbol{h}$ has two coordinates of opposite sign and $\delta(\boldsymbol{h}) = 0$ otherwise (that is, when all coordinates of $\boldsymbol{h}$ have the same sign, we need one more hyperplane and it contains only the point $\mathbf{0}$).

Marsaglia (1968) pointed out that by applying the general convex body theorem of Minkowski, we obtain the upper bound

$$n_s \leq n_s^*(n) \stackrel{\text{def}}{=} (s!\, n)^{1/s}, \tag{3.24}$$

where n is the density of the lattice L_s. For an MRG, we have $n = m^k$.

The numbers d_s, ℓ_s, and n_s are just three ways of assessing the uniformity of a lattice L_s. There are many more that we shall not discuss here. They include, for example, the Beyer quotient (Beyer, Roof, and Williamson 1971, Afflerbach and Grothe 1985) and some discrepancy measures used when lattices are used for QMC integration (Sloan and Joe 1994, Hickernell 1998b, L'Ecuyer 2009).

Lacunary indices. The lattice properties discussed so far hold as well for the point sets Ψ_I formed by values at arbitrary lags defined by a fixed set of indices $I = \{i_1, \cdots, i_s\}$. One has $\Psi_I = L_I \cap [0,1)^s$ for some s-dimensional lattice L_I. We can define and compute for the lattice L_I the counterparts of d_s, ℓ_s, and n_s, which we denote by d_I, ℓ_I, and n_I, respectively. That is, d_I is the minimal distance between any two points (or the length of a shortest nonzero vector) in L_I, $1/\ell_I$ is the largest distance between successive hyperplanes for a family of hyperplanes that cover all the points of L_I, where ℓ_I is the Euclidean length of a shortest nonzero vector in the dual lattice L_I^*, and n_I is the minimal number of hyperplanes that intersect $[0,1)^s$, among all families that cover L_I.

The lattice L_I and its dual can be constructed as explained in Couture and L'Ecuyer (1996) and L'Ecuyer and Couture (1997). Essentially, L_I is the lattice generated by the $k+s$ vectors $(x_{i,i_1}, x_{i,i_2}, \ldots, x_{i,i_s})/m$ for $1 \leq i \leq k$ and $\boldsymbol{e}_i$ for $1 \leq i \leq s$. These vectors are not linearly independent in general but they can be used to construct a basis by eliminating the redundant ones.

The upper bounds on d_s, ℓ_s, and n_s are also valid for d_I, ℓ_I, and n_I when $|I| = s$. That is, for each set $I = \{i_1, \ldots, i_s\}$, we have

$$d_I \leq d_s^*(m^k) = \gamma_s^{1/2} m^{-k/s}, \tag{3.25}$$
$$\ell_I \leq \ell_s^*(m^k) = \gamma_s^{1/2} m^{k/s}, \tag{3.26}$$
$$n_I \leq n_s^*(m^k) = (s! m^k)^{1/s}. \tag{3.27}$$

♣ Give numerical examples with lacunary indices... See guide of LatMRG.

3.2.9 Figures of merit based on several projections

Note that when adding a new coordinate to a given set I, the value of d_I can only decrease, whereas ℓ_I and n_I can only increase. So for sets I of different cardinalities s (i.e., projections L_I in different number of dimensions), the values of d_I are not comparable, and similarly for ℓ_I and n_I. For this reason, when considering figures of merit that take into account several sets I of different cardinalities, it is convenient to divide each quantity d_I, ℓ_I, or n_I by its upper bound in (3.25)—(3.27), to obtain a standardized value between 0 and 1 that can be compared across different values of s.

Such a figure of merit can be defined by selecting a class $\mathcal{J}$ of sets I and taking the worst (minimum) standardized value over this class. For example, if we use the quantity ℓ_I, this gives a figure of merit of the form

$$M_{\mathcal{J}} = \min_{I \in \mathcal{J}} \ell_I / \ell_{|I|}^*(m^k), \tag{3.28}$$

[4] which takes into account all index sets $I \in \mathcal{J}$. There are similar definitions using d_I and n_I. In all cases, a small value of $M_{\mathcal{J}}$ means that for at least one of the selected index sets I, the lattice L_I has a bad lattice structure. We want $M_{\mathcal{J}}$ to be as close to 1 as possible.

In Eq. (3.28), we assume that for $s > 8$, the constants γ_s in the definition of $\ell_s^*(m^{k/s})$ are replaced by their bounds, either $\bar{\gamma}_s$ or $\underline{\gamma}_s$. Since we believe that the lower bound $\underline{\gamma}_s$ should be closer to the exact γ_s than Rogers' upper bound (it is likely that $\underline{\gamma}_s = \gamma_s$ for several values of s), we prefer using this lower bound to standardize in (3.28). If ℓ_I ever exceeds $\underline{\gamma}_s m^{k/s}$, we would have found a better lattice than the best one known so far!

The criterion $M_{\mathcal{J}}$ has been widely used since Fishman and Moore III (1986) by taking $\mathcal{J}$ as the sets of the form $I = \{0, \ldots, s-1\}$ for $s \leq s_1$, where s_1 is an arbitrary integer. This choice considers vectors of successive output values, in dimensions up to s_1. L'Ecuyer and Lemieux (2000) generalized this criterion by considering also pairs, triples, quadruples, etc., of non-successive coordinates whose indices are not too far apart. They suggested taking

$$\begin{aligned} \mathcal{J} \;=\; & \{\{0, 1, \ldots, i\} : i < s_1\} \\ & \cup \{\{i_1, i_2\} : 0 = i_1 < i_2 < s_2\} \\ & \cup \cdots \\ & \cup \{\{i_1, \ldots, i_d\} : 0 = i_1 < \ldots < i_d < s_d\} \end{aligned}$$

for some positive integers $d, s_1, \ldots, s_d$. We shall denote by $M_{s_1, s_2, \ldots, s_d}$ the criterion $M_{\mathcal{J}}$ with this choice of $\mathcal{J}$. In practice, d must remain small (no more than 3 or 4), otherwise computing $M_{\mathcal{J}}$ becomes too time consuming.

[4] From Pierre: Attention: This is correct only if the number of points in each projection is m^k. We should impose this condition. Otherwise, in general it could be smaller for some projections!

The criterion (3.28) can be generalized by giving an arbitrary weight $w_I \geq 0$ to each set $I \in \mathcal{J}$. This gives:

$$\tilde{M}_{\mathcal{J}} = \min_{I \in \mathcal{J}} w_I \ell_I / \ell^*_{|I|}(m^k). \tag{3.29}$$

In Section 6.10.9, Eq. (6.89), we will examine a more general form of this criterion, and another one in which the min is replaced by the sum of the reversed terms.

Example 3.16 Let $m = 101$ and $k = 3$, as in Example 3.5. The prime factorization of $m^k - 1$ is $2^2 \times 5^2 \times 10303$, so among the $m^k - 1 = 1030300$ nonzero possibilities for the vector of coefficients (a_1, a_2, a_3), the number of vectors for which the recurrence has full period is $\phi(m^k - 1)/3 = (m^k - 1)(1/2)(4/5)(10302/10303)/3 = 137360$. For each of them, we computed M_8, the figure of merit obtained by taking $\mathcal{J}$ as the class of sets of the form $I = \{0, \ldots, s-1\}$ for $s = 4, \ldots, 8$. The vectors $(a_1, a_2, a_3) = (28, 49, 8)$ and $(73, 49, 93)$ were co-winners among all full-period generators, with $M_8 = 0.73047$. □

Example 3.17 Fishman and Moore III (1986) performed an exhaustive search to find all full-period LCGs with modulus $2^{31} - 1$ and with $M_6 \geq 0.8$. The best multiplier according to this criterion is $a = 742938285$, with $M_6 = 0.8320$. The widely-used multiplier $a = 16807$, on the other hand, had $M_6 = 0.3375$. □

Example 3.18 Computerized searches for good MRGs with respect to the criterion M_{s_1} are reported by L'Ecuyer, Blouin, and Couture (1993), L'Ecuyer and Andres (1997), L'Ecuyer (1999a), for example, for s_1 as high as 45. □

Example 3.19 L'Ecuyer and Lemieux (2000), Table 1, provide a list of good LCGs with respect to the criteria $M_{32,24,12,8}$ and $M_{32,24,16,12}$, for prime moduli ranging from 1021 to 131071. These small LCGs should not be used as RNGs; their intended use is as lattice rules for quasi-Monte Carlo integration. □

3.2.10 Bounds on ℓ_I and n_I in terms of the MRG coefficients

Propositions 3.8 and 3.9, taken from Couture and L'Ecuyer (1994) and L'Ecuyer (1997), provide upper bounds on ℓ_I and n_I in terms of the coefficients a_j for the MRG and in terms of the decomposition of m as a linear combination of powers of a for the LCG (3.6). The first proposition implies that the sum of squares of the coefficients a_j *must be large* if we want to have any chance of having a good lattice structure.

Proposition 3.8 *If I contains all indices i such that $a_{k-i} \neq 0$ (including $i = k$, because $a_0 = -1$) and if we put $a_i = 0$ for $i < 0$, then the vector $\boldsymbol{h} = (-a_{k-i_1}, \ldots, -a_{k-i_s})^{\mathrm{t}}$ belongs to the dual lattice L_I^*. Therefore,*

$$\ell_I^2 \leq \|\boldsymbol{h}\|_2^2 = 1 + a_1^2 + \cdots + a_k^2 \tag{3.30}$$

and

$$n_I \leq \|\boldsymbol{h}\|_1 - 1 = |a_1| + \cdots + |a_k|, \tag{3.31}$$

which implies that the nonzero points of $\Psi_I = L_I \cap [0,1)^s$ *are contained in at most* $\sum_{i\in I} |a_i|$ *equidistant parallel hyperplanes.*

Proof. For the special case where $I = \{0,\dots,s-1\}$, consider the vector $\boldsymbol{h} = (-a_k,\dots,-a_1,1,0,\dots,0)^\mathsf{t}$. For any vector $\boldsymbol{v} = (v_0,v_1,\dots,v_{s-1})^\mathsf{t} \in L_s$, the first $k+1$ coordinates must satisfy the recurrence $v_k = (a_1v_{k-1} + \cdots + a_kv_0) \bmod 1$, because the successive coordinates of every basis vector $\boldsymbol{v}_i$ do satisfy this recurrence. This implies that the scalar product $\boldsymbol{h}^\mathsf{t}\boldsymbol{v}$ must be an integer. Thus, $\boldsymbol{h}$ belongs to the dual lattice L_s^*. Its square Euclidean length $\|\boldsymbol{h}\|_2^2 = 1+a_1^2+\cdots+a_k^2$ provides an upper bound on ℓ_s^2 and its $\mathcal{L}_1$-length $1+|a_1|+\cdots+|a_k|$, minus 1, is an upper bound on the minimal number of hyperplanes that cover all the nonzero points. For a general set I that satisfies the conditions of the proposition, replace $\boldsymbol{h}$ by the vector $\boldsymbol{h}_I$ obtained from $\boldsymbol{h}$ by picking only the coordinates whose indices belong to I. A similar argument can be made with this $\boldsymbol{h}_I$ (Exercise 3.8).

Constructing MRGs with only two nonzero coefficients and taking these coefficients small has been a popular idea, because this makes the implementation easier and faster (L'Ecuyer, Blouin, and Couture 1993, Deng and Lin 2000, Knuth 1998). However, MRGs thus obtained have a bad structure.

Example 3.20 As a worst-case illustration, consider the widely-available additive or subtractive *lagged-Fibonacci* generator, based on the recurrence (3.2) where the two coefficients a_r and a_k are both equal to ±1:

$$x_n = (\pm x_{n-r} \pm x_{n-k}) \bmod m. \tag{3.32}$$

In this case, whenever I contains $\{0, k-r, k\}$, Proposition 3.8 tells us that $\ell_I^2 \le 3$, so the distance between the hyperplanes is at least $1/\sqrt{3}$, and that all the nonzero points of Ψ_I are covered by two hyperplanes. This happens in particular in three dimensions for $I = \{0, k-r, k\}$. This type of structure can have a dramatic effect on certain simulation problems and is a good reason for staying away from these lagged-Fibonacci generators, regardless of their parameters. □

Example 3.21 A similar problem occurs for the fast MRG proposed by Deng and Lin (2000), based on the recurrence

$$x_n = (-x_{n-1} + ax_{n-k}) \bmod m = ((m-1)x_{n-1} + ax_{n-k}) \bmod m,$$

with $a^2 < m$. If a is small, the bound $\ell_I^2 \le 1 + a^2$ implies a bad lattice structure for $I = \{0, k-1, k\}$. A more detailed analysis by L'Ecuyer and Touzin (2004) shows that this type of generator cannot have a good lattice structure even if the condition $a^2 < m$ is removed. □

Example 3.22 Another special case proposed by Deng and Xu (2003) has the form

$$x_n = a(x_{n-j_2} + \cdots + x_{n-j_s}) \bmod m. \tag{3.33}$$

In this case, for $k = j_s$ and $I = \{0, k-j_{s-1},\dots,k-j_2,k\}$, the vectors $(1,a,\dots,a)$ and $(a^*,1,\dots,1)$ both belong to the dual lattice L_I^*, where a^* is the multiplicative inverse of a modulo m. So neither a nor a^* should be small. □

To get around this structural problem when I contains certain sets of indices, Lüscher (1994) and Knuth (1998) recommend to skip some of the output values to break up the bad vectors. For the lagged-Fibonacci generator, for example, one can output k successive values produced by the recurrence, then skip the next d values, output the next k, skip the next d, and so on. A large value of d (e.g., $d = 5k$ or more) may get rid of the bad structure, but slows down the generator. See Wegenkittl and Matsumoto (1999) for further discussion.

The next result says that the lattice structure of an LCG is also guaranteed to be bad if the modulus m can be written as a linear combination of powers of the multiplier a, with small coefficients.

Proposition 3.9 (Couture, L'Ecuyer, and Tezuka 1993, L'Ecuyer 1997) *For the LCG, if m can be decomposed into powers of a as $m = \sum_{j=1}^{s} c_{i_j} a^{i_j}$ for some integers i_j and c_{i_j}, and $I = \{i_1, \ldots, i_s\}$, then*

$$\ell_I^2 \le \sum_{j=1}^{s} c_{i_j}^2. \tag{3.34}$$

Proof. One can verify that the vectors $\boldsymbol{w}_1 = m\boldsymbol{e}_1$ and $\boldsymbol{w}_j = m\boldsymbol{e}_j - a^{i_j}\boldsymbol{e}_1$ for $j = 2, \ldots, s$ form a basis of the dual lattice L_s^*. Then the vector

$$\boldsymbol{w} \;=\; \boldsymbol{w}_1 + \sum_{j=2}^{s} c_{i_j}\boldsymbol{w}_j = (c_0, c_{i_2}, \ldots, c_{i_s})^{\mathsf{t}}$$

belongs to the dual lattice and its square length is $\|\boldsymbol{w}\|^2 = \sum_{j=1}^{s} c_{i_j}^2$.

Example 3.23 The MWC generator defined in Section 3.2.6 is essentially equivalent to an LCG with modulus $m = a_0 + a_1 b + \cdots + a_k b^k$ and multiplier $a = b^{-1} \bmod m$. If we use the multiplier b instead of a, the LCG produces the same sequence of numbers, but in reverse order, so it has exactly the same lattice structure if we take the vector coordinates in reverse order. As a result, it follows from Proposition 3.9 that the value of ℓ_s for this LCG satisfies $\ell_s^2 \le a_0^2 + \cdots + a_k^2$ for $s \ge k$, which means that the lattice structure is necessarily bad unless the sum of squares of coefficients a_j is large, just like for MRGs. This bound also applies more generally to ℓ_I whenever I contains all values of j for which $a_j \ne 0$.

The add-with-carry and subtract-with-borrow generators of Marsaglia and Zaman (1991) (see Section 3.2.6) are a special case with $m = a^k \pm a^r \pm 1$ for some integers $a > 1$ and $k > r > 0$. These generators are still used in popular software. The previous proposition implies that for this type of generator, for $I = \{0, k-r, k\}$, we have $\ell_I \le \sqrt{3}$ and all nonzero vectors $(u_n, u_{n+k-r}, u_{n+k})$ produced by the approximating LCG lie in two parallel planes that are $1/\sqrt{3}$ apart. Because of this bad structure, these RNGs should not be used. □

3.2.11 MRG Implementation

The modulus m is often taken as a large prime number close to the largest integer directly representable on the computer (e.g., equal or near $2^{31} - 1$ for 32-bit computers). Since each x_{n-j} can be as large as $m - 1$, one must be careful in computing the right side of (3.2)

because the product $a_j x_{n-j}$ is typically not representable as an ordinary integer. Various techniques for computing this product modulo m are discussed and compared by Fishman (1996), L'Ecuyer and Côté (1991), L'Ecuyer (1999a), and L'Ecuyer and Simard (1999). Note that if $a_j = m - a'_j > 0$, using a_j is equivalent to using the negative coefficient $-a'_j$, which is sometimes more convenient from the implementation viewpoint. In what follows, we assume that a_j can be either positive or negative.

Floating-point implementation. One approach is to perform the arithmetic modulo m in 64-bit (double precision) floating-point arithmetic (L'Ecuyer 1999a). Under this representation, assuming that the usual IEEE floating-point standard is respected, all positive integers up to 2^{53} are represented exactly. Then, if each coefficient a_j is selected to satisfy $|a_j|(m-1) \le 2^{53}$, the product $|a_j| x_{n-j}$ will always be represented exactly and $z_j = |a_j| x_{n-j} \bmod m$ can be computed by the instructions

$$y = |a_j| x_{n-j}; \qquad z_j = y - m\lfloor y/m \rfloor.$$

Similarly, if $(|a_1| + \cdots + |a_k|)(m-1) \le 2^{53}$, $a_1 x_{n-1} + \cdots + a_k x_{n-k}$ will always be represented exactly.

Example 3.24 Let $k = 2$, $m = 2^{31} - 1$, $a_1 = 46325$, $a_2 = 1084587$. These values are taken from Table II of L'Ecuyer, Blouin, and Couture (1993). We have $(a_1 + a_2)(m-1) < 2^{52}$. If the generator's state (x_{n-1}, x_n) is represented in variables `x0` and `x1`, the following piece of Java code implements the recurrence:

```
static final double m = 2147483647;
static double x0, x1;

double x = 46325.0 * x1 + 1084587.0 * x0;
int k = (int)(x / m);   x -= k * m;
x0 = x1;  x1 = x;
```

□

Approximate factoring. A second technique, called *approximate factoring* (Bratley, Fox, and Schrage 1987, L'Ecuyer and Côté 1991), uses only the integer representation and works under the condition that $|a_j| = i$ or $|a_j| = \lfloor m/i \rfloor$ for some integer $0 < i < \sqrt{m}$. One precomputes $q_j = \lfloor m/|a_j| \rfloor$ and $r_j = m \bmod |a_j|$. Then, $z_j = |a_j| x_{n-j} \bmod m$ can be computed by

$$y = \lfloor x_{n-j}/q_j \rfloor; \qquad z = |a_j|(x_{n-j} - yq_j) - yr_j;$$
$$\text{if } z < 0 \text{ then } z_j = z + m \text{ else } z_j = z.$$

All quantities involved in these computations are integers between $-m$ and m, so no overflow can occur if m can be represented as an ordinary integer (e.g., $m < 2^{31}$ on a 32-bit computer). To show that this is true and that the method gives the correct result, we can write (replacing $|a_j|, x_{n-j}, q_j, r_j$ by a, x, q, r, for simplification):

$$\begin{aligned} ax \bmod m &= (ax - \lfloor x/q \rfloor m) \bmod m \\ &= (ax - \lfloor x/q \rfloor (aq + r)) \bmod m \\ &= (a(x - \lfloor x/q \rfloor q) - \lfloor x/q \rfloor r) \bmod m \\ &= (a(x \bmod q) - \lfloor x/q \rfloor r) \bmod m. \end{aligned}$$

When computing this last expression, the term $a(x \bmod q)$ never exceeds $a(q-1) < m$ whereas $\lfloor x/q \rfloor r < \lfloor (aq+r)/q \rfloor r \leq ar$, so it remains to show that $ar < m$. But if $a < \sqrt{m}$, then $ar < a^2 < m$ because $r < a$. Otherwise, we have $a = \lfloor m/i \rfloor$ for $i < a$, in which case $q = i$ and $r < q$. Implementations using this technique can be found, e.g., in Bratley, Fox, and Schrage (1987), L'Ecuyer (1988), L'Ecuyer, Blouin, and Couture (1993), L'Ecuyer (1996b).

```
#define norm  1.08420217248550 52e-19
#define m     9223372036854769163
#define a2    1754669720
#define q2    5256471877
#define r2    251304723
#define a3n   3182104042
#define q3    2898513661
#define r3    394451401

long long s0, s1, s2;

double MRG64bit ()
   {
   long long h, p2, p3;
   h = s0 / q3;   p3 = a3n * (s0 - h * q3) - h * r3;
   h = s1 / q2;   p2 = a2  * (s1 - h * q2) - h * r2;
   if (p3 < 0) p3 += m;
   if (p2 < 0) p2 += m - p3;  else p2 -= p3;
   if (p2 < 0) p2 += m;
   s0 = s1;  s1 = s2;  s2 = p2;
   return (p2 * norm);
   }
```

Fig. 3.2. An implementation of a 64-bit MRG using approximate factoring, in the C language.

Example 3.25 Figure 3.2 gives an example of an MRG implementation in 64-bit integer arithmetic, using approximate factoring, in the C language. The modulus and coefficients are $m = 2^{63} - 6645$, $a_1 = 0$, $a_2 = 1754669720$, and $a_3 = -3182104042$. The values of r_j, q_j, and $1/m$ are precomputed and defined as constants in the code. The implementation assumes that all integers from $-m_1$ and m_1 are represented exactly in the "`long long`" type. The variables `s0, s1, s2` must be initialized to non-negative integers less than m and not all zero before the first call. This MRG is *one component* of a combined MRG proposed by L'Ecuyer (1999a). (It is not intended to be used alone.) □

Power-of-two decomposition. The *powers-of-two decomposition* approach selects coefficients a_j that can be written as a sum or difference of a small number of powers of 2 (Wu 1997, L'Ecuyer and Simard 1999, L'Ecuyer and Touzin 2000). For example, one may take $a_j = \pm 2^q \pm 2^r$ and $m = 2^e - h$ for some positive integers q, r, e, and h. To compute $y = 2^q x \bmod m$, decompose $x = z_0 + 2^{e-q} z_1$ (where $z_0 = x \bmod 2^{e-q}$) and observe that

$$y = 2^q(z_0 + 2^{e-q}z_1) \bmod (2^e - h) = (2^q z_0 + hz_1) \bmod (2^e - h).$$

Suppose now that

$$h < 2^q \quad \text{and} \quad h(2^q - (h+1)2^{-e+q}) < m. \tag{3.35}$$

Then $2^q z_0 \le 2^e - 2^q < m$ and $hz_1 \le h(m-1)/2^{e-q} = h(2^e - h - 1)/2^{e-q} = h(2^q - (h+1)2^{-e+q}) < m$, so each of the two terms in $2^q z_0 + hz_1$ is less than m. To compute y, obtain $2^q z_0$ by shifting z_0 by q positions to the left, then add h times z_1, and subtract m if the result exceeds $m-1$. If h is a power of 2, this requires only shifts, additions, and subtractions (no multiplication). Intermediate results never exceed $2m-1$. Things simplify further if $q = 0$ or $q = 1$ or $h = 1$. For $h = 1$, y is obtained simply by swapping the blocks of bits z_0 and z_1 (Wu 1997).

To multiply x by $a = \pm 2^q \pm 2^r$, repeat this operation with r instead of q, and add (or subtract) the results modulo m. Figure 3.3 gives an example of how this technique can be implemented in C, for $m = 2^{30} - 35$ and $a = 2^{15} + 2^{13}$.

```
#define m     1073741789  /* 2^30 - 35   */
#define h     35
#define q     15
#define emq   15          /* e - q       */
#define mask1 32767       /* 2^(e-q) - 1 */
#define r     13
#define emr   17          /* e - r       */
#define mask2 131071      /* 2^(e-r) - 1 */
#define norm  1.0 / m

long x;

double axmodm () {
   unsigned long k, x0, x1;
   x0 = x & mask1;   x1 = x >> emq;
   k = (x0 << q) + h * x1;
   x0 = x & mask2;   x1 = x >> emr;
   k += (x0 << r) + h * x1;
   if (k < m) x = k;
   else if (k < m * 2) x = k - m;
   else x = k - m * 2;
   return x * norm;
  }
```

Fig. 3.3. Implementation of an LCG with $a = 2^{15} + 2^{13}$ and $m = 2^{30} - 35$

L'Ecuyer and Simard (1999) pointed out that LCGs with parameters of the form $m = 2^e - 1$ and $a = \pm 2^q \pm 2^r$ have bad statistical properties because the recurrence does not "mix the bits" well enough. However, good and fast combined MRGs can be obtained via the power-of-two decomposition method, as explained in L'Ecuyer and Touzin (2000).

Equal coefficients. Another idea for improving efficiency is to take all nonzero coefficients a_j equal to the same constant a (Marsaglia 1996, Deng and Xu 2003). Then, computing the right side of (3.2) requires a single multiplication. Deng and Xu (2003) provide specific parameter sets and concrete implementations for MRGs of this type, for prime m near 2^{31}, and $k = 102$, 120, and 1511. Example 3.22 shows a limitation on the quality of the lattice structure for this type of RNG.

Power-of-two modulus. Taking m equal to a power of two is a bad idea for LCGs and MRGs, but it is generally the way to go for MWC generators. When $m = 2^e$, the "mod m" operation is very easy to perform: just keep the e least significant bits and mask-out all others.

♣ Elaborate on MWC implementation ...

3.2.12 Combined MRGs and LCGs

The conditions that make MRG implementations run faster (e.g., only two nonzero coefficients both close to zero) are generally in conflict with those required for having a good lattice structure and statistical robustness. *Combined MRGs* are one solution to this problem. Consider C distinct MRGs evolving in parallel, based on the recurrences

$$x_{c,n} = (a_{c,1}x_{c,n-1} + \cdots + a_{c,k}x_{c,n-k}) \bmod m_c \tag{3.36}$$

where $a_{c,k} \neq 0$, for $c = 1, \ldots, C$. Let $\delta_1, \ldots, \delta_C$ be arbitrary integers,

$$z_n = (\delta_1 x_{1,n} + \cdots + \delta_C x_{C,n}) \bmod m_1, \qquad u_n = z_n/m_1, \tag{3.37}$$

and

$$w_n = (\delta_1 x_{1,n}/m_1 + \cdots + \delta_C x_{C,n}/m_C) \bmod 1. \tag{3.38}$$

This defines two RNGs, with output sequences $\{u_n, n \geq 0\}$ and $\{w_n, n \geq 0\}$.

Suppose that the m_c are pairwise relatively prime, that each recurrence (3.36) is purely periodic with period ρ_c, and that δ_c and m_c have no common factor, for each c. Let $m = m_1 \cdots m_C$ and let ρ be the least common multiple of $\rho_1, \ldots, \rho_C$. Under these conditions, the following was proved by L'Ecuyer and Tezuka (1991) for $k = 1$ and L'Ecuyer (1996a) for $k > 1$.

Proposition 3.10 *(a) the sequence (3.38) is exactly equivalent to the output sequence of an MRG with (composite) modulus m, with coefficients a_j given by*

$$a_j = \left(\sum_{c=1}^{C} \frac{a_{c,j} n_c m}{m_c} \right) \bmod m \tag{3.39}$$

for $j = 1, \ldots, k$, where $n_c = (m/m_c)^{-1} \bmod m_c = (m/m_c)^{m_c - 2} \bmod m_c$ for each c.

(b) the two sequences in (3.37) and (3.38) have period ρ; and

(c) if both sequences have the same initial state, then $u_n = w_n + \epsilon_n$ where $\max_{n \geq 0} |\epsilon_n|$ can be bounded explicitly by a constant $\epsilon > 0$ that is very small when the m_c are close to one another. For the case where $C = 2$, $\delta_1 = -\delta_2 = 1$, and $m_1 > m_2$, we have $\epsilon = (m_1 - m_2)(m_2 - 1)/(m_1 m_2)$.

Thus, combining MRGs can be viewed as a practical way of implementing an MRG with a large m and several large nonzero coefficients. The idea is to cleverly select the components so that: (1) each one is easy to implement efficiently (e.g., has only two small nonzero coefficients) and (2) the MRG that corresponds to the combination has a good lattice

structure. If each m_c is prime and if each component c has maximal period $\rho_c = m_c^k - 1$, then each ρ_c is even and ρ cannot exceed $\rho_1 \cdots \rho_C / 2^{C-1}$. For $C = 2$, for example, the best we can achieve is $\rho = (m_1^k - 1)(m_2^k - 1)/2 \approx m/2$. The generator will then have two large cycles of length ρ, and three other cycles of respective lengths $m_1^k - 1$, $m_2^k - 1$, and 1. These shorter cycles correspond to the situation where one of the two components is in the zero state.

Tables of good parameters for combined MRGs of different sizes that reach this upper bound are given in L'Ecuyer (1999a) and L'Ecuyer and Touzin (2000), together with C implementations.

Example 3.26 The `MRG32k3a` generator proposed by L'Ecuyer (1999a) is a combined MRG with $C = 2$ components of order $k = 3$. The components are defined by (3.36), with $m_1 = 2^{32} - 209$, $a_{1,1} = 0$, $a_{1,2} = 1403580$, $a_{1,3} = -810728$, $m_2 = 2^{32} - 22853$, $a_{2,1} = 527612$, $a_{2,2} = 0$, $a_{2,3} = -1370589$. The combination is defined by $z_n = (x_{1,n} - x_{2,n}) \bmod m_1$. The MRG that corresponds to this combination has order $k = 3$, modulus $m = m_1 m_2 = 18446645023178547541$ and multipliers $a_1 = 18169668471252892557$, $a_2 = 3186860506199273833$, and $a_3 = 8738613264398222622$. Its period is $(m_1^3 - 1)(m_2^3 - 1)/2 \approx 2^{191} \approx 3.1 \times 10^{57}$. The output is defined by $u_n = z_n/(m_1 + 1)$ if $z_n > 0$ and $u_n = z_n/(m_1 + 1)$ otherwise.

This generator was found by a computerized search of several hours. In the search, the values of C, k, m_1, and m_2 were fixed, and the constraints $a_{1,1} = a_{2,2} = 0$ and $(|a_{c,0}| + |a_{c,1}| + |a_{c,2}|)(m_c - 1) < 2^{53}$ were imposed to make sure that the recurrence of each component could be implemented easily in standard floating-point arithmetic. The selected values of m_1 and m_2 have the properties that both $h_c = (m_c - 1)/2$ and $r_c = (m_c^3 - 1)/(m_c - 1) = m_c^2 + 1$ are prime, for $c = 1$ and 2. The conditions of Proposition 3.5 are then easier to verify. Moreover, the numbers h_1, r_1, h_2, r_2 are all distinct, which implies that $(m_1^3 - 1)$ and $(m_2^3 - 1)$ have only 2 as a common factor.

The figure of merit was the worst-case standardized spectral test value in up to 32 dimensions, i.e., we wanted to maximize $M_{\mathcal{J}}$ with $\mathcal{J} = \{\{0, \ldots, i\} : i < 32\}$. The retained generator (given above) has $M_{\mathcal{J}} = 0.6336$. It also has $M_{\mathcal{J}} = 0.6225$ if we go up to 45 dimensions instead of 32.

This particular generator is implemented in several simulation and statistical packages such as MATLAB, SAS, Arena, Automod, Witness, Simul8, ns-3, and SSJ, for instance. □

♣ Add another good example, with power of 2 decomposition.

3.3 Generators Based on Recurrences Modulo 2

3.3.1 A General Framework

It appears natural to exploit the fact that computers work in binary arithmetic and to design RNGs defined directly in terms of bit strings and sequences. We do this under the following framework, taken from L'Ecuyer and Panneton (2002) and L'Ecuyer and Panneton (2009). Let $\mathbb{F}_2$ denote the finite field with two elements, 0 and 1, in which the operations are equivalent to addition and multiplication modulo 2. Consider the RNG defined by a matrix

linear recurrence over $\mathbb{F}_2$, as follows:

$$\boldsymbol{x}_n = \boldsymbol{A}\boldsymbol{x}_{n-1}, \tag{3.40}$$

$$\boldsymbol{y}_n = \boldsymbol{B}\boldsymbol{x}_n, \tag{3.41}$$

$$u_n = \sum_{\ell=1}^{w} y_{n,\ell-1}2^{-\ell} = .y_{n,0}\ y_{n,1}\ y_{n,2}\ \cdots, \tag{3.42}$$

where $\boldsymbol{x}_n = (x_{n,0}, \ldots, x_{n,k-1})^{\mathsf{t}} \in \mathbb{F}_2^k$ is the k-bit *state vector* at step n, $\boldsymbol{y}_n = (y_{n,0}, \ldots, y_{n,w-1})^{\mathsf{t}} \in \mathbb{F}_2^w$ is the w-bit *output vector* at step n, k and w are positive integers, $\boldsymbol{A}$ is a $k \times k$ *transition matrix* with elements in $\mathbb{F}_2$, $\boldsymbol{B}$ is a $w \times k$ *output transformation matrix* with elements in $\mathbb{F}_2$, and $u_n \in [0, 1)$ is the *output* at step n. All operations in (3.40) and (3.41) are performed in $\mathbb{F}_2$.

Since there are $2^k - 1$ nonzero states, the *maximal period* for this RNG is $2^k - 1$. To study the periodicity, we start with the *characteristic polynomial* of the matrix $\boldsymbol{A}$, defined by

$$P(z) = \det(z\boldsymbol{I} - \boldsymbol{A}) = z^k - \alpha_1 z^{k-1} - \cdots - \alpha_{k-1}z - \alpha_k,$$

where $\boldsymbol{I}$ is the identity matrix and each α_j is in $\mathbb{F}_2$. The linear recurrence

$$x_n = (\alpha_1 x_{n-1} + \cdots + \alpha_k x_{n-k}) \qquad (\text{in } \mathbb{F}_2) \tag{3.43}$$

also has this characteristic polynomial $P(z)$. We will assume henceforth that $\alpha_k = 1$, so that the recurrence (3.43) has *order* k and it is purely periodic. In practice we always construct $\boldsymbol{A}$ so that this is true.

In view of its definition in (3.40), the sequence $\{\boldsymbol{x}_n, n \geq 0\}$ must satisfy the recurrence with characteristic polynomial $P(z)$ (and any other polynomial that is a multiple of the minimal polynomial $Q(z)$ of the recurrence (3.43), which is defined below); that is,

$$\boldsymbol{x}_n = (\alpha_1 \boldsymbol{x}_{n-1} + \cdots + \alpha_k \boldsymbol{x}_{n-k}) \qquad (\text{in } \mathbb{F}_2). \tag{3.44}$$

Therefore, the sequence $\{x_{n,j}, n \geq 0\}$ obeys (3.43) for each j, $0 \leq j < k$. The sequence $\{y_{n,j}, n \geq 0\}$, for $0 \leq j < w$, also obeys that same recurrence. However, these sequences may also obey recurrences of order smaller than k.

For any periodic sequence in $\mathbb{F}_2$, there is a single linear recurrence of *minimal order* obeyed by this sequence. The characteristic polynomial $Q(z)$ of this recurrence, called the *minimal polynomial* of the sequence, can be computed by the Berlekamp-Massey algorithm (Massey 1969). It is the polynomial $Q(z)$ of smallest degree such that $Q(\boldsymbol{A}) = 0$. The sequences $\{x_{n,j}, n \geq 0\}$ may have different minimal polynomials for different values of j, and also different minimal polynomials than the sequences $\{y_{n,j}, n \geq 0\}$. But all these minimal polynomials must be divisors of $P(z)$. If $P(z)$ is *irreducible* (it has no divisor other than 1 and itself), then it must be the minimal polynomial of all these sequences. Reducible polynomials $P(z)$ do occur when we combine generators (Section 3.3.3); in that case, $P(z)$ is typically the minimal polynomial of the output bit sequences $\{y_{n,j}, n \geq 0\}$ as well, but the sequences $\{x_{n,j}, n \geq 0\}$ often have minimal polynomials (divisors of $P(z)$) of much smaller degrees.

We already saw that the recurrence (3.43) has maximal period $2^k - 1$ if and only if $P(z)$ is a primitive polynomial over $\mathbb{F}_2$. This is a stronger condition than irreducibility. An irreducible

polynomial $P(z)$ over $\mathbb{F}_2$ is also primitive if and only if for all prime divisors p_i of $r = 2^k - 1$, $z^{r/p_i} \not\equiv 1 \bmod P(z)$. When k is large, a good (recommended) way to verify primitivity is to verify irreducibility first, then check the second condition. Note that if r is prime (this type of prime is called a *Mersenne prime*), the second condition is automatically verified. This is the main reason why the proposed large-period generators often have a period that is a Mersenne prime (Matsumoto and Kurita 1994, Matsumoto and Nishimura 1998). But Mersenne primes are in very limited supply, so we may sometimes decide to adopt composite values of r, especially if we want to combine two or more generators of similar orders. In that case, Proposition 3.6 suggests that we should avoid integers r whose prime factorization contains many small factors, because it reduces the chances of finding primitive polynomials. The primitivity condition is also more expensive to verify when r has more distinct factors. Simple algorithms to verify irreducibility and primitivity of polynomials over $\mathbb{F}_2$ are given in L'Ecuyer and Panneton (2009).

When constructing an RNG, we usually fix k and impose a special structure on $\boldsymbol{A}$, so that a fast implementation of the RNG is available. Then we search for matrices $\boldsymbol{A}$ that satisfy these constraints and have a primitive characteristic polynomial. For combined generators (Section 3.3.3), we may do this separately for each component, retain a short list of maximal-period candidates for each component, and then analyze the uniformity of all combinations obtained by taking one component from each list.

3.3.2 Jumping Ahead

Jumping ahead in (3.40) from $\boldsymbol{x}_n$ to $\boldsymbol{x}_{n+\nu}$ for large ν can be done in the same way as for the MRG (Section 3.2.3), at least in principle. This is fine if k is small, but becomes slow for large values of k, say over a few hundreds. Precomputing the matrix $\boldsymbol{A}^\nu \bmod 2$ requires $O(k^3 \log \nu)$ operations if the matrix multiplications are done by the straightforward method, and $O(k^2)$ words of memory are needed to store this binary matrix. Then, each multiplication of $\boldsymbol{x}_n$ by this binary matrix (modulo 2) requires $O(k^2)$ operations.

Haramoto et al. (2008) have proposed a more efficient method. It represents the state $\boldsymbol{x}_{n+\nu}$ as $g_\nu(\boldsymbol{A})\boldsymbol{x}_n$, where $g_\nu(z) = \sum_{j=0}^{k-1} d_j z^j$ is a precomputed polynomial of degree less than k, with coefficients in $\mathbb{F}_2$. The product

$$g_\nu(\boldsymbol{A})\boldsymbol{x}_n = \sum_{j=0}^{k-1} d_j \boldsymbol{A}^j \boldsymbol{x}_n = \sum_{j=0}^{k-1} d_j \boldsymbol{x}_{n+j}$$

can be computed simply by running the generator for $k-1$ steps to obtain $\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{n+k-1}$ and adding (modulo 2) the $\boldsymbol{x}_{n+j}$'s for which $d_j = 1$. For large k, the cost is dominated by these additions. Their number can be reduced (e.g., by a factor of about 4 when $k = 19937$) by using a sliding window technique, as explained in Haramoto et al. (2008). This method still requires $O(k^2)$ operations but with a smaller hidden constant and (most importantly) much less memory than the standard matrix multiplication. For k in the thousands, it remains rather slow. Adopting a combined generator, as discussed next, is a good way to speed up the jumping-ahead when k is large; in that case, the ν-step jumping-ahead is done separately for each component.

3.3.3 Combined $\mathbb{F}_2$-Linear Generators

Linear generators over $\mathbb{F}_2$ can be combined in a similar way as MRGs, and for similar reasons. The idea is to select simple components that can run fast, and combine them in a way that the combination has a more complicated structure and a guaranteed good uniformity of the point sets Ψ_I, for the sets of indexes I deemed important.

We consider the following class of combinations. For some integer $C > 1$, we take C distinct recurrences of the form (3.40)–(3.41), where the cth recurrence has parameters $(k, w, \boldsymbol{A}, \boldsymbol{B}) = (k_c, w, \boldsymbol{A}_c, \boldsymbol{B}_c)$ and state $\boldsymbol{x}_{c,n}$ at step n, for $c = 1, \ldots, C$. The output of the combined generator at step n is defined by

$$\begin{aligned} \boldsymbol{y}_n &= \boldsymbol{B}_1\boldsymbol{x}_{1,n} \oplus \cdots \oplus \boldsymbol{B}_C\boldsymbol{x}_{C,n}, \\ u_n &= \sum_{\ell=1}^{w} y_{n,\ell-1} 2^{-\ell}, \end{aligned}$$

where $\oplus$ denotes the bitwise exclusive-or (xor) operation. The resulting combined generator is equivalent to the generator (3.40)–(3.42) with $k = k_1 + \cdots + k_C$, $\boldsymbol{A} = \text{diag}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_C)$, and $\boldsymbol{B} = (\boldsymbol{B}_1, \ldots, \boldsymbol{B}_C)$. If $P_c(z)$ is the characteristic polynomial of $\boldsymbol{A}_c$ for each c, then the characteristic polynomial of $\boldsymbol{A}$ is $P(z) = P_1(z) \cdots P_C(z)$. This polynomial is obviously reducible, so the combined RNG cannot have maximal period $2^k - 1$. Its period ρ is in fact the least common multiple of the periods ρ_c of its components (Tezuka and L'Ecuyer 1991, Tezuka 1995). So if we select the parameters so that each component has maximal period $\rho_c = 2^{k_c} - 1$ and the ρ_c are *pairwise relatively prime* (no pair has a common factor), then the combined generator has period $\rho = \prod_{c=1}^{C}(2^{k_c} - 1)$, the product of the periods of the components. This is very close to $2^k - 1$ when the k_c's are not too small. As an illustration, one may have four components of periods $2^{63} - 1$, $2^{58} - 1$, $2^{55} - 1$, $2^{47} - 1$, so the state of each component fits a 64-bit integer, and the overall period is near 2^{223}. The only states that are not visited by the combined generator in its main cycle are those in which at least one component is in the zero state. Constructions of this form can be found in Tezuka and L'Ecuyer (1991), Wang and Compagner (1993), L'Ecuyer (1999b), Tezuka (1995). Many of the best linear generators over $\mathbb{F}_2$ are defined by such combinations.

3.3.4 Measures of Uniformity

Easily computable uniformity measures for the point sets Ψ_I produced by $\mathbb{F}_2$-linear RNGs are defined by exploiting the fact that Ψ_I is a linear space, as was the case for MRGs (L'Ecuyer 1996b, L'Ecuyer and Panneton 2009, Tezuka 1995). More specifically, suppose we select an arbitrary vector $\boldsymbol{q} = (q_1, \ldots, q_s)$ of non-negative integers, and we partition the unit hypercube $[0, 1)^s$ into 2^{q_j} intervals of the same length along axis j, for each j. This determines a partition of $[0, 1)^s$ into $2^{q_1+\cdots+q_s}$ rectangular boxes of the same size and shape. We would like the points of Ψ_I to be evenly spread between those boxes. The best that can happen is that Ψ_I has exactly 2^t points in each box, where the $t = k - q_1 - \cdots - q_s \geq 0$. We then say that Ψ_I is ***q**-equidistributed*. This means that among the 2^k points $(u_{i_1}, \ldots, u_{i_s})$ of Ψ_I, if we consider all $(k - t)$-bit vectors formed by the q_j most significant bits of u_{i_j} for $j = 1, \ldots, s$,

each of the 2^{k-t} possibilities occurs exactly the same number of times. Of course, this is possible only if $q_1 + \cdots + q_s \le k$.

Example 3.27 The point set P_n displayed in Figure Figure 1.15 has the same type of linear structure as the set Ψ_2 from an $\mathbb{F}_2$-linear generator with $k = 8$. The figure illustrates the fact that this point set is $\boldsymbol{q}$-equidistributed for $\boldsymbol{q} = (3, 5)$ (left panel) and $\boldsymbol{q} = (6, 2)$ (right panel). □

When $k - t$ (the number of rectangles) is large, $\boldsymbol{q}$-equidistribution might seem very hard to achieve. It is indeed extremely unlikely for an arbitrary point set, but here it turns out to occur naturally because Ψ_I is a linear space, as we now explain. Recall that the box in which a point falls is determined by $k - t$ bits, with q_j bits taken from the jth coordinate. If $I = \{i_1, \ldots, i_s\}$, these q_j bits are the q_j most significant bits of $\boldsymbol{y}_{i_j}$, which are the first q_j bits of the vector $\boldsymbol{B}\boldsymbol{x}_{i_j} = \boldsymbol{B}\boldsymbol{A}^{i_j}\boldsymbol{x}_0$ when the initial state is $\boldsymbol{x}_0$. Then the binary vector $\boldsymbol{z}_0 \in \mathbb{F}_2^{k-t}$ that contains these $k - t$ bits can be written as

$$\boldsymbol{z}_0 = \boldsymbol{M}_{\boldsymbol{q}}\boldsymbol{x}_0$$

where $\boldsymbol{M}_{\boldsymbol{q}}$ is the $(k - t) \times k$ binary matrix formed by the first q_1 rows of $\boldsymbol{B}\boldsymbol{A}^{i_1}$, followed by the first q_2 rows of $\boldsymbol{B}\boldsymbol{A}^{i_2}$, ..., and ending with the first q_s rows of $\boldsymbol{B}\boldsymbol{A}^{i_s}$.

Clearly, Ψ_I is $\boldsymbol{q}$-equidistributed if and only if $\boldsymbol{z}_0$ takes each of its 2^{k-t} possible values exactly 2^t times when $\boldsymbol{x}_0$ runs through its 2^k possibilities. Obviously, this can happen only if the linear mapping defined by $\boldsymbol{M}_{\boldsymbol{q}}$ is surjective from $\mathbb{F}_2^k$ to $\mathbb{F}_2^{k-t}$, which occurs if and only if this matrix has full rank $k - t$ (i.e., its $k - t$ rows are linearly independent). In that case, the image of the linear mapping has dimension $k - t$ over $\mathbb{F}_2$, and therefore the kernel (the linear subspace of vectors that are mapped to 0) must have dimension t. It then follows from standard linear algebra that 2^t distinct vectors from $\mathbb{F}_2^k$ are mapped to each rectangle, so $\boldsymbol{q}$-equidistribution holds. We have just shown that Ψ_I is $\boldsymbol{q}$-equidistributed if and only if $\boldsymbol{M}_{\boldsymbol{q}}$ has rank $k - t$. Thus, $\boldsymbol{q}$-equidistribution can easily be verified by constructing the matrix $\boldsymbol{M}_{\boldsymbol{q}}$ and checking its rank via (binary) Gaussian elimination. This is a major motivation for adopting this measure of uniformity: it is easily computable even when k is large.

The matrix $\boldsymbol{M}_{\boldsymbol{q}}$ that corresponds to Ψ_I can be constructed as follows. For $j \in \{1, \ldots, k\}$, start the generator in initial state $\boldsymbol{x}_0 = \boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is the unit vector with a 1 in position j and zeros elsewhere, and run the generator for i_s steps. Record the q_1 most significant bits of the output at step i_1, the q_2 most significant bits of the output at step i_2, ..., and the q_s most significant bits of the output at step i_s. These bits form the jth column of the matrix $\boldsymbol{M}_{\boldsymbol{q}}$. Repeat this for all columns j.

Ideally, we would like to have $\boldsymbol{q}$-equidistribution for most of the vectors $\boldsymbol{q}$ that satisfy the condition $q_1 + \cdots + q_s \le k$. But when k is very large (say, over a few hundreds), checking $\boldsymbol{q}$-equidistribution for all those vectors is practically infeasible, because there are just too many of them. For this reason, it is customary to consider only a smaller class of vectors $\boldsymbol{q}$, namely those for which all the coordinates q_j are equal to a given constant $\ell \ge 1$. That is, we look at the ℓ most significant bits for each coordinate, which amounts to partitioning the unit cube $[0, 1)^s$ into $2^{s\ell}$ cubic boxes. If each of those cubes contains the same number of points from Ψ_I, i.e., if Ψ_I is $\boldsymbol{q}$-equidistributed for $\boldsymbol{q} = (\ell, \ldots, \ell)$, then it is called *s-distributed with ℓ bits of accuracy*. The largest value of ℓ for which this holds is called the *resolution* of the

set Ψ_I and is denoted by ℓ_I. It cannot exceed $\ell_s^* = \min(\lfloor k/s \rfloor, w)$. We define the *resolution gap* of Ψ_I as

$$\delta_I = \ell_s^* - \ell_I.$$

We want it to be as small as possible for each I.

Similar to what we did in Section 3.2.9, potential figures of merit can then be defined by

$$\Delta_{\mathcal{J},\infty} = \max_{I \in \mathcal{J}} \omega_I \delta_I \quad \text{and} \quad \Delta_{\mathcal{J},1} = \sum_{I \in \mathcal{J}} \omega_I \delta_I$$

for some non-negative weights ω_I, where $\mathcal{J}$ is a preselected class of index sets I. The weights are often taken all equal to 1.

We also denote by s_ℓ the largest dimension s for which Ψ_s is s-distributed with ℓ bits of accuracy, and we define the *dimension gap* for ℓ bits of accuracy as

$$\tilde{\delta}_\ell = s_\ell^* - s_\ell,$$

where $s_\ell^* = \lfloor k/\ell \rfloor$ is an upper bound on s_ℓ. We may then consider the *worst-case weighted dimension gap* and the *weighted sum of dimension gaps*, defined as

$$\tilde{\Delta}_\infty = \max_{1 \le \ell \le w} \omega_\ell \tilde{\delta}_\ell \quad \text{and} \quad \tilde{\Delta}_1 = \sum_{\ell=1}^{w} \omega_\ell \tilde{\delta}_\ell$$

for some non-negative weights ω_ℓ, as alternative figures of merit for our generators. Often, the weights are all 1 and the word "weighted" is removed from these definitions.

When $\tilde{\Delta}_\infty = \tilde{\Delta}_1 = 0$ with $\omega_\ell = 1$ for all $\ell \le w$, the RNG is said to be *maximally equidistributed* (ME) or *asymptotically random* for the word size w (L'Ecuyer 1996b, Tezuka 1995, Tootill, Robinson, and Eagle 1973). This property ensures perfect equidistribution of all sets Ψ_s, for any partition of the unit hypercube into subcubes of equal sizes, as long as $\ell \le w$ and the number of subcubes does not exceed 2^k, the number of points in Ψ_s.

The ME property puts no constraint on what could happen with the points when $s\ell > k$. For example, if $k = 31$, the ME property would imply that for $s = 8$ we have equidistribution for $\ell = 3$ bits of resolution, with $2^7 = 128$ points per box, but it could happen that there are still 128 points per box when we divide further by taking $\ell = 4$. This would give 2^{24} boxes with 128 points each, and $2^{32} - 2^{24}$ empty boxes. To prevent this type of behavior, L'Ecuyer 1996b proposed an additional requirement called the collision-free property. For $q_1 + \cdots + q_s \ge k$, we say that Ψ_I is ***q*-collision-free** (CF) if no box contains more than one point. This occurs if and only if $\boldsymbol{M_q}$ has rank k, where $\boldsymbol{M_q}$ is still a matrix with $k - t$ rows and k columns, but t is now negative. If Ψ_s is ME and also $(\ell, \ldots, \ell)$-collision-free whenever $s\ell \ge k$, we say that the RNG is *collision-free* (CF).

Large-period ME (or almost ME) and ME-CF generators are given by L'Ecuyer (1999b), L'Ecuyer and Panneton (2002), Panneton and L'Ecuyer (2004), and Panneton, L'Ecuyer, and Matsumoto (2006), for example.

For a combined generator as in Section 3.3.3, $\boldsymbol{M_q}$ can be constructed by first constructing the corresponding matrices $\boldsymbol{M_q}^{(c)}$ for the individual components, and simply juxtaposing these matrices, as suggested in L'Ecuyer (1999b). This is yet another advantage of combined

generators: the matrices $\boldsymbol{M_q}$ are easier to construct, by decomposition. To describe how this is done, let $\Psi_I^{(c)}$ be the point set that corresponds to component c alone, and let

$$\boldsymbol{x}_0 = \begin{pmatrix} \boldsymbol{x}_0^{(1)} \\ \vdots \\ \boldsymbol{x}_0^{(C)} \end{pmatrix}$$

where $\boldsymbol{x}_0^{(c)}$ is the initial state for component c. If $\boldsymbol{z}_0^{(c)}$ is the $(k-t)$-bit vector relevant for the $\boldsymbol{q}$-equidistribution of $\Psi_I^{(c)}$ alone, then we have $\boldsymbol{z}_0^{(c)} = \boldsymbol{M_q}^{(c)}\boldsymbol{x}_0^{(c)}$ for some $(k-t) \times k_c$ binary matrix $\boldsymbol{M_q}^{(c)}$ that can be constructed as explained earlier. Note that the point set Ψ_I can be written as the direct sum

$$\Psi_I = \Psi_I^{(1)} \oplus \cdots \oplus \Psi_I^{(C)} = \{\boldsymbol{u} = \boldsymbol{u}^{(1)} \oplus \cdots \oplus \boldsymbol{u}^{(C)} \mid \boldsymbol{u}^{(c)} \in \Psi_I^{(c)} \text{ for each } c\},$$

where $\oplus$ denotes the bitwise sum of binary expansions, coordinate by coordinate, and observe that

$$\boldsymbol{M}_q\boldsymbol{x}_0 = \boldsymbol{z}_0 = \boldsymbol{z}_0^{(1)} \oplus \cdots \oplus \boldsymbol{z}_0^{(C)} = \boldsymbol{M_q}^{(1)}\boldsymbol{x}_0^{(1)} \oplus \cdots \oplus \boldsymbol{M_q}^{(C)}\boldsymbol{x}_0^{(C)}.$$

This means that $\boldsymbol{M_q}$ is just the juxtaposition of $\boldsymbol{M_q}^{(1)}, \cdots, \boldsymbol{M_q}^{(C)}$. That is, $\boldsymbol{M_q}^{(1)}$ gives the first k_1 columns of $\boldsymbol{M_q}$, $\boldsymbol{M_q}^{(2)}$ gives the next k_2 columns, and so on.

For very large values of k, the matrix $\boldsymbol{M_q}$ is expensive to construct and reduce. An alternative method studied in Couture and L'Ecuyer (2000), based on the computation of the shortest nonzero vector in a lattice of formal series, can be used to verify the $(\ell, \ldots, \ell)$-equidistribution, and is more efficient than computing the rank of $\boldsymbol{M}_q$ when k is very large; see Section 3.3.5.

The figures of merit defined above look at the most significant bits of the output values, but give little importance to the least significant bits. We could of course extend them so that they also measure the equidistribution of the least significant bits, simply by using different bits to construct the output values and computing the corresponding $\boldsymbol{q}$-equidistributions via the matrices $\boldsymbol{M_q}$ that correspond to those bits. But these computations are cumbersome and expensive in general, because there are too many ways of selecting which bits are to be considered. The Tausworthe (or LFSR) generators defined in Section 3.3.6 have the interesting property that if all output values are multiplied by a given power of two, modulo 1, all equidistribution properties remain unchanged. In other words, their least significant bits have the same equidistribution as the most significant ones. Panneton and L'Ecuyer (2010) call such generators *resolution-stationary* and prove that a linear generator over $\mathbb{F}_2$ is resolution-stationary if and only if it is equivalent to a Tausworthe generator.

Aside from good equidistribution properties, linear generators over $\mathbb{F}_2$ are also required to have characteristic polynomials $P(z)$ whose number N_1 of nonzero coefficients is not too far from half the degree, i.e., near $k/2$ (Compagner 1991, Wang and Compagner 1993). One intuitive reason for this is that if N_1 is very small and if the state $\boldsymbol{x}_n$ contains many 0's and only a few 1's, then there is a high likelihood that the $N_1 - 1$ bits used to determine any given new bit of the next state are all zero, in which case this new bit will also be zero. So the next state is likely to contain much more 0's than 1's, and this may go on for a large number of steps. Cryptologists would say that the recurrence has *low diffusion capacity* and Markov chain experts would call it *slowly mixing*.

An illustration of this with the Mersenne twister can be found in Panneton, L'Ecuyer, and Matsumoto (2006). In particular, generators for which $P(z)$ is a trinomial or a pentanomial, which have been often used in the past, should be avoided. They fail simple statistical tests (Lindholm 1968, Matsumoto and Kurita 1996). The fraction N_1/k of nonzero coefficients in $P(z)$ can be used as a secondary figure of merit for an RNG.

Other measures of uniformity are popular in the context where k is small and the entire point set Ψ_s is used for quasi-Monte Carlo integration (see Section 6.10.2); these measures can be computed only when k is small.

♣ Add examples: take a small LFSR, show how to construct $\boldsymbol{M_q}$ for one or two vectors $\boldsymbol{q}$, show how to compute the resolution and dimension gaps. Show how to construct $\boldsymbol{M_q}$ for a combined generator with two components.

♣ Add an example where the gaps are zero (ME-CF): see later.

3.3.5 Lattice Structure in Spaces of Polynomials and Formal Series

The equidistribution measures studied in Section 3.3.4 do not apply to MRGs, because the relevant bits cannot be expressed as a linear function of the bits of the MRG's state. In general, MRGs do not have this type of equidistribution property in base 2. Reciprocally, $\mathbb{F}_2$-linear generators defined via (3.40)–(3.42) do not have a lattice structure in the real space like MRGs do. However, they do have a lattice structure in a space of formal series, as explained by Tezuka (1995), Couture and L'Ecuyer (2000), and L'Ecuyer and Panneton (2009). The real space $\mathbb{R}$ is replaced by the space $\mathbb{L}_2$ of formal power series with coefficients in $\mathbb{F}_2$, of the form $\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}$ for some integer ω. In that setting, the lattices have the form

$$\mathcal{L}_s = \left\{ \boldsymbol{v}(z) = \sum_{j=1}^{s} h_j(z)\boldsymbol{v}_j(z) \text{ such that each } h_j(z) \in \mathbb{F}_2[z] \right\},$$

where $\mathbb{F}_2[z]$ is the ring of polynomials with coefficients in $\mathbb{F}_2$, and the basis vectors $\boldsymbol{v}_j(z)$ are in $\mathbb{L}_2^s$. The $(\ell, \ldots, \ell)$-equidistribution of Ψ_s can be verified by computing a shortest vectors in the dual lattice, with an appropriate definition of vector length (or norm). Much of the theory and algorithms developed for lattices in the real space (Sections 3.2.7 and 3.2.8) translates to these different types of lattices (Couture and L'Ecuyer 2000, L'Ecuyer and Panneton 2009) and is especially useful when k is very large.

3.3.6 The LFSR Generator

Perhaps the oldest and best-known type of $\mathbb{F}_2$-linear generator is the *Tausworthe* or *linear feedback shift register* (LFSR) generator (Tausworthe 1965, L'Ecuyer 1996b, Tezuka 1995). It is defined by a linear recurrence modulo 2, and two positive integers w and s. At every s steps of the recurrence, a block of w successive bits is taken from the sequence: [5]

[5] From Pierre: This s differs from from the s that denotes the dimension, used earlier. Should probably replace s by another symbol.

$$x_n = a_1 x_{n-1} + \cdots + a_k x_{n-k}, \tag{3.45}$$

$$u_n = \sum_{\ell=1}^{w} x_{ns+\ell-1} 2^{-\ell}. \tag{3.46}$$

where $a_1, \ldots, a_k$ are in $\mathbb{F}_2$ and $a_k = 1$. This fits our framework by taking $\boldsymbol{A} = (\boldsymbol{A}_0)^s$ (in $\mathbb{F}_2$) where

$$\boldsymbol{A}_0 = \begin{pmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ a_k & a_{k-1} & \ldots & a_1 \end{pmatrix}, \tag{3.47}$$

and the blank entries in the matrix are zeros. If $w \le k$, the matrix $\boldsymbol{B}$ just contains the first w rows of the $k \times k$ identity matrix. But we may also have $w > k$; this often happens when the LFSR is a component of a combined generator. In that case, an equivalent generator can be obtained by expanding $\boldsymbol{A}$ into a $w \times w$ matrix in the right way (L'Ecuyer and Panneton 2009).

Here, $P(z)$ is the characteristic polynomial of the matrix $\boldsymbol{A} = (\boldsymbol{A}_0)^s$, not that of the recurrence (3.45), and the choice of s is important for determining the quality of this generator. The characteristic polynomial of $\boldsymbol{A}_0$ is $Q(z) = z^k - a_1 z^{k-1} - \cdots - a_k$. A frequently encountered case is when a single a_j is nonzero in addition to a_k; then, $Q(z)$ is a trinomial and we have a *trinomial-based* LFSR generator. Typically, s is small to make the implementation efficient. These trinomial-based generators are known to have important statistical weaknesses (Matsumoto and Kurita 1996, Tezuka 1995) but they can be used a components of combined LFSR generators (Tezuka and L'Ecuyer 1991, Wang and Compagner 1993, L'Ecuyer 1996b). That is, component c will be a LFSR whose characteristic polynomial $Q(z)$ is $Q_c(z) = z^{k_c} - z^{k_c - q_c} - 1$, and the combination is made by an exclusive-or as in Section 3.3.3. The matrix $\boldsymbol{A}$ of the combined generator has the form $\boldsymbol{A} = \text{diag}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_C)$ where $\boldsymbol{A}_c = \boldsymbol{A}_{0,c}^{s_c}$ and $\boldsymbol{A}_{0,c}$ is the matrix $\boldsymbol{A}_0$ associated with component c, for each c. It can be written equivalently as $\boldsymbol{A} = \boldsymbol{A}_0^s$ where $\boldsymbol{A}_0 = \text{diag}(\boldsymbol{A}_{0,1}, \ldots, \boldsymbol{A}_{0,C})$ and s is an integer that satisfies Equation (3) of L'Ecuyer (1996b). This means that the combined LFSR is equivalent to an LFSR with $Q(z) = Q_1(z) \cdots Q_C(z)$ and this value of s.

Tables of parameters for maximally equidistributed combined LFSR generators, and implementations for 32-bit and 64-bit computers, can be found in L'Ecuyer (1999b). These generators are among the fastest ones currently available.

Example 3.28 Consider the combined LFSR generator with $C = 4$ components whose recurrences (3.45) have the characteristic polynomials: $Q_1(z) = z^{31} - z^6 - 1$, $Q_2(z) = z^{29} - z^2 - 1$, $Q_3(z) = z^{28} - z^{13} - 1$, and $Q_4(z) = z^{25} - z^3 - 1$, and whose values of s in (3.46) are $s_1 = 18$, $s_2 = 2$, $s_3 = 7$, and $s_4 = 13$, respectively. All these polynomials $Q_c(z)$ are primitive, so each component has period $2^{k_c} - 1$ where k_c is the degree of $Q_c(z)$.

The recurrence (3.45) for the combined generator has characteristic polynomial $Q(z) = Q_1(z)Q_2(z)Q_3(z)Q_4(z)$ of degree 113, with 58 coefficients equal to 0 and 55 equal to 1. Its period length is $(2^{31} - 1)(2^{29} - 1)(2^{28} - 1)(2^{25} - 1) \approx 2^{113}$, because the numbers $2^{k_c} - 1$ are pairwise relatively prime. This combined generator is ME. It was proposed by L'Ecuyer (1999b), who provides an implementation under the name of `lfsr113`. □

We outline an algorithm that implements an LFSR generator efficiently under certain constraints on the parameters. It generalizes the algorithm given by Tezuka and L'Ecuyer (1991). Suppose that in (3.45), $a_j = 1$ for $j \in \{j_1, \ldots, j_d\}$ and $a_j = 0$ otherwise, where $k/2 \le j_1 < \cdots < j_d = k \le w$ and $0 < s \le j_1$. We work directly with the w-bit vectors $\boldsymbol{y}_n = (x_{ns}, \ldots, x_{ns+w-1})$, assuming that w is the computer's word length. Under the given conditions, a left shift of $\boldsymbol{y}_n$ by $k - j_i$ bits, denoted $\boldsymbol{y}_n \ll (k - j_i)$, gives a vector that contains the first $w - k + j_i$ bits of $\boldsymbol{y}_{n+k-j_i}$ followed by $k - j_i$ zeros (for $i = d$, $j_i = k$ so there is no shift). Adding these d shifted vectors by a bitwise xor, for $j = 1, \ldots, d$, gives a vector $\tilde{\boldsymbol{y}}$ that contains the first $w - k + j_1$ bits of $\boldsymbol{y}_{n+k} = \boldsymbol{y}_{n+k-j_1} \oplus \cdots \oplus \boldsymbol{y}_{n+k-j_d}$ followed by $k - j_1$ other bits (which do not matter). Now we shift $\tilde{\boldsymbol{y}}$ by $k - s$ positions to the right, denoted $\tilde{\boldsymbol{y}} \gg (k - s)$; this gives $k - s$ zeros followed by the last $w - k + s$ bits of $\boldsymbol{y}_{n+s}$ (the $k - j_1$ bits that do not matter have disappeared, because $s \ge j_1$). Zeroing the last $w - k$ bits of $\boldsymbol{y}_n$ and then shifting it to the left by s bits gives the first $k - s$ bits of $\boldsymbol{y}_{n+s}$. Adding this to $\tilde{\boldsymbol{y}}$ then gives $\boldsymbol{y}_{n+s}$. This is summarized by the following algorithm, in which & denotes a bitwise "and" and `mask` contains k 1's followed by $w - k$ 0's.

Algorithm LFSR [One step of an LFSR generator]:
Let $\tilde{\boldsymbol{y}} = \boldsymbol{y}_n$;
For $i = 2, \ldots, d$, let $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{y}} \oplus (\boldsymbol{y}_n \ll (k - j_i))$;
Let $\boldsymbol{y}_{n+s} = (\tilde{\boldsymbol{y}} \gg (k - s)) \oplus ((\boldsymbol{y}_n \,\&\, \texttt{mask}) \ll s)$;

This algorithm will work properly if $\boldsymbol{y}_0$ has been initialized to a valid state, which means that the values $x_k, \ldots, x_{w-1}$ must satisfy the recurrence $x_j = a_1 x_{j-1} + \cdots + a_k x_{j-k}$ for $j = k, \ldots, w - 1$. For that, we can take $(x_0, \ldots, x_{k-1})$ as an arbitrary nonzero vector, and then compute $x_k, \ldots, x_{w-1}$ from the recurrence. L'Ecuyer (1996b) explains how to implement this. An even simpler method to get a valid state, when k is close to w, is to just apply Algorithm LFSR once to an arbitrary k-bit vector.

♣ – Example of a tiny LFSR generator + display point set.

Example 3.29 The combined generator of Example 3.28 was obtained by L'Ecuyer (1999b) after first selecting $C = 4$ and the degrees k_c of the characteristic polynomials $Q_c(z)$ so that $k_c < 32$ for each c and the numbers $2^{k_c} - 1$ are pairwise relatively prime. This gave $k_1 = 31$, $k_2 = 29$, $k_3 = 28$, and $k_4 = 25$. Then for each c, the author made a list of all primitive polynomials of the form $Q_c(z) = z^{k_c} - z^{q_c} - 1$ with $0 < 2q_c < k_c$, and for each one a list of all step sizes s_c satisfying $0 < s_c \le k_c - q_c < k_c \le w = 32$ and $\gcd(s_c, 2^{k_c} - 1) = 1$. The corresponding recurrence (3.45) has the two nonzero coefficients $a_{k_c} = a_{k_c - q_c} = 1$, period $2^{k_c} - 1$, and can be implemented efficiently via Algorithm LFSR. Then we examined all the combinations that can be obtained by taking one component from these lists for each c; there are approximately 3.28 million such combinations. We found that 4744 of them give ME-CF generators. The `lfsr113` generator is one of them. □

Example 3.30 We illustrate in detail the implementation of the first component of `lfsr113`, for which $k_c = 31$, $q_c = 6$, and $s_c = 18$. In the notation of Algorithm LFSR, we have $w = 32$, $k = 31$, $d = 2$, $k - j_2 = 6$, and $s = 18$. The algorithm can then be written as:

$$\begin{aligned} \boldsymbol{y} &= (\boldsymbol{x}_{n-1} \ll 6) \oplus \boldsymbol{x}_{n-1}, \\ \boldsymbol{x}_n &= ((\boldsymbol{x}_{n-1} \text{ with last bit at 0}) \ll 18) \oplus (\boldsymbol{y} \gg 13). \end{aligned}$$

We illustrate this algorithm for a given initial state $\boldsymbol{x}_{n-1}$. The bits in orange are discarded. The state $\boldsymbol{x}_n$ after the transition is shown in red. It is the juxtaposition of the two blocks of bits in blue.

$$
\begin{array}{ll}
\boldsymbol{x}_{n-1} = & \mid 00010100101001101100110110100101 \\
& 100101 \mid 00101001101100110110100101 \\
\boldsymbol{y} = & \mid 00111101000101011010010011100101 \\
\boldsymbol{y} \gg 13 = & \mid \quad\quad\quad\quad\quad 00111101000101011010010011100101 \\
\boldsymbol{x}_{n-1} & \mid 00010100101001101100110110100100 \\
00010100101001101 & 1001101101001000 \\
\boldsymbol{x}_n = & \mid 00110111010010001111010001010110 1
\end{array}
$$

Note that the first 31 bits of $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$, visit all integers from 1 to $2^{31} - 1$ exactly once before returning to $\boldsymbol{x}_0$. □

Example 3.31 Figure 3.4 gives an implementation of the `lfsr113` generator, taken from L'Ecuyer (1999b), and where each of the four components is implemented via Algorithm LFSR. □

```
unsigned long z1, z2, z3, z4;  /* Contain the state. */

double lfsr113 ()
   {          /* Generates numbers between 0 and 1. */
   unsigned long b;
   b  = (((z1 <<  6) ^ z1) >> 13);
   z1 = (((z1 & 4294967294) << 18) ^ b);
   b  = (((z2 <<  2) ^ z2) >> 27);
   z2 = (((z2 & 4294967288) <<  2) ^ b);
   b  = (((z3 << 13) ^ z3) >> 21);
   z3 = (((z3 & 4294967280) <<  7) ^ b);
   b  = (((z4 <<  3) ^ z4) >> 12);
   z4 = (((z4 & 4294967168) << 13) ^ b);
   return ((z1 ^ z2 ^ z3 ^ z4) * 2.3283064365387e-10);
   }
```

Fig. 3.4. An implementation of `lfsr113` in the C language.

3.3.7 The GFSR, Twisted GFSR, and Mersenne Twister

Here we take $\boldsymbol{A}$ as a $pq \times pq$ matrix with the general form

$$
\boldsymbol{A} = \begin{pmatrix}
\boldsymbol{S}_1 & \boldsymbol{S}_2 & & \boldsymbol{S}_{q-1} & \boldsymbol{S}_q \\
\boldsymbol{I}_p & & & & \\
& \boldsymbol{I}_p & & & \\
& & \ddots & & \\
& & & \boldsymbol{I}_p &
\end{pmatrix}
$$

for some positive integers p and q, where $\boldsymbol{I}_p$ is the $p \times p$ identity matrix, and each $\boldsymbol{S}_j$ is a $p \times p$ matrix. Often, $w = p$ and $\boldsymbol{B}$ contains the first w rows of the $pq \times pq$ identity matrix. If

$\boldsymbol{S}_r = \boldsymbol{S}_q = \boldsymbol{I}_p$ for some r and all the other $\boldsymbol{S}_j$'s are zero, this generator is the trinomial-based *generalized feedback shift register* (GFSR), introduced by Lewis and Payne (1973), for which

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-r} \oplus \boldsymbol{x}_{n-q}$$

(a bitwise exclusive-or) and $\boldsymbol{x}_n$ gives the w bits of u_n. This is an extremely fast RNG. However, its state occupies pq bits whereas its period cannot exceed $2^q - 1$ (much less than 2^{pq}), because each bit of $\boldsymbol{x}_n$ follows the same binary recurrence of order $k = q$, with characteristic polynomial $P(z) = z^q - z^{q-r} - 1$.

More generally, we can define

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-r_1} \oplus \boldsymbol{x}_{n-r_2} \oplus \cdots \oplus \boldsymbol{x}_{n-r_d}$$

where $r_d = q$, so that each bit of $\boldsymbol{x}_n$ follows a recurrence in $\mathbb{F}_2$ whose characteristic polynomial $P(z)$ has $d + 1$ nonzero terms. This corresponds to taking $\boldsymbol{S}_j = \boldsymbol{I}_p$ for $j \in \{r_1, \ldots, r_d\}$ and $\boldsymbol{S}_j = \boldsymbol{0}$ otherwise. However, the period is still bounded by $2^q - 1$, whereas considering the pq-bit state, we should expect a period close to 2^{pq}.

This was the main motivation for the *twisted GFSR* (TGFSR) generator, originally introduced by Matsumoto and Kurita (1992) with $w = p$, $\boldsymbol{S}_q$ defined as the transpose of $\boldsymbol{A}_0$ in (3.47) with k replaced by p, $\boldsymbol{S}_r = \boldsymbol{I}_p$, and all the other $\boldsymbol{S}_j$'s are zero. The characteristic polynomial of $\boldsymbol{A}$ is then $P(z) = P_S(z^q + z^{q-r})$, where $P_S(\zeta) = \zeta^p - a_p\zeta^{p-1} - \cdots - a_1$ is the characteristic polynomial of $\boldsymbol{S}_q$, and its degree is $k = pq$. If the parameters are selected so that $P(z)$ is primitive over $\mathbb{F}_2$, then this TGFSR has period $2^k - 1$. The matrix $\boldsymbol{B}$ was originally selected simply as the first w rows of the identity matrix.

Matsumoto and Kurita (1994) pointed out important weaknesses of this original TGFSR, and introduced an improved version that uses a well-chosen matrix $\boldsymbol{B}$ whose rows differ from those of the identity. They called *tempering* the operations implemented by this matrix. Their purpose is to improve the uniformity of the points produced by the RNG.

The *Mersenne twister* (Matsumoto and Nishimura 1998, Nishimura 2000) (MT) is a variant of the TGFSR where k is slightly less than pq and can be a prime number It uses a pq-bit vector to store the k-bit state, where $k = pq - r$ is selected so that $r < p$ and $2^k - 1$ is a Mersenne prime. The matrix $\boldsymbol{A}$ is a $(pq-r) \times (pq-r)$ matrix similar to that of the TGFSR and the implementation is also quite similar. The main reason for using a k of that form is to simplify the search for primitive characteristic polynomials, as discussed in Section 3.3.1. With $k = pq$ it is impossible to have a Mersenne prime because $2^{pq} - 1$ is divisible by $2^p - 1$ and $2^q - 1$. A specific instance proposed by Matsumoto and Nishimura (1998), and named `MT19937`, has become quite popular; it is fast and has the huge period of $2^{19937} - 1$.

Panneton, L'Ecuyer, and Matsumoto (2006) underline and illustrate one weakness of this RNG: if the generator starts in (or reaches) a state that has very few ones, it may take up to several hundred thousands steps before the ratio of ones in the output and/or the average output value are approximately 1/2. For example, for `MT19937`, if we average the output values at steps $n + 1$ to $n + 100$ (a moving average) and average this over all 19937 initial states $\boldsymbol{x}_0$ that have a single bit at one, then we need at least $n > 700,000$ before the average gets close to 1/2 as it should be. This is graphically illustrated by Panneton, L'Ecuyer, and

Matsumoto (2006). Likewise, if two states differ by a single bit, or by only a few bits, a very large number of steps are required on average before the states or the outputs differ by about half of their bits. The source of the problem is that this RNG has a (huge) 19937-bit state and few of these bits are modified from one step to the next, as explained near the end of Section 3.3.4. It has only $N_1 = 135$ nonzero coefficients out of 19938 in its characteristic polynomial. Moreover, the figure of merit $\tilde{\Delta}_1$ takes the large value 6750 for this generator.

It has been proved that the TGFSR and Mersenne twister construction methods used in Matsumoto and Kurita (1994), Matsumoto and Nishimura (1998) *cannot* provide ME generators in general. They typically have large equidistribution gaps. But combining them via a bitwise xor *can* yield generators with the ME property. Examples of ME combined TGFSR generators with periods near 2^{466} and 2^{1250} are given by L'Ecuyer and Panneton (2002). These generators have the additional property that the resolution gaps δ_I are also zero for a class of index sets I of small cardinality and whose elements are not too far apart. These combined RNGs are of course a bit slower than their original (uncombined) counterparts.

3.3.8 The WELL RNGs

The acronym WELL stands for *well-equidistributed long-period linear*. These RNGs were developed by Panneton (2004) and are described by Panneton, L'Ecuyer, and Matsumoto (2006). The idea of the WELL project was to "sprinkle" a small number of very simple operations on w-bit words (where w is taken as the size of the computer word), such as xor, shift, bit mask, etc., into the matrix $\boldsymbol{A}$ in a way that the resulting RNG has maximal period, runs about as fast as the Mersenne twister, has a characteristic polynomial with around 50% nonzero coefficients, and has the best possible equidistribution properties under these constraints.

The state $\boldsymbol{x}_n = (\boldsymbol{v}_{n,0}^{\mathsf{t}}, \ldots, \boldsymbol{v}_{n,r-1}^{\mathsf{t}})^{\mathsf{t}}$ is comprised of r blocks of $w = 32$ bits $\boldsymbol{v}_{n,j}$, and the recurrence is defined by a set of linear transformations that apply to these blocks, as described in Panneton, L'Ecuyer, and Matsumoto (2006). The transformations modify $\boldsymbol{v}_{n,0}$ and $\boldsymbol{v}_{n,1}$ by using several of the other blocks. They are selected so that $P(z)$, a polynomial of degree $k = rw - p$, is primitive over $\mathbb{F}_2$. The output is defined by $\boldsymbol{y}_n = \boldsymbol{v}_{n,0}$.

The authors list specific parameters for WELL generators with periods ranging from $2^{512}-1$ to $2^{44497}-1$. Many of them are ME and the others are nearly ME. Their characteristic polynomials have approximately 50% coefficients equal to 1. These RNGs have much better diffusion capacity than the Mersenne twister and have comparable speed.

3.3.9 Xorshift Generators

Marsaglia (2003) proposed a class of very fast RNGs whose recurrence can be implemented by a small number of xorshift operations only, where a *xorshift operation* consists in replacing a w-bit block in the state by a (left or right) shifted version of itself (by a positions, where $0 < a < w$) xored with the original block:

$$\boldsymbol{x} = (\boldsymbol{x} \gg a) \oplus \boldsymbol{x}.$$

The constant w is the computer's word size (usually 32 or 64). The specific generators he proposed in his paper use three xorshift operations at each step. Since xorshifts are linear operations in $\mathbb{F}_2$, these generators fit our setting.

Panneton and L'Ecuyer (2005) analyzed the theoretical properties of a general class of xorshift generators that contains those proposed by Marsaglia. They studied maximal-period conditions, limits on the equidistribution, and submitted xorshift generators to empirical statistical testing. They concluded that generators built from only three xorshifts are unsafe, and they came up with generators based on 7 and 13 xorshifts, whose speed is only 20% slower than those with three xorshifts to generate $U(0,1)$ real numbers. Aside from the tests that detect $\mathbb{F}_2$-linearity, these RNGs pass other standard statistical tests.

Brent (2004) has proposed a family of generators that combine a xorshift RNG with a Weyl generator. The resulting generator is no longer $\mathbb{F}_2$-linear and it behaves well empirically (L'Ecuyer and Simard 2007).

3.3.10 Examples

♣ To be done.

3.4 Nonlinear RNGs

All RNGs discussed so far are based on linear recurrences and their structure may be deemed too regular. There are at least two ways of getting rid of this regular linear structure: (1) use a nonlinear transition function f or (2) keep the transition function linear but use a nonlinear output function g. Several types of nonlinear RNGs have been proposed over the years; see, e.g., Blum, Blum, and Schub (1986), Eichenauer-Herrmann (1995), Eichenauer-Herrmann, Herrmann, and Wegenkittl (1998), Hellekalek and Wegenkittl (2003), Knuth (1998), L'Ecuyer (1994b), Niederreiter and Shparlinski (2002), and Tezuka (1995). Their nonlinear mappings are defined in various ways by multiplicative inversion in a finite field, quadratic and cubic functions in the finite ring of integers modulo m, and other more complicated transformations. Many of them have output sequences that tend to behave much like i.i.d. $U(0,1)$ sequences even over their entire period, in contrast with "good" linear RNGs, whose point sets Ψ_s are much more regular than typical random points (Eichenauer-Herrmann, Hermann, and Wegenkittl 1998, L'Ecuyer and Hellekalek 1998, L'Ecuyer and Granger-Piché 2003, Niederreiter and Shparlinski 2002). On the other hand, their statistical properties have been analyzed only empirically or via asymptotic theoretical results. For specific nonlinear RNGs, the uniformity of the point sets Ψ_s is very difficult to measure theoretically. Moreover, the nonlinear RNGs are generally significantly slower than the linear ones. The RNGs recommended for cryptology are all nonlinear.

An interesting idea for adding nonlinearity without incurring an excessive speed penalty is to combine a small nonlinear generator with a fast long-period linear one (Aiello, Rajagopalan, and Venkatesan 1998, L'Ecuyer and Granger-Piché 2003). L'Ecuyer and Granger-

Piché (2003) show how to do this while ensuring theoretically the good uniformity properties of Ψ_s for the combined generator. A very fast implementation can be achieved by using precomputed tables for the nonlinear component. Empirical studies suggest that mixed linear-nonlinear combined generators are more robust than the linear ones with respect to statistical tests, because of their less regular structure.

♣ Add an example of this.

Several authors have proposed various ways of combining RNGs to produce streams of random numbers with less regularity and better "randomness" properties; see, e.g., Collings (1987), Knuth (1998), Gentle (2003), Law and Kelton (2000), L'Ecuyer (1994b), Fishman (1996), Marsaglia (1985), and other references given there. This includes *shuffling* the output sequence of one generator using another one (or the same one), alternating between several streams, or just adding them in different ways. Most of these techniques are heuristics. They usually improve the uniformity (empirically), but they can also make it worse. For *random variables* in the mathematical sense, certain types of combinations (e.g., addition modulo 1) can *provably* improve the uniformity, and some authors have used this fact to argue that combined RNGs are provably better than their components alone (Brown and Solomon 1979, Deng and George 1990, Marsaglia 1985, Gentle 2003), but this argument is faulty because the output sequences of RNGs are deterministic, not sequences of independent random variables. To assess the quality of a combined generator, one must analyze the mathematical structure of the combined generator itself, as in L'Ecuyer (1996b), L'Ecuyer (1996a), L'Ecuyer and Granger-Piché (2003), Tezuka (1995), rather than the structure of its components.

♣ Multiplicative lagged-Fibonacci. Kiss generator from Marsaglia. Combined cubic generators.

3.4.1 Speed Comparisons

Table 3.2 reports the speed of some RNGs available in the SSJ simulation package (L'Ecuyer and Buist 2005). The timings are for the SSJ implementations on a 2.4 GHz 64-bit AMD-Athlon computer with SUN's JDK 1.5, C implementations on the same processor, and C implementations running on a 2.8 GHz 32-bit Intel processor. The first and second columns of the table give the generator's name and its approximate period. All these generators are implemented for a 32-bit computer, although the C implementation of the two MRG generators (last two lines) used on the 64-bit computer was different; it exploits the 64-bit arithmetic. The SSJ implementations of all generators have more overhead because they support multiple streams. The C implementations do not. The jumping ahead in SSJ is implemented via a multiplication by $\boldsymbol{A}^\nu$ as explained in Section 3.3.2. For the combined LFSR generators, the linear recurrence that corresponds to the matrix $\boldsymbol{A}^\nu$ is implemented directly using the algorithm of Section 3.3.6, for each component of the combination. It is much faster for this reason. Column 3 of the table gives the CPU time (sec) to generate 10^9 random numbers and add them up, whereas column 4 gives the CPU time needed to jump ahead 10^6 times by a very large number of steps (to get a new stream), in SSJ. For comparison, columns 5 and 6 give the time to generate 10^9 numbers with the C implementation available in TestU01 (L'Ecuyer and Simard 2007), first on the same computer (gen. 64), and then on the 32-bit computer (gen. 32).

Table 3.2. CPU time (sec) to generate 10^9 random numbers, and CPU time to jump ahead 10^6 times, with some RNGs available in SSJ

RNG	$\rho \approx$	CPU time in SSJ (Java)		CPU time in C	
		gen.	jump	gen. 64	gen. 32
LFSR113	2^{113}	31	0.1	10	39
LFSR258	2^{258}	35	0.2	12	58
WELL512	2^{512}	33	234	12	38
WELL1024	2^{1024}	34	917	11	37
MT19937	2^{19937}	36	—	16	42
MRG31k3p	2^{185}	51	0.9	21	71
MRG32k3a	2^{191}	70	1.1	21	99

The first five RNGs are $\mathbb{F}_2$-linear and the last two are combined multiple recursive generators (MRGs). The first two are combined LFSRs proposed by L'Ecuyer (1999b) for 32-bit and 64-bit computers, with four and five components, respectively. The two WELL RNGs are proposed in Panneton, L'Ecuyer, and Matsumoto (2006). Other WELL generators with much longer periods (up to nearly 2^{44497}) proposed in that paper have approximately the same speed as those given here to generate random numbers, but are much slower than WELL1024 for jumping ahead because of their larger value of k. For the Mersenne twister `MT19937` of Matsumoto and Nishimura (1998), jumping ahead is also slow and is not implemented in SSJ. All these $\mathbb{F}_2$-linear RNGs have roughly the same speed for generating random numbers. Other ones with about the same speed are also proposed by Panneton and L'Ecuyer (2004) and Matsumoto and Kurita (1994), e.g., with periods near 2^{800}.

The timings of the two MRGs in the table are reported for comparison. The first one (`MRG31k3p`) was proposed by L'Ecuyer and Touzin (2000) while the second one (`MRG32k3a`) was proposed by L'Ecuyer (1999a) and is used in several simulation packages to provide multiple streams and substreams. This latter RNG has been heavily tested over the years and is very robust. On the other hand, the $\mathbb{F}_2$-linear generators are faster.

3.5 Statistical Tests

For now, replace this section by the article of L'Ecuyer and Simard (2007). The section should be rewritten based on that paper.

As mentioned earlier, a statistical test for RNGs is defined by a random variable X whose distribution under $\mathcal{H}_0$ can be well approximated. When X takes the value x, we define the *right and left p-values* of the test by

$$p_{\mathrm{r}} = \mathbb{P}[X \geq x \mid \mathcal{H}_0] \quad \text{and} \quad p_{\mathrm{l}} = \mathbb{P}[X \leq x \mid \mathcal{H}_0].$$

When testing RNGs, there is no need to prespecify the level of the test. If any of the right or left p-value is extremely close to zero, e.g., less than 10^{-15}, then it is clear that $\mathcal{H}_0$ (and the RNG) must be rejected. When a *suspicious* p-value is obtained, e.g., near 10^{-2} or 10^{-3},

one can just repeat this particular test a few more times, perhaps with a larger sample size. Almost always, things will then clarify.

Most tests are defined by partitioning the possible realizations of $(u_0, \ldots, u_{\tau-1})$ into a finite number of subsets (where the integer τ can be random or deterministic), computing the probability p_j of each subset j under $\mathcal{H}_0$, and measuring the discrepancy between these probabilities and empirical frequencies from realizations simulated by the RNG.

A special case that immediately comes to mind is to take $\tau = s$ (a constant) and cut the interval $[0, 1)$ into d equal segments for some positive integer d, to partition the hypercube $[0, 1)^s$ into $k = d^s$ subcubes of volume $1/k$. We then generate n points $\boldsymbol{u}_i = (u_{ti}, \ldots, u_{ti+t-1}) \in [0, 1)^s$, for $i = 0, \ldots, n-1$, and count the number N_j of points falling in subcube j, for $j = 0, \ldots, k-1$. Any measure of distance (or divergence) between the numbers N_j and their expectations n/k can define a test statistic X. The tests thus defined are generally called *serial tests* of uniformity (Knuth 1998, L'Ecuyer, Simard, and Wegenkittl 2002). They can be *sparse* (if $n/k \ll 1$), or *dense* (if $n/k \gg 1$), or somewhere in between. There are also *overlapping* versions, where the points are defined by $\boldsymbol{u}_i = (u_i, \ldots, u_{i+t-1})$ for $i = 0, \ldots, n-1$ (they have overlapping coordinates).

Special instances for which the distribution under $\mathcal{H}_0$ is well-known are the chi-square, the (negative) empirical entropy, and the number of collisions (L'Ecuyer and Hellekalek 1998, L'Ecuyer, Simard, and Wegenkittl 2002, Read and Cressie 1988). For the latter, the test statistic X is the number of times a point falls in a subcube that already had a point in it. Its distribution under $\mathcal{H}_0$ is approximately Poisson with mean $\lambda_1 = n^2/(2k)$, if n is large and λ_1 not too large.

A variant is the *birthday spacings* test, defined as follows (Marsaglia 1985, Knuth 1998, L'Ecuyer and Simard 2001). Let $I_{(1)} \le \cdots \le I_{(n)}$ be the numbers of the subcubes that contain the points, sorted by increasing order. Define the *spacings* $S_j = I_{(j+1)} - I_{(j)}$, for $j = 1, \ldots, n-1$, and let X be the number of collisions between these spacings. Under $\mathcal{H}_0$, X is approximately Poisson with mean $\lambda_2 = n^3/(4k)$, if n is large and λ_2 is small.

Consider now a MRG, for which Ψ_s has a regular lattice structure. Because of this regularity the points of Ψ_s will tend to be more evenly distributed among the subcubes than random points. For a well-chosen k and large enough n, we expect the collision test to detect this: it is likely that there will be too few collisions. In fact, the same applies to any RNG whose set Ψ_s is very evenly distributed. When a birthday spacings test with a very large k is applied to a MRG, the numbers of the subcubes that contain one point of Ψ_s tend to be too evenly spaced and the test detects this by finding too many collisions.

These specific interactions between the test and the structure of the RNG lead to systematic patterns in the p-values of the tests. To illustrate this, suppose that we take k slightly larger than the cardinality of Ψ_s (so $k \approx \rho$) and that due to the excessive regularity, no collision is observed in the collision test ($X = 0$). The left p-value will then be $p_1 \approx \mathbb{P}[X \le 0 \mid X \sim \text{Poisson}(\lambda_1)] = \exp[-n^2/(2k)]$. For this p-value to be smaller than a given ϵ, we need a sample size n proportional to the square root of the period ρ. And after that, p_1 decreases exponentially fast in n^2.

Example 3.32 ♣ .... □

Extensive experiments with LCGs, MRGs, and LFSR generators confirms that this is actually what happens with these RNGs (L'Ecuyer and Hellekalek 1998, L'Ecuyer 2001, L'Ecuyer, Simard, and Wegenkittl 2002). For example, if we take $\epsilon = 10^{-15}$ and define n_0 as the minimal sample size n for which $p_1 < \epsilon$, we find that $n_0 \approx 16\rho^{1/2}$ (very roughly) for LCGs that behave well in the spectral test as well as for LFSR generators. For the birthday spacings test, the rule for LCGs is $n_0 \approx 16\rho^{1/3}$ instead (L'Ecuyer and Simard 2001). So to be safe with respect to these tests, the period ρ must be so large that generating more than $\rho^{1/3}$ numbers is practically unfeasible. This certainly disqualifies all LCGs with modulus smaller than 2^{100} or so.

Example 3.33 ♣ .... □

Other types of tests for RNGs include tests based on the closest pairs of points among n points generated in the hypercube, tests based on random walks on the real line or over the integers, tests based on the linear complexity of a binary sequence, tests based on the simulation of dices or poker hands, and many others (Knuth 1998, L'Ecuyer and Simard 2002, Marsaglia 1996, Rukhin et al. 2001, Vattulainen, Ala-Nissila, and Kankaala 1995).

When testing RNGs, there is no specific alternative hypothesis to $\mathcal{H}_0$. Different tests are needed to detect different types of departures from $\mathcal{H}_0$. *Test suites* for RNGs include a selection of tests, with predetermined parameters and sample sizes. The best known are *DIEHARD* (Marsaglia 1996), the *NIST* test suite (Rukhin et al. 2001), and the *TestU01* library (L'Ecuyer and Simard 2002, L'Ecuyer and Simard 2007). The latter implements a large selection of tests in the C language and provides a variety of test suites, some designed for i.i.d. $U(0,1)$ output sequences and others for strings of bits.

3.6 RNG Software

When we apply test suites to RNGs currently found in commercial software (statistical and simulation software, spreadsheets, etc.), we find that many of them fail the tests spectacularly (L'Ecuyer 1997, L'Ecuyer and Simard 2007). There is no reason to use these poor RNGs, because there are also several good ones that are fast, portable, and pass all these test suites with flying colors. Among them we recommend, for example, the combined MRGs, combined LFSRs, and Mersenne twisters proposed in L'Ecuyer (1999b), L'Ecuyer (1999a), L'Ecuyer and Panneton (2002), Matsumoto and Nishimura (1998), and Nishimura (2000).

A convenient object-oriented software package with multiple streams and substreams of random numbers, is described in L'Ecuyer et al. (2002) and is available in Java, C, and C++, at `http://www.iro.umontreal.ca/~lecuyer`.

♣ This section must be expanded. Provide concrete examples of RNGs with multiple streams, with details. Several are available in SSJ.

3.7 Exercises

3.1 Suppose that a stream of bits $\{B_j,\, j \geq 0\}$ can be described by a Markov chain with the two states $\{0,1\}$, and with transition probabilities $\mathbb{P}[B_{j+1} = 1 \mid B_j = 0] = p$ and $\mathbb{P}[B_{j+1} = 0 \mid B_j = 1] = q$, where $0 < p, q < 1$. This simple model is often a good description of binary sequences produced by physical devices. Define the (asymptotic) bias and lag-1 correlation as $\beta = \lim_{j\to\infty} \mathbb{E}[B_j] - 1/2$ and $\rho_1 = \lim_{j\to\infty} \mathrm{Corr}[B_{j-1}, B_j]$.

(a) Give explicit formulas for β and ρ_1 as functions of p and q. Hint: Compute the steady-state probabilities of the Markov chain.

(b) To reduce the bias and correlation, suppose we define another Markov chain $\{C_j,\, j \geq 0\}$ via $C_j = B_{2j} \oplus B_{2j+1}$. What are the bias and lag-1 correlation for this new sequence? Hint: Define a Markov chain whose state at step j is (B_{2j}, B_{2j+1}) and compute its steady-state probabilities.

(c) A second way of modifying the sequence is as follows. For $j = 0, 1, 2, \ldots$, if $(B_{2j}, B_{2j+1}) = (1,0)$, output 1; if $(B_{2j}, B_{2j+1}) = (0,1)$, output 0; and if $(B_{2j}, B_{2j+1}) = (0,0)$ or $(1,1)$, output nothing and go to the next j. This clever transformation was proposed long ago by von Neumann (1951). Let $\{D_i,\, i \geq 0\}$ be the output sequence thus obtained. That is, when we output nothing, we increase j by 1, but not i. What are the bias and lag-1 correlation for this sequence?

3.2 The aim of this exercise is to prove Proposition 3.2.

(a) Show that for any $\boldsymbol{x}_n \in \mathbb{Z}_m^k$ and $\tilde{s}_n(z)$ defined by Eq. (3.7), $p_n(z) = P(z)\tilde{s}_n(z)$ is a polynomial of degree less than k, whose coefficients are given by Eq. (3.9) and Eq. (3.10).

(b) Show that the three sets $\mathbb{Z}_m^k$, $\mathcal{L}(P)$, and $\mathbb{Z}_m[z]/(P)$ have the same cardinality and that the mappings $\boldsymbol{x}_n \to \tilde{s}_n(z) \to p_n(z)$ define bijections between these sets.

3.3 Show that in the special case where m is prime and $k = 1$, the conditions of Proposition 3.5 reduce to the single condition that a_1 is a *primitive element modulo* m, i.e., that $a_1^{(m-1)/q} \bmod m \neq 1$ for each prime factor q of $m - 1$.

3.4 Prove Proposition 3.4. [6]

3.5 For $m = 2^{31} - 1$ and $k = 3$, what is the proportion of vectors $(a_1, \ldots, a_k)$ (among the $m^k - 1$ nonzero possibilities) for which the LRS (3.2) has full period $m^k - 1$? And for $k = 5$? (The required factorizations can be found in L'Ecuyer, Blouin, and Couture 1993).

3.6 Prove that for an MRG, the vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_s$ defined by $\boldsymbol{w}_i = m\boldsymbol{e}_i$ for $i \leq k$ and $\boldsymbol{w}_i = \boldsymbol{e}_i - (x_{1,i-1}, \ldots, x_{k,i-1}, 0, \ldots, 0)^{\mathsf{t}}$ for $i > k$ are indeed the columns of $\boldsymbol{V}^{-1}$. Then prove that these vectors form a basis of the dual lattice L_s^*.

3.7 Consider an MRG with $k = 3$, $a_2 = 0$, and arbitrary values of m, a_1 and a_3. We are interested in the lattice L_s, and its dual L_s^*, for $s = 5$. Write down explicitly a basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_5$ for L_5 and a basis $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_5$ for L_5^*.

[6] From Pierre: Should give some hints...

3.8 Complete the proof of Proposition 3.8 for a general set I.

3.9 Consider an MRG of order 2 with recurrence

$$x_n = (a_1 x_{n-1} + a_2 x_{n-2}) \bmod m$$

and an MRG of order k with recurrence

$$x_n = (a_1 x_{n-r} + a_2 x_{n-k}) \bmod m,$$

where $0 < a_1, a_2 < m$ and $1 \le r < k$. Show that in any dimension $s > k$, the length of a shortest vector in the dual lattice L_s^* is exactly the same for these two MRGs.

3.10 ♣ À traduire...

Soit un MRG basé sur la récurrence

$$x_n = (x_{n-1} + a x_{n-2}) \bmod m$$

et $u_n = x_n/m$, où $m = 2^{31} - 1$ et a est un entier positif. On veut étudier l'uniformité de l'ensemble Ψ_3 des points de la forme (u_n, u_{n+1}, u_{n+2}) produits par ce générateur, à partir de toutes les valeurs initiales possibles.

(a) Trouvez une base du réseau L_s défini en (3.19), selon la méthode décrite à la Section 3.2.7, pour $s = 3$. Trouvez ensuite la base duale correspondante, tel que décrit à la Section 3.2.8.1.

(b) Prouvez directement, sans invoquer la Proposition 3.8, que le vecteur $(a, 1, -1)$ appartient au réseau dual L_3^*. Que peut-on en déduire à propos de la distance entre les plans parallèles équidistants qui recouvrent L_3? Et à propos de la mesure normalisée $\ell_3/\ell_3^*(m^2)$? Et à propos de n_3, le nombre minimal de plans pour recouvrir $L_3 \cap (0,1)^3$? Qu'est-ce que cela implique pour l'uniformité de l'ensemble de points Ψ_3 et pour la qualité du générateur si a est petit, comme par exemple $a < 25$? (Pour $s = 3$, la constante de Hermite est $\gamma_s = 2^{1/3}$.)

(c) Supposons que l'on partitionne le cube unitaire $[0,1)^3$ en $k = 10^9$ sous-cubes en divisant chaque axe en 1000 parties égales. On génère ensuite $n = 10^6$ points $(u_{3i}, u_{3i+1}, u_{3i+2})$, $i = 0, \ldots, 10^6 - 1$, en utilisant ce générateur, puis on regarde dans quel sous-cube chacun des points tombe, et on compte le nombre de collisions C, i.e., le nombre de fois qu'un point tombe dans un sous-cube déjà visité. Si C est beaucoup trop grand ou beaucoup trop petit comparativement à sa "valeur attendue", on dira que le générateur échoue le *test de collision*.

Si les n points étaient vraiment indépendants et suivaient la loi uniforme sur $[0,1)^3$, la variable aléatoire C suivrait approximativement quelle loi avec quel paramètre? Pensez-vous que ce générateur échouera ce test si $a < 25$? Expliquez pourquoi.

3.11 ♣ À traduire... On considère un générateur LFSR combiné à deux composantes. Pour $c = 1, 2$ la composante c est un LFSR défini par (3.45) et (3.46), avec les paramètres $k = k_c$, $s = s_c$, et seulement deux coefficients non nuls dans la récurrence, $a_{k_c} = a_{k_c - q_c} = 1$ (les autres coefficients sont 0). Le polynôme caractéristique de la matrice $\boldsymbol{A}_0$ pour la composante c est donc $Q_c(z) = z^{k_c} - z^{q_c} - 1$ (modulo 2). La combinaison se fait tel que défini à la Section 3.3.6.

On choisit les paramètres suivants: $(k_1, q_1, s_1) = (29, 2, 18)$ et $(k_2, q_2, s_2) = (28, 9, 14)$. Dans chacun des deux cas, on peut montrer que le polynôme caractéristique est primitif.

(a) Quelle est la période du générateur combiné? Quel est le polynôme caractéristique de la matrice $\boldsymbol{A}_0$ qui lui correspond? Quelle est cette matrice? Est-ce que la matrice $\boldsymbol{A}$ a une forme particulière? Laquelle?

(b) Faites une implantation de ce générateur, en Java ou en C, en utilisant l'algorithme LFSR donné à la Section 3.3.6. Vous pouvez vous inspirer du code du générateur `lfsr113`. Utilisez des entiers de 32 bits pour l'implantation.

(c) Pour initializez l'état de chacune des deux composantes à un état valide, avec les paramètres choisis, il suffit d'initialiser l'état de chacune des deux composantes à un entier supérieur à 8, puis de faire une itération de l'algorithme. Initialisez l'état de chacune des deux composantes à 12345, sautez la première valeur générée, puis imprimez les 10 valeurs suivantes. Assurez-vous bien qu'elles sont dans l'intervalle $[0, 1)$.

(d) Expliquez, sans l'implanter, mais en donnant assez de détails pour qu'un programmeur qui ne connait rien aux RNGs puisse l'implanter facilement en suivant vos instructions, ce qu'il faut faire pour mesurer l'équidistribution de ce générateur combiné, en 3 dimensions. Expliquez en détail comment construire la matrice requise et quoi faire avec. Dans le meilleur des cas, on pourrait avoir l'équidistribution pour combien de bits en 3 dimensions, dans ce cas-ci?

4. Non-Uniform Random Variate Generation

Here we are interested in *imitating* or *simulating* random variables and random vectors from various distributions, as well as stochastic processes and other kinds of random objects such as matrices, graphs, points on a sphere, geometric structures, etc.

The primary requirement for any method is *correctness*. In practice, the generated random variate X (or random object) does not always have *exactly* the required distribution, because this would sometimes be much too costly or even impossible. But we must have a *good approximation* and, preferably, some understanding of the quality of that approximation. *Robustness* is also important: when the accuracy depends on the parameters of the distribution, it must be good *uniformly* over the entire range of parameter values that we are interested in.

The method must also be *efficient* both in terms of speed and memory usage. Often, it is possible to increase the speed by using more memory (e.g, for larger precomputed tables) or by relaxing the accuracy requirements. Some methods need a one-time setup to compute constants and construct tables. The setup time may be significant but well worth spending when amortized by a large number of subsequent calls to the generator. For example, it makes sense to invest in an expensive setup if we plan to make a million calls to a given generator and if the investment can really speed up the generator, but not if we expect to make only a few calls.

In general, compromises must be made between simplicity of the algorithm, quality of the approximation, robustness with respect to the distribution parameters, and efficiency (generation speed, memory requirements, and setup time).

Compatibility with variance reduction techniques (Chapter 6) is another important issue in many situations. We may be willing to sacrifice some speed to preserve inversion, because the gain in efficiency obtained via the variance reduction methods may more than compensate (sometimes by orders of magnitude) for the slower generator.

In the forthcoming sections, we consider the case of a unidimensional random variable X.

4.1 Inversion

The inversion method, defined in Section 1.3.5, returns

$$X = F^{-1}(U) = \min\{x \mid F(x) \geq U\}$$

where $U \sim U(0,1)$. This should be the method of choice for generating non-uniform random variates in a majority of situations. The fact that X is a monotone (non-decreasing) func-

tion of U makes this method compatible with important variance reduction techniques such as common random numbers, antithetic variates, latin hypercube sampling, and randomized quasi-Monte Carlo (Chapter 6). To maximize or minimize the correlation between two random variables, given their individual distributions, we must generate them by inversion (explicitly or implicitly), according to Fréchet's bounds (Section 2.10.1, Theorem 2.6). To sample from a multivariate distribution defined via a copula, we usually sample a vector with uniform marginals from the copula, and then we must apply inversion to recover the desired marginals (Section 2.10.8). To generate a *sorted* i.i.d. sample $X_{(1)}, \ldots, X_{(n)}$ from distribution function F, we can generate a sorted i.i.d. $U(0,1)$ sample $U_{(1)}, \ldots, U_{(n)}$ and apply F^{-1} to these $U_{(i)}$'s. Sampling from truncated distributions is also easier if we use inversion (Section 2.8.26).

Inversion is easy to implement for certain distributions, e.g., when an analytic expression is available for the inverse distribution function F^{-1} or in the case of a discrete distribution over a small set of integers. There are distributions (e.g., the normal, Student, chi-square) for which there is no closed-form expression for F^{-1} but good numerical approximations are available. When the distribution has only scale and location parameters, it suffices to approximate F^{-1} for a standardized version of the distribution.

Example 4.1 For the normal distribution, once we have an efficient method for evaluating Φ^{-1}, the inverse distribution function of a $N(0,1)$ random variable, a normal with mean μ and variance σ^2 can be generated by $X = \sigma\Phi^{-1}(U) + \mu$. There is no closed-form formula for either Φ or Φ^{-1}, but Blair, Edwards, and Johnson (1976) have developed a quick and accurate approximation method for Φ^{-1} based on a rational Chebyshev approximation. It gives 16 decimal digits of accuracy everywhere when using 64-bit floating point numbers. This method is highly recommendable. Marsaglia, Zaman, and Marsaglia (1994) propose a slightly faster algorithm (by about 20% according to some experiments with C and Java implementations) based on table lookups, but it returns only 6 decimal digits of accuracy and returns wrong (meaningless) numbers when U is outside the interval $[1.2 \times 10^{-10}, 1 - 1.2 \times 10^{-10}]$ (i.e., $|X| > 6.33$). □

When shape parameters are involved (e.g., the gamma and beta distributions), things are more complicated because F^{-1} then depends on the parameters in a more fundamental manner.

Sometimes, the probability model itself can be selected in a way that makes inversion easier. For example, one can fit a parametric, highly-flexible, and easily computable inverse distribution function F^{-1} to the data, directly or indirectly (Nelson and Yamnitsky 1998 and Section 2.9.2).

In the remainder of this section, we discuss how inversion can be implemented when there is no closed-form formula for F^{-1}, first for discrete distributions, then for continuous ones. In general, inversion can be formulated as a root-finding problem: for a given U, we need to find X such that $F(X) - U = 0$. Standard iterative root-finding algorithms can be used to solve this equation for X, provided that we have (at least) a good algorithm to compute F. But finding the root X by an iterative method each time we generate X can be quite expensive.

4.1.1 Inversion for Discrete Distributions

Sequential search. Suppose X has a discrete distribution over the real numbers $x_0 < \cdots < x_{k-1}$, where k can be finite or infinite. To generate X by inversion, we generate $U \sim U(0,1)$, find $I = \min\{i \mid F(x_i) \geq U\}$, and return $X = x_I$. The efficiency of this approach depends on how we implement the search for I. The simplest implementation is by *sequential search*: Just try $i = 0, 1, 2, \ldots$ in succession until we get $F(x_i) \geq U$. This give:

Sequential search (needs $O(k)$ iterations in the worst case);
generate $U \sim U(0,1)$;
let $i = 0$;
while $F(x_i) < U$ do $i = i + 1$;
return x_i.

If we plan to generate several copies of X with the same distribution, we should precompute the pairs $(x_i, F(x_i))$ and store them in a *look-up table*, in a one-time initialization step. If k is infinite or too large, only a finite portion of the table, that contains most of the probability mass, would be precomputed. The other values are computed only when needed. That is, if the table contains $(x_i, F(x_i))$ for $k_0 \leq i < k_1$, additional values of $F(x_i)$ for $i \geq k_1$ or $i < k_0$ need to be computed only when $U > F(x_{k_1-1})$ or $U \leq F(x_{k_0})$. If $U > F(x_{k_1-1})$, this can be done in the "while" loop by adding $p_i = \mathbb{P}[X = x_i]$ to the current value of $F(x_i)$, which is stored in a temporary variable. Likewise, the values of $F(x_i)$ for $i < k_0$ are computed only when $U \leq F(x_{k_0})$, by subtracting the relevant p_i's. Usually, we would take $k_0 = 0$, but if the probabilities p_i are extremely small for small i, it may be better to take $k_0 > 0$. For example, if X is Poisson with mean $\lambda = 2500$, then the probability that X takes a value outside the interval $[2200, 2799]$ is approximately 2.1×10^{-9}, so we could take $k_0 = 2200$ and $k_1 = 2800$. Of course, we may prefer to store more or fewer values, depending perhaps on how many copies of X we plan to generate, how much memory we are ready to spend, how large is the memory cache of our computer, etc. Taking a larger look-up table may actually *slow down* the generator at some point, because the table would not fit into the memory cache or would use too much of it. This is often the most important reason for keeping the table small. At the other extreme, we may choose *not* to precompute and store any values (e.g., if a single copy of X has to be generated); this means taking $k_1 = 0$.

Sequential search can be slow when k is large; in the worst case, we may have to do $k - 1$ turns into the "while" loop. The exact number of iterations in the loop (the number of times the code "$i = i + 1$" inside the loop is executed) is I, so we will do $\mathbb{E}[I]$ iterations on average. If X is distributed over the integers $\{0, 1, 2, \ldots\}$, then $I = X$ and the expected number of iterations is $\mathbb{E}[X]$, so sequential search is fine if $\mathbb{E}[X]$ is small, but not if it is large. For example, if X has the Poisson distribution with mean λ, we need λ iterations on average. If X is binomial with parameters (n, p), the expected number of iterations is np.

When the x_i's have negligible probabilities for small values of i, as in the Poisson example mentioned earlier, it makes more sense to start the sequential search from around the mean or median of the distribution, rather than starting it at $i = 0$. We preselect a fixed index m such that x_m is near the middle of the distribution (for instance, we may take $m = \lfloor \lambda \rfloor$ for the Poisson distribution with mean λ), and do the following:

Sequential search from near the center;
generate $U \sim U(0,1)$;
let $i = m$;
while $F(x_i) < U$ do $i = i + 1$;
while $F(x_{i-1}) \geq U$ do $i = i - 1$;
return x_i.

Binary search. When $\mathbb{E}[I]$ is large, *binary search* can beat sequential search. Suppose that $k < \infty$ and that all pairs $(x_i, F(x_i))$ have been precomputed. Then the algorithm is:

Binary search (needs either $\lfloor \log_2 k \rfloor$ or $\lceil \log_2 k \rceil$ iterations);
generate $U \sim U(0,1)$;
let $i = 0$ and $j = k$;
while $i < j - 1$ do
  $m = \lfloor (i + j)/2 \rfloor$;
  if $F(x_{m-1}) < U$ then $i = m$ else $j = m$;
  /* Invariant: at this stage, I is in $\{i, \ldots, j-1\}$. */
return x_i.

This algorithm always requires either $\lfloor \log_2 k \rfloor$ or $\lceil \log_2 k \rceil$ iterations (Brassard and Bratley 1988), i.e., approximately $\log_2 k$ iterations *both in the worst case and on average*. Despite the better worst-case bound, binary search is not always preferable to sequential search, because (1) each iteration requires more work and (2) the *average* number of iterations can be smaller for sequential search than for binary search; i.e., we may have $\mathbb{E}[I] < \log_2 k$ regardless of the size of k. This would happen for example if most of the probability mass is concentrated on the first couple of values.

The binary search algorithm can be modified in many different ways. For example, if $k = \infty$, we can start with an arbitrary value of j, double it until $F(x_{j-1}) \geq U$, and start the algorithm with this j and $i = j/2$. Only a finite portion of the table, that contains most of the probability mass, would be precomputed in this case. The other values are computed only when needed. This can also be done if k is finite but large.

Indexed search. The fastest inversion techniques when k is large use indexing or buckets to speed up the search (Chen and Asau 1974, Bratley, Fox, and Schrage 1987, Devroye 1986). We can partition the interval $(0,1)$ into c subintervals of equal sizes and start the search at a pre-tabulated initial value of i that depends on the subinterval in which U falls. Define $i_s = \inf\{i : F(x_i) \geq s/c\}$ for $s = 0, \ldots, c$, so that when $U \in [s/c, (s+1)/c)$, the final value of i must be in the set $\{i_s, \ldots, i_{s+1}\}$. After generating U, we can compute the random variable $S = \lfloor cU \rfloor$, the number of the subinterval to which U belongs, and search for I in the corresponding set by sequential or binary search. This gives:

Indexed search (combined with sequential search);
generate $U \sim U(0,1)$;
let $s = \lfloor cU \rfloor$ and $i = i_s$;
while $F(x_i) < U$ do $i = i + 1$;

return x_i.

Indexed search (combined with binary search);
generate $U \sim U(0,1)$;
let $s = \lfloor cU \rfloor$, $i = i_s$, and $j = i_{s+1}$;
while $i < j - 1$ do
  $m = \lfloor (i + j)/2 \rfloor$;
  if $F(x_{m-1}) < U$ then $i = m$ else $j = m$;
return x_i.

By conditioning on S we obtain that for both versions, the *expected* number of iterations of the "while" loop cannot exceed

$$\sum_{s=0}^{c-1} \mathbb{P}[S = s](i_{s+1} - i_s) = \frac{1}{c} \sum_{s=0}^{c-1} (i_{s+1} - i_s) = \frac{k}{c}.$$

Although tighter (and more complicated) bounds can be developed, this one already gives enough information for our purpose. With a larger value of c, the search is faster (on the average) but the setup is more costly and a larger table is required. The best compromise depends on the situation (e.g., the value of k, the number of variates we plan to generate, etc.). If k is not excessively large, we may take $c = k$ or even $c > k$, for example, to get an extremely fast algorithm. For $c = 1$, we recover the basic sequential and binary search algorithms given earlier.

A well-implemented indexed search with a large enough c is competitive with the fastest algorithms, including the alias method (Section 4.2), and it has the advantage of implementing inversion. It is hard to beat. When k/c is not too large, the sequential version is generally preferable because it has smaller overhead.

All the search methods discussed here can be easily adapted to several related situations. For example, to generate a random variate from a piecewise-linear (or piecewise-polynomial) distribution by inversion, after generating $U \sim U(0,1)$, we must first determine what piece corresponds to this U. This can be implemented by any of the search methods that we have examined. Then we perform an interpolation step on the appropriate piece (Bratley, Fox, and Schrage 1987).

4.1.2 Inversion for Continuous Distributions

For arbitrary discrete distributions, inversion can be exact, in the sense that the returned value of X is not an approximation. For general continuous distributions however, $F(X) - U = 0$ can often be solved only approximately. A measure of accuracy of the approximation can be defined in different ways, depending on the distribution, among other things (Devroye 1986). Suppose $\tilde{X}$ is our approximation to the exact solution X. If X takes its values in a bounded interval $[a, b]$, for example, the *absolute error* $|\tilde{X} - X|$ could be the right thing to check. But if the distribution of X has an infinite tail, its density far in the tail must be close to zero and a very small change in U can give a large variation in X in that area, so controlling the absolute error $|\tilde{X} - X|$ becomes impractical. Then we can use $|F(\tilde{X}) - U|$

as an alternative measure of error. Sometimes, we may want to bound the *relative error* $|\tilde{X} - X|/|X|$ instead. When solving $F(X) - U = 0$ approximately by an iterative method, the stopping criterion can be based on any of these error measures.

Again, we distinguish (a) the case where it is worth investing in an expensive initialization, because the generator is to be used many times with the same parameters, and (b) the case where the generator will be called only a few times or will be called with different shape parameters of the distribution. Case (b) often occurs in an optimization procedure that constantly changes the parameter of the distribution, for example. We start with this second case, where no table is precomputed.

Search Without Tables. When an algorithm is available for computing F, the simplest and most robust method for finding a root x^* of $F(X) - U = 0$ is binary search. In its basic form it requires that we start with an interval guaranteed to contain the root x^*. This interval is then halved recursively until its size is small enough or the increase of F over that interval is small enough. Of course, we would like the initial interval to be as short as possible. But for distributions with infinite (or very wide) support, selecting a short initial interval that contains x^* is not always straightforward. Then, we can guess some initial interval and enlarge it if it does not contain x^*.

We now give an algorithm that implements this idea for an arbitrary continuous distribution over $\mathbb{R}$. It is adapted from Cheng (1998). In this algorithm, we first select two numbers $x_{\min} < x_{\max}$ such that $x_{\min} < 0$ if $F(0) > 0$, $x_{\max} > 0$ if $F(0) < 1$, $F(x_{\min})$ is close to 0, and $F(x_{\max})$ is close to 1. If the density is nonzero only on a bounded interval, we can select $x_{\min}$ and $x_{\max}$ as the limits of that interval. We then have $F(x_{\min}) = 0$ and $F(x_{\max}) = 1$, so in that case the first two while loops can be removed and the numbers $x_{\min}$ and $x_{\max}$ can have any sign. After the initialization (i.e., in the final "while" loop), the interval $[x_1, x_2]$ always contains the root x^*. Here, we stop when either the interval length $x_2 - x_1$ or the change in F over that interval, is deemed small enough. For this, we select two accuracy thresholds ϵ_{x} and ϵ_{u}, which are small positive real numbers typically between 10^{-8} and 10^{-15}.

Binary search for continuous distribution;
generate $U \sim U(0,1)$;
let $x_1 = x_{\min}$ and $x_2 = x_{\max}$;
while $F(x_2) < U$ do
  $x_1 = x_2$ and $x_2 = 2x_2$; /* valid if $x_2 > 0$ */
while $F(x_1) > U$ do
  $x_2 = x_1$ and $x_1 = 2x_1$; /* valid if $x_1 < 0$ */
while $(x_2 - x_1 > \epsilon_{\mathrm{x}})$ and $(F(x_2) - F(x_1) > \epsilon_{\mathrm{u}})$ do
  $x = (x_1 + x_2)/2$;
  if $F(x) < U$ then $x_1 = x$ else $x_2 = x$;
  /* Invariant: at this stage, X always belongs to $[x_1, x_2]$. */
return $x = (x_1 + x_2)/2$.

Binary search is simple and robust. It works even if the density is zero in some interval or if F is discontinuous (then, $F(x) - U$ may not converge to 0 because $F(x) = U$ may have no solution, but x nevertheless converges to $F^{-1}(U)$). However, it is not very efficient: we only gain one bit of accuracy per iteration. If we start with an interval of length 1, we need for

example 52 iterations (and 52 evaluations of F) to get a value of x with 52 bits of accuracy. When F is smooth, other techniques such as the *false position*, *secant*, and *Newton-Raphson* algorithms converge much faster.

In the *false position* method (also called *regula falsi*), instead of taking the next point in the middle of the current interval, we interpolate linearly between $(x_1, F(x_1))$ and $(x_2, F(x_2))$ and take the point x at which the linear interpolation equals U. That is, we replace "$x = (x_1 + x_2)/2$" in the "while" loop by

$$x = x_1 + (x_2 - x_1)\frac{U - F(x_1)}{F(x_2) - F(x_1)},$$

hoping that this number is closer to the root x^* than the middle value.

For the *secant method*, x is computed in the same way, but it always replaces the least recently computed value between x_1 and x_2, i.e., we never update the same side of the interval twice in a row. With this approach, the interval $[x_1, x_2]$ does not always contain solution x^*, whereas with the false position and binary search methods, it always does. The secant method is faster when F is smooth and we are close to the solution, but otherwise less robust; it may fail if F is too wavy.

The *Newton-Raphson* method does not use an interval $[x_1, x_2]$ that contains the solution x^*. It starts with an initial guess of x^*, say x, and updates the guess at each iteration by taking the point where the tangent to the graph of F at the current x crosses level U. In other words, it uses a linear extrapolation whose slope is obtained by evaluating the derivative of F at x. This derivative is simply the density $f(x)$. This method is guaranteed to converge if it is restricted to an area where the density f is always increasing and we start from the right of x^*, or the density f is always decreasing and we start from the left of x^*. If this condition is satisfied and if an efficient algorithm is available to compute the density, this is the method of choice. For a unimodal density with mode at $x = x_m$, for example, if $U \geq F(x_m)$ we can apply the method by starting in the interval $[x_m, \infty)$ and otherwise starting in the interval $(-\infty, x_m]$. A very simple choice in this case is to start at x_m, as in the following algorithm, although starting closer to the solution would make the convergence much faster in general.

Inversion by Newton-Raphson for continuous distribution;
generate $U \sim U(0, 1)$;
let $x = x_m$;
while $|F(x) - U| > \epsilon_u$ do
    $x = x - (F(x) - U)/f(x)$;
return x.

If the density f has bounded derivative, Newton-Raphson's method converges quadratically: If ϵ_n is the absolute error $|x - x^*|$ at iteration n, then

$$\epsilon_{n+1} \approx \epsilon_n^2 (f'(x^*)/(2f(x^*)))$$

when we are close enough to the solution x^*. If $f'(x^*)/[2f(x^*)]$ is not too large, this means (very roughly) that each iteration squares the error, i.e., doubles the number of digits of accuracy in the solution. For the secant method, on the other hand, we have

$$\epsilon_{n+1} \approx \epsilon_n^{\gamma}[f'(x^*)/(2f(x^*))]^{1/\gamma}$$

where $\gamma = (1+\sqrt{5})/2 \approx 1.618$ is the *golden ratio*, so each iteration (very roughly) multiplies the number of digits of accuracy by 1.618. Practically speaking, this is almost as good as Newton-Raphson and there is no need to evaluate the density. For comparison, each iteration of binary search only adds one bit of accuracy. These convergence rates are asymptotic, i.e., valid in the limit when we converge to the solution. The false position method often performs quite well in practice and has the advantage that it always brackets the root, but in terms of its guaranteed convergence, we can prove no better than $\epsilon_{n+1} \leq K\epsilon_n$ for some constant K.

In situations where convergence does not seem to occur with the fast methods, e.g., if f' changes too rapidly, we can always switch to the slower but more reliable binary search. A good starting point is often an important ingredient for the fast methods. In some cases, we may have to start with a few binary search iterations to find a reasonable starting point. Hörmann, Leydold, and Derflinger (2004) suggest switching from the false position method to binary search if the method has failed to converge after 50 iterations, and using *one step* of binary search whenever the same side of the interval was updated in the last two steps of the false position method.

A more refined and robust root-finding algorithm is the method of Brent (1971); see also Brent (1973), Chapter 4, and Press et al. (1992), Chapter 9. Brent's method provides superlinear convergence while being more robust than the secant and Newton-Raphson methods. We recommend it as an efficient general-purpose root-finding method for monotone functions. It maintains an interval $[x_1, x_2]$ that always contains the solution. At each step, if function evaluations are available at three distinct points, inverse quadratic interpolation is used to find the next trial point x, otherwise the secant method is used. If this new trial point x does not bring sufficient improvement (as measured by a specific set of criteria), it is discarded and we switch to bisection (which is slower but more robust) for this step.

It is sometimes worthwhile to make a change of variable of the form $X = \varphi(Y)$, replacing the equation $F(X) - U = 0$ by $F(\varphi(Y)) - U = 0$ (Devroye 1986, page 31). If φ is well-chosen, $G = F \circ \varphi$ could be much smoother and nicer than F and the standard root-finding methods would then work better when applied to find a root Y of $G(Y) - U = 0$ than when applied directly to $F(X) - U = 0$. This can happen, for example, if $\varphi : [0,1] \to \mathbb{R}$ is a rough (and very simple) approximation of F^{-1}, so $G = F \circ \varphi$ is close to the identity function over the interval $[0,1]$.

Using Precomputed Tables. If we plan to generate many copies of X from the same distribution, it may be worth spending time to precompute an index, mapping each value of $U \in (0,1)$ to a short interval that contains X. Like in the discrete case, we may select an integer $c > 0$, then compute (by any accurate method) and tabulate the values $x_s = F^{-1}(s/c)$ for $s = 0, \ldots, c$. To generate X, we first generate U and compute $S = \lfloor cU \rfloor$. Then we know that $X \in [x_S, x_{S+1}]$, so we can start the false position method or binary search with this interval, or start Newton-Raphson from somewhere in that interval. The choice of c certainly depends on the situation, but it is rarely profitable to go beyond a few hundred. Observe that once we are close enough to the exact solution X, doubling c only adds one bit of accuracy to the initial solution x (by halving the intervals) whereas one additional Newton-Raphson iteration suffices to *double* the number of bits of accuracy. The main purpose of the index is to make sure that we do not start too far from the solution.

Here we have separated $[0, 1)$ into c intervals of equal probabilities. Ahrens and Kohrt (1981) argue that this is not necessarily the best choice, because this often leads to intervals for X that are too large in the tails of the distribution. It is often better to use intervals with smaller probabilities in the tails.

Precomputed tables to avoid evaluating F at generation time. The function F is often costly to compute or just unavailable, We can avoid evaluating it when we generate X by constructing an explicit approximation for F^{-1}. The methods discussed here are for distributions with bounded support $[a, b]$, i.e., we assume that $F(a) = 0$, $F(b) = 1$, and a and b are finite. But they can be applied to most other distributions by truncating the tails far enough for the error to be negligible.

Hörmann and Leydold (2003) propose a method that approximates F^{-1} by *piecewise-cubic Hermite interpolation*, given a function that computes F. This interpolating function is a variant of a cubic spline that matches the value of F and its derivative f at selected points $a = x_0 < x_1 < \cdots < x_c = b$ and is a polynomial of degree 3 over each interval $[x_s, x_{s+1}]$. Distributions with unbounded support can be handled by truncating the "numerically negligible" part of each tail. The points $x_0, x_1, \ldots, x_c$ and the value of c are selected in the initialization step by an adaptive method based on a heuristic approximation of the maximum interpolation error. The coefficients of the polynomials for the different intervals are computed and stored in a table, together with the values of x_s and $F(x_s)$. To generate a value of X, their algorithm generates $U \sim U(0, 1)$, uses indexed search to find the interval number S that corresponds to this U, then evaluates at U the cubic polynomial for this interval. Since F is never reevaluated during the generation step, this method generally requires a much larger c than when the table is used to find starting points for an iterative method, for a comparable accuracy.

Derflinger, Hörmann, and Leydold (2010) provide an algorithm that constructs an approximation of F^{-1} to the accuracy specified by the user, for the situation where only the density f of X is available, again under the assumption that the distribution has bounded support. This algorithm is an improvement over similar methods in Ahrens and Kohrt (1981), who proposed techniques based on piecewise-polynomial interpolation using Chebyshev polynomials. These methods require a rather expensive and complicated initialization, but once the setup is done, they are extremely fast.

4.2 The Alias and Acceptance-Complement Methods

To generate a random variate X by inversion over the finite set $\{x_0, \ldots, x_{k-1}\}$, sequential and binary search require $O(k)$ and $O(\log k)$ time, respectively. Indexed search with an index of size $O(k)$ requires only $O(1)$ time per variate on average, but $O(k)$ or $O(\log k)$ (depending on how we make the search inside each interval) in the worst case. The *alias method* (Walker 1974, Walker 1977) can generate X in $O(1)$ time per variate *in the worst case*, after a table setup that takes $O(k)$ time and space. In principle, this is better than the indexed search, especially if we are not willing to have a long generating time for X once in a while. An

important drawback, however, is that *it does not implement inversion*; the transformation from U to X is not monotone. It is also usually not faster on average than indexed search with the same table size. This is nevertheless an elegant and interesting method that deserves examination.

To explain the principle, consider a bar diagram of the distribution, where each index i has a bar of height $p_i = \mathbb{P}[X = x_i]$. The idea is to "equalize" the bars so that they all have height $1/k$, by cutting-off bar pieces and transferring them to other bars. We can think of it as "moving the mountains into the lakes." This is done in a way that in the new diagram, each bar i contains one piece of size q_i (say) from the original bar i and one piece of size $1/k - q_i$ from another bar whose index j, denoted $A(i)$, is called the *alias* value of i. The setup procedure initializes two tables, A and R, where $A(i)$ is the alias value of i and $R(i) = (i-1)/k + q_i$. This $R(i)$ can be interpreted as follows: when $U \in [(i-1)/k, i/k)$, the generated value corresponds to one of the two pieces in bar i; it is the value from the original piece if and only if $U < R(i)$ (this happens with probability q_i). Thus, to generate X after the initialization step, we do:

Generating X by the alias method;
generate $U \sim U(0,1)$;
let $i = \lceil kU \rceil$;
if $U < R(i)$ return x_i else return $x_{A(i)}$;

The following method, due to Kronmal and Peterson (1979), initializes the tables A and R.

Initialization for the alias method;
let $H = \{i : p_i > 1/k\}$ (the set of "high" bars);
let $L = \{j : p_j < 1/k\}$ (the set of "low" bars);
let $q_i = p_i$ for all i;
while $L \neq \emptyset$ do
  choose any $i \in H$ and $j \in L$;
  remove j from L and let $R(j) = (j-1)/k + q_j$;
  let $A(j) = i$ and $q_i = q_i - (1/k - q_j)$;
  if $q_i \leq 1/k$ remove i from H;
  if $q_i < 1/k$ add i to L;

At each iteration of this setup procedure, we select a "low" bar j and fill it exactly up to level $1/k$ by cutting a piece from bar i and moving it to bar j. Then j can be removed from the set L. The cardinality of L decreases by 1 at each iteration, which implies that the procedure converges after a finite number of iterations. At the end, since there are k bars with total height 1 and none is lower than $1/k$, all the bars must be at height $1/k$. During the algorithm, all the bars in $H \cup L$ always contain a single piece (from the original bar only). A second piece can be added only when the bar is removed from L. Thus, no bar can contain more than two pieces at the end.

To approximate the monotonicity more closely, at each iteration we can (as a heuristic) select the values i and j so that $|j - i|$ is as small as possible.

♣ Add figure + example.

There is a version of the alias method for continuous distributions, called the *acceptance-complement* method (Kronmal and Peterson 1984, Devroye 1986, Gentle 2003). The idea is to decompose the density f of the target distribution as the convex combination of two densities f_1 and f_2, $f = wf_1 + (1-w)f_2$ for some real number $w \in (0,1)$, in a way that $wf_1 \le g$ for some other density g and so that it is easy to generate from g and f_2. The algorithm works as follows: Generate X from density g and $U \sim U(0,1)$; if $Ug(X) \le wf_1(Y)$ return X, otherwise generate a new X from density f_2 and return it.

4.3 Composition and Convolution

Suppose F is a *mixture* of several distributions, i.e., $F(x) = \sum_j p_j F_j(x)$, or more generally $F(x) = \int F_y(x)dH(y)$. To generate X from F, one can generate $J = j$ with probability p_j, or Y from H, then generate X from F_J or F_Y. This *composition algorithm* is useful for generating from *compound* distributions such as the hyperexponential and from more general mixtures.

It is also frequently used to design efficient algorithms for generating from complicated densities or more generally for generating random points in complicated surfaces and solids. The idea is to partition the surface or solid into pieces, where piece j has proportion p_j. We then select a piece, giving piece j a probability p_j for each j, then draw a random point over that piece. If the partition is defined so that it is fast and easy to generate from the large pieces, then this method can be fast on average. In Section 4.4.2, we will see how to combine this technique with the rejection method.

A dual method to composition is the *convolution method*, which can be used when $X = Y_1 + Y_2 + \cdots + Y_k$, where the Y_i's are independent with specified distributions. With this method, one just generates the Y_i's and sum up. This requires at least k uniforms. Examples of random variables that can be expressed as sums like this include the hypoexponential (a sum of exponentials), Erlang (a sum of exponentials with the same mean), the binomial (a sum of Bernoulli random variables), the chi-square (a sum of squares of standard normals), and many others.

4.4 The Rejection Method

4.4.1 The principle

Generating from a density f. This technique was introduced by von Neumann (1951) and is the most important variate generation method after inversion. Suppose we want to generate a continuous random variable X having a complicated density f. Let

$$\mathcal{S}(f) = \{(x,y) \in \mathbb{R}^2 : 0 \le y \le f(x)\},$$

the surface under the curve of f.

Proposition 4.1 *If (X, Y) is a random point uniformly distributed in $\mathcal{S}(f)$, then X has density f. Conversely, if X has density f and the conditional distribution of Y given X is $U(0, f(X))$, then (X, Y) is uniformly distributed over $\mathcal{S}(f)$.*

Proof. Since the total area of $\mathcal{S}(f)$ is 1, if (X, Y) is uniform over $\mathcal{S}(f)$ then $\mathbb{P}[X \le x]$ is equal to the area of $\{(z, y) \in \mathcal{S}(f) : z \le x\}$, which is $\int_{-\infty}^{x} f(z)dz$. This proves that X has density f. Conversely, under the conditions of the second part, for any $(x, y) \in \mathbb{R}^2$,

$$\begin{aligned}\mathbb{P}[X \le x,\, Y \le y] &= \int_{-\infty}^{x} \mathbb{P}[Y \le y \mid X = z] f(z)dz \\ &= \int_{-\infty}^{x} \min(1,\, y/f(z)) f(z)dz \\ &= \int_{-\infty}^{x} \min(f(z),\, y)dz,\end{aligned}$$

which is the area of the part of $\mathcal{S}(f)$ that is below y and to the left of x. This implies that for any measurable subset $\mathcal{D}$ of $\mathcal{S}(f)$, the probability that (X, Y) belongs to $\mathcal{D}$ is proportional to the area of $\mathcal{D}$. That is, (X, Y) is uniformly distributed over $\mathcal{S}(f)$.

So we can generate X simply by generating (X, Y) uniformly in $\mathcal{S}(f)$, but how do we do that when this surface has a complicated shape? The idea is simple: Pick a larger surface $\mathcal{B}$ that contains $\mathcal{S}(f)$ and has a much simpler shape (a rectangular box, for example), and generate a random point (X, Y) uniformly over $\mathcal{B}$. If (X, Y) belongs to $\mathcal{S}(f)$, take it as a random point from $\mathcal{S}(f)$, otherwise try again. Conditional on $(X, Y) \in \mathcal{S}(f)$, this point is indeed uniformly distributed over $\mathcal{S}(f)$, as Proposition 4.2 will tell us. Since the successive attempts until the first success are independent, the first point (X, Y) that belongs to $\mathcal{S}(f)$ is uniformly distributed over $\mathcal{S}(f)$ and its first coordinate is a random variate with density f, from Proposition 4.1.

The general rejection idea. The next proposition deserves to be introduced in a more general setting than what we just described: We replace the surface $\mathcal{S}(f)$ by an arbitrary set $\mathcal{A}$ in the d-dimensional real space $\mathbb{R}^d$, with $0 < \text{vol}(\mathcal{A}) < \infty$, where $\text{vol}(\mathcal{A})$ denotes the volume of $\mathcal{A}$ (in two dimensions, it is the area of the surface $\mathcal{A}$). (All sets considered here are assumed measurable, implicitly.) Let $\mathcal{B} \subset \mathbb{R}^d$ be another set such that $\mathcal{A} \subseteq \mathcal{B}$ and $\text{vol}(\mathcal{B}) < \infty$, and selected so that it is easy to generate a random point uniformly in $\mathcal{B}$. For example, $\mathcal{A}$ can be a solid with complicated shape and $\mathcal{B}$ a rectangular box that contains $\mathcal{A}$. To generate a random point uniformly in $\mathcal{A}$, we generate one in $\mathcal{B}$, return it if it belongs to $\mathcal{A}$, and otherwise try again, until the generated point belongs to $\mathcal{A}$. This is the *rejection method*. The successive attempts being independent, the retained point is distributed according to the conditional distribution given that the point falls in $\mathcal{A}$.

Proposition 4.2 *Suppose a random point $\boldsymbol{X}$ is generated uniformly in the set $\mathcal{B}$. Then, conditional on the event $\{\boldsymbol{X} \in \mathcal{A}\}$, $\boldsymbol{X}$ is uniformly distributed in $\mathcal{A}$.*

Proof. For any set $\mathcal{D} \subseteq \mathcal{A}$, we have $\mathbb{P}[\boldsymbol{X} \in \mathcal{D}] = \text{vol}(\mathcal{D})/\text{vol}(\mathcal{B})$ and therefore

$$\mathbb{P}[\boldsymbol{X} \in \mathcal{D} \mid \boldsymbol{X} \in \mathcal{A}] = \frac{\mathbb{P}[\boldsymbol{X} \in \mathcal{D} \cap \mathcal{A}]}{\mathbb{P}[\boldsymbol{X} \in \mathcal{A}]} = \frac{\text{vol}(\mathcal{D})/\text{vol}(\mathcal{B})}{\text{vol}(\mathcal{A})/\text{vol}(\mathcal{B})} = \frac{\text{vol}(\mathcal{D})}{\text{vol}(\mathcal{A})}.$$

This means that the conditional distribution of $\boldsymbol{X}$ given that $\boldsymbol{X} \in \mathcal{A}$ is uniform over $\mathcal{A}$.

Rejection algorithm with a hat function. The standard way of constructing a set $\mathcal{B}$ that contains $\mathcal{S}(f)$ for the rejection method is to select another density g and a constant $a \geq 1$ such that

$$f(x) \leq h(x) \stackrel{\text{def}}{=} ag(x)$$

for all x, and such that generating variates from the density g is easy. The function h is called a *hat function* or *majoring function* and g is the corresponding *sampling density*. Then we apply the above setting with $\mathcal{A} = \mathcal{S}(f)$ and

$$\mathcal{B} = \mathcal{S}(h) = \{(x, y) \in \mathbb{R}^2 : 0 \leq y \leq h(x)\},$$

the surface under h. This gives the following *rejection* (or *acceptance-rejection*) algorithm to generates X with density f (e.g., Devroye 1986, Hörmann, Leydold, and Derflinger 2004):

Algorithm 4 : Rejection algorithm

repeat
  generate X from the density g and $V \sim U(0,1)$, independent;
until $Vh(X) \leq f(X)$;
return X.

Proposition 4.3 *This procedure generates a random variate X with density f.*

Proof. From the second part of Proposition 4.1 with f replaced by g, each pair $(X, Vg(X))$ generated in the "while" loop is uniformly distributed in the surface under g. Then, each pair $(X, Vh(X))$ is uniformly distributed in B, the surface under h, because h is just a rescaling of g. By taking $\mathcal{A} = \mathcal{S}(f)$ in Proposition 4.2, we then get that when we exit the loop, the pair $(X, Vh(X))$ is uniformly distributed in $\mathcal{S}(f)$. The result then follows from the first part of Proposition 4.1.

To give additional insight, we provide a second, more self-contained, proof. If F is the distribution function corresponding to density f, we have

$$\begin{aligned}
& P[X \leq x \text{ and } X \text{ is accepted}] \\
= \; & P[X \leq x \text{ and } V \leq f(X)/h(X)] \\
= \; & \int_{-\infty}^{\infty} P[z \leq x \text{ and } V \leq f(z)/h(z) \mid X = z] g(z) dz \\
= \; & \int_{-\infty}^{x} [f(z)/h(z)] g(z) dz \\
= \; & \int_{-\infty}^{x} [f(z)/a] dz \;=\; F(x)/a.
\end{aligned}$$

From that, we obtain

$$P[X \le x \mid X \text{ is accepted}] = \frac{P[X \le x \text{ and } X \text{ is accepted}]}{P[X \text{ is accepted}]} = \frac{F(x)/a}{1/a} = F(x).$$

Thus, the conditional distribution function of X is F, the correct one.

The number R of turns into the "repeat" loop of the rejection algorithm is one plus a geometric random variable with parameter $1/a$, so $\mathbb{E}[R] = a$. Thus, we want $a \ge 1$ to be as small as possible, i.e., we want to minimize the area between f and h. There is generally a compromise between bringing a close to 1 and keeping g simple.

Unknown normalization constant. In our description of the rejection method, we have assumed that f is known and can be computed easily. But in reality we only need to know f up to a multiplicative constant $\kappa > 0$, i.e., it is sufficient to know $\tilde{f} = \kappa f$ even if we do not know κ. We just replace f by $\tilde{f}$ in the algorithm and make sure that a is large enough so that $\tilde{f}(x) \le h(x) \stackrel{\text{def}}{=} ag(x)$ for all x. Then the geometric random variable $R - 1$ has parameter κ/a, and $\mathbb{E}[R] = a/\kappa$.

♣ Applications of this in statistics... see Evans and Swartz (2000).

Examples. The following examples are oversimplified compared with the most efficient rejections methods currently available, but they illustrate the main ideas without getting into excessively complicated constructions.

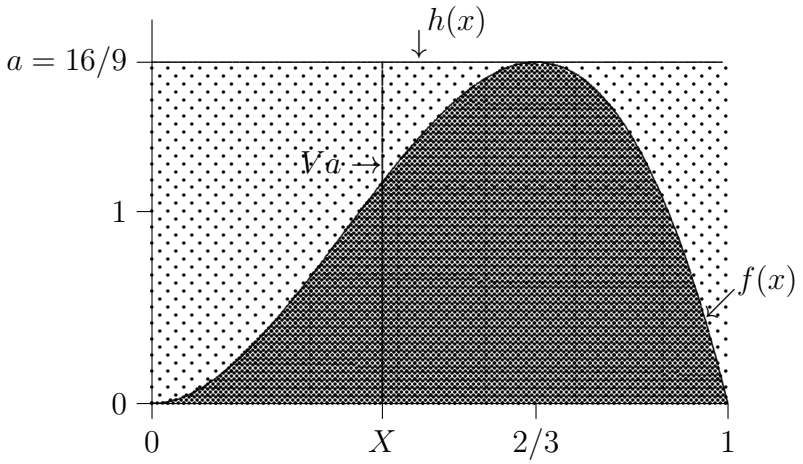


Fig. 4.1. A rejection method with uniform hat function for the Beta(3, 2) density.

Example 4.2 Suppose we want to generate X from the beta distribution with parameters $(\alpha, \beta) = (3, 2)$, over the interval $(0, 1)$. The density is $f(x) = 12x^2(1-x)$. It has a maximum at $x_* = 2/3$ (this is easily found by finding x such that $f'(x) = 0$) and its maximum value is

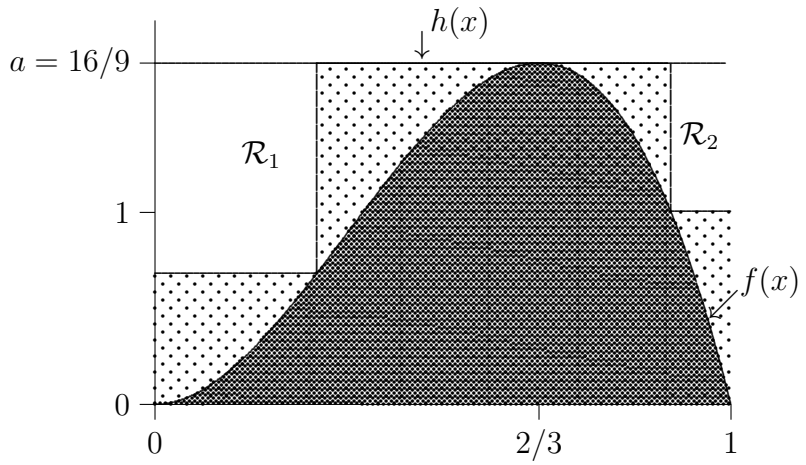


Fig. 4.2. A second hat function for the Beta(3, 2) density.

$f(2/3) = 12(2/3)^2(1/3) = 16/9 \approx 1.77778$. We can thus take $h(x) = a = 16/9$ for $0 \le x \le 1$ as a hat function. The corresponding sampling density g is uniform over $[0, 1]$ and $a = 16/9$ is the smallest a that we can choose for a uniform sampling density. This is illustrated in Figure 4.1, where $\mathcal{S}(f)$ is the darkly-shaded area and $\mathcal{B} = \mathcal{S}(h)$ is the surface under the top horizontal line. At each turn of the loop, we generate $X \sim U(0, 1)$, then $V \sim U(0, 1)$, and accept X if and only if $aV \le f(X)$, i.e., if (X, aV) falls into the dark area. The acceptance probability is $9/16$ and the expected number of turns into the loop is $\mathbb{E}[R] = a = 16/9$.

To reduce the value of a, we can take a slightly less simplistic hat function, as in Figure 4.2. This hat function has the form

$$h(x) = \begin{cases} f(x_1) & \text{for } x < x_1; \\ 16/9 & \text{for } x_1 \le x \le x_2; \\ f(x_2) & \text{for } x > x_2, \end{cases}$$

where x_1 and x_2 are two constants such that $0 < x_1 < 2/3 < x_2 < 1$. The rejection area (between f and h) is thus reduced by $x_1(16/9 - f(x_1)) + (1 - x_2)(16/9 - f(x_2))$, which is the combined area of $\mathcal{R}_1$ and $\mathcal{R}_2$ in Figure 4.2. To minimize a, we want to maximize this reduction, i.e., select the values of x_1 and x_2 that maximize $x_1(16/9 - f(x_1))$ and $(1 - x_2)(16/9 - f(x_2))$, respectively. These values turn out to be $x_1 = 0.281023$ and $x_2 = 0.89538$. The area of $\mathcal{R}_1 \cup \mathcal{R}_2$ is then 0.38881 and a is reduced from 1.77778 to 1.38997. Since the sampling density $g = h/a$ is piecewise-constant, the corresponding distribution function G and its inverse are piecewise-linear, so it is easy to generate from it by inversion (Exercise 4.7).

Of course, we could use more than three pieces in our piecewise-constant hat function h, to further reduce a. For arbitrary breakpoints $0 = x_0 < x_1 < x_2 < \cdots < x_c = 1$, we can take

$$h(x) = \begin{cases} f(x_j) & \text{if } x_{j-1} < x \le x_j < x_*; \\ f(x_{j-1}) & \text{if } x_* < x_{j-1} < x \le x_j; \\ 16/9 & \text{otherwise, i.e., if } x_{j-1} < x, x_* \le x_j. \end{cases}$$

If the breakpoints are optimized to minimize a for any given c, then $a \to 1$ when $c \to \infty$.

Why not take the hat function piecewise-linear instead of piecewise-constant? For a given number of breakpoints, this can certainly give a smaller surface between f and h, and thus a smaller value of a. However, a piecewise-linear sampling density means a quadratic distribution function, whose inversion involves taking a square root, which is more expensive than computing a linear function. So even if we reduce a, we may lose on overall efficiency. □

Example 4.3 Consider generating a gamma random variate X with parameters (α, λ), for $\alpha \geq 1$. Since λ is just a scale parameter, we can take it equal to 1 and then divide the generated random variate by λ to rescale it correctly. So, suppose $\lambda = 1$. The density is then

$$f(x) = \frac{x^{\alpha-1}e^{-x}}{\Gamma(\alpha)} \qquad \text{for } x > 0.$$

Fishman (1976) proposes the exponential hat function $h(x) = ag(x)$ where $g(x) = \exp(-x/\alpha)/\alpha$, the density of an exponential with mean α, from which we can easily generate a random variate by inversion. The minimal value of a for which $f(x) \leq h(x)$ for all x is $a = \alpha^\alpha \exp(1-\alpha)/\Gamma(\alpha)$. This gives $a = 1$ for $\alpha = 1$ (the exponential case), $a \approx 1.83$ for $\alpha = 3$, $a \approx 4.18$ for $\alpha = 15$, and a increases with α. This method is not very efficient, but it can be used as a "quick-and-simple" solution when α is small. An easy improvement would be to replace the exponential hat by a constant function equal to $f(x_m)$ over the interval $[0, x_m]$, where x_m is the mode of the density f. But in any case, there are better (although more complicated) rejection methods for the gamma distribution (Hörmann, Leydold, and Derflinger 2004).

To illustrate the fact that there is no need to know the normalization constant $\kappa = 1/\Gamma(\alpha)$ in this example, suppose we only know that f is *proportional* to $\tilde{f}$ defined by $\tilde{f}(x) = x^{\alpha-1}e^{-x}$ for $x > 0$, but we don't know the proportionality constant. Then we select a sampling density g and a constant a such that $\tilde{f}(x) \leq ag(x)$ for all x. We sample X from g and $V \sim U(0,1)$ (independently) until we get $\tilde{f}(X) \leq Vag(X)$, and return this last value of X. □

Example 4.4 (From Devroye 1986, page 44, with small modifications.) Let f be the standard normal density, $f(x) = (2\pi)^{-1/2}e^{-x^2/2}$, and suppose we take h proportional to the Laplace density $g(x) = (1/2)e^{-|x|}$, for $x \in \mathbb{R}$. By taking $a = \sqrt{2e/\pi}$, we obtain

$$h(x) = ag(x) = (2e/\pi)^{1/2}(1/2)e^{-|x|} = (2\pi)^{-1/2}e^{-|x|+1/2}.$$

Then,

$$2\ln(h(x)/f(x)) = x^2 - 2|x| + 1 = (1-|x|)^2 \geq 0,$$

which implies that $h(x) \geq f(x)$ for all x, so h is a valid hat function. To generate X from the Laplace density, we can generate $U \sim U(0,1)$ and take $X = -\ln(2U-1)$ if $U \geq 1/2$ and $X = \ln(1-2U)$ if $U < 1/2$. This is not inversion but it is equivalent to generating an exponential with a random sign, which is the same as a Laplace random variate. The condition $Vh(X) \leq f(X)$ is then the first inequality in:

$$2\ln V \leq 2\ln(f(X)/h(X)) = (1-|X|)^2 = (1+\ln|2U-1|)^2.$$

Putting these pieces together gives the algorithm:

A normal generator by rejection from the Laplace density;
repeat
generate $U \sim U(0,1)$ and $V \sim U(0,1)$, independent;
let $X = \ln|2U - 1|$;
until $2\ln V \le (1+X)^2$;
if $U < 1/2$ return X else return $-X$.

The expected number of pairs (U, V) that need to be generated is $a = \sqrt{2e/\pi} \approx 1.35$. □

Discrete distributions. The rejection method works for discrete distributions as well; it suffices to replace densities by probability mass functions. The sampling density g in this case can be either discrete or continuous. In the latter case, assuming that X takes integer values and $f(x) = \mathbb{P}[X = x]$ defines the probability mass function, the acceptance condition can be either $Vh(\lfloor X \rfloor) \le f(\lfloor X \rfloor)$ or $Vh(X) \le f(\lfloor X \rfloor)$, and the returned value is $\lfloor X \rfloor$ (Exercise 4.9). See also Devroye (1986) and Chapter 10 of Hörmann, Leydold, and Derflinger (2004) for further details and other variants.

Squeeze functions. Often, the density $f(x)$ is expensive to compute, so we would like to reduce the number of times we have to evaluate it when checking for rejection in the algorithm. This can be achieved by using *squeeze functions* s_1 and s_2 that are faster to evaluate and such that

$$0 \le s_1(x) \le f(x) \le s_2(x) \le h(x)$$

for all x. To verify the condition $Vh(X) \le f(X)$, we first check if $Vh(X) \le s_1(X)$, in which case we accept Y immediately, otherwise we check if $Vh(X) \ge s_2(X)$, in which case we reject X immediately. The value of $f(X)$ must be computed only when $Vh(X)$ falls between the two squeezes. Sequences of embedded squeezes can also be used, where the primary ones are the least expensive to compute, the secondary ones are a little more expensive but closer to f, etc. In practice, a squeeze s_1 below f is often used, but a squeeze s_2 above f is much less common.

Example 4.5 For the beta distribution in Example 4.2, suppose we use the piecewise-constant hat function with breakpoints $0 = x_0 < x_1 < x_2 < \cdots < x_c = 1$ defined there. For a squeeze s_1, we can take another piecewise-constant function with the same breakpoints, but for which $s_1(x) \le f(x)$ for all x. The largest function s_1 that satisfies these constraints is defined by $s_1(x) = \min(f(x_{j-1}),\, f(x_j))$ for $x_{j-1} < x \le x_j$. This gives

$$s_1(x) = \begin{cases} f(x_{j-1}) & \text{if } x_{j-1} < x \le x_j < x_*; \\ f(x_j) & \text{if } x_* < x_{j-1} < x \le x_j; \\ \min(f(x_{j-1}),\, f(x_j)) & \text{otherwise.} \end{cases}$$

This squeeze is constant over each piece and it can be evaluated faster than the beta density. □

4.4.2 Rejection with composition and recycling

The general idea here is to partition the surface $S(h)$ under the hat function into k pieces $S_0, \ldots, S_{k-1}$, where S_i takes a proportion p_i of the total surface. The pieces are constructed so that some of them are entirely contained in $S(f)$, the surface under f. Suppose the first k_0 pieces have this property. To generate X, we first generate a piece number I with the probabilities p_i (using, e.g., indexed search), then we generate a random point (X, Y) in that piece. If $I < k_0$, we return X immediately (no need to check for the acceptance condition). If $I \geq k_0$, we accept X if $Y \leq f(X)$, otherwise we start all over again. It is often convenient and safe to use the same uniform U to generate both I and X, and there is no need to generate Y when $I < k_0$.

Example 4.6 Consider again the beta distribution as in Examples 4.2 and 4.5, with piecewise-constant hat and squeeze functions with breakpoints $0 = x_0 < x_1 < x_2 < \cdots < x_c = 1$. For $j = 0, \ldots, c-1$, we split the surface under the hat function in the interval $[x_j, x_{j+1}]$ in two rectangular boxes: S_{2j} is the surface under the squeeze and S_{2j+1} is the surface between the squeeze and the hat. For $i = 0, \ldots, 2c-1$, let p_i be the area of box i divided by the total area under h and $R_i = p_0 + \cdots + p_i$ the total probability of the boxes with index smaller or equal to i. Note that $p_0 = p_{2c-2} = 0$ and that box i is over the interval $[x_j, x_{j+1}]$ where $j = \lfloor i/2 \rfloor$.

♣ Add a figure...

To generate X, we generate $U \sim U(0,1)$ and use it first to select the box number $I = \min\{i : R_i \geq U\}$. The left side of this box is at $x = x_J$ where $J = \lfloor I/2 \rfloor$. Conditional on I, $U - R_{I-1}$ has the uniform distribution over $[0, p_I]$ and we can use its value to generate X in the interval $[x_J, x_{J+1}]$: just take X so that a proportion $(U - R_{I-1})/p_I$ of the area of box S_I is on the left of X. This means selecting X so that $(U - R_{I-1})/p_I = (X - x_J)/(x_{J+1} - x_J)$, which gives

$$X = x_J + (U - R_{I-1})(x_{J+1} - x_J)/p_I.$$

Here, the uniform U used to generate I is *recycled* to generate X as well. Note that if each interval $[x_j, x_{j+1}]$ had only one box, this would be equivalent to generating the proposed X (before acceptance) by inversion from U. If I is *even* (box I is under the squeeze) X is accepted immediately. If I is *odd* (box I is between the squeeze and the hat), X is accepted with probability $(f(X) - s_1(X))/(h(X) - s_1(X))$. Summarizing, we get:

Rejection with composition for the beta distribution;
repeat
  generate $U \sim U(0,1)$;
  let $I = \min\{i : R_i \geq U\}$ and $J = \lfloor I/2 \rfloor$;
  let $X = x_J + (U - R_{I-1})(x_{J+1} - x_J)/p_I$;
  if I is even, return X;
  else
    generate $V \sim U(0,1)$;
    if $V \leq (f(X) - s_1(X))/(h(X) - s_1(X))$ return X;
until false.

□

4.5 Change of variables

4.5.1 General formulation

A key tool for developing efficient variate generation algorithms is a *change of variable*, i.e., a transformation of the coordinates. The principle applies in an arbitrary number of dimensions but here we describe it in two dimensions only (a) to keep the notation simpler and (b) because the changes of variables used in the context of random variate generation are often two-dimensional. In the rejection method, for example, we want to generate a random pair (X, Y) uniformly in the surface under the hat function.

The idea is to express the continuous random vector (X, Y) of interest as a function φ of another continuous random vector (U, V), where $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ is a one-to-one mapping between the two spaces. That is, we consider the change of variable

$$(x, y) = (\varphi_1(u, v),\ \varphi_2(u, v)) = \varphi(u, v)$$

where $\varphi_1 : \mathbb{R}^2 \to \mathbb{R}$ and $\varphi_2 : \mathbb{R}^2 \to \mathbb{R}$. Standard calculus tells us that (X, Y) has density f if and only if (U, V) has density t given by

$$t(u, v) = f(\varphi_1(u, v),\ \varphi_2(u, v))|J(u, v)|, \tag{4.1}$$

where $J(u, v)$, the Jacobian of the transformation φ, defined as the determinant of the matrix of partial derivatives of x and y as functions of u and v:

$$J(u, v) = \frac{\partial\varphi_1(u, v)}{\partial u}\frac{\partial\varphi_2(u, v)}{\partial v} - \frac{\partial\varphi_2(u, v)}{\partial u}\frac{\partial\varphi_1(u, v)}{\partial v}.$$

The idea is to generate (U, V) from the density t and compute $(X, Y) = \varphi(U, V)$ to get a random vector with the desired density f. Note that if t is the uniform density over some finite region $\mathcal{C}$, then (X, Y) will be uniformly distributed over $\varphi(\mathcal{C})$ if and only if $|J(u, v)|$ is constant over $\mathcal{C}$.

Example 4.7 (The Box-Muller method for normal random variates.) We want to generate a standard normal random variate X. Box and Muller (1958) proposed to exploit the fact that it is easier to generate a standard bivariate normal (X, Y), with density

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

over $\mathbb{R}^2$, by changing the Cartesian coordinates (x, y) to polar coordinates (r, θ) via $r^2 = x^2 + y^2$ and $\cos\theta = x/r$. The inverse transform is given by $x = r\cos\theta$ and $y = r\sin\theta$. So the idea is to generate the polar coordinates (R, Θ) from the appropriate density $t(r, \theta)$, and return the transformed coordinates $(X, Y) = (R\cos\Theta,\ R\sin\Theta)$.

♣ Add figure.

To find $t(r, \theta)$, note that the Jacobian of the transformation $(r, \theta) \to (x, y)$ has determinant

$$(\cos\theta)(r\cos\theta) - (\sin\theta)(-r\sin\theta) = r(\cos^2\theta + \sin^2\theta) = r,$$

so we have

$$t(r,\theta) = r f(r\cos\theta,\, r\sin\theta) = (r/2\pi)e^{-r^2/2}.$$

Since this density does not depend on θ, Θ must have the uniform density over $[0, 2\pi]$. Integrating with respect to θ over the interval $[0, 2\pi]$, we find that R has density

$$f_R(r) = re^{-r^2/2} \quad \text{for } r \geq 0.$$

The corresponding distribution function is $F_R(r) = 1 - e^{-r^2/2}$ and its inverse is defined by

$$F_R^{-1}(u) = \sqrt{-2\ln(1-u))}.$$

From this, it is easy to generate (R, Θ). Transforming back to Cartesian coordinates, to obtain a pair (X, Y) of independent standard normal random variates. If a single variate is needed, we can save the other for the next call, so the following algorithm needs to be executed n times to generate $2n$ normal random variates:

Box-Muller method to generate a pair of independent normals;
Generate $U_1 \sim U(0,1)$ and $U_2 \sim U(0,1)$, independently;
let $\Theta = 2\pi U_1$ and $R = \sqrt{-2\ln(1-U_2)}$;
return $(X, Y) = (R\cos\Theta,\, R\sin\Theta)$.

This change of variable is nice and elegant, but the method turns out to be slower in practice than inversion based on a well-crafted approximation of Φ^{-1}, because the sine, cosine, logarithm, and square root are too expensive to compute. However, one can get rid of the sine and cosine as follows (Marsaglia 1962).

Consider the point $(V_1, V_2) = (D\cos\Theta,\, D\sin\Theta)$, where R is replaced by $D = \sqrt{U_2}$. This point, located at distance D from the origin, turns out to be uniformly distributed in the disk of radius 1 centered at the origin (Exercise 4.13). We have $X = R\cos\Theta = RV_1/D$ and $Y = R\sin\Theta = RV_2/D$. We can generate (V_1, V_2) directly by generating random points in the square $[-1, 1]^2$ and retain the first one that falls inside the disk, i.e., for which $U_2 = D^2 = V_1^2 + V_2^2 < 1$. Then, we can compute $R/D = \sqrt{-2\ln(1-U_2)/U_2}$, $X = V_1 R/D$, and $Y = V_2 R/D$. This is known as the *polar method*. □

Transforming from Cartesian to polar coordinates is useful for developing elegant variate generation methods for several distributions other than the normal. See, e.g., Devroye (1996) for other examples, including the Student and beta distributions.

4.5.2 Rejection with a change of variable

In the standard rejection method, the aim is to generate a random point (X, Y) uniformly in $\mathcal{S}(f)$, the surface under f. With a change of variable, we generate instead a random point (U, V) uniformly in another surface $\mathcal{C}$, and transform it by some one-to-one mapping $\varphi : \mathcal{C} \to \mathcal{S}(f)$. To make sure that $(X, Y) = \varphi(U, V)$ is uniformly distributed over $\mathcal{S}(f)$ when (U, V) is uniformly distributed over $\mathcal{C}$. we assume that the Jacobian $J(u, v)$ is constant. If $|J(u, v)| \equiv K$, this means that φ maps each piece of area ϵ in $\mathcal{C}$ to a piece of area ϵ/K in $\mathcal{S}(f)$, and vice-versa,

Transposing the rejection method in the space of (U, V) coordinates, we seek a set $\bar{\mathcal{C}}$ that contains $\mathcal{C}$, from which it is easy to sample uniformly, and whose area is not much larger than that of $\mathcal{C}$. Typically, the set $\bar{\mathcal{C}}$ will be defined by choosing a hat function h in the (u, v) space such that $0 \le v \le h(u)$ whenever $(u, v) \in \mathcal{C}$. This gives:

Rejection method in transformed coordinates;
repeat
  generate (U, V) uniformly in $\bar{\mathcal{C}}$;
until $(U, V) \in \mathcal{C}$;
return $X = \varphi_1(U, V)$.

Proposition 4.4 *This algorithm generates a random variate X with density f.*

The goal here is to find a transformation φ and a set $\bar{\mathcal{C}}$ such that (a) it is easy to generate a point (U, V) uniformly in $\bar{\mathcal{C}}$, to check if it belongs to $\mathcal{C}$, and to compute $X = \varphi_1(U, V)$, and (b) the acceptance probability, given by $\text{area}(\mathcal{C})/\text{area}(\bar{\mathcal{C}})$, is near 1. We discuss special cases in what follows.

4.5.3 A univariate change of variable

Consider a one-to-one differentiable transformation $\tau : \mathbb{R} \to \mathbb{R}$ and put $X = \tau(U)$. If X has density f then $U = \tau^{-1}(X)$ has density t defined by $t(u) = f(\tau(u))\tau'(u)$. To generate X from f, we can generate U from t and return $X = \tau(U)$.

In the context of the rejection method, this means generating a random point (U, V) on the surface $S(t)$ under t instead of a random point (X, Y) on the surface $\mathcal{S}(f)$ under f. We can define the one-to-one correspondence $(X, Y) = \varphi(U, V)$, by $X = \varphi_1(U, V) = \tau(U)$ and $Y = \varphi_2(U, V) = V/\tau'(U)$. The Jacobian of this transformation is easily seen to be $J(u, v) \equiv 1$ (Exercise 4.11). Therefore, (X, Y) is uniformly distributed over $\mathcal{S}(f)$ if and only if (U, V) is uniformly distributed over

$$\mathcal{S}(t) = \varphi^{-1}(\mathcal{S}(f)) = \{(u, v) : 0 \le v \le f(\tau(u))\tau'(u)\}.$$

A *transformed density rejection* algorithm generates a random point in $\mathcal{S}(t)$ by the rejection method, using a hat function for the density t.

If τ^{-1} approximates F, the distribution function of X, then U has approximately the $U(0, 1)$ distribution. In that case, if $U = \tau^{-1}(X)$ takes its values only in $[0, 1]$, U can often be generated efficiently by rejection from a uniform hat function $h(u) = a$ for $0 \le u \le 1$, where a is close to 1. This approach is known as *almost-exact inversion* (Devroye 1986).

Example 4.8 (Wallace 1976, Hörmann, Leydold, and Derflinger 2004). Suppose we want to generate X from the positive part of the standard normal density, so $f(x)$ is proportional to $\tilde{f}(x) = \exp(-x^2/2)$, for $x > 0$. (This can be used to generate a normal random variate by adding a random sign.) Wallace (1976) suggests the transformation $\tau : [0, 1) \to [0, \infty)$ defined by $\tau(u) = 1.22u(1 + 0.14/(1 - u))$. We have $\tau'(u) = 1.22(1 + 0.14/(1 - u)^2)$ and the density $t(u)$ is proportional to $\tilde{t}(u) = \tilde{f}(\tau(u))\tau'(u)/1.22$, whose maximum value on the interval $(0, 1)$ is $\tilde{a} = 1.17054363007$, attained at $u = 0.73791696460$. We can then use the

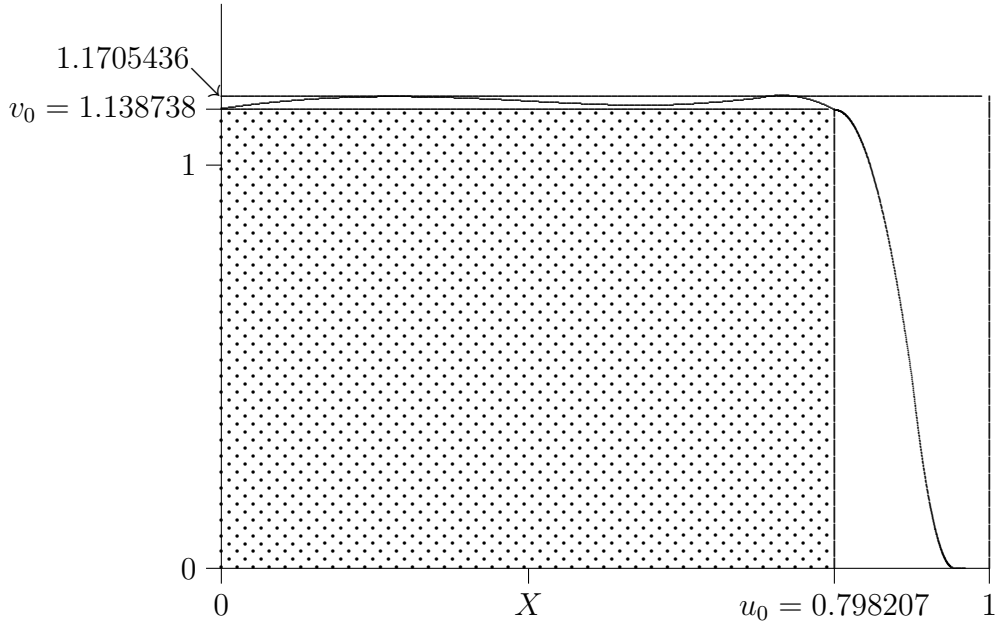


Fig. 4.3. Example 4.8: A rejection method with uniform hat function for the transformed normal density t.

constant hat function $h(u) = \tilde{a}$ to generate U by the rejection method and transform it into X. For that, we generate $U \sim U(0,1)$ and accept it with probability $\tilde{t}(U)/\tilde{a}$. We have $\int_0^1 \tilde{t}(u)du \approx 1.02730667035$. The acceptance probability is this integral divided by $\tilde{a}$, namely $1/a \approx 0.8776$.

Hörmann and Derflinger (1994) replace the constants 1.22 and 0.14 by different numbers that increase the acceptance probability to $1/a \approx 0.8904$. To speed up the algorithm, they also add a rectangular squeeze as follows: Select a constant $0 < u_0 < 1$ and let $v_0 = \min_{0 \le u \le u_0} \tilde{t}(u)/\tilde{a}$. Whenever $U \le u_0$ and $V \le v_0$, we already know that U will be accepted and there is no need to compute the expression in the "until" statement. The constant u_0 can be chosen to maximize the area below the squeeze, $u_0 v_0$. With the same numbers as above, this gives $u_0 = 0.798207256116$ and $v_0 = 1.138738153941$, and the probability of falling under the squeeze is $u_0 v_0/\tilde{a} \approx 0.7765$.

Collecting these ingredients gives the following *transformed density rejection* algorithm:

Transformed rejection to generate a positive standard normal;
repeat
  generate $U \sim U(0,1)$ and $V \sim U(0,1)$;
  let $X = 1.22\,U(1 + 0.14/(1-U))$;
  if $U \le 0.798207256116$ and $V \le 1.138738153941/1.17054363007$ return X;
until $1.17054363007\,V \le \exp(-X^2/2)(1 + 0.14/(1-u)^2)$;
return X.

□

Transformations of the form $U = \ln X$ for $X > 0$ (i.e., $\tau(U) = \exp(U)$ for $U \in \mathbb{R}$) and $U = X^{-1/2}$ for $X > 0$ (i.e, $\tau(U) = 1/U^2$ for $U > 0$) are often used to transform the density f into a *concave* density t for U. The concavity of t makes it easier to construct good hat and squeeze functions (often piecewise linear) for it. This class of transformed density rejection methods has several variants and extensions (Devroye 1986, Evans and Swartz 2000, Hörmann, Leydold, and Derflinger 2004).

♣ Other examples...

4.5.4 A generalized ratio-of-uniforms method

We consider the class of invertible transformations $(X, Y) = \varphi(U, V)$ defined by

$$\begin{aligned} X &= \varphi_1(U,V) = (U - b(V))/g'(V), \\ Y &= \varphi_2(U,V) = g(V), \end{aligned}$$

where $g : [0, \infty) \to [0, \infty)$ is a differentiable and invertible (increasing) function with $g(0) = 0$, and b is an arbitrary function. The inverse of φ is defined by $v = g^{-1}(y)$ and $u = g'(v)x - b(v)$. For any density f and constant $\kappa > 0$, this transformation defines a bijection between the sets

$$\mathcal{C} = \left\{ (u,v) \in \mathbb{R}^2 : 0 \le v \le g^{-1}\left(\kappa f\left(\frac{u - b(v)}{g'(v)}\right)\right)\right\}$$

and $S(\kappa f) = \{(x, y) \in \mathbb{R}^2 : 0 \le y \le \kappa f(x)\}$, and its Jacobian is equal to 1 (Exercise 4.11). It then follows from Proposition 4.4 that to generate X with density f, it suffices to generate a point (U, V) uniformly in $\mathcal{C}$, and return $X = \varphi_1(U, V)$.

Kinderman and Monahan (1977) have originally proposed this technique with $g'(v) = v$ and $b(V) = 0$, i.e., $X = U/V$ and $Y = V^2/2$. To generate (U, V) uniformly in $\mathcal{C}$, they generate (U, V) uniformly in a rectangular box $\bar{\mathcal{C}}$ that contains $\mathcal{C}$, by generating two independent uniforms U and V, until $(U, V) \in \mathcal{C}$. Their algorithm then returns $X = U/V$. For this reason, they call it the *ratio-of-uniforms* method. The generalization considered here is from Wakefield, Gelfand, and Smith (1991).

Rejection is the usual approach for generating (U, V) uniformly over $\mathcal{C}$. We define a region $\bar{\mathcal{C}}$ that contains $\mathcal{C}$ and in which it is easy to generate a point uniformly (for example, a rectangular box or a polygonal region). We then generate (U, V) uniformly over $\bar{\mathcal{C}}$, until it belongs to $\mathcal{C}$. If there is another region $\underline{\mathcal{C}}$ contained in $\mathcal{C}$ and for which it is very fast to check if $(U, V) \in \underline{\mathcal{C}}$, this $\underline{\mathcal{C}}$ can be used as a squeeze to accelerate the verification that $(U, V) \in \mathcal{C}$. Several special cases and refinements are described in Devroye (1986), Gentle (2003), Hörmann, Leydold, and Derflinger (2004), and other references given there.

Since $(x, y) \in S(\kappa f)$ implies that $y \le \kappa f(x)$, for each point $(u, v) \in \mathcal{C}$ we must have $0 \le v = g^{-1}(y) \le g^{-1}(\kappa f(x))$. So v can never exceed

$$v^+ = \sup_{-\infty < x < \infty} g^{-1}(\kappa f(x)).$$

Moreover, since $u = g'(v)x - b(v)$, it must lie in the interval from

$$u^- = \inf_{-\infty<x<\infty} \inf_{0\le y\le \kappa f(x)} g'(g^{-1}(y))x - b(g^{-1}(y))$$

to

$$u^+ = \sup_{-\infty<x<\infty} \sup_{0\le y\le \kappa f(x)} g'(g^{-1}(y))x - b(g^{-1}(y)).$$

When these quantities are finite, we can use the *minimal bounding rectangle* $[u^-, u^+] \times [0, v^+]$ as the bounding region $\bar{\mathcal{C}}$.

Example 4.9 As in Example 4.3, we want to generate X from the gamma distribution with parameters $(\alpha, 1)$, for $\alpha \ge 1$. Its density is proportional to

$$\kappa f(x) = x^{\alpha-1}e^{-x} \qquad \text{for } x > 0$$

and has a maximum at $x = \alpha - 1$. We use the transformation $X = U/V + \alpha - 1$ and $Y = V^2/2$. That is, $g'(V) = V$ and $b(V) = (\alpha - 1)V$.

Here $g^{-1}(\kappa f(x)) = (2\kappa f(x))^{1/2} = (2\kappa x^{\alpha-1}e^{-x})^{1/2}$ has a maximum at $x = \alpha - 1$, the mode of the density. We have

$$v^+ = \sqrt{2\kappa(\alpha-1)^{\alpha-1}e^{-\alpha+1}}.$$

We also have

$$g'(g^{-1}(y))x - b(g^{-1}(y)) = \sqrt{2y}x - (\alpha-1)\sqrt{2y} = \sqrt{2y}(x - \alpha + 1),$$

from which we find, by combining with $0 \le y \le \kappa f(x) = x^{\alpha-1}e^{-x}$, that

$$u^- = \inf_{x\ge 0} \sqrt{2x^{\alpha-1}e^{-x}}(x - \alpha + 1) < 0$$

and

$$u^+ = \sup_{x\ge 0} \sqrt{2x^{\alpha-1}e^{-x}}(x - \alpha + 1) > 0.$$

These values are finite. They can be computed numerically and the rectangular box $[u^-, u^+] \times [0, v^+]$ can be used for $\bar{\mathcal{C}}$. □

Example 4.10 .... □

4.6 Thinning a Point Process with Time-Varying Rate

Thinning is a variant of the rejection method, useful to generate events from a non-homogeneous Poisson process with complicated rate function. Suppose the process has rate $\lambda(t)$ at time t, with $\lambda(t) \le \bar{\lambda}$ for all t, where $\bar{\lambda}$ is a finite constant. One can generate Poisson *pseudo-arrivals* at constant rate $\bar{\lambda}$ by generating i.i.d. exponential interarrival times with mean $1/\bar{\lambda}$. A pseudo-arrival at time t is accepted (becomes an arrival) with probability $\lambda(t)/\bar{\lambda}$ and is rejected with probability $1 - \lambda(t)/\bar{\lambda}$. That is, for each pseudo-arrival we generate $V \sim U(0, 1)$ independent of everything that has happened before, and turn this pseudo-arrival into an arrival if and only if $V \le \lambda(t)/\bar{\lambda}$.

Non-homogeneous Poisson processes can also be generated by inversion, as we saw in Section 2.13.2, by applying a nonlinear transformation of the time scale to standardize the process (so it becomes homogeneous with rate 1 in the new time scale), generate the arrivals by inversion for the standardized process, then transform it back to find the arrival times in the original time scale. This method can be adapted to other types of point processes with time-varying rates.

4.7 Kernel Density Estimation and Generation

Instead of selecting a hard-to-invert parametric distribution and estimating its parameters, we can estimate the density via a *kernel density estimation* method for which random variate generation is very easy (Devroye 1986, Hörmann and Leydold 2000, and Section 2.9.4). In the case of a Gaussian kernel, for example, random variates can be generated simply by selecting one observation at random from the data and adding random noise generated form a normal distribution with mean zero. This method *is not* equivalent to inversion. Because of the added noise, selecting a larger observation does not necessarily guarantee a larger value for the generated variate.

4.8 Special Techniques

Besides the general methods, elegant special techniques have been designed for a number of specific distributions. We give selected examples.

Example 4.11 (A Poisson random variate with small mean.) We want $X \sim \mathsf{Poisson}(\lambda)$ where λ is small. One way to generate X is to simulate a Poisson process with rate λ over the time interval $(0, 1]$ and count the number X of arrivals. Let A_1 be the time until the first arrival and $A_2, A_3, \ldots$ the times between the successive arrivals. These A_j's are i.i.d. exponentials with mean $1/\lambda$. The number X of arrivals during $(0, 1]$ satisfies

$$\sum_{j=1}^{X} A_j \le 1 < \sum_{j=1}^{X+1} A_j.$$

If each A_j is generated by inversion from $1 - U_j \sim U(0, 1)$, we have $A_j = -(\ln U_j)/\lambda$ and this inequality can be written as

$$-\sum_{i=1}^{X} \ln(U_i) \le \lambda < -\sum_{i=1}^{X+1} \ln(U_i),$$

i.e.,

$$\prod_{i=1}^{X} U_i \ge e^{-\lambda} > \prod_{i=1}^{X+1} U_i,$$

or

$$X = \max\left\{k \text{ such that } e^{\lambda}\prod_{i=1}^{k} U_i \geq 1\right\}.$$

This gives:

A Poisson(λ) generator;
let $X = 0$ and $p = e^{\lambda}$;
repeat
  generate $U \sim U(0,1)$;
  let $p = pU$ and $X = X + 1$
until $p < 1$;
return $X - 1$.

This simple algorithm requires $\mathbb{E}[X + 1] = \lambda$ iterations on average. It is fine for small λ but highly inefficient when λ is large. □

Example 4.12 (Random points on the surface of a d-dimensional sphere.) Generating a random point on the surface of a hypersphere centered at zero is equivalent to generating a random direction from the origin, in d dimensions. This can be achieved by generating a random point from a *radially symmetric* density in d dimensions, which is defined as a density that is constant on the surface of any sphere centered at the origin (i.e., that can be written as a function of the distance to the origin, or the sum of squares of coordinates). The standard multinormal density is the most notorious example. To generate a random point $\boldsymbol{X}$ on the d-dimensional sphere of radius r, generate a vector $\boldsymbol{Z}$ from any radially symmetric density, and normalize its length to r by defining $\boldsymbol{X} = r\boldsymbol{Z}/\|\boldsymbol{Z}\|_2$, where $\|\boldsymbol{Z}\|_2 = (Z_1^2 + \cdots + Z_d^2)^{1/2}$ is the Euclidean norm of $\boldsymbol{Z}$. The most common way of generating $\boldsymbol{Z}$ is from the standard normal distribution; in that case, $Z_1, \ldots, Z_d$ are i.i.d. $N(0,1)$ random variates.

In two dimensions, one alternative is to generate a random angle Θ uniformly over the interval $(0, 2\pi)$, and return $(X_1, X_2) = (r\cos\Theta,\ r\sin\Theta)$, which is the point at angle Θ on the circle. Note that this could be slower than the previous method, because the trigonometric functions can be more expensive to evaluate than generating the standard normals.

On the three-dimensional sphere of radius r centered at zero, it turns out (somewhat surprisingly) that any of the three coordinates of a random point on the sphere is uniformly distributed over $(-r, r)$. So we can first generate, say, the third coordinate X_3 uniformly over this interval. Suppose X_3 takes the value x_3. Then it remains to generate a random point uniformly over the circle defined by the intersection of the surface with the place $X_3 = x_3$. This circle has radius $\tilde{r} = r\cos(\arcsin x_3)$, so we can generate an angle Θ as in the two-dimensional case, and put $(X_1, X_2) = (\tilde{r}\cos\Theta,\ \tilde{r}\sin\Theta)$. □

Example 4.13 (Random points in a three-dimensional sphere.)

□

Automatic methods. Recently, there has been an effort in developing *automatic* or *black box* algorithms for generating variates from an arbitrary (known) density, and reliable software that implements these methods (Hörmann and Leydold 2000, Hörmann, Leydold, and Derflinger 2004, Leydold and Hörmann 2002, Leydold, Janka, and Hörmann 2002).

The most extensive coverage of non-uniform variate generation remains the book of Devroye (1986).

4.9 Markov Chain Monte Carlo

♣ To be done...

4.10 Exercises

4.1 Explain how to generate a random variate X from each of the following distributions, *by inversion*. Then, implement it, generate 1000 variates, compute their mean and variance, and check if the values make sense.

(a) The triangular distribution with parameters (a, b, m), whose density is positive only over the interval (a, b), attains its maximum at $x = m$, and is linear over the intervals $[a, m]$ and $[m, b]$.

(b) The Pareto distribution with parameters (α, β), whose distribution function is $F(x) = 1 - (\beta/x)^\alpha$ for $x \geq \beta$.

(c) The Cauchy distribution, for which $F(x) = 1/2 + \arctan(x)/\pi$ for $x \in \mathbb{R}$.

(d) The logistic distribution, for which $F(x) = 1/(1 + e^{-x})$ for $x \in \mathbb{R}$.

(e) The Rayleigh distribution with parameter β, for which $F(x) = 1 - \exp[-x^2/\beta]$ for $x > 0$.

4.2 Suppose we want to generate a random variate X with the same distribution as $\max(Y_1, \ldots, Y_n)$, where the Y_j are i.i.d. with distribution function F. Show that X can be generated directly via $X = F^{-1}(U^{1/n})$ where $U \sim U(0, 1)$.

4.3 Implement a fast algorithm to compute the inverse of the Poisson distribution function with parameter λ, using indexed search with an index of size $c = \lambda$. There should be an initialization procedure that constructs the tables and a procedure that returns $F^{-1}(U)$ for any given $U \in (0, 1)$. In an object-oriented programming environment, a Poisson generator of this kind would normally be implemented as an object, with a method for computing F^{-1} and a different method (that calls the first method) to generate random variates (because sometimes we only want to compute F^{-1} and not generate a random variate). The tables would be computed by the constructor. It is recommended that you implement it that way. Then, use your algorithm to generate 1000 i.i.d. Poisson random variates with mean $\lambda = 20$. Compute and print their mean and variance and check if these values make sense.

4.4 Suppose you have functions that return good approximations of the gamma and beta inverse distribution functions. Show how to use these functions to generate random variates from the Pearson type 5 and type 6 distributions by inversion.

4.5 Recall that if $X \sim \mathsf{Erlang}(k, \lambda)$ and $Y \sim \mathsf{Poisson}(\lambda x)$, then $\mathbb{P}[X \leq x] = \mathbb{P}[Y \geq k]$ (Exercise 2.12). Explain how this can be used to generate a Poisson(λ) random variate with

large λ, by inversion, if a good approximation of the inverse gamma distribution function is available.

4.6 A simple way of generating a random point in a d-dimensional sphere of radius 1 is by a rejection method as follows. Define a d-dimensional cube of side 2 that contains the sphere, and generate random points in the cube until one falls in the sphere. What is the expected number of trials with this method, as a function of the volume V_d of the d-dimensional unit sphere? Compute this expected number for $d = 2$, 5, 10, and 20. Discuss the viability of the method as a function of d. Hint: see Exercise 1.19.

4.7 (a) In Example 4.2, explain in detail how to generate from the sampling density g that corresponds to the hat function h of Figure 4.2, by inversion.

(b) Implement the algorithm for each of the two function h, in Figures 4.1 and 4.2, and compare their average execution times. Explain how you have made the comparison and how you have tested the correctness of your two implementations.

(c) Design a version of the method where the hat function is piecewise-constant with five pieces ($c = 5$) instead of three. Optimize the four breakpoints and compute the corresponding value of a.

4.8 In Example 4.3, the function was taken proportional to an exponential density of parameter θ, with $\theta = 1/\alpha$. But we could use a different value of θ. What is the minimal value of a as a function of θ, and what value of θ minimizes this a?

4.9 Suppose we use a continuous sampling distribution g to generate an *integer-valued discrete* random variate X by the rejection method, as explained near the end of Section 4.4.1. We want X to have probability mass function f, i.e., $\mathbb{P}[X = x] = f(x)$ for all integers $x \in \mathbb{Z}$. Let h be a hat function such that $h(x) = ag(x) \geq \kappa f(x)$ for all x, for some constant $\kappa > 0$.

(a) Prove that with the acceptance condition $Vh(X) \leq \kappa f(\lfloor X \rfloor)$, the returned value $\lfloor X \rfloor$ has the correct discrete distribution.

(b) Prove that this is also true with the acceptance condition $Vh(\lfloor X \rfloor) \leq \kappa f(\lfloor X \rfloor)$, if $(1/h(k)) \int_k^{k+1} h(x)dx$ has the same value for all integers k for which $f(k) > 0$ (this is the case, for instance, if h is constant between any two integers). Give a counterexample showing that if this last condition is not satisfied, the method with this second acceptance condition can be invalid.

4.10 Detail a rejection algorithm that uses a hat function proportional to a Cauchy density to generate standard normal random variates. Find the appropriate constant a.

4.11 Prove that in Section 4.5.3 and Section 4.5.4, the Jacobian of the transformation φ is equal to 1.

4.12 Detail an algorithm to generate a random point uniformly distributed inside a d-dimensional ellipsoid centered at the origin, defined by $\boldsymbol{x}^{\mathrm{t}}\boldsymbol{A}\boldsymbol{x} \leq 1$ where $\boldsymbol{A}$ is a $d \times d$ positive definite matrix.

4.13 (a) For the polar method described at the end of Example 4.7, show that if $U_2 = D^2$ is uniformly distributed over $(0, 1)$, then the point (V_1, V_2) is uniformly distributed in the disk of radius 1 centered at the origin. Use this to prove that the pair (X, Y) returned at the end is a pair of independent standard normal random variables.

(b) Implement the Box-Muller and the polar method and compare their speed by generating a few million standard normals with each of them.

♣ Simulation of Archimedean copulas in $d \geq 2$ dimensions. E.g., Gumbel copula.

♣ Truncated binormal random vector. First show that marginal is not truncated normal.

5. Output Analysis

5.1 Quality and Precision of Statistical Estimators

We estimate an unknown quantity μ by an estimator X, whose quality can be assessed in different ways. In the introduction, we adopted the *efficiency* $\text{Eff}[X]$ as a quality measure, assuming that the estimation error can be measured by the mean square error $\text{MSE}[X]$. However, *any quality measure for an estimator X is imperfect*, and $\text{Eff}[X]$ is no exception. For example, the MSE gives the same weight to negative error (underestimation) than to positive error (overestimation). It may well be that one of these two types of error is more damaging than the other. The damage is also not necessarily proportional to the square of the error (as measured by the MSE).

Another important aspect of the quality of an estimator, not measured by $\text{Eff}[X]$, is the availability of a good way to assess the estimation error. For example, if our error assessment is based on the variance of X, we need a good variance estimator. In this situation, given two unbiased estimators X and Y with similar computing costs and variances, if S_X^2 and S_Y^2 are the available (unbiased) estimators for the variances of X and Y and if $\text{Var}[S_X^2] \ll \text{Var}[S_Y^2]$, we would probably prefer X to Y even though $\text{Eff}[X] \approx \text{Eff}[Y]$. (See Exercise 1.16.)

Reporting a confidence interval is a standard way to provide an estimate of an unknown quantity, together with an assessment of the estimation error. For $0 \leq \alpha \leq 1$, a *confidence interval* with *confidence level* $1-\alpha$ (also called $100(1-\alpha)\%$ confidence interval) for an unknown quantity μ is a random interval $[I_1, I_2]$ such that $\mathbb{P}[I_1 \leq \mu \leq I_2] = 1-\alpha$. (The random variables in this statement are I_1 and I_2.) Confidence intervals are constructed for a *nominal or target confidence level* $1-\alpha$ under certain assumptions (e.g., normality), but the true value of $\mathbb{P}[I_1 \leq \mu \leq I_2]$, called the *coverage probability*, is often different and unknown, due to departure from the assumptions or because the coverage may depend on the (unknown) exact value of μ. The difference $\mathbb{P}[I_1 \leq \mu \leq I_2] - (1-\alpha)$ is called the *coverage error*. An ideal confidence interval has large coverage probability and small width. But since a smaller width generally implies a smaller coverage probability, a compromise must be made and the quality of a confidence interval must be judged based on these two conflicting criteria. For two confidence intervals having the same coverage probability one would normally prefer the one with the *smallest expected width*. Another relevant quality measure is the *variance of the width*.

Often, the word "estimator" refers to a *sequence of estimators* $\{Y_n, n \geq 1\}$, and we refer to this sequence by Y_n (a slight abuse of notation). Two examples of this are $\bar{X}_n$ and S_n^2, introduced in Section 1.4. If we want to estimate μ, then Y_n is said to be *asymptotically unbiased* if $\mathbb{E}[Y_n - \mu] \to 0$ as $n \to \infty$, *consistent* if $Y_n \to \mu$ in probability as $n \to \infty$, and

strongly consistent if $Y_n \to \mu$ with probability one as $n \to \infty$. Consistency is generally a *required* property of estimators. One has, for example, that $\bar{X}_n$ is unbiased and strongly consistent for $\mathbb{E}[X_i]$ whereas S_n^2 is unbiased and consistent for $\mathrm{Var}[X_i]$. A confidence interval $(I_{n,1}, I_{n,2})$ is *asymptotically valid* if its coverage probability converges to the desired coverage, $1-\alpha$, when $n \to \infty$.

Most of the techniques explained in this section apply to estimators obtained via simulation or by other means. An important difference between simulation data and data obtained from real-life systems is that it is generally much easier to increase the sample size in a simulation setting than in the real world. For example, for statistical studies looking at the effect of different drugs on a given human disease, sample sizes are limited. For simulation, we are limited only by the computing time we are ready to spend. We also (normally) have much better control over simulation experiments than over experiments with real-life systems. As a result, there is some difference between the appropriate statistical output analysis tools for simulation and the most popular techniques in classical statistics. For example, the computation of confidence intervals based on regenerative analysis, and the question of choosing between one long run and several smaller runs to estimate an infinite-horizon average, are important in the simulation context but are usually not considered in (classical) statistical data analysis textbooks.

Simulation can be used to estimate expectations over a finite horizon, steady-state averages over an infinite horizon, expected total discounted costs over an infinite horizon, quantiles of a distribution, linear or nonlinear functions of one or several expectations, roots of functions, solutions of optimization problems, and there are many other possibilities. Techniques for constructing estimators of these quantities, and for assessing the error of these estimators (e.g., by constructing confidence intervals), are discussed in this chapter. We also examine related questions such as how to compute confidence intervals for multivariate performance measures (i.e., for vectors), and methods for comparing two or more systems.

It is important to emphasize that the error estimates here take into account only the simulation error, and not the modeling error, and the error made in estimating the model parameters used for the simulation. The simulation error can often be made much smaller then the other two. In that case, the confidence intervals computed by considering only the simulation error can give an overly optimistic view. One must be very careful about their interpretation.

♣ Expand this last discussion a bit. Add a section on CI methods that take into account the error in parameter estimation (e.g., via bootstrap techniques).

5.2 Estimation of a finite-horizon expectation

If the quantity μ of interest is the expectation of some random variable X that can be computed over a finite horizon (e.g., as in Eq. (2.74)), one can obtain a sample of n i.i.d. copies of X, say, $X_1, \ldots, X_n$, by performing n independent simulation runs. Obvious estimators of $\mu = \mathbb{E}[X]$ and of $\sigma^2 = \mathrm{var}[X]$ are the sample mean $\bar{X}_n$ and the sample variance S_n^2, respectively. We are back to the setup of Section 1.4.3. The classical way of computing a confidence interval in that case is to assume either that X has the normal distribution, or that n is large enough so that $\bar{X}_n$ is approximately normal because of the CLT.

5.2.1 Small samples, normal observations

We recall a standard result in elementary statistics. Part (iii) follows directly from (i–ii) and the definition of the Student distribution.

Theorem 5.1 *(e.g., Hogg and Craig 1995). If* $X_1, \ldots, X_n$ *are* i.i.d. $N(\mu, \sigma^2)$*, then*
(i) $\bar{X}_n$ *and* S_n^2 *are independent;*
(ii) $(n-1)S_n^2/\sigma^2$ *has the chi-square distribution with* $n-1$ *degrees of freedom;*
(iii) $\sqrt{n}(\bar{X}_n - \mu)/S_n$ *has the Student-t distribution with* $n-1$ *degrees of freedom.*

Part (iii) of this theorem can be used to compute a confidence interval for μ, as explained in Section 1.4.3. An interval with coverage probability $1-\alpha$ is given by $(\bar{X}_n \pm t_{n-1,1-\alpha/2} S_n/\sqrt{n})$, where $t_{n-1,1-\alpha/2} = F^{-1}(1-\alpha/2)$ and F denotes the Student$(n-1)$ distribution function. That is, $t_{n-1,1-\alpha/2}$ is the unique real number x for which the probability that a Student$(n-1)$ random variable exceeds x is $\alpha/2$, The quantity $t_{n-1,1-\alpha/2} S_n/\sqrt{n}$ is called the *half-width* of the interval.

For large n (e.g., $n \geq 30$ or so), the Student distribution is approximately identical to the standard normal, i.e., $t_{n-1,1-\alpha/2} \approx z_{1-\alpha/2} = \Phi^{-1}(1-\alpha/2)$, where Φ is the standard normal distribution function. One has $t_{n-1,1-\alpha/2} > z_{1-\alpha/2}$ and the smaller is n, the larger is the difference (so the confidence intervals are wider than with the normal approximation).

♣ Make figures.

5.2.2 Large samples, central-limit effects

If n is large, $\bar{X}_n$ may be approximately normally distributed even if X is not, because of the CLT. A confidence interval can then be computed as in Section 1.4.3.

One must be careful however: The distribution of $\bar{X}_n$ can be far from normal, e.g., if n is small, or if α is close to 0, or if the distribution of the X_i's is highly asymmetric, or if there are outliers at more than 3 or 4 standard deviations away from the mean. In any case, one must measure how the distribution of the X_i's departs from the normal distribution. The test of normality of Shapiro, Wilk, and Chen (1968) is often recommended.

♣ Berry-Essen bound: See Theorem A.16 in the appendix.

Example 5.1 Example 1.24 in Section 1.4.3 gave an illustration where the normal approximation appeared reasonable. We had $n = 1000$ i.i.d. Bernoulli(p) random variables $X_1, \ldots, X_n$ and wanted to estimate p by $\bar{X}_n$. In that example, we had $p \approx 0.88$. Suppose now that p is very close to 1 and that we have obtained 998 successes, i.e., $\bar{X}_n = 0.998$. In this case, the normal approximation to the binomial is bad; one should compute a confidence interval using the binomial probabilities directly or via the Poisson approximation for the number of failures; i.e., $\sum_{i=1}^{n}(1-X_i) \approx$ Poisson$(n(1-p))$ (see Section 5.2.3). □

♣ Add examples and exercises on this.

[1] Nelson (1992) suggests that when the X_i's are highly non-normal, a good idea is to batch the observations to improve normality. Regroup the observations into (say) 30 batches of equal sizes, take the average within each batch, and compute a confidence interval by considering these 30 averages, or *batch means*, as the observations. When n is large, Nelson (1992) advocates using batch means anyway, whatever be the distribution of the X_i's, and then test the normality of the batch means. This does not change the mean $\bar{X}_n$, neither the expectation of its variance estimator S_n^2/n, which equals $\text{Var}[X_i]/n$ with or without batching. However, batching changes the *distribution* (and generally increases the variability) of the variance estimator and of the confidence interval half-width. Exercise 5.4 illustrates this.

An alternative (and often better) approach for computing confidence intervals in the case of highly non-normal observations is the bootstrap (Section 5.9).

5.2.3 Confidence Intervals for Discrete distributions

In Example 5.1, suppose this time that p is very close to 0. We want to compute a confidence interval on $\mu = np$ from a single observation of $Y = X_1 + \cdots + X_n$, knowing that $Y \sim \text{Binomial}(n, p)$. For example, if $n = 1000$ and $Y = 2$, what is an appropriate confidence interval on p for a given α? The following general methodology allows us to compute such an interval.

Let Y be an integer-valued random variable with (unknown) continuous parameter θ, such that $P_\theta[Y \geq y]$ is a monotone function of θ for any y, where P_θ denotes the probability when the parameter value is θ. Binomial, geometric, and Poisson random variables, for example, are of this type. Suppose we want to compute a confidence interval $[I_1, I_2]$ with confidence level (approximately) $1-\alpha$ for θ. We can define the confidence interval as follows. Decompose α as $\alpha = \alpha_1 + \alpha_2$ where $\alpha_1 > 0$ and $\alpha_2 > 0$ (for example, $\alpha_1 = \alpha_2 = \alpha/2$). Here, we assume that $P_\theta[Y \geq y]$ increases with θ; for the decreasing case, just permute $\leq$ and $\geq$ in (5.1). Compute $I_1 = I_1(y)$ and $I_2 = I_2(y)$ as the solutions of the equations

$$\alpha_1 = P_{I_1}[Y \geq y] \qquad \text{and} \qquad \alpha_2 = P_{I_2}[Y \leq y], \tag{5.1}$$

where y is the observed value of Y. These equations can be solved via binary search, assuming that the probabilities in (5.1) are easy to compute when θ and y and given.

Proposition 5.2 *The coverage probability provided by the procedure just described is at least* $1 - \alpha$ *for any* θ.

Proof. Let $y^*(\theta) \stackrel{\text{def}}{=} \min\{y \in \mathbb{N} : I_1(y) \geq \theta\}$ (an integer) and $\nu \stackrel{\text{def}}{=} I_1(y^*(\theta)) \geq \theta$. One has $I_1(y) \geq \theta$ if and only if $y \geq y^*(\theta)$. The probability that the interval falls completely to the right of θ is

$$P_\theta[I_1(Y) \geq \theta] \leq P_\nu[I_1(Y) \geq \theta] = P_\nu[Y \geq y^*(\theta)] = P_{I_1(Y^*(\theta))}[Y \geq y^*(\theta)] = \alpha_1.$$

One can show that $P_\theta[I_2(Y) \leq \theta] \leq \alpha_2$ in a similar way. Then, $P_\theta[\theta \in (I_1(Y), I_2(Y))] \geq 1 - \alpha_1 - \alpha_2 = 1 - \alpha$.

[1] From Pierre: Move this to the exercises.

The *exact coverage probability* of such a confidence interval is difficult to compute because it generally depends on F_θ (and therefore on the true value of θ). For specific distributions, the exact coverage probability as a function of θ could be estimated by simulation and tabulated.

Example 5.2 As an important special case, suppose that $X_1, \ldots, X_n$ are i.i.d. with $\mathbb{P}[X_i = 1] = 1 - \mathbb{P}[X_i = 0] = p$, so that $Y = n\bar{X}_n = \sum_{i=1}^n X_i$ is $\mathsf{Binomial}(n, p)$, and that we want to compute a confidence interval on p based on the observation of Y. For any given values of $\theta \equiv p$ and of y, the probabilities in (5.1) can be computed by summing the exact binomial probabilities if y is small. If n is large and p is small, Y is approximately a Poisson random variable with mean np, so one can approximate the probabilities in (5.1) by summing the appropriate Poisson probabilities. For p close to 1, we can simply replace p and X_i by $1 - p$ and $1 - X_i$. □

In the binomial and Poisson cases, we can avoid summing probabilities by exploiting the relationship between the binomial and beta cdf's, and between the Poisson and gamma cdf's (see Sections 2.8.13 and 2.8.12). If $Y \sim \mathsf{Binomial}(n, p)$, then

$$\mathbb{P}[Y \geq y] = F_{\beta,y}(p) \tag{5.2}$$

where $F_{\beta,y}$ is the cdf of a beta random variable with parameters $(y, n - y + 1)$. To satisfy (5.1), we must find I_1 and I_2 such that $\alpha_1 = F_{\beta,y}(I_1)$ and $\alpha_2 = 1 - F_{\beta,y+1}(I_2)$. These values are given directly by $I_1 = F^{-1}_{\beta,y}(\alpha_1)$ and $I_2 = F^{-1}_{\beta,y+1}(1 - \alpha_2)$. They can be computed via a good approximation of the inverse beta distribution function. Likewise, if $Y \sim \mathsf{Poisson}(\lambda)$, then $\mathbb{P}[Y \geq y] = F_{\gamma,y}(\lambda) = \mathbb{P}[X \leq \lambda]$ where $X \sim \mathsf{Gamma}(y, 1)$, so the same method can be used with $F_{\beta,y}(p)$ replaced by $F_{\gamma,y}(\lambda)$.

5.2.4 Distribution-free confidence intervals for the mean *

Suppose a random variable X is bounded with probability 1, i.e., $\mathbb{P}[A \leq X \leq B] = 1$ for some constants $A < B$, and we want to compute a confidence interval on $\mu = \mathbb{E}[X]$ without making any further assumption about the distribution of X. This can be achieved via *Hoeffding inequalities*, given in the next theorem.

To simplify the notation, we standardize the random variable so that $A = 0$ and $B = 1$. That is, we consider $Y = (X - A)/(B - A)$. Let $\nu = \mathbb{E}[Y] = (\mu - A)/(B - A)$. A confidence interval on ν readily gives a confidence interval on μ.

Theorem 5.3 *(Hoeffding 1963). Let* $Y_1, \ldots, Y_n$ *be* i.i.d. *random variables such that* $\mathbb{P}[0 \leq Y_i \leq 1] = 1$ *and* $\mathbb{E}[Y_i] = \nu$. *Then,*

$$\begin{aligned} \mathbb{P}[\bar{Y}_n - \nu \geq \epsilon] &\leq e^{-ng(\nu)\epsilon^2} \leq e^{-2n\epsilon^2} \qquad \text{and} \\ \mathbb{P}[\nu - \bar{Y}_n \geq \epsilon] &\leq e^{-ng(1-\nu)\epsilon^2} \leq e^{-2n\epsilon^2}, \end{aligned}$$

where

$$g(\nu) = \begin{cases} \dfrac{1}{1-2\nu}\ln((1-\nu)/\nu) & \text{for } 0 < \nu < 1/2, \\ \dfrac{1}{2\nu(1-\nu)} & \text{for } 1/2 \le \nu < 1. \end{cases}$$

Note that $g(\nu)$ is continuous, decreasing in $(0, 1/2)$, increasing in $(1/2, 1)$, and its minimum is $g(1/2) = 2$.

A simple two-sided $100(1-\alpha)\%$ confidence interval on μ obtained from these inequalities is $(\bar{Y}_n - \epsilon,\ \bar{Y}_n + \epsilon)$ where $\epsilon^2 = -\ln(\alpha/2)/(2n)$. Indeed, one then has

$$\mathbb{P}[|\bar{Y}_n - \nu| \le \epsilon] \ge 1 - 2e^{-2n\epsilon^2} = 1 - \alpha.$$

The bounds involving $g(\nu)$ and $g(1-\nu)$ can be used with profit to compute a tighter confidence interval when ν is known to lie in an interval that does not contain $1/2$. For example, suppose it is known that $\nu < \nu_1$ for some known constant $\nu_1 < 1/2$, or that $\nu > \nu_1$ for some $\nu_1 > 1/2$. By taking $\epsilon^2 = -\ln(\alpha/2)/g^* n$ where

$$g^* = \min_{\nu \le \nu_1} \min(g(\nu),\ g(1-\nu)) = \min(g(\nu_1),\ g(1-\nu_1)),$$

we obtain that

$$\mathbb{P}[|\bar{Y}_n - \nu| \le \epsilon] \ge 1 - e^{-ng(\nu_1)\epsilon^2} - e^{-ng(1-\nu_1)\epsilon^2} \ge 1 - \alpha/2 - \alpha/2 = 1 - \alpha.$$

Fishman (1996), Theorem 2.4, provides a refined version of this type of confidence interval. Hoeffding (1963) gives variants of his inequality for *dependent* random variables. Several other inequalities of a similar flavor (for example, Chebyshev, Bernstein, and Bennett inequalities) can also be used to obtain distribution-free confidence intervals; see, e.g., Hoeffding (1963) and Devroye, Györfi, and Lugosi (1996). Chebyshev inequality typically yields wider (more conservative) confidence intervals than those based on Hoeffding inequality.

5.2.5 Fixed sample size vs sequential estimation

So far we saw how to compute a confidence interval for a given confidence level $1 - \alpha$ and a fixed sample size n. Then, the width $I_2 - I_1$ of the interval is a random variable and the interval may turn out to be much wider than desired. Alternatively, we may decide to fix α and a maximal acceptable width w, hoping that the required sample size N (which is now random) to obtain $I_2 - I_1 \le w$ will be reasonable. How can we predict the required value of N? We can estimate it from pilot runs or we can use sequential estimation.

Example 5.3 In Example 5.1, we obtained $S_n^2 \approx 0.1042$ and a 95% confidence interval of half-width (around) 0.020 with $n = 1000$ simulation runs. How many *additional* runs should we perform if we want to reduce the half-width to 0.005? We seek a value of n such that $1.96 S_n/\sqrt{n} \le 0.005$. Assuming that S_n^2 will not change much (it converges to $\mathrm{Var}[X]$ when $n \to \infty$), this gives $n \ge (1.96 \times S_n/0.005)^2 \approx 0.1042 \times (1.96 \times 200)^2 \approx 16011.8$. So our prediction is that we need 15012 additional simulation runs. □

For a confidence interval based on the normal distribution when the variance σ^2 is known, the minimal n that gives a *half-width less than $w/2$* for a $100(1-\alpha)\%$ confidence interval is

$$n^* = \min\left\{n > 1 : z_{1-\alpha/2}\sigma/\sqrt{n} \le w/2\right\}. \tag{5.3}$$

If σ^2 is unknown, we can replace it in (5.3) by our current best variance estimator. Under the assumption that the X_i are normally distributed, we may use the Student distribution instead of the normal, and this gives the following heuristic to approximate n^*: Select some integer n_0, perform n_0 simulation runs, compute the sample variance $S^2_{n_0}$, and compute

$$\hat{N}^* = \min\left\{n \ge n_0 : t_{n-1,1-\alpha/2}S_{n_0}/\sqrt{n} \le w/2\right\}. \tag{5.4}$$

This $\hat{N}^*$ is our prediction of the total sample size needed. The rationale is that $S^2_{n_0}$ is our best guess of S^2_n.

A *two-stage procedure* based on this heuristic computes $S^2_{n_0}$ for a moderate value of n_0 in the first stage, computes $\hat{N}^*$ in (5.4), and then performs $\hat{N}^* - n_0$ additional simulation runs in the second stage. Finally, it computes a confidence interval based on all $\hat{N}^*$ runs. This procedure applies *mutatis mutandis* to other distributions than the Student. Of course, since S^2_n is a random variable whose value may change when n increases, the $\hat{N}^*$ estimated from the first stage may turn out to be too small or too large.

A different approach is *sequential estimation*: Compute the half-width of the confidence interval after every run as soon as the sample size n exceeds some predetermined threshold n_0 (for example, n_0 could be 1/4 of a rough initial guess of n^*). Stop when the half-width is less than the desired value.

Note that this procedure is more likely to stop when S^2_n is smaller than usual than to stop when S^2_n is larger than usual. Let N denote the sample size at which we stop (a random variable). If S^2_n is positively [negatively] correlated with $\bar{X}_n$ for each n, $\bar{X}_N$ will tend to be smaller [larger] than $\mu = \mathbb{E}[X_i]$. Therefore, sequential estimation is generally *biased* for the mean. If the X_i's have the normal distribution, then $\bar{X}_n$ and S^2_n are independent (see Theorem 5.1) and the procedure is unbiased for the mean, but S^2_N is still a biased estimator of the variance. Sequential estimation may thus provide biased estimators and confidence intervals in general, and the bias for the mean is likely to be worse when the distribution of the X_i's is far from the normal (e.g., highly skewed).

A quite interesting result, on the other hand, is that when $w \to 0$ and α is fixed, all this bias goes away; everything becomes essentially the same as if we knew the variance σ^2 exactly beforehand and were using the exact minimal sample size n^* with the normal distribution. The next theorem states this *asymptotic consistency* property. For the result to hold even if $\mathbb{P}[X_1 = \cdots = X_n] > 0$ for small n, it is convenient to add $1/n$ to the variance estimator S^2_n. Otherwise, the method could stop prematurely with $S^2_n = 0$. Note that $n^* \to \infty$ when $w \to 0$.

Theorem 5.4 *(Chow and Robbins 1965) Suppose we use a sequential estimation procedure with initial sample size n_0 and stop at*

$$N = \min\left\{n \ge n_0 : z_{1-\alpha/2}\sqrt{(S^2_n + 1)/n} \le w/2\right\}. \tag{5.5}$$

If $\alpha > 0$ *is fixed and* $w \to 0$*, then we have* $N/n^* \overset{\text{w.p.1}}{\to} 1$*,* $\mathbb{E}[N]/n^* \to 1$*, and* $\mathbb{P}[|\bar{X}_N - \mu| \leq w/2] \to 1 - \alpha$*.*

This result implies that sequential estimation can be (approximately) reliable when the target half-width $w/2$ is small enough. Based on extensive empirical experiments made by Law, Kelton, and Koenig (1981), Law and Kelton (2000) argue that the procedure is usually safe when the *relative half-width* $w/|2\mu|$ does not exceed 0.15 (as a rule of thumb) and recommend that this condition should always be respected.

There is a similar version of the theorem for the case of a target relative half-width $w'/2$, when the mean μ is simply estimated by the sample mean $\bar{X}_n$. In that case, we want the interval width divided by $|\mu|$ to be no more than w'. The procedure stops at

$$N = \min\left\{n \geq n_0 : z_{1-\alpha/2}\sqrt{(S_n^2 + 1)/n} \leq \bar{X}_n w/2\right\}. \tag{5.6}$$

Other theorems give more refined results where bounds on the coverage error are also provided for certain special cases (e.g., if the X_i's are i.i.d. normal). An extensive coverage of sequential estimation procedures can be found in Ghosh, Mukhopadhyay, and Sen (2001).

Sequential estimation implies more overhead than the two-stage procedure, because the width of the confidence interval must be updated at each step. On the other hand, the coverage is always less than the target and the half-width is typically less variable. To reduce the overhead, one can update the confidence interval only at every k steps, i.e., at steps n_0, $n_0 + k$, $n_0 + 2k$, and so on, for some integer $k > 0$.

♣ Give a robust updating algorithm for the variance. See Chan, Golub, and LeVeque (1983).

5.3 Confidence regions for vectors

Charnes (1991, 1995) provides an overview of this topic. Morrison (1990) is an excellent reference on multivariate confidence intervals. Suppose that we want to estimate a vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_d)$ of d real numbers (simultaneously), and that we can compute individual confidence intervals $\mathcal{I}_1, \ldots, \mathcal{I}_d$ for these μ_j, with confidence level $1 - \alpha_j$, as usual. For each j, we would have $\mathbb{P}[\mu_j \in \mathcal{I}_j] = 1 - \alpha_j$. Then, the probability that all intervals cover simultaneously their parameters, $1 - \alpha' = \mathbb{P}[\mu_j \in \mathcal{I}_j \text{ for each } j]$, is generally very difficult to compute when the $\mathcal{I}_j$ are correlated.

What we want is a confidence region $\mathcal{I} \subset \mathbb{R}^d$ for the vector $\boldsymbol{\mu}$, with coverage probability at least $1 - \alpha$, i.e., $\mathbb{P}[\text{the random region } \mathcal{I} \text{ falls over } \boldsymbol{\mu}] \geq 1 - \alpha$, for some fixed value of α, and such that the volume of $\mathcal{I}$ is not too large.

5.3.1 Bonferroni Inequality

The *Bonferroni inequality* (5.7) provides a simple but conservative lower bound on the coverage probability of a confidence box defined as the product of univariate confidence intervals. Suppose that for each j, we have a confidence interval $\mathcal{I}_j$ with confidence level *at least* $1 - \alpha_j$

for μ_j. Define the rectangular box $\mathcal{I} = \{(\mu_1, \ldots, \mu_d) \mid \mu_1 \in \mathcal{I}_1, \ldots, \mu_d \in \mathcal{I}_d\}$, i.e., the cartesian product of $\mathcal{I}_1, \ldots, \mathcal{I}_d$, and let $\alpha = \sum_{j=1}^{d} \alpha_j < 1$. Then

$$\begin{aligned} \mathbb{P}[\boldsymbol{\mu} \in \mathcal{I}] &= \mathbb{P}[\mu_j \in \mathcal{I}_j \text{ for each } j] \\ &= 1 - \mathbb{P}[\mu_j \notin \mathcal{I}_j \text{ for some } j] \\ &\geq 1 - \sum_{j=1}^{d} \mathbb{P}[\mu_j \notin \mathcal{I}_j] \\ &\geq 1 - \sum_{j=1}^{d} \alpha_j \;=\; 1 - \alpha. \end{aligned} \tag{5.7}$$

So $\mathcal{I}$ is a confidence region with confidence level at least $1 - \alpha$. Several authors recommend this procedure because it is simple. However, it could be overly conservative for large d.

5.3.2 Confidence ellipsoids

Suppose that $\boldsymbol{X} = (X_1, \ldots, X_d)^{\mathsf{t}}$ is multinormal and that $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{X}]$. We estimate $\boldsymbol{\mu}$ by

$$\bar{\boldsymbol{X}}_n = (\bar{X}_{n1}, \ldots, \bar{X}_{nd})^{\mathsf{t}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{X}_i,$$

where $\boldsymbol{X}_i$ is the ith replicate of $\boldsymbol{X}$. We also estimate $\boldsymbol{\Sigma}$, the covariance matrix of $\boldsymbol{X}$, by

$$\hat{\boldsymbol{\Sigma}}_n = \frac{1}{n-1} \sum_{i=1}^{n} (\boldsymbol{X}_i - \bar{\boldsymbol{X}}_n)(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_n)^{\mathsf{t}}.$$

The element $\hat{\sigma}_{jk}^2$ of $\hat{\boldsymbol{\Sigma}}_n$ is the estimated covariance between X_j and X_k and $\hat{\rho}_{jk} = \hat{\sigma}_{jk}^2 / (\hat{\sigma}_{ii} \hat{\sigma}_{kk})$ is the estimated correlation.

When $\boldsymbol{X}$ is multinormal, the random variable

$$F_{d,n-d} = \frac{n(n-d)}{(n-1)d} (\bar{\boldsymbol{X}}_n - \boldsymbol{\mu})^{\mathsf{t}} \hat{\boldsymbol{\Sigma}}_n^{\;-1} (\bar{\boldsymbol{X}}_n - \boldsymbol{\mu}) \tag{5.8}$$

has the F distribution with $(d, n-d)$ degrees of freedom. To compute a confidence region with confidence level $1 - \alpha$, one finds the constant x such that $\mathbb{P}[F_{d,n-d} > x] = \alpha$, and the set of values of $\boldsymbol{\mu}$ for which the quadratic form on the right side of (5.8) is less than x is the confidence region. This region is a d-dimensional ellipsoid centered at the sample mean $\bar{\boldsymbol{X}}_n$. When $n \to \infty$, $d\, F_{d,n-d}$ converges to a chi-square with d degrees of freedom, and one can conveniently use this chi-square approximation when n is large.

In fact, when $n \to \infty$, $\boldsymbol{X}$ does not have to be multinormal. In general, we have $\hat{\boldsymbol{\Sigma}}_n \overset{\text{w.p.1}}{\to} \boldsymbol{\Sigma}$ and

$$n(\bar{\boldsymbol{X}}_n - \boldsymbol{\mu})^{\mathsf{t}} \hat{\boldsymbol{\Sigma}}_n^{\;-1} (\bar{\boldsymbol{X}}_n - \boldsymbol{\mu}) \;\Rightarrow\; \chi^2(d)$$

when $d \to \infty$. This can be used to construct a confidence ellipsoid for $\boldsymbol{\mu}$ when n is large. It generalizes CLT-based confidence intervals to the multivariate setting. The confidence ellipsoid for confidence level $1 - \alpha$ is the set of vectors $\boldsymbol{\mu}$ that satisfy $n(\bar{\boldsymbol{X}}_n - \boldsymbol{\mu})^{\mathsf{t}} \hat{\boldsymbol{\Sigma}}_n^{\;-1} (\bar{\boldsymbol{X}}_n - \boldsymbol{\mu}) \leq x$, where x is the $1-\alpha$ quantile of the chi-square distribution with d degrees of freedom.

5.4 Confidence intervals for functions of expectations

Very frequently, the quantity to be estimated is not written directly as an expectation, but can be written as a function of one or more expectations. This happens when we estimate a variance, or a covariance, or the ratio of two expectations, for example. The delta method is a general tool to handle this type of situation when the sample size n is large.

Example 5.4 For a simple one-dimensional case, suppose that $Y_1, \ldots, Y_n$ are i.i.d. with mean μ and we want to compute a confidence interval on $g(\mu)$ for some increasing function g. If n is large enough, we can use the CLT applied to $\bar{Y}_n$ to obtain a confidence interval $[I_1, I_2]$ on μ. Then $[g(I_1), g(I_2)]$ is a confidence interval on $g(\mu)$ with exactly the same confidence level $1-\alpha$. This is true even though $\mathbb{E}[g(\bar{Y}_n)] \neq g(\mu)$ in general. For example, for $g(y) = \ln y$, $\ln(\bar{Y}_n)$ is a biased estimator of $\ln \mu$, but if $[I_1, I_2]$ is a 95% confidence interval on μ, then $[\ln I_1, \ln I_2]$ is a 95% confidence interval on $\ln \mu$. □

Things become more complicated when g is a function of several expectations. A common example is when we want a confidence interval on the ratio $\mathbb{E}[X]/\mathbb{E}[Y]$ from an i.i.d. sample $(X_1, Y_1), \ldots, (X_n, Y_n)$ of the pair (X, Y). A reasonable (strongly consistent) estimator for the ratio is $\bar{X}_n/\bar{Y}_n$, but how do we compute a confidence interval? The remainder of this section will answer this type of question.

5.4.1 The delta method

If $\{\boldsymbol{Y}_n = (Y_{1n}, \ldots, Y_{dn}),\ n \geq 0\}$ is a (deterministic) sequence of vectors converging to a vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_d)$ and if $g : \mathbb{R}^d \to \mathbb{R}$ is a continuous function, then $g(\boldsymbol{Y}_n) \to g(\boldsymbol{\mu})$. Suppose now that the $\boldsymbol{Y}_n$'s are random vectors and that we are interested in the *distribution* of $g(\boldsymbol{Y}_n)$ for large n. More specifically, let us assume that $r(n)(\boldsymbol{Y}_n - \boldsymbol{\mu})$ converges in distribution to some random variable $\boldsymbol{Y}$, and that $r(n) \to \infty$, when $n \to \infty$. A special case frequently encountered is when $\boldsymbol{Y}_n$ is an average of n random vectors that obeys a central-limit theorem; then $r(n) = \sqrt{n}$ and $\boldsymbol{Y}$ has the d-dimensional normal distribution with mean $\mathbf{0}$ and some covariance matrix $\boldsymbol{\Sigma}_{\mathrm{y}}$.

A key tool for studying the asymptotic distribution of $g(\boldsymbol{Y}_n)$ is the *delta theorem*, stated below, which is based simply on a first-order Taylor approximation of g around $g(\boldsymbol{\mu})$:

$$g(\boldsymbol{Y}_n) - g(\boldsymbol{\mu}) = \nabla g(\boldsymbol{\mu})(\boldsymbol{Y}_n - \boldsymbol{\mu}) + o(\|\boldsymbol{Y}_n - \boldsymbol{\mu}\|),$$

where $\nabla g(\boldsymbol{\mu}) = (\partial g(\boldsymbol{\mu})/\partial \mu_1,\ \ldots,\ \partial g(\boldsymbol{\mu})/\partial \mu_d)^{\mathsf{t}}$ is the gradient (vector of partial derivatives) of g at $\boldsymbol{\mu}$, and $g : \mathbb{R}^d \to \mathbb{R}$ is assumed to be continuously differentiable in a neighborhood of $\boldsymbol{\mu}$. [2] Multiplying both sides by $r(n)$, we obtain:

Theorem 5.5 (The delta theorem). *Under the above assumptions, if* $r(n)(\boldsymbol{Y}_n - \boldsymbol{\mu}) \Rightarrow \boldsymbol{Y}$ *when* $n \to \infty$, *then*

[2] From Pierre: Just differentiable might be enough provided that we have uniform integrability in a neighborhood of $g(\boldsymbol{\mu})$. Check this.

$$r(n)(g(\boldsymbol{Y}_n) - g(\boldsymbol{\mu})) \;\Rightarrow\; (\nabla g(\boldsymbol{\mu}))^{\mathrm{t}}\boldsymbol{Y} \quad \text{as } n \to \infty. \tag{5.9}$$

More general versions of this theorem can be found, e.g., in King (1989) and Rubinstein and Shapiro (1993).

Corollary 5.6 *Suppose $\boldsymbol{Y}_n$ obeys a CLT of the form $\sqrt{n}(\boldsymbol{Y}_n - \boldsymbol{\mu}) \;\Rightarrow\; N(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathrm{y}})$ as $n \to \infty$. This is the case, in particular, if $\boldsymbol{Y}_n$ is the average of n* i.i.d. *random vectors with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}_{\mathrm{y}}$. Then, the delta theorem yields the following CLT for $g(\boldsymbol{Y}_n)$:*

$$\frac{\sqrt{n}(g(\boldsymbol{Y}_n) - g(\boldsymbol{\mu}))}{\sigma_g} \;\Rightarrow\; N(0,1) \quad \text{as } n \to \infty, \tag{5.10}$$

where

$$\sigma_g^2 = (\nabla g(\boldsymbol{\mu}))^{\mathrm{t}}\boldsymbol{\Sigma}_{\mathrm{y}}\nabla g(\boldsymbol{\mu}). \tag{5.11}$$

The CLT (5.10) also holds if σ_g is replaced by a strongly consistent estimator.

This CLT can be used to construct asymptotically valid confidence intervals for several quantities of interest, as we will illustrate in the following subsections.

In principle, a more accurate estimate than $g(\boldsymbol{Y}_n)$ could be obtained by taking another term in the Taylor expansion of g, if g is twice continuously differentiable:

$$g(\boldsymbol{Y}_n) - g(\boldsymbol{\mu}) = (\nabla g(\boldsymbol{\mu}))^{\mathrm{t}}(\boldsymbol{Y}_n - \boldsymbol{\mu}) + (\boldsymbol{Y}_n - \boldsymbol{\mu})^{\mathrm{t}}\boldsymbol{H}(\boldsymbol{\mu})(\boldsymbol{Y}_n - \boldsymbol{\mu})/2 + o(\|\boldsymbol{Y}_n - \boldsymbol{\mu}\|^2),$$

where $\boldsymbol{H}$ is the Hessian matrix of g at $\boldsymbol{\mu}$. If a good estimator of the correction term $(\boldsymbol{Y}_n - \boldsymbol{\mu})^{\mathrm{t}}\boldsymbol{H}(\boldsymbol{\mu})(\boldsymbol{Y}_n - \boldsymbol{\mu})/2$ is available, then we can subtract it to reduce the bias.

For example, in the case where $\boldsymbol{Y}_n = \bar{\boldsymbol{X}}_n$, the average of n i.i.d. random variables $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$, then the correction term is

$$\frac{(\bar{\boldsymbol{X}}_n - \boldsymbol{\mu})^{\mathrm{t}}\boldsymbol{H}(\boldsymbol{\mu})(\bar{\boldsymbol{X}}_n - \boldsymbol{\mu})}{2} = \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{X}_i - \boldsymbol{\mu})^{\mathrm{t}}\boldsymbol{H}(\boldsymbol{\mu})(\boldsymbol{X}_j - \boldsymbol{\mu}).$$

Given that $\mathbb{E}[(\boldsymbol{X}_i - \boldsymbol{\mu})^{\mathrm{t}}\boldsymbol{H}(\boldsymbol{\mu})(\boldsymbol{X}_j - \boldsymbol{\mu})] = 0$ for $i \neq j$, this correction term can be estimated by

$$\frac{1}{2n(n-1)}\sum_{i=1}^{n}(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_n)^{\mathrm{t}}\boldsymbol{H}(\bar{\boldsymbol{X}}_n)(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_n).$$

This leads to the following estimator of $g(\boldsymbol{\mu})$, with (asymptotically) lower bias:

$$g(\boldsymbol{\mu}) \approx g(\bar{\boldsymbol{X}}_n) - \frac{1}{2n(n-1)}\sum_{i=1}^{n}(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_n)^{\mathrm{t}}\boldsymbol{H}(\bar{\boldsymbol{X}}_n)(\boldsymbol{X}_i - \bar{\boldsymbol{X}}_n). \tag{5.12}$$

However, this estimator may have a larger variance and larger MSE than $g(\bar{\boldsymbol{X}}_n)$, because of the variance of the correction, so it is not necessarily better. It also requires computation of the Hessian, which often makes it more cumbersome to compute. Moreover, the correction term does not change the CLT of Corollary 5.6, because it converges to zero typically as

$O(1/n)$, whereas the error converges as $O(\sqrt{n})$ in probability. That is, the correction becomes negligible compared with the standard deviation when n is large. Thus, this correction should be seen as a heuristic, and could be worthwhile only when n is not too large.

Example 5.5 In Example 5.4, we estimate $g(\mu) = \log \mu$ by $\log(\bar{Y}_n)$. Here μ has a single dimension and we have $\nabla g(\mu) = \partial \ln \mu / \partial \mu = 1/\mu$ and $\boldsymbol{H}$ contains the single element $\partial^2 \ln \mu / \partial \mu^2 = -1/\mu^2$. If we assume that $\sqrt{n}(\bar{Y}_n - \mu)/S_n \Rightarrow N(0,1)$ as $n \to \infty$, where S_n^2 is the empirical variance of the Y_i's, and we apply Corollary 5.6 to this situation, we obtain that $\sqrt{n}(\ln(\bar{Y}_n) - \ln \mu)/\sigma_g \Rightarrow N(0,1)$ where $\sigma_g^2 = \mathrm{Var}[Y_i]/\mu^2$. Given that $S_n^2/(\bar{Y}_n)^2 \overset{\text{w.p.1}}{\to} \sigma_g^2$, we also have that $\sqrt{n}(\ln(\bar{Y}_n) - \ln \mu)\bar{Y}_n/S_n \Rightarrow N(0,1)$ when $n \to \infty$. A confidence interval on $\ln \mu$ based on this approximation, with confidence level $1-\alpha$, would be $(\ln(\bar{Y}_n) \pm \Phi^{-1}(1-\alpha/2)S_n/(\bar{Y}_n\sqrt{n}))$. This interval is based on a linear approximation of the logarithm around $\bar{Y}_n$, and it differs from that of Example 5.4.

If we add the quadratic term in the Taylor expansion, then the corrected estimator becomes

$$\ln(\bar{Y}_n) + \frac{1}{2n(n-1)(\bar{Y}_n)^2}\sum_{j=1}^{n}(Y_j - \bar{Y}_n)^2 = \ln(\bar{Y}_n) + \frac{1}{2n(n-1)}\sum_{j=1}^{n}(Y_j/\bar{Y}_n - 1)^2,$$

and the confidence interval is simply shifted by the correction.

♣ Exercise: Try to construct a parameterized example (by selecting the distribution of Y_j in parametric form) where the variance of the correction term can be arbitrarily large. Idea: Take $\mu = \mathbb{E}[Y_i] = 1$, so $\bar{Y}_n \approx 1$ and the correction term is approximately $1/n$ times the empirical variance of the Y_i's. Then, observe that the variance of the empirical variance is related to the fourth moment of Y_i (see end of Section 5.4.3), and select a distribution of Y_i for which the second moment is 1 (say) and the fourth moment is arbitrarily large. □

5.4.2 Confidence interval for a ratio of expectations

Let $(X_1, Y_1), \ldots, (X_n, Y_n)$ be i.i.d. replicates of the random vector (X, Y) and suppose we estimate $\nu = \mathbb{E}[X]/\mathbb{E}[Y]$ by

$$\hat{\nu}_n = \frac{\bar{X}_n}{\bar{Y}_n} = \frac{\sum_{i=1}^n X_i}{\sum_{i=1}^n Y_i}.$$

This estimator is biased, but it is easily seen to be strongly consistent by applying the strong law of large numbers to $\bar{X}_n$ and $\bar{Y}_n$.

Denote $\mu_1 = \mathbb{E}[X]$, $\mu_2 = \mathbb{E}[Y]$, $g(\mu_1, \mu_2) = \mu_1/\mu_2$, $\sigma_1^2 = \mathrm{Var}[X]$, $\sigma_2^2 = \mathrm{Var}[Y]$, and $\sigma_{12} = \mathrm{Cov}[X, Y]$. We assume that all these quantities are finite and that $\mu_2 \neq 0$, $0 < \sigma_1^2 < \infty$, and $0 < \sigma_2^2 < \infty$. Write $\hat{\nu}_n = g(\bar{X}_n, \bar{Y}_n)$ where $g(y_1, y_2) = y_1/y_2$. The gradient of g is $\nabla g(\mu_1, \mu_2) = (1/\mu_2, -\mu_1/\mu_2^2)^{\mathsf{t}}$.

We know from the ordinary CLT that

$$\sqrt{n}(\bar{X}_n - \mu_1, \bar{Y}_n - \mu_2)^{\mathsf{t}} \Rightarrow (W_1, W_2)^{\mathsf{t}} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$$

where

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}.$$

Then the delta theorem tells us that

$$\sqrt{n}(\hat{\nu}_n - \nu) \;\Rightarrow\; (W_1, W_2) \cdot \nabla g(\mu_1, \mu_2) = W_1/\mu_2 - W_2\mu_1/\mu_2^2 \sim N(0, \sigma_g^2) \tag{5.13}$$

where

$$\begin{aligned} \sigma_g^2 &= (\nabla g(\mu_1, \mu_2))^{\mathsf{t}} \boldsymbol{\Sigma} \nabla g(\mu_1, \mu_2) \\ &= \sigma_1^2/\mu_2^2 + \sigma_2^2\mu_1^2/\mu_2^4 - 2\sigma_{12}\mu_1/\mu_2^3 \\ &= (\sigma_1^2 + \sigma_2^2\nu^2 - 2\sigma_{12}\nu)/\mu_2^2. \end{aligned} \tag{5.14}$$

To compute a confidence interval for ν using this limit theorem, we need an estimator of the variance constant σ_g^2. Any consistent estimator will do and will yield an asymptotically valid confidence interval for ν. An obvious candidate is the variance estimator $\hat{\sigma}_{g,n}^2$ obtained by replacing the mean, the variances, and the covariance in (5.14) by their sample counterparts. That is,

$$\hat{\sigma}_{g,n}^2 = (\hat{\sigma}_1^2 + \hat{\sigma}_2^2\hat{\nu}_n^2 - 2\hat{\sigma}_{12}\hat{\nu}_n)/(\bar{Y}_n)^2, \tag{5.15}$$

where

$$\begin{aligned} \hat{\sigma}_1^2 &= \frac{1}{n-1}\sum_{j=1}^{n}(X_j - \bar{X}_n)^2, \\ \hat{\sigma}_2^2 &= \frac{1}{n-1}\sum_{j=1}^{n}(Y_j - \bar{Y}_n)^2, \\ \hat{\sigma}_{12} &= \frac{1}{n-1}\sum_{j=1}^{n}(X_j - \bar{X}_n)(Y_j - \bar{Y}_n). \end{aligned}$$

Putting all of this together, we obtain the following CLT for ratios of expectations:

Theorem 5.7 *Under the assumptions made in this example, we have that*

$$\frac{\sqrt{n}(\hat{\nu}_n - \nu)}{\hat{\sigma}_{g,n}} \;\Rightarrow\; \frac{\sqrt{n}(\hat{\nu}_n - \nu)}{\sigma_g} \;\Rightarrow\; N(0,1) \qquad \text{as } n \to \infty. \tag{5.16}$$

The *classical confidence interval* for ν, with nominal confidence level $1-\alpha$, is then the interval centered at $\hat{\nu}_n$ and with half-width $r = z_{1-\alpha/2}\hat{\sigma}_{g,n}/\sqrt{n}$. More refined methods for computing a confidence interval on a ratio are discussed in Example 5.8.

A more direct approach for deriving the CLT in (5.16), for the special case of a ratio of expectation, is as follows. Define the random variables

$$Z_j = X_j - \nu Y_j,$$

which are i.i.d. with mean 0 and variance

$$\sigma_z^2 = \mathrm{Var}[Z_j] = \mathrm{Var}[X_j] + \nu^2 \mathrm{Var}[Y_j] - 2\nu \mathrm{Cov}(X_j, Y_j). \tag{5.17}$$

By applying the ordinary CLT to the Z_j's, we get

$$\frac{\sqrt{n}\bar{Y}_n(\hat{\nu}_n - \nu)}{\sigma_z} = \frac{\sqrt{n}\bar{Z}_n}{\sigma_z} \;\Rightarrow\; N(0,1) \qquad \text{as } n \to \infty. \tag{5.18}$$

This is equivalent to (5.16), because $\sigma_z/\bar{Y}_n \overset{\text{w.p.1}}{\to} \sigma_z/\mu_2 = \sigma_g$ when $n \to \infty$. The obvious estimator of σ_z^2 is

$$\hat{\sigma}_{z,n}^2 = \hat{\sigma}_1^2 + \hat{\sigma}_2^2\hat{\nu}_n^2 - 2\hat{\sigma}_{12}\hat{\nu}_n. \tag{5.19}$$

5.4.3 Confidence Interval on the Variance

Normal case. In the case where $X_1, \ldots, X_n$ are i.i.d. normal with variance σ^2, the classical way of computing a confidence interval on σ^2 is via Theorem 5.1, which says that $(n-1)S_n^2/\sigma^2$ has the chi-square distribution with $n-1$ degrees of freedom. To determine a $100(1-\alpha)\%$ confidence interval for σ^2, select two numbers x_1 and x_2 such that

$$\mathbb{P}[x_1 < \chi_{n-1}^2 < x_2] = 1 - \alpha,$$

where χ_{n-1}^2 is a chi-square random variable with $n-1$ degrees of freedom. It is customary (but not necessary) to select x_1 and x_2 so that

$$\mathbb{P}[\chi_{n-1}^2 < x_1] = \mathbb{P}[\chi_{n-1}^2 > x_2] = \alpha/2.$$

Define the confidence interval as

$$[I_1, I_2] = [(n-1)S_n^2/x_2,\ (n-1)S_n^2/x_1]. \tag{5.20}$$

One has

$$\begin{aligned} \mathbb{P}[I_1 \le \sigma^2 \le I_2] &= \mathbb{P}[(n-1)S_n^2/x_2 \le \sigma^2 \le (n-1)S_n^2/x_1] \\ &= \mathbb{P}[x_1 \le (n-1)S_n^2/\sigma^2 \le x_2] \\ &= 1 - \alpha, \end{aligned}$$

so the interval has the correct coverage under the normality assumption.

♣ Make figure.

Table 5.1 provides the boundaries $(n-1)/x_2$ and $(n-1)/x_1$ of a confidence interval on σ^2/S_n^2, for a few values of α and n. They indicate, for example, that for $n = 1000$, a 90% confidence interval on σ^2 would be $[0.930\, S_n^2,\ 1.077\, S_n^2]$, whereas for $n = 30$ it would be $[0.66\, S_n^2,\ 1.56\, S_n^2]$.

Large-sample case. If the distribution of the X_i's differ significantly from the normal, then $(n-1)S_n^2/\sigma^2$ no longer has the chi-square distribution, and a confidence interval based on the chi-square may have poor coverage. If n is large, we can use the delta theorem to construct a confidence interval as follows. Write

Table 5.1. The boundaries $(n-1)/x_2$ and $(n-1)/x_1$ of a confidence interval on σ^2/S_n^2 for some values of α and n.

	$\alpha = 0.02$		$\alpha = 0.10$	
n	$(n-1)/x_2$	$(n-1)/x_1$	$(n-1)/x_2$	$(n-1)/x_1$
10	0.388	3.518	0.492	2.284
30	0.570	1.939	0.663	1.568
100	0.729	1.413	0.796	1.270
300	0.831	1.216	0.876	1.146
1000	0.902	1.111	0.930	1.077

$$\frac{(n-1)S_n^2}{n\sigma^2} = g(Y_{1n}, Y_{2n}) = \frac{(Y_{2n} - Y_{1n}^2)}{\sigma^2}$$

where $Y_{1n} = \bar{X}_n$, $Y_{2n} = (1/n)\sum_{i=1}^n X_i^2$, and $g(y_1, y_2) = (y_2 - y_1^2)/\sigma^2$. We have $\nabla g(y_1, y_2) = (-2y_1, 1)/\sigma^2$. Denoting $\mu_k = \mathbb{E}[X_i^k]$, this gives $\boldsymbol{\mu} = (\mu_1, \mu_2)^\mathsf{t}$, $g(\boldsymbol{\mu}) = (\mu_2 - \mu_1^2)/\sigma^2$, $\nabla g(\boldsymbol{\mu}) = (-2\mu_1, 1)/\sigma^2$,

$$\boldsymbol{\Sigma}_y = \begin{pmatrix} \mathrm{Var}[X_i] & \mathrm{Cov}[X_i, X_i^2] \\ \mathrm{Cov}[X_i, X_i^2] & \mathrm{Var}[X_i^2] \end{pmatrix} = \begin{pmatrix} \mu_2 - \mu_1^2 & \mu_3 - \mu_1\mu_2 \\ \mu_3 - \mu_1\mu_2 & \mu_4 - \mu_2^2 \end{pmatrix},$$

and by computing (5.11), we find

$$\sigma_g^2 = \mathbb{E}[(X_i - \mu)^4]/\sigma^4.$$

If $M_4 = (1/n)\sum_{i=1}^n (X_i - \bar{X}_n)^4$ is the empirical fourth centered moment, [3] then we have the CLT:

$$\frac{\sqrt{n}(S_n^2/\sigma^2 - 1)}{\sqrt{M_4/S_n^4}} \Rightarrow \frac{\sqrt{n}(S_n^2/\sigma^2 - 1)}{\sigma_g} \Rightarrow N(0,1) \qquad \text{as } n \to \infty. \tag{5.21}$$

In this CLT, M_4/S_n^4 can of course be replaced by any consistent estimator of the standardized fourth moment $\mathbb{E}[(X_i - \mu)^4]/\sigma^4$ (the excess kurtosis plus 3). Unfortunately, however, estimators of the fourth moment are sometimes very noisy.

5.4.4 Confidence Intervals for the Ratio of Two Variances

Normal case. The need to compute a confidence interval on the *ratio of two variances* occurs, for example, in the context of analysis of variance and for comparing the efficiencies of two different estimators. Suppose that $X_1, \ldots, X_m$ are i.i.d. normal with variance σ_x^2, that $Y_1, \ldots, Y_n$ are i.i.d. normal with variance σ_y^2, that the X_i's are independent of the Y_j's, and that the sample variances are $S_{x,m}^2$ and $S_{y,n}^2$, respectively. Then

$$F_{m-1,n-1} = \frac{S_{x,m}^2/\sigma_x^2}{S_{y,n}^2/\sigma_y^2} = \frac{S_{x,m}^2\sigma_y^2}{S_{y,n}^2\sigma_x^2}$$

[3] From Pierre: We also need M_4 to have finite expectation and perhaps some form of uniform integrability; check this.

has the F distribution with $(m-1, n-1)$ degrees of freedom. This follows directly from the fact that $F_{m-1,n-1}$ is a ratio of two independent chi-square random variables divide by their degrees of freedom $m-1$ and $n-1$, respectively. To determine a $100(1-\alpha)\%$ confidence interval for σ_x^2/σ_y^2, select x_1 and x_2 such that

$$\mathbb{P}[x_1 < F_{m-1,n-1} < x_2] = 1-\alpha$$

and define the confidence interval as

$$[I_1, I_2] = \left[\frac{1}{x_2}\frac{S_{x,m}^2}{S_{y,n}^2}, \frac{1}{x_1}\frac{S_{x,m}^2}{S_{y,n}^2}\right].$$

Example 5.6 Suppose that two unbiased estimators X and Y of a quantity μ have known computing costs $C(X)$ and $C(Y)$, respectively. Independent i.i.d. samples of size m for X and size n for Y have sample variances $S_{x,m}^2$ and $S_{y,n}^2$, respectively. If we suppose that X and Y have the normal distribution, a confidence interval $[I_1, I_2]$ of confidence level $1-\alpha$ for σ_x^2/σ_y^2 can be computed as just explained, using the F distribution. This provides a confidence interval for the ratio of efficiencies $\text{Eff}[Y]/\text{Eff}[X] = \sigma_x^2 C(X)/[\sigma_y^2 C(Y)]$, given by $[I_1 C(X)/C(Y),\ I_2 C(X)/C(Y)]$. □

Large-sample case. ♣ For large n, we can also use the delta theorem with $\boldsymbol{\mu} = (\mu_{x,1}, \mu_{x,2}, \mu_{y,1}, \mu_{y,2})^t$ where $\mu_{x,j} = \mathbb{E}[X^j]$, $\mu_{y,j} = \mathbb{E}[Y^j]$, $g(\boldsymbol{\mu}) = [\mu_{x,2} - \mu_{x,1}^2]/[\mu_{y,2} - \mu_{y,1}^2]$.

5.4.5 Confidence intervals on the covariance and correlation

Sometimes, we might want to estimate the covariances or correlations between the performance measures for their own sake. For example, the covariance between the average wait of corporate customers and the average wait of private customers in a day (Charnes 1991), or between the average waiting time and number of abandonments in the call center example.

♣ To compute a confidence interval on $\text{Cov}[X, Y]$ for large n, we can use the delta theorem with $\boldsymbol{\mu} = (\mu_x, \mu_x, \mu_{xy})^t$ where $\mu_x = \mathbb{E}[X]$, $\mu_y = \mathbb{E}[Y]$, $\mu_{xy} = \mathbb{E}[XY]$, $g(\boldsymbol{\mu}) = \mu_{xy} - \mu_x\mu_y$.

♣ To test absence of correlation: von Newmann test; see Alexopoulos (1998), page 253.

5.5 Relative performance: comparing systems

5.5.1 Confidence intervals on the difference between two means

We have n_1 i.i.d. observations $X_{11}, \ldots, X_{1,n_1}$, with mean μ_1, and n_2 i.i.d. observations $X_{21}, \ldots, X_{2,n_2}$, with mean μ_2. We want a confidence interval on the difference $\mu_2 - \mu_1$. The paired-t method and the Welch method are two classical approaches for computing such an interval, under the assumption that the X_{ji}'s are normally distributed. In the Welch method, the X_{1i}'s must be independent of the X_{2i}'s, whereas the paired-t method allows correlation

but requires $n_1 = n_2$. Both methods are based on a normality assumption. Which of the two methods is most appropriate depends on the situation, but in simulation, the paired-t method is used much more frequently, because normally we prefer to use common random numbers and it is easy to have $n_2 = n_1$.

The paired-t method. Let $n_1 = n_2 = n$. Define $Z_i = X_{2i} - X_{1i}$ for $1 \le i \le n$,

$$\bar{Z}_n = \frac{1}{n}\sum_{i=1}^{n} Z_i, \qquad \text{and} \qquad S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Z_i - \bar{Z})^2.$$

Assume that the Z_i are approximately i.i.d. normal. Then, $\sqrt{n}[\bar{Z}_n - (\mu_2 - \mu_1)]/S_n$ follows approximately the Student-t distribution with $n-1$ degrees of freedom. It is then easy to compute a confidence interval on $\mu_2 - \mu_1$, just as in Section 5.2.1.

The X_{1i}'s need not be independent of the X_{2i}'s in this setup. Regardless of the distribution, one has

$$\mathrm{Var}[Z_i] = \mathrm{Var}[X_{2i}] + \mathrm{Var}[X_{1i}] - 2\mathrm{Cov}[X_{1i}, X_{2i}],$$

so it is better if $\mathrm{Cov}[X_{1i}, X_{2i}] > 0$ than if (X_{1i}, X_{2i}) are independent, because this reduces the variance of $\bar{Z}_n$. For this reason it is common practice to deliberately induce such a positive correlation when estimating a difference (see Sections 1.7 and 6.4).

If we apply the delta theorem to the function $g(\mu_1, \mu_2) = \mu_2 - \mu_1$, we find $\sigma_g^2 = \mathrm{Var}[Z_i]$ and $\sqrt{n}[\bar{Z}_n - (\mu_2 - \mu_1)]/S_n \Rightarrow N(0,1)$, which agrees with the fact that the Student distribution converges to the standard normal when $n \to \infty$.

The Welch method. For this method, the X_{1i} and X_{2i} are assumed independent and normally distributed, but one may have $n_1 \neq n_2$. This setup can be useful, for example, to compare a real-life system with a simulation model. Define

$$\bar{X}_{(k)} = \frac{1}{n_k}\sum_{i=1}^{n_k} X_{ki} \qquad \text{and} \qquad S_{(k)}^2 = \frac{1}{n_k - 1}\sum_{i=1}^{n_k}(X_{ki} - \bar{X}_{(k)})^2,$$

for $k = 1, 2$. Then,

$$\frac{\bar{X}_{(2)} - \bar{X}_{(1)} - (\mu_2 - \mu_1)}{[S_{(1)}^2/n_1 + S_{(2)}^2/n_2]^{1/2}}$$

has approximately the Student distribution with a number of degrees of freedom approximately equal to

$$\hat{\ell} = \frac{[S_{(1)}^2/n_1 + S_{(2)}^2/n_2]^2}{[S_{(1)}^2/n_1]^2/(n_1 - 1) + [S_{(2)}^2/n_2]^2/(n_2 - 1)}.$$

5.5.2 Comparing more than two systems

Suppose we want to compare $d > 2$ pairs of systems. To obtain simultaneous confidence intervals for d differences, we can use the Bonferroni inequality (Section 5.3.1). With a confidence level of $1 - \alpha/d$ for each individual interval, the overall confidence that all the intervals cover simultaneously the true differences is at least $1 - \alpha$. Note that the d individual

confidence intervals may be computed by different techniques, with different sample sizes, etc.

♣ Add a section on Ranking and selection. References: Law and Kelton (2000), Goldsman, Nelson, and Schmeiser (1991), Goldsman and Nelson (1998), Dudewicz (1995).

5.6 Estimating a root or a minimum of a function

Confidence intervals are often sought for quantities that are not expressed as the mean or variance of a random variable. Estimating the root or the maximum of a function are prime examples of this.

Suppose that we want to *estimate a root* $\theta = \theta^* \in \mathbb{R}$ to the equation $f(\boldsymbol{\mu}, \theta) = 0$, where $\boldsymbol{\mu}$ is unknown and estimated by $\boldsymbol{Y}_n$. We estimate θ^* by $\hat{\theta}_n$, a root of $f(\boldsymbol{Y}_n, \theta) = 0$. Thus, we have $f(\boldsymbol{\mu}, \theta^*) = f(\boldsymbol{Y}_n, \hat{\theta}_n) = 0$. If f is differentiable around $(\boldsymbol{\mu}, \theta^*)$ and if $(\boldsymbol{Y}_n, \hat{\theta}_n)$ is close enough to $(\boldsymbol{\mu}, \theta^*)$, Taylor expansions around $(\boldsymbol{\mu}, \theta^*)$ and around $(\boldsymbol{Y}_n, \hat{\theta}_n)$ give

$$f(\boldsymbol{\mu}, \hat{\theta}_n) = \nabla_\theta f(\boldsymbol{\mu}, \theta^*)(\hat{\theta}_n - \theta^*) + o(|\hat{\theta}_n - \theta^*|)$$

and

$$f(\boldsymbol{\mu}, \hat{\theta}_n) = (\nabla_{\boldsymbol{\mu}} f(\boldsymbol{Y}_n, \hat{\theta}_n))^{\mathsf{t}}(\boldsymbol{Y}_n - \boldsymbol{\mu}) + o(\|\boldsymbol{Y}_n - \boldsymbol{\mu}\|),$$

respectively. If we assume that $r(n)(\boldsymbol{Y}_n - \boldsymbol{\mu}) \Rightarrow \boldsymbol{Y}$ as in the delta method, then by equaling these two expressions and multiplying by $r(n)$, we obtain

$$r(n)\nabla_\theta f(\boldsymbol{\mu}, \theta^*)(\hat{\theta}_n - \theta^*) \Rightarrow r(n)(\nabla_{\boldsymbol{\mu}} f(\boldsymbol{Y}_n, \hat{\theta}_n))^{\mathsf{t}}(\boldsymbol{Y}_n - \boldsymbol{\mu}) \Rightarrow (\nabla_{\boldsymbol{\mu}} f(\boldsymbol{\mu}, \theta^*))^{\mathsf{t}}\boldsymbol{Y}$$

when $n \to \infty$.

In the common situation of Corollary 5.6, where $\sqrt{n}(\boldsymbol{Y}_n - \boldsymbol{\mu}) \Rightarrow N(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathrm{y}})$ as $n \to \infty$, then we obtain the following CLT for $\hat{\theta}_n$:

$$\frac{\sqrt{n}(\hat{\theta}_n - \theta^*)}{\hat{\sigma}_\theta} \Rightarrow N(0, 1) \quad \text{as } n \to \infty, \tag{5.22}$$

where

$$\sigma_\theta^2 = \frac{(\nabla_{\boldsymbol{\mu}} f(\boldsymbol{\mu}, \theta^*))^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{y}} \nabla_{\boldsymbol{\mu}} f(\boldsymbol{\mu}, \theta^*))}{\nabla_\theta f(\boldsymbol{\mu}, \theta^*)} \tag{5.23}$$

as $n \to \infty$, and the same holds when σ_θ^2 is replaced by a strongly consistent estimator, say

$$\hat{\sigma}_\theta^2 = \frac{(\nabla_{\boldsymbol{\mu}} f(\boldsymbol{Y}_n, \hat{\theta}_n))^{\mathsf{t}} \hat{\boldsymbol{\Sigma}}_{\mathrm{y}} \nabla_{\boldsymbol{\mu}} f(\boldsymbol{Y}_n, \hat{\theta}_n))}{\nabla_\theta f(\boldsymbol{Y}_n, \hat{\theta}_n)} \tag{5.24}$$

if this estimator is strongly consistent.

Estimating the minimum of a smooth one-dimensional function f is equivalent to finding a root of its derivative, if we assume that f is strictly decreasing on the left of the minimum, and strictly increasing on the right. Estimating the maximum is similar if f is strictly increasing on the left and strictly decreasing on the right.

5.7 Estimating quantiles

A prime example of estimating the root of a function is quantile estimation, as we saw briefly in Example 1.12. Consider a random variable X with distribution function F. For $0 \leq q \leq 1$, the qth *quantile* of this distribution is defined as

$$\xi_q = F^{-1}(q) = \inf\{x : F(x) \geq q\}. \tag{5.25}$$

To estimate ξ_q, the general idea is estimate F by some function $\tilde{F}$, and estimate the quantile ξ_q by $\tilde{F}^{-1}(q)$. There is a wide variety of choices for $\tilde{F}$.

Let $X_{(1)}, \ldots, X_{(n)}$ be a sample of n i.i.d. replicates of X, sorted by increasing order, and let $\hat{F}_n$ be their empirical distribution, defined as in Eq. (2.18), with a jump of size $1/n$ at each observation. Perhaps the most straightforward estimator of ξ_q is the *empirical quantile*

$$\hat{\xi}_{q,n} = \hat{F}_n^{-1}(q) = \inf\{x : \hat{F}_n(x) \geq q\} = X_{(\lceil nq \rceil)}. \tag{5.26}$$

This estimator is biased in general, but it is strongly consistent and obeys a CLT:

Theorem 5.8 *(e.g., Serfling 1980, pages 75 and 77, or David 1981, Theorem 9.2, or Schervish 1995, Section 7.2).*
(i) For each q, $\hat{\xi}_{q,n} \overset{\text{w.p.1}}{\to} \xi_q$ when $n \to \infty$.
(ii) If X has a strictly positive and finite density f (the derivative of F) in a neighborhood of ξ_q, then

$$\frac{\sqrt{n}(\hat{\xi}_{q,n} - \xi_q) f(\xi_q)}{\sqrt{q(1-q)}} \;\Rightarrow\; N(0,1) \qquad \text{as } n \to \infty. \tag{5.27}$$

[4]

This CLT indicates that $\hat{\xi}_{q,n}$ is a noisy estimator when the density is small around ξ_q. The intuitive explanation is simple: when the derivative of F is small, a small change in u brings a large change in $x = F^{-1}(u)$. In principle, we could use (5.27) to compute a confidence interval on ξ_q; however this requires a good estimator of the density $f(\xi_q)$, which is usually not easy to obtain when X is the output of a complicated simulation model.

A non-asymptotic way of computing a confidence interval for ξ_q is based on the following reasoning. Assume that $\mathbb{P}[X = \xi_q] = 0$ (i.e., F is continuous at ξ_q). Then, since $\mathbb{P}[X < \xi_q] = q$, the number B of observations $X_{(i)}$ smaller than ξ_q is a binomial random variable with parameters (n, q). If $1 \leq j < k \leq n$, one has $X_{(j)} < \xi_q \leq X_{(k)}$ if and only if $j \leq B < k$. Thus,

$$\mathbb{P}[X_{(j)} < \xi_q \leq X_{(k)}] \;=\; \mathbb{P}[j \leq B < k] \;=\; \sum_{i=j}^{k-1} \binom{n}{i} q^i (1-q)^{n-i}.$$

[4] From Pierre: This can be derived by using the Delta Theorem with $g(\mu) = F^{-1}(q)$. See freakonometrics.hypotheses.org/2352 or www.math.mcgill.ca/dstephens/OldCourses/556-2006/Math556-Median.pdf.

For given n, q, and α, if we want a two-sided confidence interval, we can select values of j and k such that this probability is close to the desired confidence level $1-\alpha$, and $(k+j)/2$ is not too far from nq. The confidence interval is then $(X_{(j)}, X_{(k)})$. For a one-sided confidence interval, e.g., of the form $[I_1, \infty)$, we can take $(X_{(j)}, \infty)$, where j satistifes $\mathbb{P}[B < j] \approx \alpha$.

When n is large and q is not too close to 0 or 1, we can approximate the binomial distribution of B by the normal. That is, $(B - nq)/\sqrt{nq(1-q)} \approx Z$ where $Z \sim N(0,1)$. In this case, for a two-sided interval, we obtain $j = \lfloor nq + 1 - \delta \rfloor$ and $k = \lfloor nq + 1 + \delta \rfloor$, where $\delta = \sqrt{nq(1-q)}\Phi^{-1}(1-\alpha/2)$. [5]

To improve the quality of the quantile estimator, we may replace the step function $\hat{F}_n$ by a smoother estimator of F in (5.26). For example, one could think of using one of the quasi-empirical distributions discussed in Section 2.9.1, or the distribution that corresponds to a kernel density estimator. Then, the estimated quantile is not necessarily equal to one of the observations. In all these cases, the quantile estimator remains biased.

Avramidis and Wilson (1998) consider for instance the quantile estimator $\tilde{\xi}_{q,n}$ obtained when F is estimated by the piecewise-linear function $\tilde{F}_n$ that interpolates the points $\{(X_{(i)}, (i-1/2)/n),\ i = 1, \ldots, n\}$, that is linear between these points, and has jumps of size 1/2 at $X_{(1)}$ and at $X_{(n)}$. They show that the bias and the variance are in $O(1/n)$ for both $\hat{\xi}_{q,n}$ and $\tilde{\xi}_{q,n}$, so the squared bias becomes negligible compared with the variance (e.g., in the MSE expression) for large n. They also show both theoretically and empirically that $\tilde{\xi}_{q,n}$ typically has less bias than $\hat{\xi}_{q,n}$, especially when ξ_q happens to be in an area where the density is small (for example, in the tail of the distribution).

For more on quantile estimation, the reader may consult Conover (1980), David (1981), Hogg and Craig (1995), Avramidis and Wilson (1998) and Weron (2004).

Example 5.7 Banks, mutual funds, pension funds, and insurance companies have huge amounts of money to invest in different assets. Their *investment portfolio* is the vector giving the number of units of each asset they have. It is good practice to diversify the portfolio, i.e., avoid putting a large percentage of the fund on a single asset (or on highly-correlated assets), to reduce the risk of a big loss in the event of a large drop in the value of this single asset.

Managers and regulators of these institutions are very much interested in measuring the risk of investment portfolios. One simple and widely used measure is the *value-at-risk* (VaR), defined as the value x_p such that $\mathbb{P}[L > x_p] = p$, for a fixed probability p, where L is the *decrease* in the value of the portfolio over a given (fixed) time horizon. In other words, if $V(\zeta)$ is the portfolio value at time ζ, and T is the time horizon, then the VaR x_p is the $(1-p)$th quantile of the distribution of $L = V(0) - V(T)$. A popular value of p is $p = 0.01$. Regulatory agencies may require the banks to compute their VaR over a two-week horizon, for example. For insurance companies and pension funds, the time horizon is typically much longer. The widespread use of the VaR as the main measure of risk is certainly open to well-deserved criticism: a single quantile only tells a small amount of information about the distribution of L. For example, if $x_{0.01} = 10^7$ dollars, then all we know is that we have 1% chance of loosing more than 10 million dollars. We may have 0.5% chance of loosing 10 billion dollars, or maybe the maximum we can loose is 20 million dollars, we do not know. A complementary

[5] From Pierre: Add details here. Exercise? Why the +1 in both places?

measure such as $\mathbb{E}[L \mid L > x_p]$, the expected loss conditional on loosing more than the VaR, for example, would provide more information.

To estimate the VaR, a stochastic model must be built to describe the evolution of the vector of asset prices, whose coordinates are generally *dependent*. A portfolio may contain several thousand assets. A common way of reducing the dimension is to define a model where a stochastic process describes the evolution of a (smaller) vector of *risk factors*, and where the value of each asset is a predefined function of the values of the risk factors, with some added noise. Of course, reliable and accurate models for the evolution of asset prices are more difficult to obtain for longer time horizons. See, e.g., Weron (2004) for further details. There are models for which the VaR can be computed exactly or approximately by analytic formulas or numerical methods. But in many cases, it has to be estimated by simulation. For small p, this amounts to estimating a quantile in the tail of the distribution, and straightforward Monte Carlo is rather inefficient for doing this because few observations will fall in the tail. Importance sampling is an appropriate method for improving the efficiency in this context. The idea is to change the probability laws of the model in a way that loosing more than x_p is no longer a rare event. We will return to this in Chapter 6. □

5.8 Functional estimation

One might be interested in estimating the entire distribution (or density) of a random variable X instead of only its expectation or a specific quantile. For this, several form of quasi-empirical distributions and density estimators are available (see Sections 2.9 and 2.9.4). However, a confidence interval for the entire function is generally difficult to obtain.

♣ To be continued.

Another example of functional estimation is when the model depends on a parameter θ, and we want to estimate $\mu(\theta) = \mathbb{E}_\theta[X]$ as a function of θ, where $\mathbb{E}_\theta$ is the expectation for the given value of θ.

5.9 Bootstrap confidence intervals

A general class of approaches for constructing confidence intervals (and more generally for estimating the distribution of some estimator) without assuming any particular distribution for the observations are based on the *bootstrap principle*. These are in fact Monte Carlo methods that *resample* from the original sample to estimate the distribution of interest. This is a primary example of *using simulation for statistics*, as opposed to using statistics for simulation (which is what most of this chapter is about).

Basic nonparametric bootstrap. Suppose we have an i.i.d. sample of random vectors $X_1, \ldots, X_n$ from distribution F in $\mathbb{R}^d$ and we use $Y = g(X_1, \ldots, X_n)$ as an estimator of some unknown real-valued quantity θ. For example, if $d = 1$, we may have $Y = \bar{X}_n$ with $\theta = \mu$, or $Y = S_n^2$ with $\theta = \sigma^2$, but there are many other possibilities. (The X_i's are not in boldface but they can be either scalars of vectors.) We do not assume that $\mathbb{E}[Y] = \theta$. On the

other hand, we assume that Y does not change when we permute the observations X_i, so we may rewrite, for example, $Y = g(X_{(1)}, \ldots, X_{(n)})$ if the X_i are real-valued.

Let $K_n(F, \cdot)$ be the distribution function of the *error* $Y - \theta$:

$$K_n(F, z) = \mathbb{P}[Y - \theta \le z] \qquad \text{for } z \in \mathbb{R}.$$

If we knew this distribution, an exact confidence interval for θ at level $1 - \alpha_1 - \alpha_2$ could be computed as

$$(I_1, I_2) = (Y - K_n^{-1}(F, 1 - \alpha_1),\ Y - K_n^{-1}(F, \alpha_2)), \tag{5.28}$$

where $K_n^{-1}(F, q)$ is the qth quantile of the distribution $K_n(F, \cdot)$. Indeed, one has $\mathbb{P}[I_1 > \theta] = \mathbb{P}[Y - \theta > K_n^{-1}(F, 1 - \alpha_1)] = 1 - K_n(F,\ K_n^{-1}(F, 1 - \alpha_1)) = \alpha_1$. Similarly, $\mathbb{P}[I_2 < \theta] = \alpha_2$. For a one-sided interval, it suffices to take $\alpha_1 = 0$ or $\alpha_2 = 0$.

But in most practical situations, the distribution function $K_n(F, \cdot)$ of $Y - \theta$ is unknown and often quite complicated. How can we have an idea of that distribution? Conceptually, in the case where $\mathbb{E}[Y] = \theta$, we could think of replicating the experiment to obtain m i.i.d. copies of Y, say $Y_1, \ldots, Y_m$, and look at the empirical distribution of $Y_j - \bar{Y}_m$ for $j = 1, \ldots, m$. This solution is generally unacceptable, because if we could obtain an i.i.d. sample of size mn, we would rather use it to compute an estimator Y_{mn} based on all mn observations, and we are back to the same problem, with n replaced by mn.

Let $x_1, \ldots, x_n$ be the values taken by the random variables $X_1, \ldots, X_n$ and let $y = g(x_1, \ldots, x_n)$. The idea of the bootstrap is to obtain "simulated" additional copies of Y by resampling from the same observations $x_1, \ldots, x_n$. The *basic nonparametric bootstrap* procedure works as follows. Draw n random observations $X_1^*, \ldots, X_n^*$ *with replacement* from $\{x_1, \ldots, x_n\}$ (thus, $X_1^*, \ldots, X_n^*$ is an i.i.d. sample of size n taken from the empirical distribution $\hat{F}_n$ defined in Section 2.9) and compute $Y^* = g(X_1^*, \ldots, X_n^*)$. Repeat this m times (with the same $\hat{F}_n$), independently, and let $Y_1^*, \ldots, Y_m^*$ be the m copies of Y^* thus obtained. What we are actually doing is replicating the experiment, but assuming that the observations X_i have distribution $\hat{F}_n$ instead of F. Resampling observations from $\hat{F}_n$ is typically much cheaper than making new simulation runs to generate observations from F, especially when each simulation run involves a lot of computations.

Let $\hat{K}_{n,m}$ denote the empirical distribution of $Y_1^* - y, \ldots, Y_m^* - y$. When $m \to \infty$, this distribution function converges w.p.1 to the distribution function of $Y^* - y$, which happens to be $K_n(\hat{F}_n, \cdot)$. The procedure returns the confidence interval obtained after replacing $K_n(F, \cdot)$ by $\hat{K}_{n,m}$ in (5.28), namely

$$(y - \hat{K}_{n,m}^{-1}(1 - \alpha_1),\ y - \hat{K}_{n,m}^{-1}(\alpha_2)). \tag{5.29}$$

This is equivalent to replacing F by $\hat{F}_n$ in (5.28), and then approximating $K_n(\hat{F}_n, \cdot)$ by $\hat{K}_{n,m}$. Each of these two approximations is a source of error. The second error vanishes when $m \to \infty$ and the first vanishes when $n \to \infty$.

Observe that the qth quantile of the empirical distribution $\hat{K}_{n,m}$ is $Y^*_{(\lceil mq \rceil)} - y$, where $Y^*_{(1)}, \ldots, Y^*_{(m)}$ are the Y_j^* sorted by increasing order. Thus, the confidence interval in (5.29) can be rewritten as

$$(2y - Y^*_{(\lceil m(1-\alpha_1) \rceil)},\ 2y - Y^*_{(\lceil m\alpha_2 \rceil)}). \tag{5.30}$$

There are situations where $K_n^{-1}(\hat{F}_n, \cdot)$ can be computed in closed form, so it can be used in (5.29) in place of $\hat{K}_{n,m}^{-1}$ and there is no need to perform the m bootstrap simulations. This

eliminates one source of error. One example of this is when X_j has a continuous univariate distribution and we want a two-sided confidence interval on its mean $\mu = \mathbb{E}[X_j]$. Explicit formulas for the confidence interval boundaries in this case are given by Hall (1988), page 945. These formulas involve the first four empirical moments of the X_j (the mean, variance, skewness, and kurtosis).

Nonparametric bootstrap-*t*. There are many other types of bootstrap procedures. The *nonparametric bootstrap-t* is one variant that often works better than the basic bootstrap, but it requires an estimator of the variance of Y. The difference is that it works with the distribution $J_n(F, \cdot)$ of the *studentized statistic* $(Y - \theta)/S$, where $S^2 = h^2(X_1, \ldots, X_n)$ is an estimator of $\mathrm{Var}[Y]$. In this case, an exact confidence interval of level $(1 - \alpha_1 - \alpha_2)$ for θ would be

$$(I_1, I_2) = (Y - J_n^{-1}(F, 1 - \alpha_1)S,\ Y - J_n^{-1}(F, \alpha_2)S). \tag{5.31}$$

The nonparametric bootstrap-t algorithm generates n observations $X_1^*, \ldots, X_n^*$ for each of the m bootstrap samples as before, but this time computes $Y^* = g(X_1^*, \ldots, X_n^*)$, $S^* = h(X_1^*, \ldots, X_n^*)$, and $Z^* = (Y^* - y)/S^*$. Let $Z_1^*, \ldots, Z_m^*$ be the m replicates of Z^* and $\hat{J}_{n,m}$ their empirical distribution. To compute the confidence interval, we simply replace $J_n(F, \cdot)$ by $\hat{J}_{n,m}(\cdot)$ in (5.31). This gives the interval

$$\begin{aligned}(I_1, I_2) &= (y - \hat{J}_{n,m}^{-1}(1 - \alpha_1)S,\ y - \hat{J}_{n,m}^{-1}(\alpha_2)S) \\ &= (y - Z^*_{(\lceil m(1-\alpha_1)\rceil)}S,\ y - Z^*_{(\lceil m\alpha_2\rceil)}S).\end{aligned} \tag{5.32}$$

Coverage and width. Suppose that $m = \infty$, $\theta = g(\mathbb{E}[X_i])$ and $Y = g(\bar{X}_n)$ where g is a smooth function, and the asymptotic variance of Y is $\sigma^2 = h^2(\mathbb{E}[X_i])/n$ for a fixed function h. The fact that the variance must be a function of $\mathbb{E}[X_i]$ is not restrictive, because we can add artificial components to the vector X_i as needed. For instance, if $\theta = \mathbb{E}[W_i]$ and $\sigma^2 = \mathrm{Var}[W_i]$ for a univariate random variable W_i, then we can set $X_i = (W_i, W_i^2)$.

For the two types of bootstrap confidence intervals discussed so far, we may distinguish the one-sided and two-sided intervals. For each of these four types of intervals, under the above assumptions and additional regularity conditions on the distribution F detailed in his paper, Hall (1988) has shown that the coverage error of a confidence interval of level $1 - \alpha$ converges as

$$n^{-\gamma} p(z_{1-\alpha})\phi(z_{1-\alpha}) + O(n^{-\gamma-1/2}),$$

where p is a polynomial, γ is a constant, and both depend only on the type of confidence interval considered, ϕ is the standard normal density, and $z_{1-\alpha}$ is the $(1 - \alpha)$th quantile of the standard normal distribution. One has $\gamma = 1/2$ for the one-sided basic bootstrap confidence interval and $\gamma = 1$ for the other three types of intervals. Thus, in the one-sided case, the bootstrap-t interval has a much better asymptotic behavior than the basic bootstrap. Empirically, the bootstrap-t often performs better than the basic bootstrap even for two-sided intervals. But it can be used only when a good variance estimator is available.

Hall (1986) has also shown that the choice of m has little impact on the coverage error, but that a small m gives a larger variance for the width of the confidence interval. In practice, a typical value is $m = 1000$.

Other types of bootstraps. The nonparametric basic bootstrap and bootstrap-t were constructed by replacing F by $\hat{F}_n$. But as we have seen earlier, there can be better estimators of F than $\hat{F}_n$. This includes, for example, the smoother quasi-empirical distributions, as well as estimators of F based on density estimators. These estimators of F can be used in place of $\hat{F}_n$ to define improved variants of the bootstrap. On the other hand, the bootstrap sampling may become significantly more expensive with these smoother distributions.

In *parametric bootstrap* procedures, F is assumed to belong to a class of distributions F_θ parameterized by θ. Let $\hat{\theta}_n = Y = g(X_1, \ldots, X_n)$ be an estimator of θ computed from the data. Each bootstrap sample is generated from the parametric distribution $F_{\hat{\theta}_n}$ instead of from $\hat{F}_n$, and is used to estimate the parameter θ by $Y^* = g(X_1^*, \ldots, X_n^*)$. The distribution of these m bootstrap parameter estimates $Y_1^*, \ldots, Y_m^*$ is used to make inference about the distribution of the estimator $\hat{\theta}_n$ in the same way as for the nonparametric bootstrap. In particular, a confidence interval on θ is obtained from (5.29) for the *parametric basic bootstrap* and by (5.32) for the *parametric bootstrap-t*.

Other bootstrap confidence interval methods studied in the literature include the percentile method, its improvements the BC, BCa, and ABC methods, double bootstrap techniques, etc. We refer the reader to Efron and Tibshirani (1994), Hall (1992), Léger, Politis, and Romano (1992), and Chernick (1999) for more extensive coverages.

Example 5.8 Choquet, L'Ecuyer, and Léger (1999) have studied and experimented different types of bootstrap methods for computing a confidence interval for a ratio of expectations. They recommend a nonparametric bootstrap-t procedure in which they use the mean and variance estimators $\hat{\nu}_n$ and $\hat{\sigma}_{g,n}$ defined here. In their experiments, this bootstrap performed at least as well as all the other methods they tried, and much better than the classical confidence interval described in Section 5.4.2, especially for one-sided intervals. □

♣ Examples: (1) CI for highly non-normal case; (2) ratio of expectations; (3) $P(X > Y)$, see Léger, Politis, and Romano (1992), page 379. (4) parametric: CI on λ for exponential dist. (5) Estimating θ for a $U(0, \theta)$ distribution: nonparametric bootstrap fails, but parametric bootstrap ok.

♣ See Schervish (1995), Section 5.3, for additional insight, examples, and exercises.

5.10 Estimation of Steady-State Performance: The Setup

In this section, we switch our attention to the problem of estimating a long-term average over an infinite time horizon. Our analysis is for a discrete time model, although we briefly outline how everything translates to the continuous-time setting. We start by assuming (momentarily) that the process is stationary. The initial bias problem will be dealt with later.

5.10.1 Autocorrelation in stationary stochastic processes

Asymptotic variance constant. Consider a weakly stationary discrete-time process $\{C_i, i \geq 1\}$, where $\mathbb{E}[C_i] = \mu$, $\text{Var}[C_i] = \sigma^2 > 0$, and ρ_k is the autocorrelation of lag k, defined in Section 2.17. Suppose we estimate the mean μ by the average of the first n observations,

$$\bar{C}_n = \frac{1}{n}\sum_{i=1}^{n} C_i.$$

Then, $\mathbb{E}[\bar{C}_n] = \mu$ and

$$\text{Var}[\bar{C}_n] \;=\; \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\text{Cov}[C_i, C_j] = \frac{\sigma^2(1+\gamma_n)}{n} = \frac{\sigma_n^2}{n} \tag{5.33}$$

where $\sigma_n^2 = \sigma^2(1+\gamma_n)$ and

$$\gamma_n = \frac{2}{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\rho_{j-i} = \frac{2}{n}\sum_{k=1}^{n-1}(n-k)\rho_k. \tag{5.34}$$

Let

$$\gamma \stackrel{\text{def}}{=} \lim_{n\to\infty}\gamma_n = 2\sum_{k=1}^{\infty}\rho_k \tag{5.35}$$

if this limit exists. If $\gamma < \infty$, then $\text{Var}[\bar{C}_n] \in O(1/n)$, with *asymptotic variance constant*

$$\sigma_\infty^2 \stackrel{\text{def}}{=} \lim_{n\to\infty}\sigma_n^2 = \lim_{n\to\infty} n\text{Var}[\bar{C}_n] = \sigma^2(1+\gamma). \tag{5.36}$$

The constant γ can be interpreted as an overall measure of autocorrelation between the C_i's. The ratio $\sigma_\infty^2/\sigma^2 = 1+\gamma$ is sometimes called the *variance inflation factor*. When the C_i's are i.i.d. one has $\gamma = 0$.

[6]

Central-limit theorems. If $\bar{C}_n$ obeys a CLT and if a consistent estimator $\hat{\sigma}_n^2$ is available for σ_∞^2, one can compute an asymptotically valid confidence interval for μ by exploiting the fact that $\sqrt{n}(\bar{C}_n - \mu)/\hat{\sigma}_n \;\Rightarrow\; N(0,1)$ when $n \to \infty$. At this point, we need the following two things: (a) conditions under which such a CLT holds and (b) a good estimator of σ_∞^2. Proposals for (b) are discussed later in this chapter. For (a), we will find it more convenient to work with a *functional CLT (FCLT)*, stated in Assumption 5.1.

We now drop the explicit assumption that the process $\{C_j, j \geq 1\}$ is stationary. The FCLT assumption below implies a form of asymptotic stationarity. We will see later how this assumption can be verified in practice. For arbitrary constants μ and σ_∞ and for each n, we define the continuous-time process

$$W_n(t) = \frac{\lfloor nt \rfloor(\bar{C}_{\lfloor nt \rfloor} - \mu)}{\sigma_\infty\sqrt{n}}, \qquad 0 \leq t \leq 1, \tag{5.37}$$

[6] From Pierre: Here, perhaps we should start with an ergodic Markov chain over a finite number of states.

where $\bar{C}_0 = \mu$ by convention, so $W_n(0) = 0$. Let $D[0,1]$ denote the space of right-continuous real-valued functions over $[0,1]$, with left-hand limits.

Assumption 5.1 (FCLT) For some finite constants μ and $\sigma_\infty > 0$, as $n \to \infty$, the process W_n converges in distribution in $D[0,1]$ to a standard Brownian motion process W. □

This assumption implies of course that $W_n(1) \Rightarrow W(1)$, which immediately implies the ordinary CLT:

$$\frac{\sqrt{n}(\bar{C}_n - \mu)}{\sigma_\infty} = W_n(1) \Rightarrow W(1) \sim N(0,1). \tag{5.38}$$

If a consistent estimator $\hat{\sigma}_n^2$ of σ_∞^2 is available, we also have

$$\frac{\sqrt{n}(\bar{C}_n - \mu)}{\hat{\sigma}_n} \Rightarrow N(0,1) \tag{5.39}$$

and this can be used to construct an asymptotically valid confidence interval for μ.

The FCLT assumption implies the following limit theorem (for a proof, see Foley and Golsman 1999, Theorem 1). Define the process

$$T_n(t) = \frac{\lfloor nt \rfloor (\bar{C}_n - \bar{C}_{\lfloor nt \rfloor})}{\sigma_\infty \sqrt{n}}, \qquad 0 \le t \le 1. \tag{5.40}$$

Note that $T_n(0) = T_n(1) = 0$.

Theorem 5.9 *Under Assumption 5.1, the process T_n converges in distribution to the process B defined by $B(t) = tW(1) - W(t)$, which is a standard Brownian bridge independent of $W(1)$.*

Verifying the FCLT assumption. Assumption 5.1 and Theorem 5.9 give a lot of information on the behavior of the average process $\{\bar{C}_j,\ j \ge 1\}$, but how can we verify the FCLT assumption? It turns out that this assumption holds under great generality, provided that the C_j's do not have strong long-range dependence. A classical sufficient condition that implies Assumption 5.1 can be expressed in terms of the phi-mixing property, which we now define. Roughly speaking, phi-mixing means that $\{C_{j+n}, C_{j+n+1}, \dots\}$ becomes independent of $\{C_1, \dots, C_j\}$ in the limit when $n \to \infty$.

Definition 5.1 For $1 \le i \le j \le \infty$, let $\mathcal{F}_{i,j}$ denote the sigma-field generated by $C_i, \dots, C_j$, which can be interpreted as the set of events whose occurence can be determined by observing only $C_i, \dots, C_j$. The process $\{C_j,\ j \ge 1\}$ is called *phi-mixing* if there is a sequence of real numbers $\{\varphi_n,\ n \ge 1\}$ decreasing to 0, such that for each $j \ge 1$ and $n \ge 1$, each $E_1 \in \mathcal{F}_{1,j}$ with $\mathbb{P}[E_1] > 0$, and each $E_2 \in \mathcal{F}_{j+n,\infty}$,

$$|\mathbb{P}[E_2 \mid E_1] - \mathbb{P}[E_2]| \le \varphi_n.$$

□

Theorem 5.10 (Billingsley 1968, Theorem 20.1) *If* $\{C_j,\ j \geq 1\}$ *is strongly stationary and phi-mixing with* $\sum_{n=1}^{\infty} \sqrt{\varphi_n} < \infty$, *then* $\sum_{k=1}^{\infty} |\rho_k| < \infty$, $\gamma < \infty$, *and Assumption 5.1 holds.*

Assumption 5.1 can also hold under less restrictive conditions. For example, the process $\{C_j,\ j \geq 1\}$ can be stationary only in the limit as $n \to \infty$. Other sets of sufficient conditions for that FCLT are mentioned, e.g., in Glynn and Iglehart (1990).

Lyapunov test functions. Phi-mixing is one of the weakest conditions, but it is generally hard to verify directly. We now introduce the stronger concept of v-uniform ergodicity for Markov chains, which holds under conditions that can be checked via Lyapunov test functions. The definitions are given in terms of a *discrete-time Markov chain* $\{Y_j,\ j \geq 0\}$ evolving over a general state space $\mathbb{S}$. Usually, the cost process $\{C_j,\ j \geq 1\}$ is not a Markov chain but it can be defined in terms of the Markov chain by $C_j = h(Y_j)$ for $j \geq 1$, for some function h, where Y_j may represent the state of the system at step j.

For more details, see, e.g., Meyn and Tweedie (1993) and Henderson (2001).

Definition 5.2 A Markov chain $\{Y_j,\ j \geq 0\}$ is *v-uniformly ergodic* for a function $v : \mathbb{S} \to [1, \infty)$ if there is a probability measure π over $\mathbb{S}$ such that

$$\lim_{n\to\infty} \sup_{y\in\mathbb{S}} \sup_{h:|h|\leq v} (\mathbb{E}[h(Y_n) \mid Y_0 = y] - \mathbb{E}_\pi[h(Y)]) = 0,$$

where $h : \mathbb{S} \to \mathbb{R}$ and $\mathbb{E}_\pi$ denotes the expectation when Y obeys π. □

The usual way of verifying v-uniformly ergodicity is to find a *test function* v that satisfies the conditions of the following theorem.

Theorem 5.11 *An aperiodic discrete-time Markov chain* $\{Y_j,\ j \geq 0\}$ *is v-uniformly ergodic if it satisfies the following (sufficient)* Lyapunov conditions*: There exists a set* $B \subseteq \mathbb{S}$, *real numbers* $0 < a < 1$, $c > 0$, $\delta > 0$, *an integer* $m \geq 1$, *a probability measure* Q *on* $\mathbb{S}$, *and a function* $v : \mathbb{S} \to [1, \infty)$ *such that*

$$\mathbb{P}[Y_m \in \cdot \mid Y_0 = y] \geq \delta Q(\cdot) \quad \text{for all } y \in B \text{ and} \tag{5.41}$$

$$\mathbb{E}[v(Y_1) \mid Y_0 = y] \leq av(y) + c\mathbb{I}[y \in B] \quad \text{for all } y \in \mathbb{S}. \tag{5.42}$$

Theorem 5.12 *If the chain* $\{Y_j,\ j \geq 0\}$ *is v-uniformly ergodic, then*

(i) It has a unique stationary probability distribution π *over* $\mathbb{S}$.

(ii) (Strong law of large numbers.) If $h : \mathbb{S} \to \mathbb{R}$, $\mathbb{E}_\pi[|h(Y)|] < \infty$, $\mu = \mathbb{E}_\pi[h(Y)]$, *and* $C_j = h(Y_j)$, *then*

$$\lim_{n\to\infty} \bar{C}_n \overset{\text{w.p.1}}{\to} \mu.$$

(iii) (Functional central limit theorem.) Let σ_∞ *be defined as in (5.36). If* $|h(y)| \leq v(y)$ *for all* $y \in \mathbb{S}$ *and* $\sigma_\infty > 0$, *then the process* $\{C_j,\ j \geq 1\}$ *satisfies the FCLT in Assumption 5.1 with* $\mu = \mathbb{E}_\pi[h(Y)]$ *and this* σ_∞.

(iv) (Convergence of the bias as $O(1/n)$.) Under the conditions of (iii), if the initial state Y_0 has probability distribution π_0 over $\mathbb{S}$ and if $\mathbb{E}_{\pi_0}[v(Y_0)] < \infty$, then

$$\sum_{j=1}^{\infty} \mathbb{E}_{\pi_0}[C_j - \mu] = \kappa$$

for some $\kappa < \infty$, and

$$\mathbb{E}_{\pi_0}\left[\bar{C}_n - \mu\right] = \kappa/n + O(\alpha^n)$$

for some constant $\alpha < 1$.

5.10.2 Continuous-time setup

The previous discussion, written in terms of a discrete-time process, can be reformulated in terms of a continuous-time process $\{C(t), t \geq 0\}$ as follows (see also Billingsley 1968, page 178). We define

$$\begin{aligned} \bar{C}(t) &= \frac{1}{t}\int_0^t C(s)ds, \qquad t \geq 0, \\ W_n(t) &= \frac{nt(\bar{C}(nt) - \mu)}{\sigma_\infty \sqrt{n}}, \qquad 0 \leq t \leq 1, \end{aligned}$$

and Assumption 5.1 remains the same. In the phi-mixing condition, φ_n is replaced by $\varphi(t)$, $t \geq 0$, and one must have $|\mathbb{P}[E_2 \mid E_1] - \mathbb{P}[E_2]| \leq \varphi(t)$ whenever E_1 depends only on $\{C(u), 0 \leq u \leq s\}$ and E_2 depends only on $\{C(u), s + t \leq u < \infty\}$. The continuous-time version of Theorem 5.10 requires that $\int_0^\infty \sqrt{\varphi(t)}dt < \infty$. In the stationary case, we have $\mu = \mathbb{E}[C(t)]$, $\sigma^2 = \mathrm{Var}[C(t)]$, $\sigma^2(t) = t\mathrm{Var}[\bar{C}(t)]$, and

$$\sigma_\infty^2 = \lim_{t\to\infty} \sigma^2(t) = 2\int_0^\infty \mathrm{Cov}[C(0), C(t)]dt.$$

The discrete-time case is a special case of the continuous-time one: It suffices to define $C(t) = C_i$ for $i - 1 \leq t < i$. On the other hand, the continuous-time process can be approximated by a discrete-time process defined by $C_i = \int_{i-1}^i C(t)dt$ for $i \geq 1$. Billingsley (1968) uses this approximation to show that the properties proved for the discrete-time process transfer to the continuous-time case.

5.10.3 Average cost per unit of time

The discrete-time setup defined so far applies directly if C_i represents the cost at the ith step of a discrete-time simulation, and if we are interested in the average cost per step. In the more general case where the event times do not satisfy $t_i = i$, and where μ is the steady-state average *cost per unit of time*, things are more complicated, because the number of events in a given time interval is random. However, if the time horizon t is deterministic, then under fairly reasonable conditions that are satisfied for typical systems, there is a constant $\lambda > 0$

such that $N(t)/t \to \lambda$ with probability 1 as $t \to \infty$, where $N(t)$ is the number of events during $[0, t]$. This λ is the average number of events per unit of time. In this case, $\bar{v} = \lambda\mu$, where $\bar{v}$ is the average cost per unit of time, defined in (2.79), and μ is the average cost per event. If we define

$$\tilde{\sigma}_\infty^2 = \lambda\sigma_\infty^2, \tag{5.43}$$

we have the following CLT, which is similar to (5.39) except that the number of steps n is replaced by the time t (this rescales the time) and the variance constant is multiplied by λ to take the change of time scale into account.

Theorem 5.13 *Under Assumption 5.1, if $N(t)/t \to \lambda$ with probability 1 where $0 < \lambda < \infty$, and if some variance estimator $\hat{\sigma}_t^2$ satisfies $\hat{\sigma}_t^2 \to \tilde{\sigma}_\infty^2$ in probability, when $t \to \infty$, then*

$$\frac{\sqrt{t}[V_{N(t)}/t - \bar{v}]}{\hat{\sigma}_t} \quad \Rightarrow \quad N(0,1) \qquad \text{when } t \to \infty.$$

Proof. For large t, one has, a.s.,

$$\begin{aligned}
\frac{\sqrt{t}[V_{N(t)}/t - \bar{v}]}{\hat{\sigma}_t} &\approx \frac{\sqrt{N(t)/\lambda}[\lambda V_{N(t)}/N(t) - \lambda\mu]}{\tilde{\sigma}_\infty} \\
&= \frac{\sqrt{N(t)}[V_{N(t)}/N(t) - \mu]}{\sigma_\infty} \\
&\Rightarrow N(0,1) \qquad \text{as } t \to \infty
\end{aligned}$$

from Theorem 5.10 and because $N(t) \to \infty$ when $t \to \infty$.

For more on event vs time averages for point process models, see, e.g., Brémaud, Kannurpatti, and Mazumdar (1992).

5.10.4 Examples

Example 5.9 Let $\{C_j,\, j \geq 0\}$ be an *autoregressive process of order 1* (AR(1) process) defined by the recurrence

$$C_{j+1} = \mu + a_1(C_j - \mu) + \epsilon_j, \qquad j \geq 0, \tag{5.44}$$

where $-1 < a_1 < 1$, the ϵ_j are i.i.d. $N(0, \sigma_\epsilon^2)$, and $C_0 \sim N(\mu, \sigma^2)$ where $\sigma^2 = \sigma_\epsilon^2/(1 - a_1^2)$. This process is strictly stationary with $C_j \sim N(\mu, \sigma^2)$, $\rho_j = a_1^j$, and $\sigma_\infty^2 = \sigma^2(1 + a_1)/(1 - a_1)$. The ratio σ_∞^2/σ^2 becomes arbitrarily large when a_1 approaches 1. When $a_1 < 0$, the autocorrelation process alternates in sign and $\sigma_\infty^2 < \sigma^2$. For a detailed analysis of this process, see Box, Jenkins, and Reinsel (1994). The Lyapunov conditions of Theorem 5.11 are not hard to verify for this example. □

Example 5.10 Consider an $M/M/1$ queue with arrival rate λ and service rate μ. It is well-known (e.g., Taylor and Karlin 1998) that the *equilibrium distribution* of the process

$\{W_j,\, j \geq 1\}$ of customer's waiting times is given by $F_w(x) = \mathbb{P}[W_j \leq x] = 1 - \rho e^{-(\mu-\lambda)x}$ for $x \geq 0$, where $\rho = \lambda/\mu$. If W_1 has this distribution, then each W_j has it as well so the process is stationary. One can then easily deduce that

$$w = \mathbb{E}[W_j] = \frac{\rho}{\mu(1-\rho)} \quad \text{and} \quad \sigma^2 = \mathrm{Var}[W_j] = \frac{\rho(2-\rho)}{\mu^2(1-\rho)^2}.$$

The autocorrelation function for the W_j's is given by (see Blomqvist 1967, Song and Schmeiser 1995, Whitt 1989)

$$\begin{aligned}\rho_j &= \frac{(1-\rho)^3(1+\rho)}{\rho^3(2-\rho)} \sum_{k=j+3}^{\infty} \frac{\rho^k}{(1+\rho)^{2k}} \frac{(2k-3)!}{k!(k-2)!}(k-j-1)(k-j-2) \\ &= \frac{(1-\rho)^3(1+\rho)}{2\pi\rho^3(2-\rho)} \int_0^{4\rho/(1+\rho)^2} \frac{x^{j+3/2}[4\rho(1+\rho)^{-2} - x]^{1/2}}{(1-x)^3} dx\end{aligned}$$

for $j \geq 1$. One has $\lim_{j\to\infty} \rho_{j+1}/\rho_j = 4\rho/(1+\rho)^2 < 1$ and

$$\sigma_\infty^2 = \frac{\rho(\rho^3 - 4\rho^2 + 5\rho + 2)}{\mu^2(1-\rho)^4}$$

(Daley 1968).

Here, ρ_j is (asymptotically) exponentially monotone decreasing in j, but when ρ is close to 1, it decreases very slowly and $\sigma_\infty^2 \approx 4\rho/[\mu^2(1-\rho)^4] \approx 4\sigma^2/(1-\rho)^2 \gg \sigma^2$ (i.e., the mean process $\bar{W}_n$ is very noisy). For example, if $\mu = 1$ and $\rho = 0.9$, then $\rho_{j+1}/\rho_j \to 3.6/3.61 \approx 0.9972$, $\rho_{200} \approx 0.30$, $\sigma^2 = 99$ and $\sigma_\infty^2 = 39890$. Whitt (1989) gives explicit expressions for σ_∞^2 for other processes such as the queue length, workload, and number of customers in the system as a function of time, in a $M/M/1$ queue. He also gives approximations for $M/G/1$ queues. Such approximations of σ_∞^2 permit one to estimate how much simulation time is needed to obtain a confidence interval of a given width.

The equilibrium distribution of the number of customers in the system for the $M/M/1$ queue is the geometric distribution defined by $\mathbb{P}[L = \ell] = \rho(1-\rho)^\ell$ for $\ell \geq 0$. If $L(t)$ denotes the number of customers in the system at time t and if $L(0)$ follows this distribution, then $L(t)$ follows the same distribution for all $t \geq 0$ and W_i also follows its equilibrium distribution for all $i \geq 1$.

Note that starting this process deterministically at $W_1 = w$ *does not* yield a stationary process; instead one has $\mathbb{E}[W_j] < \mu$ for $j \geq 2$, and $\mathbb{E}[W_j]$ converges to μ from below when $j \to \infty$. For a fixed λ, the bias $\mathbb{E}[W_j - \mu]$ increases with ρ. The intuitive explanation is that in steady state, W_1 has a chance of taking a huge value, in which case $W_2, W_3, \ldots$ will also take huge values, and this possibility is ruled out when we fix W_1. □

Example 5.11 Let $\{Y_j,\, j \geq 0\}$ be an irreducible, discrete-time Markov chain evolving over the finite state space $\{1, \ldots, k\}$, with transition matrix $\boldsymbol{P}$ and equilibrium probability vector $\boldsymbol{\pi}$ (see Section A.18). That is, $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)$ is a row vector such that $\boldsymbol{\pi} = \boldsymbol{\pi}\boldsymbol{P}$, $\pi_1 + \cdots + \pi_k = 1$, and $\pi_i \geq 0$ for each i. Assume that the initial state is selected from this equilibrium distribution, i.e., $\mathbb{P}[Y_0 = i] = \pi_i$. Suppose that a cost c_i is incurred each time the chain visits state i and let $\boldsymbol{c} = (c_1, \ldots, c_k)^{\mathsf{t}}$. Let $\boldsymbol{D}$ be the diagonal matrix with diagonal elements $\pi_1, \ldots, \pi_k$ and let $\boldsymbol{\Pi}$ be the square matrix with each line equal to the vector $\boldsymbol{\pi}$.

For $j \geq 1$, the cost at step j is $C_j = c_{X_j}$ and $\mu = \boldsymbol{\pi c}$ is the average cost per step in the long run. It can be shown (see, e.g., Steiger and Wilson 2001a) that

$$\sigma^2 \rho_j = \mathrm{Cov}[X_0, X_j] = \boldsymbol{c}^{\mathrm{t}} \boldsymbol{D}(\boldsymbol{P}^j - \boldsymbol{\Pi})\boldsymbol{c}$$

for $j \geq 0$ and

$$\sigma_\infty^2 = \boldsymbol{c}^{\mathrm{t}} \boldsymbol{D}[\boldsymbol{I} - \boldsymbol{\Pi} + 2(\boldsymbol{I} - \boldsymbol{P}\boldsymbol{\Pi})^{-1}(\boldsymbol{P} - \boldsymbol{\Pi})]\boldsymbol{c}.$$

Here, ρ_j converges to 0 at rate $O(\beta^j)$ where $\beta < 1$ is the second largest eigenvalue (in absolute value) of the matrix $\boldsymbol{P}$. This implies that $n\mathrm{Var}[\bar{C}_n] = O(1/(1-\beta))$. ♣ Add details here. □

5.10.5 Generating the initial state from the equilibrium distribution

If one can generate the initial state of the simulation from the equilibrium distribution, as we saw in Examples 5.9, 5.10, and 5.11, then the cost process of interest starts in steady-state and there is no initial bias. For most complex models encountered in practice, generating a state from the equilibrium distribution is extremely hard to achieve. However, recent progress has been made in this direction by exploiting the concept of *coupling* of stochastic process trajectories. Roughly speaking, coupling occurs when two of more sample paths that may have started from different initial states reach the same state at a given point in time.

♣ Perfect simulation via coupling from the past: papers by Propp and Wilson, etc. Permits one to generate a state *exactly* from the steady-state distribution. Add a section on this in Chapter 4, and refer to it here. Give an example with a queueing system.

5.10.6 Initial Bias Detection and Reduction

Suppose that, as is often the case, the process is not stationary from the beginning but only in the asymptotic sense, i.e., $(\mu_i, \sigma_i^2, \rho_{i,i+j}) \to (\mu, \sigma^2, \rho_j)$ as $i \to \infty$. One then faces the problem of trying to detect and reduce the absolute bias $|\mathbb{E}[\bar{C}_n] - \mu|$ or $|\mathbb{E}[V_{N(t)}]/t - \bar{v}|$ associated with the choice of initial state. This problem does not have a satisfactory general solution. It is typically addressed by heuristics and we discuss one of these heuristics in this section. The regenerative approach studied in Section 5.12 circumvents this initial bias problem, but does not remove the bias completely.

Discrete time. We start again with the special case of a *discrete-time cost process* $\{C_i, i \geq 1\}$, for which our aim is to estimate μ, the steady-state average cost per step. Typically, when the initial state is *not* chosen according to the equilibrium distribution, the absolute bias $|\mathbb{E}[C_i] - \mu|$ is bounded by an exponentially decreasing function of i. Theorem 5.12 (iv), for example, provides general ergodicity conditions under which this is true. In particular, if the state evolves as an aperiodic discrete-time Markov chain with countable state space and probability transition matrix $\boldsymbol{P}$, and if the cost at step i is a bounded function of the state, then $|\mathbb{E}[C_i] - \mu| \in O(\beta^i)$ where $\beta < 1$ is the second largest eigenvalue (in absolute value) of the matrix $\boldsymbol{P}$. This implies that $|\mathbb{E}[\bar{C}_n] - \mu| \in O(1/(n(1-\beta)))$.

Suppose that in order to reduce the bias of the steady-state mean estimator, we discard the first n_0 observations of the process and take the average of the next $n - n_0$ observations. The estimator becomes

$$\bar{C}_{n_0,n} = \frac{1}{n - n_0} \sum_{j=n_0+1}^{n} C_j. \tag{5.45}$$

The discarded part of the process (the first n_0 steps) is called the *warm-up* part. If we assume that $|\mathbb{E}[C_i] - \mu| \leq \kappa_0 \beta^i$ for some constants $\kappa_0 < \infty$ and $\beta < 1$, then the (absolute) bias of the estimator (5.45) is bounded as follows:

$$|\mathbb{E}[\bar{C}_{n_0,n}] - \mu| \leq \frac{\kappa_0}{n - n_0} \sum_{i=n_0+1}^{n} \beta^i = \frac{\kappa_0 \beta^{n_0+1}(1 - \beta^{n-n_0})}{(n - n_0)(1 - \beta)}.$$

Assuming that $n \gg n_0$, this bias decreases as

$$O\left(\frac{\beta^{n_0+1}}{(n - n_0)(1 - \beta)}\right) \approx O\left(\frac{\beta^{n_0}}{n(1 - \beta)}\right)$$

as a function of n_0 and n, i.e., exponentially in n_0 and linearly in n. However, β may be very close to 1, in which case n_0 must be taken very large for the bias to be negligible. On the other hand, for a given value of n, increasing n_0 increases the variance of $\bar{C}_{n_0,n}$. Finding an n_0 that approximately minimizes the MSE is generally too difficult. One must settle with heuristics. Since the variance is in $O(1/(n - n_0))$ whereas the squared bias is in $O(\beta^{2(n_0+1)}/[(n - n_0)(1 - \beta)]^2)$, the contribution of the squared bias to the MSE eventually becomes negligible compared with that of the variance when n gets large. These asymptotics cannot be used to find the optimal n_0 in practical applications because the hidden constants in the $O(\cdot)$ notation are unknown.

Continuous time. For the general case of an infinite-horizon average *cost per unit of time*, $\bar{v}$ in (2.79), we can simulate up to a fixed time horizon t (instead of a fixed number of observations n) and discard the observations up to a fixed *warm-up time* $t_0 < t$. The truncated time-average estimator is then

$$\frac{V_{N(t)} - V_{N(t_0)}}{t - t_0} = \frac{1}{t - t_0} \sum_{j=N(t_0)+1}^{N(t)} C_j. \tag{5.46}$$

Under reasonable conditions (e.g., $N(t)/t$ converges to a positive constant with probability 1 when $t \to \infty$), the bias typically behaves in a similar way as a function of t and t_0 as it behaves in the discrete-time case as a function of n and n_0, i.e., it decreases as $O(\tilde{\beta}^{t_0}/[t(1 - \tilde{\beta})])$ for some constant $\tilde{\beta} < 1$.

Heuristics. In practice, we need a concrete way of selecting n_0 (or t_0). Two questions arise: (1) How should we choose n_0? (2) For a given pair (n_0, n), how can we test if the initial bias that remains is negligible? All available methods that (partially) answer these questions are heuristic. It seems impossible to design an efficient universal procedure.

Example 5.12 Each day, our nuclear plant can blow up, with probability 10^{-15}. As long as it does not blow up, we pay 0 per day, but when it blows up, we pay 1 per day for all the following days, forever. We simulate the plant in discrete-time, day by day. In this case, most procedures will, with high probability, choose n_0 and n less than the blowing-up time and return 0 as an estimate of the steady-state average cost. □

We now describe an heuristic rule proposed by Welch (1983) to estimate a time n_0 where the effect of the initial bias seems to have dissipated. The basic idea is to evaluate, graphically, when the average cost process appears to flat out as a function of time. This is typically difficult to see by looking at a single sample path, because of the noise (or variability) in the trajectory. The trick is to smooth out the process in two ways: (a) by averaging many sample paths and (b) by taking a moving average of successive observations in time, and replacing the process by this moving average. Averages tend to be much less noisy than individual observations. We explain the procedure in the discrete-time setup. It is easy to generalize to continuous time.

Welch heuristic for initialization bias 1. Make k independent simulation runs, each of length n_1. Let C_{ij} be the observation (cost) at step j of replication i.
2. Define $\bar{C}_j = \sum_{i=1}^{k} C_{ij}/k$, the average for the cost at step j, over the k runs. Note that this $\bar{C}_j$ has the same expectation and less variance than C_{ij}.
3. To smooth out high frequency oscillations (and keep low frequencies) in the process $\{\bar{C}_j,\, j \geq 1\}$, compute a moving average, with window size w:

$$\bar{C}_j(w) = \frac{1}{2w+1} \sum_{s=-w}^{w} \bar{C}_{j+s}, \qquad \text{for } j = w+1, \ldots, n_1 - w.$$

4. Plot $\bar{C}_j(w)$ versus j and let n_0 be the value of j at which the process appears to have (reasonably) converged. (This is of course a matter of subjective judgment.)

Recommendations: Take k at least 5 or 10, and a very large n_1. Experiment with different values of w (like for choosing the widths of the rectangles when drawing an histogram). If the result is not satisfactory, increase k and/or n_1 as needed. Numerical illustrations are give by Law and Kelton (2000).

Statistical tests for detecting whether or not there remains significant bias due to the initial state have been proposed in the literature. See, e.g., Cash et al. (1992) and other references given there. The idea is to perform a formal test of hypothesis on $\mathcal{H}_0$: "$C_{n_0+1}, \ldots, C_n$ all have the same expectation." For example, certain tests compare the variability of the process over its first portion, say from steps n_0 to n_0', to the variability over the second portion, from n_0' to n, using an F statistic. The null hypothesis $\mathcal{H}_0$ is rejected if the first portion has significantly more variability. Different ways of measuring the variability give rise to different tests.

5.10.7 Truncated Horizon: One long run or multiple runs

After n_0 (or t_0) has been selected, one must perform the *production runs* per se. The general framework is: Make k independent runs of lengths $T_1, \ldots, T_k$, where the time horizons T_i are either deterministic or random. For run i, remove the warm-up part of length t_0 and compute the time-average over the remaining part,

$$X_i = \frac{1}{T_i - t_0} \sum_{j=N(t_0)+1}^{N(T_i)} C_j.$$

An overall estimator of the steady-state mean is then $\bar{X}_k = (1/k)\sum_{i=1}^{k} X_i$, and a confidence interval can be computed by considering these X_i's as i.i.d. observations. This is the *replication-deletion approach*.

The remainder of the discussion is in terms of the discrete-time framework, where

$$X_i = \frac{1}{n - n_0} \sum_{j=n_0+1}^{n} C_j$$

for some constants $n_0 < n$. For a given computing budget kn, how should we choose k? Typically, $k = 1$ gives the best deal, because there is only one warm-up piece to discard (e.g., Exercise 1.32). It is more difficult to obtain a good variance estimator (for computing a confidence interval) when $k = 1$ than when $k \gg 1$, but this is feasible. We explain how in Section 5.11.

There are cases where $k > 1$ wins due to the conjuncture of slowly diminishing positive autocorrelations between the C_j's and a quickly vanishing bias (see Whitt 1991). The intuitive explanation is that independent observations provide more information than positively correlated ones, so multiple runs beat a single run in this situation as long as the bias is not important enough to offset the variance reduction provided by the independence.

Example 5.13 As an extreme worst-case example, consider a process $\{C_i,\, i \geq 1\}$ for which $C_1 = 1$ with probability p and $C_1 = 0$ with probability $1 - p$, and $C_{i+1} = C_i$ for all $i > 0$. The expected average cost is p and one has $\mathbb{E}[C_i] = p$ for all i (no bias). Here, for a total computing budget of B observations, it is optimal to make $k = B$ runs of length 1. □

Example 5.14 Suppose that $|\mathbb{E}[C_j] - \mu| = \kappa_0 \beta^j$ and $\mathrm{Cov}[C_i,\, C_{i+j}] = \sigma^2 \rho_j = \sigma^2 \alpha^j$ for $j \geq 0$, where $\beta < 1$, $0 < \alpha < 1$, $\kappa_0 > 0$, and $\sigma^2 > 0$ are constants. Our computing budget B allows for k runs of length $n = B/k$ and we want to choose k to minimize

$$\begin{aligned} &\mathrm{MSE}[\bar{X}_k] \\ &= \left(\frac{\kappa_0(\beta^{n_0+1} - \beta^{n+1})}{(n - n_0)(1 - \beta)}\right)^2 + \frac{\sigma^2}{(n - n_0)k}\left(1 + 2\sum_{j=1}^{n-n_0-1} \frac{\alpha^j (n - n_0 - j)}{n - n_0}\right). \end{aligned}$$

The optimal k in this case can be larger than 1 if β and κ_0 are small and if σ^2 and α are large. □

When Assumption 5.1 is in force, the CLT guarantees that $X_1, \ldots, X_k$ are i.i.d. normal when $n \to \infty$ for n_0 fixed. The bias $\mathbb{E}[X_i] - \mu$ also vanishes when $n \to \infty$. On the other hand, if n_0 and n are fixed and $k \to \infty$, the bias remains and the CLT becomes:

$$\frac{\sqrt{k}(\bar{X}_k - \mu)}{\sigma_{n_0,n}} \;\Rightarrow\; N(0,1) + \frac{\sqrt{k}\beta_{n_0,n}}{\sigma_{n_0,n}} \tag{5.47}$$

where $\beta_{n_0,n} = \mathbb{E}[X_i] - \mu$ and $\sigma^2_{n_0,n} = \mathrm{Var}[X_i]$. Computing a confidence interval by assuming that the X_i are i.i.d. normal with mean μ is asymptotically valid only if $\sqrt{k}\beta_{n_0,n}/\sigma_{n_0,n} \to 0$ when $k \to \infty$. In other words, n must increase with k at a fast enough rate so that the bias converges to zero faster than the standard deviation of $\bar{X}_k$ (which is proportional to the average width of a confidence interval).

Example 5.15 Consider again Example 5.14, where

$$\beta_{n_0,n} = \frac{\kappa_0(\beta^{n_0+1} - \beta^{n+1})}{(n-n_0)(1-\beta)} \;\in\; O\left(\frac{\beta^{n_0}}{n-n_0}\right)$$

and

$$\frac{\sigma^2_{n_0,n}}{k} = \frac{\sigma^2}{(n-n_0)k}\left(1 + 2\sum_{j=1}^{n-n_0-1}\frac{\alpha^j(n-n_0-j)}{n-n_0}\right) \;\in\; O\left(\frac{1}{(n-n_0)k}\right),$$

so that

$$\frac{\sqrt{k}\;\beta_{n_0,n}}{\sigma_{n_0,n}} \;\in\; O\left(\sqrt{k/(n-n_0)}\;\beta^{n_0}\right).$$

The latter converges to zero when $k \to \infty$ if and only if $\ln(k/(n-n_0))/2 + n_0 \ln(\beta) \to -\infty$, if and only if

$$\frac{\ln[k/(n-n_0)]}{2\ln\beta} + n_0 \to \infty \tag{5.48}$$

(observe than $\ln\beta < 0$). Fishman (2001), in his theorems 6.4 and 6.1 (ii), gives the *sufficient condition* $n_0/\ln[k/(n-n_0)] \to \infty$, which is a bit stronger than needed. In the case where n_0 is fixed, (5.48) holds if and only if $k/n \to 0$. □

It is common practice to use the *same simulation* runs for computing the X_i's and for determining n_0. Formally, this is incorrect; the production runs that compute the X_i's should be independent of the pilot runs made to determine n_0. Moreover, $k > 1$ is preferred in the Welch procedure, whereas $k = 1$ is usually preferred for the production run(s). From a practical viewpoint, however, using the same runs may be acceptable when $n_0 \ll n$.

5.11 Confidence intervals using a single long run

Suppose we perform a single long simulation run and the estimator $\bar{V}$ of the average cost $\bar{v}$ has the form (5.45) or (5.46). Assuming that this estimator is approximately normally distributed and that the initialization bias that remains can be neglected, we now need a good estimator of $\mathrm{Var}[\bar{V}]$. The quality of this variance estimator can be measured by its MSE

(i.e., variance plus squared bias), assuming that its computing cost is negligible compared with the cost of performing the simulation. In this section, we shall assume that the cost process $\{C_j,\, j \geq 1\}$ is stationary, with mean $\mathbb{E}[C_j] = \mu$. This should be approximately true if we consider the process only after a sufficiently long warmup. The mean μ is estimated by the empirical mean

$$\bar{C}_n = \frac{1}{n}\sum_{i=1}^{n} C_i,$$

where n is the length of the truncated horizon.

5.11.1 Batch Means

The simplest and most popular approach for estimating the variance and computing a confidence interval in the context of a single run is the *batch means* method. The idea is to regroup the (correlated) observations into large batches so that the batch means (the averages within the different batches) are approximately independent and normally distributed.

[7] More specifically, in the discrete-time case, the n observations are divided into k *batches* of size $\ell = n/k$ (we assume that k divides n). The average cost over the ith batch of ℓ successive observations,

$$X_i = \frac{1}{\ell}\sum_{j=\ell(i-1)+1}^{\ell i} C_j,$$

is called the ith *batch mean*, and the global average cost (the *grand mean*) is

$$\bar{X}_k = \frac{1}{k}\sum_{i=1}^{k} X_i = \bar{C}_n.$$

If the average is with respect to (continuous) time, the costs are incurred at random times, and the simulation ends at a deterministic time T, one divides the time interval $(0, T]$ into k equal segments $(T_{i-1}, T_i]$, where $T_i = T/k$, for $i = 1, \ldots, k$. The average cost per unit of time over the ith interval (the ith batch mean) is

$$X_i = \frac{k}{T}\sum_{j=N(T_{i-1})+1}^{N(T_i)} C_j$$

and the average cost over the entire interval $(0, T]$ is

$$\bar{X}_k = \frac{1}{k}\sum_{i=1}^{k} X_i = \frac{1}{T}\sum_{j=1}^{N(T)} C_j.$$

In its simplest form, the batch means method *assumes* that the X_i's are i.i.d. normally distributed with mean $\bar{v}$, and computes a confidence interval for $\bar{v}$ under this assumption.

[7] From Pierre: Should insist more on the fact that the batching is only to estimate the variance, not the mean.

Under reasonable conditions (see, e.g., Glynn and Iglehart 1990), this assumption is almost true when the size of the intervals, $\ell = (T - t_0)/k$, is large enough. In practice, one often expects that the X_i's are slightly positively correlated, and that the coverage probability of the confidence interval is somewhat less than the nominal value $1 - \alpha$. It is sometimes recommended to choose T as large as possible and k no more than 30.

The estimator of $\mathrm{Var}[\bar{X}_k]$ under the batch means setting is

$$\frac{S_k^2}{k} = \frac{1}{k(k-1)} \sum_{i=1}^{k} (X_i - \bar{X}_k)^2,$$

i.e., the empirical variance of the X_i's, divided by k. The confidence interval is computed by assuming that $\sqrt{k}(\bar{X}_k - \mu)/S_k$ has the Student distribution with $k - 1$ degrees of freedom.

A smaller k (larger batches) typically implies less correlated batch means X_i, less bias in the variance estimator, and a distribution of the X_i's closer to the normal. But it also means a larger variance of the variance estimator S_k^2/k, giving more noisy confidence intervals for $\bar{v}$. There is thus a compromise to be made and the optimal k (which minimizes the MSE of S_k^2/k) depends on the simulation length $T - t_0$ and also on the problem (via the correlation structure between the successive observations). The optimal k as a function of T is hard to find in practice. A number of heuristics have been proposed to *determine k adaptively* during the simulation. Some are implemented in public-domain software (Fishman 2001, Steiger and Wilson 2000).

To simplify things in what follows, we shall assume a discrete-time framework with $t_0 = 0$, $T = n = k\ell$, and no initial bias. Note that both k and ℓ must increase fast enough with n if we want the variance estimator to be consistent. Rigorous theoretical analyzes of these issues can be found in Glynn and Iglehart (1990) and Damerdji (1994), in the discrete-time setup. Sufficient conditions on the growth rates of k and ℓ to guarantee that $nS_k^2/k \to \sigma_\infty^2$ with probability 1 when $n \to \infty$ are given by Damerdji (1994).

It is also shown in Song and Schmeiser (1995) and Chien, Goldsman, and Melamed (1997) that under appropriate conditions (if the process $\{C_j,\, j \geq 1\}$ is ϕ-mixing), one has $\mathbb{E}[nS_k^2/k] - \sigma_\infty^2 \in O(1/\ell)$ and $\mathrm{Var}[nS_k^2/k] \in O(1/k)$. Based on an analysis of Chien et al. (1989), Fishman (2001) suggests that both k and ℓ should grow as $O(1/\sqrt{n})$, because this gives the fastest convergence rate of $\sqrt{k}(\bar{X}_k - \mu)/S_k$ to the $N(0, 1)$ distribution.

Fishman (1998) proposes a public-domain software program called LABATCH.2 for choosing k dynamically as a function of n using one of the two rules LBATCH and ABATCH. The ABATCH rule starts by running the simulation with fixed values of n and k. At each step, it tests the hypothesis $\mathcal{H}_0$ that $X_1, \ldots, X_k$ are independent, then doubles the value of n; if $\mathcal{H}_0$ is not rejected, k is multiplied by $\sqrt{2}$, otherwise k remains unchanged. The LBATCH rule operates similarly, except that whenever $\mathcal{H}_0$ is not rejected, k remains unchanged forever. The independence of the batch averages are tested using the von Neumann ratio (Fishman 2001, page 260).

Steiger and Wilson (2001b) propose an alternative procedure called ASAP which differs from ABATCH in the following way: when the independence test fails, ASAP applies the Shapiro-Wilk test of multivariate normality to the batch means, and if the latter test passes it delivers a confidence interval with a correlation correction based on an inverted Cornish-Fisher expansion of the studentized statistic $\sqrt{k}(\bar{X}_k - \mu)/S_k$ to which an ARMA times

series model is adjusted. This takes advantage of the fact that approximate normality is sometimes achieved at smaller batch sizes than approximate independence. The correlation adjustment improves the validity of the confidence interval when some remaining dependence is detected. The procedure returns a confidence interval with the required precision if it can, otherwise it returns the estimated number of additional batches that the user should collect. An improved version of the procedure, called ASAP2, is described by Steiger et al. (2002). It uses an enhanced version of the multivariate normality test and disregards correlation tests. It starts with 256 batches, multiplies the batch size by $\sqrt{2}$ at each step until the normality test is passed, then fits an AR(1) time series model to the batch means and computes a correlation-adjusted confidence interval using an inverted Cornish-Fisher expansion as in ASAP. Computational experiments in a variety of settings (Steiger and Wilson 2000, Steiger et al. 2002) suggest that the method generally performs very well compared with other algorithms, including LBATCH and ABATCH.

♣ Variants: Overlapping batch means, weighted batch means. For more on batch means, including asymptotic analysis and automatic batching methods, see Alexopoulos (1998), Alexopoulos and Kim (2002), Fishman (1998), Fishman (1996), Steiger and Wilson (2000) and other references therein.

5.11.2 Spectral Analysis

We explain this method in the discrete-time context. Suppose that the cost process $\{C_i, i \geq 0\}$ is weakly stationary, with $\mathbb{E}[C_i] = \mu$ for all i. This implies, in particular, that the initial bias has been removed or is negligible, so we can forget about the warm-up (to simplify the notation). As usual, we estimate μ by the grand mean

$$\bar{C}_n = \frac{1}{n}\sum_{i=1}^{n} C_i,$$

where n is the length of the truncated horizon. The *spectral method* tries to estimate the variance constant via the formulas (5.33) and (5.34).

A naive idea might be to just replace the variance σ^2 and the autocorrelations ρ_k in these formulas by their sample counterparts S_n^2 and

$$\hat{\rho}_k = \frac{1}{(n-k)S_n^2}\sum_{i=1}^{n-k}(C_i - \bar{C}_n)(C_{i+k} - \bar{C}_n),$$

to obtain the variance estimator

$$\widehat{\mathrm{Var}}[\bar{C}_n] = \frac{S_n^2}{n}\left(1 + 2\sum_{k=1}^{n-1}(n-k)\hat{\rho}_k\right). \tag{5.49}$$

Unfortunately, this does not work because this estimator is always equal to 0 (Exercise 5.17). In fact, the $\hat{\rho}_k$'s are very bad estimators of the ρ_k's for the large values of k, and they cancel out the contribution of the earlier terms.

A second idea is to assume that the short-lag autocorrelations are really the most important and that the long-lag ones are 0. Then, replace the factors $(n-k)$ that multiply the $\hat{\rho}_k$'s in (5.49) by other *weights* w_k and stop the sum at some $m \ll n$. This gives the (more general) estimator

$$\widehat{\mathrm{Var}}[\bar{C}_n] = \frac{S_n^2}{n}\left(1 + 2\sum_{k=1}^{m} w_k\hat{\rho}_k\right). \tag{5.50}$$

Then, under certain assumptions, $(\bar{C}_n - \mu)/(\widehat{\mathrm{Var}}[\bar{C}_n])^{1/2}$ has approximately the Student distribution with $(8/3)(n/m)$ degrees of freedom. A recommended choice for w_k is the *Tukey-Hanning window*, $w_k = (1+\cos(\pi k/m))/2$. For the details, see the references given in Bratley, Fox, and Schrage (1987) and Alexopoulos (1998). This is related to the classical topic of *spectral analysis* of stochastic processes.

Another idea might be to first batch the observations as in the batch means method, and then apply the spectral method to the batch means instead of to the original observations. It is highly likely that between the batches, only the autocorrelations of lag 1 will be non-negligible, so one may just take $m = 1$. In this context, it could also make sense to take more (i.e., smaller) batches than usual.

5.11.3 Standardized time series

♣ See Glynn and Iglehart (1990), Tokol et al. (1998), and Foley and Goldsman (2000).

5.12 Regenerative Simulation

The theory of regenerative processes offers powerful tools to verify whether the limits (2.79)–(2.82) exist and to express them as ratios of finite-horizon expectations. Our coverage here is inspired by Sigman and Wolff (1993). We begin with classical regenerative processes and their properties, then briefly discuss more general types of regenerative processes such as Harris-recurrent processes.

5.12.1 Classical Regenerative Processes

Definition 5.3 A stochastic process $\{Y(t),\ t \geq 0\}$ is called *regenerative* (in the classical sense) if there exists a random variable $\tau_1 > 0$ such that the process $\{Y(\tau_1 + t),\ t \geq 0\}$ is stochastically equivalent to $\{Y(t),\ t \geq 0\}$, and is independent of τ_1 and of $\{Y(t),\ t < \tau_1\}$. The random variable τ_1 is called a *regeneration epoch*, and the time-interval $(0, \tau_1]$, together with the trajectory of the process over this interval, is called a *regenerative cycle*. These definitions stand for the discrete case as well; t and τ_1 are then restricted to the set of integers. [8] □

The term *stochastically equivalent* means that its behavior is governed by the same probability laws. It is intuitively clear that if $\{Y(t),\ t \geq 0\}$ is regenerative, then $\{Y(\tau_1 + t),\ t \geq 0\}$ is also regenerative with some regeneration epoch τ_2, and so on. We then have an infinite sequence of regeneration epochs $0 = \tau_0 < \tau_1 < \tau_2 \ldots$ and regenerative cycles. All those regenerative cycles are i.i.d. in the sense that the trajectory of $\{Y(t), t \geq 0\}$ over any such cycle is stochastically equivalent to that over any other cycle, and also independent. In that sense, at each τ_j, the process starts afresh. These epochs τ_j need not be stopping times. If $\mathbb{E}[\tau_1] < \infty$, the process is also called *positive recurrent*.

Sometimes, the cycles are independent and stochastically equivalent according to the definition, except that the first cycle is not equivalent to the remaining ones. Then, the process is regenerative only from time τ_1, and we call it *delayed regenerative*. If the process is positive recurrent, that little "defect" in the first cycle does not affect the time-average properties, and can be neglected in the long run.

Proposition 5.14 *Functions of regenerative processes are regenerative with the same regeneration epochs. That is, if $\{Y(t),\ t \geq 0\}$ is a regenerative process with regeneration points $\tau_1, \tau_2, \ldots$, and if $\{C(t),\ t \geq 0\}$ is another process defined by $C(t) = f(Y(t))$ for some (measurable) function f, then this second process also regenerates at $\tau_1, \tau_2, \ldots$.*

Example 5.16 Let $\{Y_i,\ i \geq 0\}$ be a (discrete) Markov chain defined over the finite state space $\{1, \ldots, K\}$, with transition probabilities $P[j,k] = \mathbb{P}[Y_i = k \mid Y_{i-1} = j]$, and suppose that all the states are accessible from each other. Let $Y_0 = j$. Then this process is positive recurrent regenerative, with regeneration at the epochs i for which $Y_i = j$. If we start from a different state, say $Y_0 = k$, and keep the same definition for the regenerative epochs, we have a delayed regenerative process.

[8] From Pierre: Give the definition for that case.

Now, suppose that all the rows of the matrix P are the same, i.e., $P[j,k]$ is independent of j. Then, the process is regenerative with regeneration epochs at each step ($\tau_i = i$ for each i). This illustrates the fact that the process need not visit a fixed state j at each regeneration epoch. [9] □

Example 5.17 Consider again the $GI/GI/1$ example, assuming that the queue is stable. Recall that the sequence of waiting times $\{W_i,\ i \geq 0\}$ obeys Lindley's equation (1.39). It follows that if $W_1 = 0$ and if the random variables $S_i - A_i$ are i.i.d., $\{W_i,\ i \geq 0\}$ is regenerative with regeneration epochs at each index i such that $W_i = 0$. The process $\{Q(t),\ t \geq 0\}$, giving the number of customers in the queue as a function of time, is also regenerative, with regeneration epochs occuring at the times when a customers arrives while the system is empty. At such times, a new interarrival time and a new service time are generated and the process (stochastically) starts afresh. Observe that the times at which a customer leaves the system empty cannot be taken as regeneration epochs in general, because at such an epoch, the distribution of the time until the next arrival depends on the elapsed time since the last arrival (unless the interarrival times are exponentially distributed); therefore the cycles would not be independent nor stochastically equivalent with that choice. □

Example 5.18 In the call center example, suppose that the center operates for an unlimited sequence of (independent and stochastically equivalent) days. If t denotes the total time elapsed since 0:00 (midnight) the first day and if $Q(t)$ is the number of calls in the queue at time t, then $\{Q(t),\ t \geq 0\}$ regenerates at $\tau_j = t_0 + 24j$ for $j = 1, 2, \ldots$, if t is in hours, where t_0 can be any time outside the operating hours of the center (i.e, if the center opens at 8 AM and closes at 9 PM, t_0 can be any number in $[0, 8] \cup [21, 24]$). On the other hand, if X_j and Z_j denote the number of calls received and the number of abandonments on day j, respectively, then the processes $\{X_j,\ j \geq 0\}$ and $\{Z_j,\ j \geq 0\}$ are both i.i.d., so they are both regenerative with $\tau_j = j$ for each j. □

5.12.2 Renewal Reward Theorem

In the general model of Section 2.19, let the cost process $\{C_i,\ i \geq 0\}$ be regenerative, where τ_j denotes the simulation time at which the jth regeneration epoch occurs. For each $j \geq 1$, let $X_j = V_{N(\tau_j)} - V_{N(\tau_{j-1})}$, the cost incurred during the jth regenerative cycle, and $Y_j = \tau_j - \tau_{j-1}$, the duration of that cycle.

Theorem 5.15 (Renewal reward theorem.) *Suppose that* $\mathbb{E}[Y_j] > 0$ *and* $\mathbb{E}[|X_j|] < \infty$. *Then,*

$$\bar{v} \stackrel{\text{def}}{=} \lim_{t\to\infty} \frac{\mathbb{E}[V_{N(t)}]}{t} = \frac{\mathbb{E}[X_j]}{\mathbb{E}[Y_j]}; \tag{5.51}$$

and

$$\bar{v} \stackrel{\text{w.p.1}}{=} \lim_{t\to\infty} \frac{V_{N(t)}}{t}. \tag{5.52}$$

[9] From Pierre: Check this... with the definition given, we must have $X_{\tau_i} = x_0$.

The proof can be found, e.g., in Wolff (1989). Eq. (5.51) is called the *expected value version* of the renewal reward theorem, while (5.52) is the *sample path version*. The main consequence of the theorem, in the simulation context, is that $\bar{v}$ can be expressed as a ratio of two expectations and that each of these expectations can be estimated by a simulation over a finite (random) horizon.

5.12.3 Confidence Intervals

We saw in Section 5.4.2 how to obtain a confidence interval for a ratio of expectations. In the case where the number n of regenerative cycles is fixed in advance, this can be used directly to compute a confidence interval on $\bar{v}$.

Often, we would rather decide on the total simulation time t, and let the number of cycles be random. Define $M(t) = \sup\{n \geq 0 : \tau_n \leq t\}$, the number of regenerative cycles completed by time t. When time t is reached, a first possibility is to discard the ongoing cycle and use the formulas above with $n = M(t)$. A second possibility is to complete the ongoing cycle, so that $n = M(t) + 1$. Meketon and Heidelberger (1982) have shown (under mild regularity conditions) that the bias is $O(1/t)$ in the first case and $O(1/t^2)$ in the second case, and that the variance is $O(1/t)$ in both cases. Completing the ongoing cycle is thus recommended in general. [10] This is one advantage of the regenerative method over using a deterministic truncated horizon t.

Under the conditions of Section 5.4.2, we have $M(t) \to \infty$, $M(t)\bar{Y}_{M(t)}/t \to 1$, and $(M(t)+1)\bar{Y}_{M(t)+1}/t \to 1$, with probability 1, as $t \to \infty$. Therefore, we can replace n by either $M(t)$ or $M(t)+1$, and $n \to \infty$ by $t \to \infty$, to obtain the following variant of the CLT:

Theorem 5.16 *Under the conditions of Theorem 5.7,*

$$\frac{\sqrt{M(t)}(\hat{\mu}_{M(t)} - \mu)}{\hat{\sigma}_{z,n}} \Rightarrow \frac{\sqrt{t/\bar{Y}_{M(t)}}(\hat{\mu}_{M(t)} - \mu)}{\hat{\sigma}_{z,n}} \Rightarrow \frac{\sqrt{t/\mathbb{E}[Y_1]}(\hat{\mu}_{M(t)} - \mu)}{\hat{\sigma}_{z,n}} \Rightarrow N(0,1) \quad (5.53)$$

as $t \to \infty$, where $\hat{\sigma}^2_{z,n}$ is defined as in (5.19) with $n = M(t)$, and similarly for $M(t)$ replaced by $M(t)+1$.

♣ Should have exercises on this.

What if we replace the mean estimator $\hat{\mu}_{M(t)}$ by the total cost up to time t, divided by t, where t is deterministic and is generally not a regeneration point? Under mild additional conditions on the cost structure (e.g., it suffices that all the costs are non-negative), the CLT still holds for $t \to \infty$. In the latter case, we obtain a re-expression of Theorem 5.13 in Section 5.10.1, with $\tilde{\sigma}^2_\infty = \lambda\sigma^2_\infty = \sigma^2_z/\mathbb{E}[Y_1]$. In this context, the regenerative method can be reinterpreted as a way of estimating the variance constant $\tilde{\sigma}^2_\infty$, via the estimator $\hat{\sigma}^2_{z,n}/\bar{Y}_{M(t)}$.

♣ Give more details and state a theorem.

[11] There are a number of ways of *reducing the bias* of a ratio estimator from $O(1/n)$ to $O(1/n^2)$ when n (the number of cycles) is fixed. One of them is the jackknife (Efron 1982,

[10] From Pierre: Not quite sure if this is good... What about the MSE? The variance constant?

[11] From Pierre: The following should be moved to the section on ratio estimation and delta theorem.

Shao and Tu 1995). ♣ Define and explain in a more general setting (earlier). Typically, it gives rise to an estimator with less bias but more variance, so the MSE is not necessarily reduced, and the natural variance estimator under jackknifing tends to underestimate the variance. Bootstrap methods offer a good way of reducing the MSE and computing confidence intervals having better coverage than those provided by the classical method (see Choquet, L'Ecuyer, and Léger 1999).

♣ Add examples.

5.12.4 Discounted Costs

Suppose now that the costs are discounted at rate $\rho > 0$ and that we want to estimate v_ρ^∞, defined in (2.84), for some initial model state $\mathcal{S}_0$. If we assume that $v_\rho^\infty = \mathbb{E}[V_\rho^\infty]$ and that the process $\{C_i,\, i \geq 0\}$ is regenerative as above, we obtain

$$\begin{aligned} V_\rho^\infty &= V_{\rho,N(\tau_1)} + \sum_{i=N(\tau_1)+1}^{\infty} e^{-\rho t_i} C_i \\ &= V_{\rho,N(\tau_1)} + e^{-\rho\tau_1} \sum_{i=N(\tau_1)+1}^{\infty} e^{-\rho(t_i-\tau_1)} C_i. \end{aligned}$$

Taking expectations yields

$$\mathbb{E}[V_\rho^\infty] = E[V_{\rho,N(\tau_1)}] + E\left[e^{-\rho\tau_1}\right] \mathbb{E}[V_\rho^\infty].$$

That is,

$$v_\rho^\infty = \mathbb{E}[V_\rho^\infty] = \frac{E[V_{\rho,N(\tau_1)}]}{1 - E\left[e^{-\rho\tau_1}\right]}.$$

Again, we are back to a ratio of two finite-horizon expectations. It should be pointed out that these expectations, as well as the definition of the regeneration epochs, generally depend on the initial state $\mathcal{S}_0$.

♣ Examples.

5.12.5 Harris-recurrent and m-dependent regenerative processes

A more general definition of regeneration removes the requirement that $\{Y(\tau_1 + t),\ t \geq 0\}$ is independent of $\{Y(t),\ t < \tau_1\}$:

Definition 5.4 For an integer $m \geq 0$, a process is called *m-dependent regenerative* if there exists a sequence of random variables $0 \leq \tau_1 \leq \tau_2 \cdots$ such that for each $j \geq 1$, the process $\{Y(\tau_j + t),\ t \geq 0\}$ is stochastically equivalent to $\{Y(t),\ t \geq 0\}$, and $\{Y(\tau_{j+m} + t),\ t \geq 0\}$ is independent of $\{Y(t),\ t < \tau_j\}$. □

It turns out that by relaxing the independence assumption in this way, a much larger class of applications can be covered, and most interesting properties of regenerative processes

can still be derived. However, additional assumptions are required for some of the properties, namely to make sure that the process is ergodic. We refer the reader to Asmussen (1987) and Meyn and Tweedie (1993). For $m = 0$, we are back to the classical definition of a regenerative process. The important class of processes called *Harris recurrent*, which covers essentially all real-life situations for which it makes sense to consider steady-state measures, are one-dependent regenerative processes. This can be shown by applying a *splitting technique* due to Athreya and Ney (1978) (see also Meyn and Tweedie 1993).

♣ Properties of one-dependent regenerative processes: renewal reward theorem, etc. Harris recurrent models and splitting technique. Coupling.

♣ Examples: $GI/GI/c$ queue, storage process, etc.

♣ More general ergodic models (Meyn and Tweedie 1993). See Henderson and Glynn. Links with Lyapunov.

5.13 Exercises

5.1 In Section 5.2.1, a $100(1-\alpha)\%$ confidence interval for μ is $(I_1, I_2) = (\bar{X}_n + \delta,\ \bar{X}_n - \delta)$, where $\delta = t_{n-1,1-\alpha/2} S_n/\sqrt{n}$.

(a) Explain why it is correct to say that $\mathbb{P}[I_1 < \mu < I_2] = 1 - \alpha$. Hint: What are the random variables in that statement?

(b) Suppose we obtain the values $I_1 = 3.154$ and $I_2 = 3.562$. Explain why it is *incorrect* to say that $\mathbb{P}[3.154 < \mu < 3.562] = 1 - \alpha$. Hint: Where is the random variable here?

5.2 In Example 1.29 (Section 1.4.4), suppose that μ is small but unknown, and is estimated by $\bar{X}_n$, the average of $X_1, \ldots, X_n$, which are n i.i.d. copies of X. Let $Y = X_1 + \cdots + X_n$. For $n = 10000$, suppose that we have obtained $Y = 2$.

(a) Compute a 95% confidence interval on μ using the binomial distribution for Y.

(b) Same thing, using the Poisson approximation for the distribution of Y.

(c) Same thing, but using Hoeffding inequality, assuming that μ is known a priori to be less than $1/1000$.

(d) With such a large value of n, why not compute a confidence interval by assuming that $\bar{X}_n$ has approximately the normal distribution?

5.3 In the previous exercise, with the method of Section 5.2.3, the coverage probability is at least 95% and its exact value depends on the value of μ. Explain how we could estimate this coverage probability by simulation for different values of μ. Design a functional estimation method along the lines of Exercise 1.39.

5.4 Suppose we have n i.i.d. observations $X_n, \ldots, X_n$ and want to compute a confidence interval on the mean μ. We can regroup these observations into k batches of n/k observations each, consider the batch means $Y_1, \ldots, Y_k$ as normally distributed, and compute a confidence interval by assuming that $\bar{Y}_k$ has a Student$(k-1)$ distribution.

Perform the following experiment. Generate $n = 200$ i.i.d. exponentials with mean 1, and compute a 95% confidence interval as above with $k = 10, 20, 100, 200$. Repeat this experiment

$r = 1000$ times. For each value of k, estimate from your data: (i) The coverage probability of the confidence interval; (ii) the mean and the variance of the confidence interval half-width. Make also an histogram of the distribution of half-widths, for each k, and compare. What are your conclusions regarding the choice of k? Explain the advantages and disadvantages of taking a small k.

5.5 Let X be a random variable with unknown distribution, taking its values in the interval $[0, 10]$. Suppose we know for sure that $\mu = \mathbb{E}[X]$ is less than $\nu_1 < 1/2$. We want to compute a 95% *left-sided confidence interval* on μ; that is, we seek $\epsilon > 0$ such that $\mathbb{P}[\mu \leq \bar{Y}_n + \epsilon] \geq 1 - \alpha$, where $\bar{Y}_n$ is the sample mean of n i.i.d. copies of Y.

(a) Give an expression for such an ϵ based on Theorem 5.3.

(b) What is the confidence interval if $\nu_1 = 0.2$, $n = 1000$, and $\bar{Y}_n = 0.08$?

5.6 The aim of this exercise is to experiment with the robustness of confidence intervals on the variance based on the normal approximation as explained in Section 5.4.3.

(a) Repeat the following experiment 1000 times, independently:

1. Generate $n = 100$ i.i.d. random variables with the exponential distribution with mean 1.

2. Pretend that you don't know the true variance σ^2 of these observations, and compute a confidence interval with confidence level $1 - \alpha = 0.95$ for σ^2 by assuming (wrongly) that $(n-1)S_n^2/\sigma^2$ has the chi-square distribution with $n - 1$ degrees of freedom.

3. Check if the confidence interval covers the true value $\sigma^2 = 1$ and compute its width W.

The proportion $\hat{p}$ of the 1000 confidence intervals that contain 1 is an estimator of the exact coverage probability p of the confidence interval. Compute a 95% confidence interval for p. Compute also a 95% confidence interval on the average width, $\mathbb{E}[W]$, of the confidence interval on σ^2. Plot, on a single chart, the empirical cdf of (i) the left side of the confidence interval, (ii) the right side of the interval, (iii) the variance estimator, and (iv) the cdf of $Y/(n-1)$ where Y has the chi-square distribution with $n - 1$ degrees of freedom. Note that if $(n-1)S_n^2/\sigma^2$ had exactly the $\chi^2(n-1)$ distribution, then S_n^2 would have exactly the same distribution as $Y/(n-1)$. Discuss what you observe.

(b) Repeat the same exercise with $n = 1000$, and compare. Is the approximation improving when n increases? You may also try larger values of n.

(c) Redo the same experiment as in (a), but this time compute the 95% confidence intervals on σ^2 (in step 2) via the basic nonparametric bootstrap method, with $m = 1000$ resamples, and without the plots. Discuss you results.

5.7 We want to estimate the expected average cost per day for some system, assuming that the days are independent. We have already simulated $n = 1000$ days and obtained the sample mean $\bar{X}_n = 1073.5$ and the sample variance $S_n^2 = 5890.0$. How many additional simulation runs should we make if we want a 95% confidence interval with half-width at most 2?

5.8 You will perform a simple experiment with sequential estimation. Let $X \sim \mathsf{Weibull}(\alpha, 1)$ and pretend you do not know $\mu = \mathbb{E}[X]$ and want to estimate it by simulation, with sequential estimation. The target confidence level and interval half-width are $1-\delta$ and $w/2$, respectively. Compute a 95% confidence interval for μ by sequential estimation, with $n_0 = 20$, and check if the interval covers μ. Repeat this $m = 5000$ times and estimate the true coverage probability by the proportion of those m confidence intervals that cover μ. Give a confidence interval on that proportion.

Repeat this experiment with all nine combinations of the following values: $\alpha = 0.2$, 1.0, 3.0, and $w/\mu = 0.01$, 0.1, and 0.3.

5.9 We have a portfolio of five assets whose values are assumed (somewhat simplistically) to follow independent geometric Brownian motions with parameters $\sigma = 0.3$, $r = 0.05$, and $S(0) = 50$. We have one unit of each asset and we want to estimate the value-at-risk (VaR) x_p for time $T = 0.1$, with $p = 0.01$. The loss in this case is $L = \sum_{j=1}^{5}(50 - S_j(T))$. Recall that x_p is defined via $\mathbb{P}[L > x_p] = p$.

(a) Compute a 95% confidence interval on x_p, based on a sample of $n = 1000$ independent values of L generated by standard Monte Carlo, with each of the following three methods: (i) the method based on the binomial distribution; (ii) the method in which the binomial distribution is approximated by a normal; (iii) the basic nonparametric bootstrap.

(b) Redo the same experiment with $n = 10000$. Compare and discuss the results.

(c) What if you are asked to estimate it for $p = 0.001$ instead?

5.10 To compute a confidence interval for the difference between two means (Section 5.5.1), one may decide to use the Welch and the paired-t methods *simultaneously* (assuming that the assumptions of both methods hold), and retain the shortest of the two confidence intervals. This is cheating. Explain why one should expect the true coverage probability to be less than the nominal coverage $1 - \alpha$ if one does this.

5.11 Suppose we estimate a mean μ by the sample mean $\bar{X}_n$ of n identically distributed observations $X_1, \ldots, X_n$, with $\mathbb{E}[X_i] = \mu$ and $\mathrm{Var}[X_i] = \sigma^2$.

(a) If the X_i's are positively correlated, does it give us *more* confidence or *less* confidence in our estimator than if they are independent?

(b) Explain why it is not possible that all pairs (X_i, X_j), $i \neq j$, are highly negatively correlated if $n \geq 3$.

(c) If some pairs are positively correlated and others are negatively correlated, give a necessary and sufficient condition, in terms of the $n(n-1)/2$ correlations, under which $\bar{X}_n$ has less variance than when the X_i's are independent.

5.12 In Section 5.3.2, what happens with the confidence ellipsoid if $\hat{\sigma}_{jk} = 0$ for all $j \neq k$? What happens if $d = 2$ and $\hat{\sigma}_{12} = 1$?

5.13 For Example 5.9, show that $C_j \sim N(\mu, \sigma^2)$, $\rho_j = a_1^j$, and $\sigma_\infty^2 = \sigma^2(1 + a_1)/(1 - a_1)$. Then, verify the Lyapunov conditions of Theorem 5.11.

5.14 For the $M/M/1$ queue in Example 5.10, compute γ for $\rho = 0.2$, 0.5, and 0.8.

5.15 ♣ Construct a concrete example where $\gamma < 0$.

(e.g., inventory system as in Law and Kelton 2000.)

5.16 In the context of estimating the infinite-horizon average cost for a discrete-time process, suppose we adopt the strategy described in Section 5.10.6, using a truncated horizon of n steps, with a warm-up of n_0 steps. Suppose that the variance and absolute bias of the mean estimator thus obtained are bounded by $\kappa_1/(n - n_0)$ and $\kappa_0\beta^{n_0}/(n - n_0)$, respectively, for some constants κ_0 and κ_1. This gives an upper bound on the MSE.

How would you compute the n_0 that minimizes this upper bound? What is the asymptotically optimal n_0 as a function of n, as $n \to \infty$? (In practice, unfortunately, the constant κ_0 and κ_1 are unknown. And even if they were known, this could be used only to minimize the upper bound, not necessarily the MSE.)

5.17 Prove that $\widehat{\text{Var}}[\bar{Y}_n]$ in (5.49) is always 0.

5.18 The purpose of this exercise is to explore, with a simple model, different approaches for computing a confidence interval for the steady-state mean. Consider an $M/M/1$ queue with arrival rate $\lambda = 1$ and service rate $\mu > 1$. The performance measure is the steady-state average waiting time in the queue, $w = \mathbb{E}[W_i]$, assuming that W_1 has the equilibrium distribution given in Example 5.10. We simulate this system for n customers (via Lindley's recurrence) and estimate w by $\bar{W}_n$. To estimate the variance, we first use the batch means method: regroup the W_i's into k batches of n/k observations each and let $X_1, \ldots, X_k$ be the batch means. Then, two possible approaches are: (A) assume that the X_i's are i.i.d. normal; or (B) estimate the autocorrelation of lag 1 between the X_i's, ♣ Explain how!

Instead of doing this, one should use the ASAP2 method of Steiner et al. assume that the autocorrelations of lag > 1 are negligible, and use the spectral analysis estimator (5.50) with $m = w_1 = 1$.

(a) Implement these two approaches and try them with $\mu = 1.5$, $n = 10^4$, and $k = 20, 50, 100, 500$, by repeating the following experiment 1000 times:

1. Simulate n customers.
2. Compute an estimate of $\text{Var}[\bar{W}_n]$, and a confidence interval for w, using each of the two method (A) and (B), and each value of k.
3. In each case, check if the confidence interval contains w.

For each case, compute the proportion $\hat{p}$ of the 1000 confidence intervals that contain w and compute a 95% confidence interval on the true coverage probability p of the confidence intervals for w. Compute also a 95% confidence interval on the expected value of the variance estimate computed in step 2. For case (B), compute a 95% confidence interval on the lag-1 correlation, for each value of k, using all the simulation runs.

(b) Compare the two methods (A) and (B), and compare the results with the true variance of $\bar{W}_n$, which can be computed numerically (approximately) via (5.33), (5.34), and the formula given in Example 5.10, or can be estimated with high precision by performing an extremely long simulation run and taking $k \approx 50$ with method (A) or (B).

(c) What would be your conclusion and recommendation? Would it be safe to generalize your conclusions to other types of models?

5.19 Consider a $GI/GI/1$ queue with constant interarrival times, always equal to 10 seconds, and i.i.d. random service times with a mean of 8 seconds. Identify a sequence of epochs that are regeneration points in this model. The times when the system empties are not regeneration points. Why? What distribution of the interarrival times would give a model that regenerates when the system empties?

5.20 Consider 3 FIFO queues in series, e.g., 3 different machines that some manufactured parts must visit, in order. Customers (e.g., the parts) arrive to machine 1 according to a stationary Poisson process with rate 1. After completing their service at machine 1, parts go to machine 2, then to machine 3. Service times are mutually independent across machines and across customers, with distribution functions G_1, G_2 and G_3 at machines 1, 2, and 3, respectively. We are interested in the infinite-horizon steady-state time-average workload (i.e., average number of customers in the system).

(a) Is this a regenerative system? If yes, suggest one possibility for the regeneration epochs. (Of course, one cannot define the regenerative cycles for each machine by considering the 3 machines separately. Why?)

(b) Give a simple *necessary* condition on the G_i's for the regenerative cycles to have finite length. (Hint: the system cannot be regenerative if there is an infinite accumulation of parts.)

(c) Suppose now that the interarrival times are i.i.d. $U(1, 2)$, i.e., uniform over the continuous interval $(1, 2)$, that each of the first two machines has constant service times equal to 1, and that the service times at machine 3 are $U(0.5, 1)$. Is your definition of regeneration epoch suggested in (a) still valid with this new interarrival times distribution? If yes, what is the expected length of a regenerative cycle in this case? Show that we can define the regeneration epochs as the times when a part arrives to machine 2 while machine 3 is idle, and that with this definition the length of a regenerative cycle has finite expectation.

(d) What if the interarrival times are constant, all equal to 1.5? Is the expected length of a regenerative cycle still finite? If not, redefine the regeneration points so that it becomes finite.

5.21 Detail the proof of Theorem 5.16.

6. Efficiency Improvement

6.1 Introduction

Recall our definition of *efficiency* of an estimator X, given in Section 1.4.4:

$$\text{Eff}[X] = \frac{1}{\text{MSE}[X]C(X)}. \tag{6.1}$$

We pointed out the imperfectness of this (and any other) criterion at the beginning of Chapter 5; yet it constitutes a reasonable compromise. Improving efficiency with this criterion means reducing the bias, or the variance, or the computing cost. Often, several alternative estimators are unbiased and have roughly the same computational costs. In that context, improving the efficiency means reducing the variance. For that reason, we often talk of *variance reduction techniques (VRTs)*. However, efficiency can sometimes be improved by increasing the variance (and reducing the computing cost); see Fishman and Kulkarni (1992) and Glynn and Whitt (1992) for examples.

Other introductions to variance reduction can be found in Bratley, Fox, and Schrage (1987), Hammersley and Handscomb (1964), Fishman (1996), and Law and Kelton (2000). Surveys are given by Glynn (1994), Heidelberger (1995), and Wilson (1984).

Small case studies were already given in Sections 1.6 and 1.7 to introduce variance reduction concepts in an intuitive way. We start this chapter with more examples of that flavor. The theory is developed in subsequent sections. We believe it is good for the uninitiated reader to go through concrete illustrations to understand how variance can actually be reduced in simple situations before trying to digest the more general and abstract descriptions of the methods and the relevant mathematical results. The readers who already know about variance reduction and who are mostly interested in the theory can skip the next section.

6.2 Motivating Examples and Heuristics

6.2.1 Variance reduction for the call center example

We return to the call center example of Section 1.12, with the parameters given in Table 1.7, but with the following modification: We assume here that the random busyness factor B is a *discrete* random variable that takes the value b_t with probability q_t, for $t = 1, \ldots, 4$, where the q_t's and b_t's are as follows:

t	1	2	3	4
b_t	0.8	1.0	1.2	1.4
q_t	0.25	0.55	0.15	0.05

We have $\mathbb{E}[B] = 1$ and $\mathrm{Var}[B] = 0.024$. This change is to simplify the stratification method that we will apply in a moment. L'Ecuyer (2005) and L'Ecuyer and Buist (2008) explain how to apply a similar stratification method to the original example, with a gamma-distributed business factor B_i.

We simulate n days of operation of the center. Let $X_i = G_i(s)$ the number of calls who waited less than s seconds on day i and suppose we want to estimate $\mu = \mathbb{E}[G_i(s)]$. In this section, we take $s = 20$. A straightforward unbiased estimator of μ is

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i, \tag{6.2}$$

with variance $\mathrm{Var}[\bar{X}_n] = \mathrm{Var}[X_i]/n$.

We ran a simulation experiment with $n = 1000$. The sample mean and sample variance of the X_i's were $\bar{X}_n = 1518.3$ and $S_n^2 = 21615$, respectively. The estimated variance of $\bar{X}_n$ is thus 21.6. How can we reduce this variance? In what follows, we motivate and try alternative estimators, with empirical results summarized in Table 6.1. These alternatives are introduced here as heuristics, but they are special cases of general methods to be studied later.

We note that these variance estimates are noisy; they differ from the exact variances. To assess the probable error, we can compute a confidence interval for $\sigma^2 = n\mathrm{Var}[\bar{X}_n]$ by making the assumption that $(n-1)S_n^2/\sigma^2$ has approximately the χ^2_{n-1} distribution. We have seen in Section 5.4.3 that under this assumption, for $n = 1000$, a 90% confidence interval on σ^2 would be $[0.930\,S_n^2,\ 1.077\,S_n^2]$, i.e., the relative error on σ^2 is less than 8% with 90% confidence. This analysis applies to all sample variances S_n^2 in Table 6.1.

Indirect estimation. The total number of calls arriving on day i can be written as $A_i = X_i + D_i$ where D_i is the number of calls who waited at least s seconds or were lost by abandonment. We know that $a = \mathbb{E}[A_i] = 1660$, so we can write $\mu = \mathbb{E}[X_i] = \mathbb{E}[A_i - D_i] = a - \mathbb{E}[D_i]$ and μ can be estimated as well by the *indirect estimator*

$$\bar{X}_{\mathrm{i},n} = \mathbb{E}[A_i] - \bar{D}_n = a - \frac{1}{n}\sum_{i=1}^{n} D_i.$$

Since its computing time is essentially the same, this estimator is more efficient than (6.2) if and only if $\mathrm{Var}[D_i] < \mathrm{Var}[X_i]$. *Proving* the latter does not seem easy, but in our experiment with $n = 1000$, we obtained a sample variance of 18389 for the D_i's, which is about 15% smaller than for the X_i's. Thus, $\bar{X}_{\mathrm{i},n}$ seems to improve slightly over $\bar{X}_n$.

Control variable. Another idea is to exploit auxiliary information. For example if the total number of calls A_i is much larger than usual on a given day, we may expect X_i and D_i to overestimate their expectations (or perhaps the opposite if the system is overloaded...). We can make a "correction" to the estimator to take this information into account; for instance replace X_i in (6.2) by

$$X_{\mathrm{c},i} = X_i - \beta(A_i - a)$$

where β is an appropriate constant. Suppose for now that $\beta \geq 0$. When $A_i > a$, the correction term reduces the value of the estimator X_i to make up for the overestimation. When $A_i < a$, it does the opposite. We have $\mathbb{E}[X_{\mathrm{c},i}] = \mathbb{E}[X_i]$ since $\mathbb{E}[A_i - a] = 0$, so $X_{\mathrm{c},i}$ remains an unbiased estimator.

In this context, A_i is called a *control variable (CV)*. We obtain the new estimator

$$\bar{X}_{\mathrm{c},n} = \bar{X}_n - \beta(\bar{A}_n - a), \tag{6.3}$$

with variance

$$\mathrm{Var}[\bar{X}_{\mathrm{c},n}] = (\mathrm{Var}[X_i] + \beta^2 \mathrm{Var}[A_i] - 2\beta \mathrm{Cov}[A_i, X_i])/n.$$

This variance is a quadratic function of β and is minimized by taking

$$\beta = \beta^* = \mathrm{Cov}[A_i, X_i]/\mathrm{Var}[A_i].$$

If we take $\beta = 0$, we get the original estimator $\bar{X}_n$. By taking $\beta = 1$, we recover the indirect estimator $\bar{X}_{\mathrm{i},n}$, because $X_i - (A_i - a) = a - D_i$. Therefore, the CV estimator $\bar{X}_{\mathrm{c},n}$ with the optimal β must perform at least as well as any of these two estimators in terms of variance. We do not know β^* here, because the exact value of $\mathrm{Cov}[A_i, X_i]$ is unknown, but it can be estimated. The variance of A_i can be computed exactly; conditional on B_i, $A_i \sim \mathsf{Poisson}(1660 B_i)$, and therefore:

$$\begin{aligned} \mathrm{Var}[A_i] &= \mathrm{Var}[\mathbb{E}[A_i|B_i]] + \mathbb{E}[\mathrm{Var}[A_i|B_i]] = \mathrm{Var}[1660 B_i] + \mathbb{E}[1660 B_i] \\ &= 1600^2 \mathrm{Var}[B_i] + 1660 \mathbb{E}[B_i] = 67794.4. \end{aligned}$$

To estimate $\mathrm{Cov}[A_i, X_i]$ (and β^*), one approach is to simulate (say) n_0 days in a preliminary experiment. An estimator $\hat{\beta}^*$ of β^* computed from these n_0 *pilot runs* can then be used in place of β when performing the 1000 regular simulation runs (i.e., 1000 days) to estimate μ.

A second approach is to use the same $n = 1000$ simulated days to estimate $\mathrm{Cov}[A_i, X_i]$ and compute an estimator $\hat{\beta}_n^* = \beta^*$, simultaneously with $\mathbb{E}[X_i]$. This gives

$$\bar{X}_{\mathrm{ce},n} = \bar{X}_n - \frac{1}{n-1} \left[\sum_{i=1}^{n} (A_i - \bar{A}_n)(X_i - \bar{X}_n) \right] \frac{\bar{A}_n - a}{\mathrm{Var}[A_i]}. \tag{6.4}$$

as an estimator of μ. This estimator is biased for finite n, because the constant β has been replaced by an estimator $\hat{\beta}_n^*$ that is correlated with $\bar{A}_n$. However, the bias converges to 0 when $n \to \infty$.

For the results of Table 6.1, we tried both approaches and there was no significant difference. We had $\hat{\beta}_n^* \approx 0.52$. The (empirical) variance was reduced to 3305 with 1000 pilot runs and to 3310 without pilot runs, a reduction by a factor of about 6.5 in both cases, and there was no apparent bias in the "no-pilot" case. If $\mathrm{Var}[A_i]$ was unknown as well, we could estimate it in the same way as $\mathrm{Cov}[A_i, X_i]$. We tried it and this gave essentially the same result for this example.

Combining the CV with the indirect estimator gives

$$\begin{aligned}\bar{X}_{\mathrm{i,c},n} &= a - \bar{D}_n - \beta_2(\bar{A}_n - a) \\ &= \bar{A}_n - \bar{D}_n - (1+\beta_2)(\bar{A}_n - a) \\ &= \bar{X}_n - (1+\beta_2)(\bar{A}_n - a)\end{aligned}$$

for some constant β_2. Thus, $\bar{X}_{\mathrm{i,c},n}$ is exactly equivalent to $\bar{X}_{\mathrm{c},n}$ with $\beta = 1+\beta_2$. The optimal β_2 is $\beta_2^* = -\mathrm{Cov}[A_i, D_i]/\mathrm{Var}[A_i] = 1 + \beta^*$. The results in Table 6.1 agree with this equivalence.

Other random variables in the model can be used as CVs, if we know their expectations. One candidate is certainly B_i, the busyness factor. However, we do better with B_i in what follows.

Stratified sampling. Clearly, the variability of B_i is an important source of variability for X_i and D_i. Intuitively, a large variance reduction seems possible if we can control that source of variability.

Denoting $\mu_t = \mathbb{E}[X_i \mid B_i = b_t]$, we can write

$$\begin{aligned}\mu &= \mathbb{E}[X_i] = \sum_{t=1}^{4} \mathbb{P}[B_i = b_t] \cdot \mathbb{E}[X_i \mid B_i = b_t] \\ &= .25\,\mathbb{E}[X_i \mid B_i = 0.8] + .55\,\mathbb{E}[X_i \mid B_i = 1.0] \\ &\quad +.15\,\mathbb{E}[X_i \mid B_i = 1.2] + .05\,\mathbb{E}[X_i \mid B_i = 1.4] \\ &= .25\,\mu_1 + .55\,\mu_2 + .15\,\mu_3 + .05\,\mu_4.\end{aligned}$$

The idea here is to estimate the conditional expectations μ_t *separately for each* t, i.e., for each possible value of B_i. Suppose there are N_t days where $B_i = b_t$ and let $X_{t,1}, \ldots, X_{t,N_t}$ be the values of X_i for those days. We can estimate μ_t by

$$\hat{\mu}_t = \frac{1}{N_t} \sum_{i=1}^{N_t} X_{t,i}$$

for each t, and μ by

$$\bar{X}_{\mathrm{s},n} = .25\hat{\mu}_1 + .55\hat{\mu}_2 + .15\hat{\mu}_3 + .05\hat{\mu}_4. \tag{6.5}$$

We have $\mathrm{Var}[\hat{\mu}_t | N_t] = \sigma_t^2/N_t$ where $\sigma_t^2 = \mathrm{Var}[X_i \mid B_i = b_t]$. Therefore,

$$\mathrm{Var}[\bar{X}_{\mathrm{s},n} \mid N_1, N_2, N_3, N_4] = .25^2\sigma_1^2/N_1 + .55^2\sigma_2^2/N_2 + .15^2\sigma_3^2/N_3 + .05^2\sigma_4^2/N_4$$

The overall variance is reduced if the conditional variances σ_t^2 tend to be smaller than $\mathrm{Var}[X_i]$. If the B_i's are generated independently according to their probabilities, as in a straightforward simulation, then the N_t's are random variables and this scheme is called *post-stratification*.

We can also fix the B_i's in advance, for example by taking $N_t = n_t \approx nq_t$, so the number of B_i's of each kind is in proportion to the probabilities q_t (the $\approx$ sign is because each n_t must be a positive integer). This is stratified sampling with *proportional allocation*.

The *optimal allocation*, which minimizes the variance of the estimator $\bar{X}_{\mathrm{s},n}$ under the constraint that $n_1 + n_2 + n_3 + n_4 = n$, is given by

$$n_t = \frac{\sigma_t q_t}{\sum_{t=1}^{4} \sigma_t q_t} n \tag{6.6}$$

if we allow n_t to take any real value (this is proved in Section 6.8). The σ_t's in (6.6) are unknown, but they can be estimated from pilot runs. We did that with 800 pilot runs, 200 runs with each value of t, and obtained $(n_1, n_2, n_3, n_4) = (219, 512, 182, 87)$ as approximate optimal values (for n_1, n_2 and n_3, we rounded the value of the expression (6.6), then we took $n_4 = n - n_1 - n_2 - n_3$).

Stratification can be combined with other methods such as indirect estimation and estimation with a CV. The optimal allocation is not the same with the combination than with the stratification alone, because the σ_t's are not the same. If we combine stratification with the CV, things become a bit more complicated because we must estimate first the optimal CV coefficient within each stratum (the CV can have a different coefficient for each value of t), then the variance within each stratum with the optimal CV coefficient, and use this to estimate the optimal allocation between the strata. The resulting estimator has the form (6.5) with

$$\hat{\mu}_t = \frac{1}{n_t}\sum_{i=1}^{n_t} X_{\mathrm{c},t,i} = \frac{1}{n_t}\sum_{i=1}^{n_t}[X_{t,i} - \beta_t(A_{t,i} - a\,b_t)].$$

The variance constant $\sigma_t^2 = \mathrm{Var}[X_{\mathrm{c},t,i}]$ is minimized by taking

$$\beta_t = \beta_t^* = \frac{\mathrm{Cov}[A_{t,i},\, X_{t,i}]}{\mathrm{Var}[A_{t,i}]} = \frac{\mathrm{Cov}[A_i, X_i | B_i = b_t]}{a\,b_t},$$

which gives a minimal variance of $\sigma_t^2 = \mathrm{Var}[X_{t,i}] - (\beta_t^*)^2 a b_t$. These covariances and minimal variances can be estimated by pilot runs. Then the allocation that minimizes the overall variance can be computed via (6.6). From our pilot runs, we obtained $(\beta_1, \beta_2, \beta_3, \beta_4) = (1.020, 0.648, 0.224, -0.202)$ as estimated optimal coefficients and then $(n_1, n_2, n_3, n_4) = (131, 503, 247, 119)$ as estimated optimal allocation. This optimal allocation is quite different from the one without CV and also quite different from proportional allocation. It is interesting to observe that $\beta_4 < 0$, which means that $\mathrm{Cov}[A_{4,i},\, X_{4,i}] < 0$, i.e., when the call center is very busy the number of customers with good service quality eventually becomes *negatively correlated* with the number of arrivals. When there is too much traffic, everyone must wait and almost no call can be served on time!

Table 6.1 summarizes our empirical results for various stratified estimators. The subscripts "sp" and 'so" refer to stratified sampling with proportional allocation and optimal allocation, respectively. Combination of two different methods is denoted by double subscripts. For example, the estimator $\bar{X}_{\mathrm{so,c},n}$ uses the CV and stratified sampling with optimal allocation. It has the smallest estimated variance among the methods we tried: approximately 4.2% of that of the crude estimator. In other words, it provides a precision equivalent to that of the crude estimator with approximately 24 times fewer simulation runs (not counting the pilot runs). In practice, we could reuse the pilot runs together with the additional runs in the final estimator, despite the small bias due to the fact that the CV coefficients are estimated in part from the same runs. Simpler schemes that do not require pilot runs, such as the CV alone without the pilot runs, or stratification with proportional allocation without the CV, also yield significant variance reductions.

The results in the first five rows of Table 6.1 have been obtained from the *same* 1000 independent simulation runs. The results for stratification with proportional allocation, and those for optimal allocation, were obtained from two other (independent) sets of 1000 runs. Both of them used the same set of 800 pilot runs.

Table 6.1. Empirical comparison of various estimators for the call center example

Method	Estimator	Mean	$S_n^2(\pm 8\%)$	Ratio
Crude estimator	$\bar{X}_n$	1518.2	21615	1.000
Indirect	$\bar{X}_{\mathrm{i},n}$	1502.5	18389	0.851
CV A_i, with pilot runs	$\bar{X}_{\mathrm{c},n}$	1510.1	3305	0.153
CV A_i, no pilot runs	$\bar{X}_{\mathrm{ce},n}$	1510.2	3310	0.153
Indirect + CV, no pilot runs	$\bar{X}_{\mathrm{i,c},n}$	1510.1	3309	0.153
Stratification (propor.)	$\bar{X}_{\mathrm{sp},n}$	1509.5	1778	0.082
Stratification (optimal)	$\bar{X}_{\mathrm{so},n}$	1509.4	1568	0.073
Strat. (propor.) + CV	$\bar{X}_{\mathrm{sp,c},n}$	1509.2	1140	0.053
Strat. (optimal) + CV	$\bar{X}_{\mathrm{so,c},n}$	1508.3	900	0.042

This entire experiment was repeated with $n = 10^5$, using 1000 pilot runs for each t, to assess the accuracy of the variance reduction factors given in Table 6.1. The empirical variances were very similar; 21998 for the crude estimator, 17966 for the indirect estimator, 3043 (with $\hat{\beta}^* = 0.530$) for the estimator with a CV, and 885 for the stratified estimator with the CV and optimal allocation. These variance estimates are accurate to about 1% with 95% confidence. The value of the best estimator of μ was $\bar{X}_{\mathrm{so,c},n} = 1508.8$. The half-width of a 95% confidence interval on μ based on this estimator is $w/2 \approx 1.96\sqrt{885/n} \approx 0.184$. So we obtain (1508.8 ± 0.2) as an approximate 95% confidence interval on μ.

♣ How to do the stratification when B has the gamma distribution (exercice).

6.2.2 Sensitivity to the service speed

The service times (call durations) in the call center model of the previous subsection are i.i.d. exponential with mean $\theta = \theta_1 = 100$ seconds. We now want to study the effect of changing slightly the parameter θ (the mean service time) to $\theta_2 = \theta_1 - \delta$ for some small $\delta > 0$.

The distribution of $X_i = G_i(s)$ and its expectation, $\mu(\theta) = \mathbb{E}_\theta[X_i]$, now depend on θ. Suppose we want to estimate $\mu(\theta_2) - \mu(\theta_1)$. For example, the call center managers might consider increasing the speed of agents either by additional training or by asking them to handle the calls faster, and would want to evaluate the impact on the waiting times.

We simulate n days at each server speed. Let $X_{1,i}$ and $X_{2,i}$ be the number of customers who started their service after waiting less than s seconds on day i, with $\theta = \theta_1$ and $\theta = \theta_2$, respectively. Define $\Delta_i = X_{2,i} - X_{1,i}$ and let

$$\bar{\Delta}_n = \frac{1}{n}\sum_{i=1}^{n} \Delta_i$$

be the (crude) estimator of the difference $\mu(\theta_2) - \mu(\theta_1)$. To compute $X_{1,i}$ and $X_{2,i}$, we can use either (i) independent random numbers (IRNs) or (ii) common random numbers (CRNs), i.e., the same underlying uniform random numbers for the two values of θ, as we did in Section 1.7 when we looked at the effect of adding one agent for two periods. We have

$$\mathrm{Var}[\Delta_i] = \mathrm{Var}[X_{1,i}] + \mathrm{Var}[X_{2,i}] - 2\mathrm{Cov}[X_{1,i}, X_{2,i}].$$

With IRNs, the covariance term is zero. The aim of CRNs is to make this covariance positive, thus reducing the variance of Δ_i. By using the same random numbers at the same places for both systems as far as this is achievable, the responses $X_{1,i}$ and $X_{2,i}$ are expected (intuitively, at least) to be strongly positively correlated, especially if θ_1 and θ_2 are close.

To see how the same random numbers can be used at approximately the same places in this example, we distinguish four types of random numbers: Those used to generate (1) the busyness factor in the morning; (2) the interarrival times; (3) the service times; and (4) the patience times. To make sure that the same sequences of random numbers are employed within each category, we have attached a different random number stream to each category (see Section 1.12). We initialize these four generators to the same seeds for both server speeds and we generate all random variates by inversion, including the service times.

In our attempt to use the same random numbers at exactly the same places for both systems, we meet a difficulty with the synchronization: Due to the different service times, the abandonments will eventually not be the same for θ_1 and θ_2. A given call may abandon in one system and not in the other. For such a call, the service time must be generated in the first case but does not have to be generated in the second case. In this context, we can generate service times:

(a) for all calls (the abandoned calls will have dummy service times), or

(b) only for the calls that are actually answered.

The rationale for (a) is to make sure that the same calls have the same service times in both configurations, regardless of abandonments. The argument for (b) is for the sequence of service times that are effectively employed to be the same for both systems, even though these service times may belong to different calls because of different abandonment decisions. So if a call has a very long service time for parameter θ_1, this very long service time will also appear (perhaps for another call) for θ_2 under (b). Under (a), this long service time could be unused for θ_2.

Likewise, the patience time does not need to be generated for each call, but only for those whose service does not start immediately upon arrival. We can generate the patience time:

(c) for all arriving calls, again to maintain synchronization, or

(d) only when needed.

By combining these choices, we obtain four different possibilities for the synchronization strategy, and it is unclear a priori which one is best.

Table 6.2 reports empirical results for $\delta = 10$, 1, and 0.1, with $n = 10^4$. This corresponds to a reduction of the mean service time by 10%, 1%, and 0.1%, respectively. The table gives the sample mean and the sample variance of the Δ_i's for simulations with independent random numbers (IRNs), simulations with CRNs without synchronization, and simulations with CRNs with the different types of synchronizations just described. For *CRNs without synchronization*, we just took all the random numbers needed in the simulation sequentially from a single stream instead of from four different streams, and we generated random variates

Table 6.2. Effect of change in the mean service time for the call center, with CRNs, for $n = 10^4$

Method	$\delta = 10$		$\delta = 1$		$\delta = 0.1$	
	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta_i]$	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta_i]$	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta_i]$
IRN (a + c)	55.2	56913	4.98	45164	0.66	44046
IRN (a + d)	52.2	54696	7.22	45192	-1.82	45022
IRN (b + c)	50.3	56919	9.98	44241	1.50	45383
IRN (b + d)	53.7	55222	5.82	44659	1.36	44493
CRN, no sync. (a + c)						
CRN, no sync. (b + d)	56.0	3187	5.90	1204	0.19	726
CRN (a + c)	56.4	2154	6.29	37	0.62	1.8
CRN (a + d)	55.9	2161	6.08	158	0.74	53.8
CRN (b + c)	55.8	2333	6.25	104	0.63	7.9
CRN (b + d)	55.5	2323	6.44	143	0.59	35.3

only when they were needed (strategy (b + d)). Thus, for example, an inter-arrival time for one system configuration could be generated from the same random number as a service time for the other configuration. All the pairs $(\bar{\Delta}_n, \widehat{\text{Var}}[\Delta_i])$ in the table were obtained by independent simulations.

Our results indicate that the CRNs, with any of the four combinations of synchronization approaches, reduce the variance by a huge factor. The smaller δ is, the more the variance is reduced. With IRNs, we have $\text{Var}[\Delta_i] = \text{Var}[X_{1,i}] + \text{Var}[X_{2,i}] \approx 2\,\text{Var}[X_i]$ because $X_{1,i}$ and $X_{2,i}$ are independent, so the variance does not depend much on δ. (It is slightly larger for $\delta = 10$ because $\text{Var}[X_{2,i}]$ is significantly larger in that case.) With CRNs, however, the variance diminishes rapidly when $\delta \to 0$. A theoretical analysis of the convergence rate will be given in Section 6.4.2, Example 6.12.

The difference between the four IRN results is just noise. Note that as $\delta \to 0$, the difference $\mathbb{E}[\Delta_i]$ becomes closer to 0 and thus harder to estimate. When δ is very small, the finite-difference estimators with IRNs are practically useless (too noisy), whereas those with CRNs remain viable.

CRNs with (b + d) without synchronization reduces the variance much less than the good synchronization schemes, but nevertheless improves over IRNs by a significant factor. Understanding exactly why it works would require further investigation, but most of the explanation might be that (i) it takes some time before the synchronization is lost and (ii) the important variable B_i is always generated with the same random number, and thus takes a common value for both systems. With strategy (a + c), on the other hand, it turns out that for this particular example, synchronization is maintained across the two systems even with a single stream, because the two systems get a common value of B_i and the same sequence of arrival events, each call requires exactly three random numbers upon arrival, and no random numbers are needed anywhere else. For this reason, CRNs with a single stream give the same variance reduction as CRNs with the four different streams as described earlier. However, such a situation is more the exception than the rule.

Between the synchronization strategies for the CRNs, the (a + c) combination is the best performer, followed by (b + c). So it is better in this example to generate service and patience times for all calls, and simply discard the values that are not needed. The choice of strategy makes a significant difference when δ is very small, but not much when δ is larger (e.g., 10). We must underline that this observation should not be taken as a general rule: there are similar situations where the best synchronization strategy would be (b + d) instead. So in practice, one should try and compare.

To improve the performance further, we could combine the use of CRNs with stratification and a CV as in Section 6.2.1, or with other variance reduction techniques. For the stratification with "optimal" allocation, we would have to first make n_0 pilot runs in each stratum t to estimate the variance of Δ_i conditional on $B_i = b_t$, and the optimal allocation $(\hat{n}_1, \ldots, \hat{n}_4)$ for n runs. Then we can make $\max(0, \hat{n}_t - n_0)$ additional runs in stratum t for each t (if this gives a total larger than n, we can readjust the numbers downwards, proportionally or in any other reasonable way). The final estimator can be computed from all n runs. When δ is very small, there is a good chance to see $\Delta_i = 0$ for all i in the stratum with $t = 1$. This would give $\hat{n}_1 = 0$. By re-using the pilot runs, we are guaranteed that no stratum will be empty, so the estimator is well-defined.

6.3 Correlation Induction: Theory

Several variance reduction methods rely on inducing correlation between two estimators. One example is the use of CRNs to induce *positive correlation* when comparing similar systems (as in Sections 1.7 and 6.2.2). A second example is to exploit the correlation with a *control variable*. Other methods (e.g., antithetic variates and randomized quasi-Monte Carlo) try to induce *negative* correlations. In this section, we study *sufficient conditions* that guarantee a positive or negative correlation between estimators, and we look at how the correlation can be maximized, or minimized, under certain constraints. We start with the simple case where each of the two estimators is a function of a single uniform, and then generalize to more complicated functions. These results will be used in subsequent sections.

6.3.1 Covariance between functions of a single uniform

For two random variables X and Y with given distributions functions F and G, the maximal and minimal possible correlation between X and Y are the Fréchet bounds given by Theorem 2.6. The theorem also tells us how to reach these bounds, at least theoretically. The maximum correlation (or maximum covariance) is attained by generating both X and Y by inversion from a single uniform U, i.e., $X = F^{-1}(U)$ and $Y = G^{-1}(U)$ where $U \sim U(0, 1)$. To minimize the (negative) correlation, take $X = F^{-1}(U)$ and $Y = G^{-1}(1 - U)$.

Theorem 2.6 *does not* imply that X and Y must be generated by explicit inversion to maximize (or minimize) their correlation. It suffices to generate the pair from the joint distribution indicated in the second part of the theorem. For example, we may generate X *by any method* and then put $Y = G^{-1}(F(X))$. From the correlation-induction point of view, this is equivalent to inversion from $U = F(X) = G(Y)$. If F and G differ only by scale and location parameters, we can generate X by any method, then multiply X by the ratio of the

scale parameters of Y and X, and add the difference between the location parameters, to obtain Y. When we assume that certain random variates are generated by inversion, in the remainder of this chapter, we mean this general (or implicit) form of inversion.

Example 6.1 Let $X \sim N(\mu_x, \sigma_x^2)$ and $Y \sim N(\mu_y, \sigma_y^2)$, two normal random variables with different parameters. To generate the pair (X, Y) with the correct normal marginal distributions while maximizing the correlation between X and Y, we can generate $Z \sim N(0, 1)$ by any method, then define $X = \mu_x + \sigma_x Z$ and $Y = \mu_y + \sigma_y Z$. This is equivalent to generating X and Y by inversion from the common uniform $U = \Phi(Z)$, where Φ is the standard normal distribution function. Even if Z is not effectively generated by applying Φ^{-1} to a uniform U, this can be considered as inversion for the purpose of the theorems in this section. □

Theorem 2.6 can be used to optimize the correlation between simple random variables. When X and Y are estimators computed via simulation, their distributions is usually unknown and too complicated to generate them directly by inversion, so it is generally not possible to maximize or minimize the correlation between them, e.g., if they represent the performance measures of two systems that we want to compare. The usual approach is to optimize the correlation between intermediate (simple) random variables such as service times, lifetimes of components, random decisions, etc., hoping that a significant part of this correlation will be transmitted to the final estimators, with the correct sign.

Example 6.2 For the call center example, in Section 6.2.2 we compared two systems that differed only by a scale parameter of the service-time distribution. For that, we used CRNs with inversion to generate the service times. This maximizes the correlation between the service times of the corresponding customers in the two systems. Maximizing the correlation between the values of $G_i(s)$ (number of calls answered within s seconds) for the two systems is another matter. We can only hope (and test empirically) that strong positive correlations between the service times induce the desired positive correlations between the $G_i(s)$'s as well. We saw in Section 6.2.2 that this is clearly the case for this example.

If we do not use inversion to generate the service times, then the correlation is not necessarily maximized even between the service times, and this could reduce the effectiveness of CRNs. As an illustration, we repeated the experiment of Section 6.2.2 but using an acceptance-complement method to generate the service times (as in Figure 1.29) instead of inversion. For $\delta = 0.001$, with CRNs, we obtained empirical variances ranging from 665 to 770 instead of from 1.8 to 53.8 as in Table 6.2.

As an example of a situation where a non-inversion method is equivalent to inversion, suppose the α parameter of the gamma service time distribution is an integer larger than 1. Then this distribution is $\mathsf{Erlang}(k, \theta)$ with $k = \alpha$, so each service time can be generated by generating k independent exponentials with parameter θ and adding them. This is not inversion but turns out to be equivalent to inversion when combined with CRNs (Exercise 6.6). □

If two random variables X and Y are both functions of the same stream of uniforms, under what conditions are we guaranteed that $\text{Cov}[X, Y] \geq 0$? Or that $\text{Cov}[X, Y] \leq 0$? Such

conditions are developed in what follows, based on the notion of quadrant dependence. They will imply in particular that if $X = f(U_0, U_1, \cdots)$ and $Y = g(U_0, U_1, \cdots)$ where f and g are both monotone in the same direction with respect to each U_j, and $U_0, U_1, U_2, \ldots$ are arbitrary independent random variables, then $\mathrm{Cov}[X, Y] \geq 0$.

6.3.2 Quadrant dependence

Given two monotone functions f and g (nondecreasing or non-increasing) and a random vector (U, V), what are the minimal dependence conditions on the pair (U, V) to guarantee that $f(U)$ and $g(V)$ are positively correlated (or negatively correlated)? Here, U and V are not necessarily uniform, and may have different distributions. For example, U and V can represent the waiting time of a particular customer for two slightly different configurations of a system, and $f(U)$ and $g(V)$ can be the estimators of some performance measure for these two system configurations. The following concept of dependence answers our question.

Definition 6.1 The random variables U and V are *positively quadrant dependent (PQD)* if for all real numbers u and v,

$$\mathbb{P}[U \leq u, V \leq v] \geq \mathbb{P}[U \leq u] \cdot \mathbb{P}[V \leq v]. \tag{6.7}$$

We also say that (U, V) has a *PQD distribution*. If (6.7) holds with the sign of the inequality reversed, we say that U and V are *negatively quadrant dependent (NQD)*, or that (U, V) has a *NQD distribution*. Let $\mathcal{P}^+$ and $\mathcal{P}^-$ denote the families of all PQD and NQD random variable pairs, respectively. □

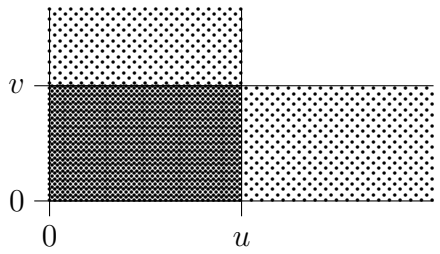


Fig. 6.1. The PQD property holds if for any (u, v), the probability that (U, V) falls into the dark area is at least as large as the product $\mathbb{P}[U \leq u] \cdot \mathbb{P}[V \leq v]$, i.e., as large as if U and V were independent. The figure illustrates the case of positive random variables, but the definition is more general.

The following summarizes useful properties of PQD and NQD distributions.

Theorem 6.1 *(Lehmann 1966, Esary, Proschan, and Walkup 1967). Let (U, V) be any pair of random variables. Then:*

(i) $(U, U) \in \mathcal{P}^+$;

(ii) $(U, V) \in \mathcal{P}^+$ *if and only if* $(U, -V) \in \mathcal{P}^-$;

(iii) *If* $(U, V) \in \mathcal{P}^+$, *and* f *and* g *are two non-decreasing functions or two non-increasing functions, then* $(f(U), g(V)) \in \mathcal{P}^+$, *and similarly for* $\mathcal{P}^-$;

(iv) *If* $(U, V) \in \mathcal{P}^+$ *then* $\mathrm{Cov}[U, V] \geq 0$, *provided that this covariance exists, and the covariance can be zero only if* U *and* V *are independent. Similarly, if* $(U, V) \in \mathcal{P}^-$ *then* $\mathrm{Cov}[U, V] \leq 0$.

(v) $(U, V) \in \mathcal{P}^+$ *if and only if* $\mathrm{Cov}[f(U), g(V)] \geq 0$ *for all nondecreasing functions* f, g *for which that covariance exists.*

6.3.3 Functions of several random variables

We now examine the dependence between two functions of several random variables. If Y_1 is a function f of a sequence of random variables, Y_2 is a function g of another sequence, and we manage to induce dependence between the two sequences, what kind of dependence can be guaranteed between Y_1 and Y_2, and under what conditions? This question is important because simulation programs are really transformations of this type. The next theorem gives a partial answer. It is a key tool to obtain *sufficient* variance reduction conditions for estimators of differences with common random numbers (Section 6.4).

Definition 6.2 Two functions f and g of a denumerable number of variables are *comonotone* [*countermonotone*] in their argument j if they are both monotone in the same direction [in opposite directions] as functions of their jth coordinate when the other coordinates are fixed. □

Theorem 6.2 *Let* $\{(U_j, V_j),\ j \geq 1\}$ *be independent pairs of random variables. Let* $Y_1 = f(U_1, U_2, \dots)$ *and* $Y_2 = g(V_1, V_2, \dots)$ *be two random variables with finite variance, and suppose that for each* $j \geq 1$, *with probability 1, one of the following three conditions holds:*

(i) $(U_j, V_j) \in \mathcal{P}^+$ *and* f *and* g *are comonotone in argument* j;

(ii) $(U_j, V_j) \in \mathcal{P}^-$ *and* f *and* g *are countermonotone in argument* j;

(iii) U_j *and* V_j *are independent.*

Then, $(Y_1, Y_2) \in \mathcal{P}^+$. *Similarly, if these conditions hold after interchanging* $\mathcal{P}^+$ *and* $\mathcal{P}^-$ *in (i)–(ii), then* $(Y_1, Y_2) \in \mathcal{P}^-$.

Proof. This is proved in Lehmann (1966) for the special case where f and g are functions of only the first n random variables, $(U_1, \dots, U_n)$ and $(V_1, \dots, V_n)$, where n is fixed and finite. Our proof for the infinite case is different and uses part (v) of Theorem 6.1. Let $Z_1 = \tilde{f}(Y_1)$ and $Z_2 = \tilde{f}(Y_2)$ where $\tilde{f}$ and $\tilde{g}$ are two nondecreasing functions such that $\mathrm{Cov}[Z_1, Z_2]$ exists. We need to show that $\mathrm{Cov}[Z_1, Z_2] \geq 0$.

Take an index j for which condition (i) of the current theorem holds. Let $\mathcal{F}^{(-j)}$ be the sigma-field generated by $\{(U_i, V_i), i \neq j\}$ and let $(Z_1^{(-j)}, Z_2^{(-j)})$ be the random variable

pair obtained when (U_j, V_j) is replaced, in the definition of (Z_1, Z_2), by a pair $(\tilde{U}_j, \tilde{V}_j)$ of independent random variables with the same marginals as (U_j, V_j). Then, $\mathbb{E}[Z_1 \mid \mathcal{F}^{(-j)}] = E[Z_1^{(-j)} \mid \mathcal{F}^{(-j)}]$, with probability 1, and similarly for Z_2. Also, $\mathrm{Cov}[Z_1, Z_2 \mid \mathcal{F}^{(-j)}] \geq 0 = \mathrm{Cov}[Z_1^{(-j)}, Z_2^{(-j)} \mid \mathcal{F}^{(-j)}]$, because when viewed as functions of (U_j, V_j) alone, for whatever $\mathcal{F}^{(-j)}$ (that is, when all other (U_i, V_i) are fixed), (Z_1, Z_2) are PQD and $(Z_1^{(-j)}, Z_2^{(-j)})$ are independent. Therefore,

$$\begin{aligned}
\mathrm{Cov}[Z_1, Z_2] &= \mathbb{E}[\mathrm{Cov}[Z_1, Z_2 \mid \mathcal{F}^{(-j)}]] + \mathrm{Cov}(\mathbb{E}[Z_1|\mathcal{F}^{(-j)}], \mathbb{E}[Z_2|\mathcal{F}^{(-j)}]) \\
&\geq \mathbb{E}[\mathrm{Cov}[Z_1^{(-j)}, Z_2^{(-j)} \mid \mathcal{F}^{(-j)}]] + \mathrm{Cov}(\mathbb{E}[Z_1^{(-j)}|\mathcal{F}^{(-j)}], \mathbb{E}[Z_2^{(-j)}|\mathcal{F}^{(-j)}]) \\
&= \mathrm{Cov}[Z_1^{(-j)}, Z_2^{(-j)}].
\end{aligned}$$

The same reasoning works if condition (ii) holds for the index j, yielding the same conclusion. Since this holds whatever the distribution of the (U_i, V_i) for $i \neq j$, and for each j that satisfies (i) or (ii), it follows that $\mathrm{Cov}[Z_1, Z_2] \geq \mathrm{Cov}[\tilde{Z}_1, \tilde{Z}_2] = 0$, where $(\tilde{Z}_1, \tilde{Z}_2)$ is the random variable pair obtained when each pair (U_j, V_j) is replaced, in the definition of (Z_1, Z_2), by a pair $(\tilde{U}_j, \tilde{V}_j)$ of independent random variables with the same marginals as (U_j, V_j).

The random variables Y_1 and Y_2 in Theorem 6.2 may actually be functions of only a finite (perhaps random) number of U_j's and V_j's, say $Y_1 = f(U_1, \ldots, U_{N_1})$ and $Y_2 = g(V_1, \ldots, V_{N_2})$, where N_1 and N_2 are random variables. This is just a special case of the theorem where f happens to be constant with respect to U_j for $j > N_1$, and similarly for g. Here, N_1 and N_2 need not be stopping times or be observable. Often, if we compare two systems with proper synchronization, $N_1 = N_2$.

6.4 Common Random Numbers

CRNs are normally used for estimating the *difference* between the expected performance measures μ_1 and μ_2 of two different (but similar) systems, as in Sections 1.7 and 6.2.2). It also makes sense when there are more than two systems to compare, although the analysis is then more complicated.

Suppose we want to estimate $\mu_2 - \mu_1$ by $\Delta = X_2 - X_1$, where $\mathbb{E}[\Delta] = \mu_2 - \mu_1$. Then,

$$\mathrm{Var}[\Delta] = \mathrm{Var}[X_2] + \mathrm{Var}[X_1] - 2\,\mathrm{Cov}[X_1, X_2].$$

If X_1 and X_2 are uncorrelated, the covariance term disappears. But if we manage to induce a positive covariance between X_1 and X_2 without changing their individual distributions, the variance (and MSE) of Δ is reduced. The standard way of inducing such a positive covariance is to use the same underlying uniform random numbers to drive the simulation for both X_1 and X_2, and to make sure that these random numbers are used (as much as possible) at exactly the same place for both systems. The latter is called *synchronization*. To be explicit, let $X_k = f_k(\boldsymbol{U}_k)$, for $k = 1, 2$, where $\boldsymbol{U}_k = (U_{k,1}, U_{k,2}, \ldots)$ represents the i.i.d. sequence of uniforms used to drive the simulation of system k. Using CRNs means taking $\boldsymbol{U}_1 = \boldsymbol{U}_2$. The rationale is that random noise (or "experimental conditions") is then the same for both systems; so the observed differences are due only to the differences between

the systems, and not to the fact that one system has been more lucky than the other in picking its random numbers. As an analogy, using CRNs is like comparing two fertilizers [or drugs] by using each of them on the same piece of land [or the same patients], at the same time. This is impossible in real life, but can be done with simulation. Using CRNs usually does not increase significantly the computing time (it sometimes reduces it, e.g., if we exploit the fact that some random variables need not be generated twice), so the efficiency is increased whenever the variance is reduced.

6.4.1 Sufficient conditions for variance reduction

CRNs do not always work: $\boldsymbol{U}_1 = \boldsymbol{U}_2$ does not guarantee that $\mathrm{Cov}(X_1, X_2) > 0$. Obviously, this depends on how the functions f_1 and f_2 are defined; i.e., how the uniforms are transformed to produce the estimators. These functions are usually very complicated, making even the sign of the covariance hard to evaluate *a priori*. The best possible case is when X_1 and X_2 are perfectly linearly correlated: $\rho(X_1, X_2) = 1$; then $\mathrm{Var}[\Delta]$ is reduced to zero. In the worst case, $\rho(X_1, X_2) = -1$ and $\mathrm{Var}[\Delta]$ is doubled compared with independent simulations. In what follows, sufficient conditions for the covariance to be non-negative are derived from the results of Section 6.3.3.

We assume that $\mathbb{E}[X_1^2] < \infty$ and $\mathbb{E}[X_2^2] < \infty$. Let Ψ^+ be any subset of arguments j for which f_1 and f_2 are comonotone with respect to their jth argument $U_{k,j}$, let Ψ^- be any subset of arguments for which they are countermonotone, and let Ψ^0 be the complement of $\Psi^+ \cup \Psi^-$. We construct $\boldsymbol{U}_2$ from $\boldsymbol{U}_1$ as follows: $U_{2,j} = U_{1,j}$ for $j \in \Psi^+$, $U_{2,j} = 1 - U_{1,j}$ for $j \in \Psi^-$, and $U_{2,j}$ independent of $U_{1,j}$ for $j \in \Psi^0$. Thus, across the two systems, we take CRNs for a subset of arguments for which the function is comonotone, complementary (or *antithetic*) random numbers for a subset of arguments for which the function is countermonotone, and independent random numbers for the other arguments. Here, CRN and antithetic can be interpreted in a wide sense, as explained in Section 6.3.1 (see Example 6.1).

Definition 6.3 A sampling scheme constructed as we just described is called *CRN-concordant*. If Ψ^0 is empty, it is called *completely CRN-concordant*. Similarly, if we permute $U_{1,j}$ and $1 - U_{1,j}$ in the description, that is, take CRNs for countermonotone coordinates and complementary random numbers for comonotone ones, the resulting sampling scheme is called *CRN-discordant*, and *completely CRN-discordant* if Ψ_0 is empty. □

The following theorem is well-known in the simulation community for the special case where Ψ^+ contains all the arguments j; see Glasserman and Yao (1992).

Theorem 6.3 *For any CRN-concordant sampling scheme,* $\mathrm{Cov}[X_1, X_2] \geq 0$*, whereas for any CRN-discordant sampling scheme,* $\mathrm{Cov}[X_1, X_2] \leq 0$*. In both cases, the covariance can be zero only if* X_1 *and* X_2 *are independent.*

Proof. Note that $(U_{1,j}, U_{1,j}) \in \mathcal{P}^+$ and $(U_{1,j}, 1 - U_{1,j}) \in \mathcal{P}^-$. By Theorem 6.2, for the CRN-concordant scheme, $(X_1, X_2) \in \mathcal{P}^+$ and the result follows. The proof for the CRN-discordant scheme is similar: by Theorem 6.2, $(X_1, X_2) \in \mathcal{P}^-$ and the result follows.

Example 6.3 For the stochastic activity network model of Example 1.4, consider two different sets of distributions for the activity durations Y_j, for the same network. The distribution function of Y_j is $F_{1,j}$ in the first configuration and $F_{2,j}$ in the second configuration. The indicator function $\mathbb{I}[T > x]$, where T is the project completion time and x is a constant, is monotone with respect to T, and T is monotone with respect to each Y_j. Therefore, if the Y_j's are generated by inversion with CRNs for the two configurations: $Y_{k,j} = F_{k,j}^{-1}(U_j)$ where the U_j's are i.i.d. $U(0,1)$, and if $X_k = f_k(\boldsymbol{U}_k) = \mathbb{I}[T_k > x]$ where T_k is the project duration for configuration k, for $k = 1, 2$, we have a completely CRN-concordant sampling scheme and thus $\text{Cov}[X_1, X_2] \geq 0$ by Theorem 6.3. □

Example 6.4 One class of models where the conditions of Theorem 6.3 can be verified from the basic building blocks of the model is that of a stochastically monotone Markov chain (Heidelberger and Iglehart 1979, Glasserman and Yao 1992). Recall that a *partially ordered set* is a set $\mathcal{X}$ together with relation $\leq$ having the following properties, for any $x, y, z \in \mathcal{X}$: reflexivity ($x \leq x$), antisymmetry ($x \leq y$ and $y \leq x$ implies $x = y$), and transitivity ($x \leq y$ and $y \leq z$ implies $x \leq z$). A subset $\mathcal{Y}$ of a partially ordered set $\mathcal{X}$ is *nondecreasing* if $y \in \mathcal{Y}$ and $y \leq x \in \mathcal{X}$ implies that $x \in \mathcal{Y}$. Roughly, a Markov chain on a partially ordered state space $\mathcal{X}$ is *stochastically monotone* if at any step, for any nondecreasing set $\mathcal{Y} \subset \mathcal{X}$, the probability that the next state is in $\mathcal{Y}$ is a nondecreasing function of the current state.

Given a stochastically monotone Markov chain on $\mathcal{X}$, we consider an additive cost X of the form (2.74) where C_i is a nondecreasing function of the chain's state at step i. Let X_1 and X_2 denote the random variable X for two versions of that chain, with different transition probabilities, where the transitions at each step of the chain are simulated by inversion with CRNs across the two versions. Then, $\text{Cov}[X_1, X_2] \geq 0$. Heidelberger and Iglehart (1979) show how this can be applied to steady-state average costs in regenerative models, and give the example of a single $GI/GI/1$ queue. Glasserman and Yao (1992) give related conditions on the basic building blocks of a generalized semi-Markov process. □

Example 6.5 The Lindley process $\{W_i,\ i \geq 1\}$, where $W_{i+1} = \max(0,\ W_i + S_i - A_i)$ and $S_i - A_i$ is independent of W_i, is a simple example of a stochastically monotone Markov chain. Indeed, $\mathbb{P}[W_{i+1} \geq x \mid W_i = w_i]$ is a non-decreasing function of w_i for any w_i. If X_1 and X_2 are two nondecreasing functions of the W_i's, for two Lindley processes simulated with CRNs (or for the same one), then $\text{Cov}[X_1, X_2] \geq 0$. □

Example 6.6 As another simple example, consider the value of a financial option with payoff $g(S(t_1), \ldots, S(t_d))$ for some function g, where $\{S(t),\ t \geq 0\}$ is a geometric Brownian motion and $0 = t_0 < t_1 < \cdots < t_d$ (Example 1.11). Here, $\{S(t_j),\ j \geq 0\}$ is a stochastically monotone Markov chain. Therefore, if two such options have payoffs defined by $X_1 = g_1(S_1(t_1), \ldots, S_1(t_d))$ and $X_2 = g_2(S_2(t_1), \ldots, S_2(t_d))$, for some non-decreasing functions g_1 and g_2, and if both are simulated with CRNs (i.e., using the same standard normals $Z_1, \ldots, Z_d$), then $\text{Cov}[X_1, X_2] \geq 0$. □

The monotonicity conditions for the choice of Ψ^+ and Ψ^- are often hard to check. Moreover, there are generally different ways of defining f_1 and f_2 as functions of $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ in a given simulation model; for example, the random numbers could be used in a different order

or transformed differently. Which sets Ψ^+ and Ψ^- are allowed depends on those definitions. The sign of $\text{Cov}[X_1, X_2]$ may be positive for some of the definitions and negative for others. It is rarely clear at the outset which definition is best, but proper synchronization seems to be an important ingredient to make the sampling scheme as close to CRN-concordant as possible (see the example in Section 6.2.2).

It is important to emphasize that *concordance is not a necessary condition, only a sufficient one*. Even if the sampling scheme is known not to be CRN-concordant (or is not provably CRN-concordant) with respect to a given j, taking CRNs for that j may still be beneficial even though Theorem 6.3 no longer applies.

For given distributions of X_1 and X_2, there is always a way of defining f_1 and f_2 such that the covariance is non-negative. Theorem 2.6 gives the optimal way: Use a single uniform to generate both X_1 and X_2 via direct inversion of their distribution functions F_1 and F_2. This is usually impossible to implement, however, because the distributions of X_1 and X_2 are generally unknown or too complicated.

Theorem 6.4 *Suppose that f_1 and f_2 are fixed right-continuous functions that are either comonotone or countermonotone with respect to each of their arguments (i.e., it is possible to choose a completely CRN-concordant scheme). Then, among all sampling schemes such that $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are two sequences of* i.i.d. *$U(0,1)$ r.v.'s and for which non-zero correlation can exist only between the corresponding elements U_{1j} and U_{2j} of $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$, the completely CRN-concordant sampling scheme maximizes the correlation between $X_1 = f_1(\boldsymbol{U}_1)$ and $X_2 = f_2(\boldsymbol{U}_2)$. Similarly, the completely CRN-discordant sampling scheme minimizes the correlation.*

Proof. This generalizes Proposition 2.2 of Glasserman and Yao (1992), whose proof is easily extended.

Example 6.7 For the call center example, we obtained convincing variance reductions empirically in Sections 1.7 and 6.2.2. Can we use Theorem 6.2 to formally prove that the variance is actually reduced, i.e., that the covariance is positive? This seems difficult. Here, each service time is an increasing function of the uniform used to generate it. However, the number $G_i(s)$ of calls answered within s seconds (or the number D_i lost or *not* answered within s seconds) in a day is not necessarily a monotone function of any given service time. A slightly longer service time might cause abandonment of a future call that previously had good service, and can thus decrease the number $G_i(s)$. But if that abandoned call has an exceptionally long service time, getting rid of it might decrease the workload enough so that more than one subsequent call joins the group of calls answered within s seconds. Thus, the overall result could be (exceptionally) a net increase of $G_i(s)$. In other words, the performance measure is not always monotone with respect to the service times. Similarly, it is not monotone with respect to the patience times. In Example 6.12, we prove in a different way that CRNs reduce the variance when δ is sufficiently small.

On the other hand, suppose we consider a simplified call center model with no abandonment and where the performance measure is the average number of calls in the queue during the opening hours. This average is nondecreasing with respect to each service time,

and non-increasing with respect to each interarrival time. CRNs then provably reduce the variance, by Theorem 6.2. □

For complex real-life models, to assess whether CRNs would work, we can make a pilot study: perform a number of replications *with CRNs* and check if $\text{Cov}(X_1, X_2) > 0$. If the cost of using CRNs is significantly different than for independent random numbers, we can compare the two estimators via their efficiencies. From the runs with CRNs, we can estimate $\text{Var}[X_1]$, $\text{Var}[X_2]$, and $\text{Cov}(X_1, X_2)$. These quantities suffice to compute the variance of Δ with and without CRNs, so there is no need to actually perform simulation runs without CRNs.

6.4.2 CRNs for very small differences

A situation where CRNs works extremely well, even without monotonicity, is when a system is parameterized by a continuous parameter θ, reacts similarly to similar values of θ, and we want to compare the performance under two values of θ that are close to each other. More specifically, suppose that the response can be written as $f(\theta, \boldsymbol{U})$, where $\boldsymbol{U}$ represents an underlying sequence of i.i.d. $U(0,1)$ r.v.'s, and $\theta \in \Upsilon$, an interval of the real line. Let $\theta_2 = \theta_1 + \delta$ where δ is small, and suppose $\theta_1, \theta_2 \in \Upsilon$. Let

$$X_1 = f(\theta_1, \boldsymbol{U}_1), \quad X_2 = f(\theta_2, \boldsymbol{U}_2), \quad \text{and } \Delta = X_2 - X_1.$$

Again, we want to estimate $\mathbb{E}[\Delta]$.

The parameter θ could be, for example, that of a service time distribution, or a conveyor speed, or a (positive) routing probability, etc. The difference δ could become very small if we are interested in estimating $\mu'(\theta)$, the derivative of $\mu(\theta) = \mathbb{E}[f(\theta, \boldsymbol{U})]$ at $\theta = \theta_1$, by the finite difference $(X_2 - X_1)/(\theta_2 - \theta_1) = \Delta/\delta$, as in Section 1.8. Here we have

$$\text{Var}[\Delta/\delta] = \frac{\text{Var}[X_1] + \text{Var}[X_2] - 2\text{Cov}[X_1, X_2]}{\delta^2}.$$

When $\delta \to 0$ (for θ_1 fixed), with independent random numbers, we have $\text{Var}[\Delta/\delta] \approx 2\text{Var}[X_1]/\delta^2 \to \infty$. This is bad news: the variance blows up quickly!

With CRNs, however, if enough continuity is present in f, the correlation between X_1 and X_2 will approach 1 and $\text{Var}[\Delta/\delta]$ may remain bounded as $\delta \to 0$. As we saw in Section 1.8, under certain conditions things work so well that we can take the limit

$$f'(\theta_1, \boldsymbol{U}) = \lim_{\delta \to 0} \frac{f(\theta_1 + \delta, \boldsymbol{U}) - f(\theta_1, \boldsymbol{U})}{\delta} = \lim_{\delta \to 0} \frac{\Delta}{\delta}$$

as an unbiased estimator of $\mu'(\theta)$. This estimator can be computed from a *single* simulation (if all goes well), whereas the finite difference estimator requires two simulations. Before detailing this in greater generality, we start with an example.

Example 6.8 We return to the Asian call option problem of Examples 1.11 and 1.21, under the GMB model, with observation times $t_1, \ldots, t_d$. Suppose we want to estimate the

derivative of the expected discounted payoff, $v(s_0, T)$, with respect to s_0, for $s_0 > 0$, using finite differences. The payoff at time T when $S(0) = s_0$ can be written as

$$Y(s_0) = e^{-rT} \max(0,\, s_0 W - K),$$

where

$$W \;=\; \frac{1}{d}\sum_{i=1}^{d} \exp\left[(r-\sigma^2/2)t_i + \sigma \sum_{j=1}^{i} \sqrt{t_j - t_{j-1}}\,\Phi^{-1}(U_j)\right]$$

and $U_1, \ldots, U_d$ are i.i.d. $U(0,1)$. With CRNs, assuming that $\delta > 0$, we have

$$\Delta = Y(s_0+\delta) - Y(s_0) = \begin{cases} 0 & \text{if } Y(s_0+\delta) = Y(s_0) = 0; \\ e^{-rT}\delta W & \text{if } Y(s_0) > 0; \\ e^{-rT}[(s_0+\delta)W - K] & \text{if } Y(s_0+\delta) > Y(s_0) = 0, \end{cases}$$

and the finite difference estimator Δ/δ satisfies $0 \le \Delta/\delta \le e^{-rT}W$, because $s_0 W - K \le 0$ when $Y(s_0) = 0$. Therefore, $\mathrm{Var}[\Delta/\delta] \le \mathbb{E}[\Delta^2/\delta^2] \le e^{-2rT}\mathbb{E}[W^2]$, which is bounded uniformly in δ, so we can take $\delta > 0$ as small as we want and the variance remains bounded.

What is the best δ to use in practice? If we stick with the finite-difference estimator, we have to be careful in the implementation because of the limited accuracy of computers: If δ is too small, $Y(s_0)$ and $Y(s_0+\delta)$ may become almost indistinguishable (on the computer) and this may cause a large error on Δ.

However, we can do much better by taking the estimator to the limit; that is, use the sample derivative

$$Y'(s_0) = \lim_{\delta\to 0} \frac{Y(s_0+\delta) - Y(s_0)}{\delta} = \begin{cases} 0 & \text{if } Y(s_0) = 0; \\ e^{-rT}W & \text{if } Y(s_0) > 0, \end{cases}$$

as an estimator of the derivative

$$v'(s_0, T) = \frac{\partial v(s_0, T)}{\partial s_0} = \frac{\partial \mathbb{E}[Y(s_0)]}{\partial s_0}.$$

Since Δ/δ is bounded uniformly in δ by the integrable function $e^{-rT}W$, the Lebesgue dominated convergence theorem guarantees that we can interchange the limit and expectation in the second equality here:

$$\mathbb{E}[Y'(s_0)] = \mathbb{E}\left[\lim_{\delta\to 0} \frac{Y(s_0+\delta) - Y(s_0)}{\delta}\right] = \lim_{\delta\to 0} \mathbb{E}\left[\frac{Y(s_0+\delta) - Y(s_0)}{\delta}\right] = v'(s_0, T).$$

That is, we have an unbiased estimator of the derivative, whose variance is bounded by $e^{-2rT}\mathbb{E}[W^2]$. □

Taking the sample derivative as a derivative estimator does not always work, as we saw in Example 1.48. There are situations where the variance of Δ/δ with CRNs increases to infinity when $\delta \to 0$, although often at a slower rate than $O(\delta^{-2})$.

Example 6.9 Suppose we replace the payoff $Y(s_0)$ in the previous example by the indicator function:

$$\tilde{Y}(s_0) = \mathbb{I}[Y(s_0) > 0] = \mathbb{I}[s_0 W > K],$$

where $K > 0$. In this case, for a given realization of W, the payoff is 0 for $s_0 \le K/W$, and 1 for $s_0 > K/W$, so the sample derivative $\tilde{Y}'(s_0)$ is 0 with probability 1. On the other hand, the derivative of $\mathbb{E}[\tilde{Y}(s_0)] = \mathbb{P}[W > K/s_0]$ with respect to s_0 is strictly positive. Thus, we no longer have an unbiased estimator.

If we use the finite-difference estimator, we have

$$\Delta = \mathbb{I}\left[s_0 \le K/W < s_0 + \delta\right] = \mathbb{I}\left[\frac{K}{s_0 + \delta} < W < \frac{K}{s_0}\right].$$

Observe that both W and K/W have a bounded and continuous density in a small enough neighborhood of s_0. Thus, $\mathbb{P}[\Delta > 0] = \kappa\delta + o(\delta)$ when δ is small, where κ is the density of K/W at s_0. Then,

$$\mathrm{Var}[\Delta/\delta] = \frac{\mathbb{P}[\Delta > 0] - \mathbb{P}^2[\Delta > 0]}{\delta^2} = \frac{\kappa\delta - (\kappa\delta)^2 + o(\delta)}{\delta^2} = \frac{\kappa}{\delta} - \kappa^2 + o(1/\delta).$$

The variance still blows up when $\delta \to 0$, but at a slower rate than with IRNs, namely as $\Theta(1/\delta)$ instead of $\Theta(1/\delta^2)$. Here, the finite-difference estimator can only take the values 0 and $1/\delta$. This second value is unbounded when $\delta \to 0$ and this is the reason why we cannot apply the dominated convergence theorem. Note that we are in the same setting as in Exercise 1.16(b). □

We now state and prove a general theorem that covers the previous examples and many more. It is followed by a corollary that provides conditions for the sample derivative to be an unbiased estimator of $\mu'(\theta)$. The result of Example 6.8 is a special case of the corollary. Example 6.9 is covered by Part (iv) with $\delta^\alpha = 0$ and $\beta = 1$ (since $\mathbb{P}[\Delta^2 > 0] = O(1/\delta)$ in that example).

Part (iii) of the theorem is related to the results of Glasserman and Yao (1992) and L'Ecuyer and Perron (1994) if we take $\alpha = 2$ (see Corollary 6.6). Part (ii) is more complicated. Its proof is based on (6.8) and on Holder's inequality. The need for (6.8) stems from the fact that the random variable Δ is unbounded a priori. When Δ is bounded, the result simplifies substantially and is given in Part (iv). The usefulness of each part will be illustrated by examples. The corollary is Proposition 3 of L'Ecuyer and Perron (1994). The second part is a special case of Theorem 1 of L'Ecuyer (1990b) and is very similar to Lemma 1 of Glasserman (1988).

Theorem 6.5 *Let* $\Delta = X_2 - X_1 = f(\theta_2, \boldsymbol{U}_2) - f(\theta_1, \boldsymbol{U}_1)$.

(i) If $\boldsymbol{U}_1$ *and* $\boldsymbol{U}_2$ *are independent and if*

$$\sup_{\theta \in \Upsilon} \mathrm{Var}[f(\theta, \boldsymbol{U})] < \infty,$$

then $\mathrm{Var}[\Delta] \in O(1)$ *as* $\delta \to 0$.

(ii) Let $\boldsymbol{U}_1 = \boldsymbol{U}_2 = \boldsymbol{U}$ *and suppose that*

$$K_z(\delta, q) \stackrel{\text{def}}{=} \sup_{\theta_1, \theta_2 \in \Upsilon} \mathbb{E}[\Delta^{2q}] \leq \delta^\nu K_0 \tag{6.8}$$

for some $q > 1$, $\nu \geq 0$, *and* $K_0 < \infty$. *If there is a random variable* Γ, *independent of* θ_1 *and* θ_2, *and some constants* α, β, K_δ, *and* K_γ *such that* $\mathbb{E}[\Gamma^2] \leq K_\gamma$ *and*

$$p(\delta, \alpha) \stackrel{\text{def}}{=} \sup_{\theta_1, \theta_2 \in \Upsilon} \mathbb{P}\{\Delta^2 > \Gamma^2 \delta^\alpha\} \leq K_\delta \delta^\beta, \tag{6.9}$$

then

$$\text{Var}[\Delta] \leq K_\gamma \delta^\alpha + K_z(\delta, q)^{1/q} (K_\delta \delta^\beta)^{1-1/q} \in O(\delta^\alpha + \delta^{[\nu+\beta(q-1)]/q}).$$

As a special case, if (6.9) holds with $\Gamma^2 \delta^\alpha = 0$, *then we can replace* δ^α *by 0 in the result (this could be seen as taking* $\alpha = \infty$*).*

(iii) Let $\boldsymbol{U}_1 = \boldsymbol{U}_2 = \boldsymbol{U}$. *Suppose that there is a random variable* Γ, *independent of* θ_1 *and* θ_2, *and a constant* K_γ *such that* $\mathbb{E}[\Gamma^2] < K_\gamma$. *If* $p(\delta, \alpha) = 0$ *for some constant* α, *then*

$$\text{Var}[\Delta] \leq K_\gamma \delta^\alpha \in O(\delta^\alpha).$$

(iv) Let $\boldsymbol{U}_1 = \boldsymbol{U}_2 = \boldsymbol{U}$. *Suppose that the random variable* $|\Delta|$ *is bounded uniformly with probability 1, i.e., there is a constant* K_h *such that* $\mathbb{P}[|\Delta| \leq K_h \text{ for all } \theta_1, \theta_2 \in \Upsilon] = 1$. *If (6.9) holds, then*

$$\text{Var}[\Delta] \leq K_\gamma \delta^\alpha + K_h^2 K_\delta \delta^\beta \in O(\delta^\alpha + \delta^\beta).$$

As in (ii), this result also holds with $\alpha = \infty$ *(i.e.,* $\delta^\alpha = 0$ *everywhere).*

Proof. For (i), we have $\text{Var}[\Delta] = \text{Var}[X_1] + \text{Var}[X_2] \in O(1)$.

For (ii), using Holder's inequality with $1/p = 1 - 1/q$, we derive:

$$\begin{aligned}
\text{Var}[\Delta] &\leq \mathbb{E}[\Delta^2] \\
&\leq \mathbb{E}[\Gamma^2 \delta^\alpha \mathbb{I}[\Delta^2 \leq \Gamma^2 \delta^\alpha]] + \mathbb{E}[\Delta^2 \mathbb{I}[\Delta^2 > \Gamma^2 \delta^\alpha]] \\
&\leq \mathbb{E}[\Gamma^2 \delta^\alpha] + (\mathbb{E}[\Delta^{2q}])^{1/q} (\mathbb{E}[\mathbb{I}^p[\Delta^2 > \Gamma^2 \delta^\alpha]])^{1/p} \\
&\leq K_\gamma \delta^\alpha + (K_z(\delta, q))^{1/q} (p(\delta, \alpha))^{1-1/q} \\
&\leq K_\gamma \delta^\alpha + (\delta^\nu K_0)^{1/q} (K_\delta \delta^\beta)^{(q-1)/q} \\
&\in O(\delta^\alpha + \delta^{[\nu+\beta(q-1)]/q}).
\end{aligned}$$

For the case where $\alpha = \infty$ (or $\delta^\alpha = 0$), we remove the first term after the second inequality, and all the corresponding terms thereafter.

For (iii), if $\Delta^2 \leq \Gamma^2 \delta^\alpha$ with probability 1,

$$\text{Var}[\Delta] \leq \mathbb{E}[\Delta^2] \leq \mathbb{E}[\Gamma^2 \delta^\alpha] \leq K_\gamma \delta^\alpha \in O(\delta^\alpha).$$

For (iv), we have

$$\begin{aligned}\mathrm{Var}[\Delta] &\le \mathbb{E}[\Delta^2] \\ &\le \mathbb{E}[\Gamma^2\delta^\alpha \mathbb{I}[\Delta^2 \le \Gamma^2\delta^\alpha]] + \mathbb{E}[\Delta^2\mathbb{I}[\Gamma^2\delta^\alpha < \Delta^2 \le K_h^2]] \\ &\le \mathbb{E}[\Gamma^2\delta^\alpha] + K_h^2 p(\delta,\alpha) \\ &\le K_\gamma\delta^\alpha + K_h^2 K_\delta \delta^\beta \\ &\in O(\delta^\alpha + \delta^\beta).\end{aligned}$$

For the case $\alpha = \infty$, we put $\delta^\alpha = 0$ everywhere.

Corollary 6.6 *Suppose that with probability 1, the function $f(\cdot, \boldsymbol{U})$ is continuous in Υ and is differentiable in $D(\boldsymbol{U}) \subseteq \Upsilon$, where $\Upsilon \setminus D(\boldsymbol{U})$ is at most a denumerable set, and*

$$\sup_{\theta \in D(\boldsymbol{U})} |f'(\theta, \boldsymbol{U})| \le \Gamma(\boldsymbol{U}), \tag{6.10}$$

where $\Gamma = \Gamma(\boldsymbol{U})$ is a random variable independent of θ such that $\mathbb{E}[\Gamma^2] \le K_\gamma < \infty$. Then, $\mathrm{Var}[\Delta] \le K_\gamma\delta^2$.

If in addition, for a given θ, $f'(\theta, \boldsymbol{U})$ exists w.p.1, then we can interchange the expectation and derivative operators:

$$\mathbb{E}[f'(\theta, \boldsymbol{U})] = \frac{\partial \mathbb{E}[f(\theta, \boldsymbol{U})]}{\partial \theta},$$

and therefore the sample derivative $f'(\theta, \boldsymbol{U})$ is an unbiased estimator of $\mu'(\theta)$.

Proof. By a generalized version of the mean value theorem (e.g., Theorem 8.5.3 of Dieudonné 1969), with probability 1, we have

$$|\Delta| = |f(\theta_2, \boldsymbol{U}) - f(\theta_1, \boldsymbol{U})| \le \delta \sup_{\theta \in [\theta_1, \theta_2] \cap D(\boldsymbol{U})} |f'(\theta, \boldsymbol{U})| \le \delta\Gamma(\boldsymbol{U}),$$

so $\Delta^2 \le \Gamma^2\delta^2$. Part (iii) of the theorem then applies with $\alpha = 2$. For the second part, it suffices to apply the dominated convergence theorem (Theorem A.2) to Δ/δ.

Suppose that $\mathrm{Var}[\Delta] = O(\delta^\alpha)$ for some $\alpha \ge 0$. Theorem 6.5 and Corollary 6.6 provide conditions under which this happens. Then, the variance of the finite difference estimator satisfies $\mathrm{Var}[\Delta/\delta] = O(\delta^{\alpha-2})$. If $\alpha = 2$ (e.g., if Corollary 6.6 applies), Δ/δ has bounded variance. If $\alpha = 1$, the variance of Δ/δ increases toward infinity at rate $O(1/\delta)$ when $\delta \to 0$. The case where $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are independent is worse: we have $\alpha = 0$ and the variance increases as $O(1/\delta^2)$ when $\delta \to 0$.

Using the sample derivative $f'(\theta, \boldsymbol{U})$ as a derivative estimator is known in the discrete-event simulation literature as *infinitesimal perturbation analysis (IPA)* (Suri 1987, L'Ecuyer 1990b, Glasserman 1991). Often, for discrete-event systems, the sample derivative is computed via recurrence equations that are not always trivial to implement and require additional work.

Example 6.10 Consider the $GI/GI/1$ queue introduced in Section 1.11. Suppose that the service time distribution G_θ depends on a continuous parameter $\theta \in \Upsilon$. When the parameter value is θ, the waiting time $W_j(\theta)$ of the jth customer obeys the Lindley equation

$$W_{j+1}(\theta) = \max(0,\ W_j(\theta) + S_j(\theta) - A_j), \tag{6.11}$$

where $S_j(\theta)$ is the service time of customer j and A_j is the time between the arrivals of customers j and $j+1$. Let $f(\theta, \boldsymbol{U}) = \bar{W}_t(\theta)$, the average waiting time of the first t customers, and $\mu(\theta) = w_t(\theta) = \mathbb{E}[\bar{W}_t(\theta)]$. Here we have $\Delta = \bar{W}_t(\theta_2) - \bar{W}_t(\theta_1)$.

For a concrete illustration, suppose that the interarrival times and service times are exponential with means 1 and θ, respectively, and that the service times are generated by inversion: $S_j(\theta) = -\theta \ln(1 - U_j)$ where the U_j are i.i.d. $U(0,1)$. Suppose also that $\Upsilon \subset (0,1)$ and that $W_1(\theta) = 0$. The derivative of $f(\theta, \boldsymbol{U})$ is

$$f'(\theta, \boldsymbol{U}) = \frac{1}{t} \sum_{j=1}^{t} W_j'(\theta),$$

where $W_1'(\theta) = 0$ and

$$W_{j+1}'(\theta) = \begin{cases} W_j'(\theta) + S_j'(\theta) & \text{if } W_j(\theta) + S_j(\theta) - A_j > 0, \\ 0 & \text{if } W_j(\theta) + S_j(\theta) - A_j < 0, \end{cases}$$

and $S_j'(\theta) = -\ln(1 - U_j)$. For a fixed $\boldsymbol{U}$, the derivative does not exist at the values of θ where $W_j(\theta) + S_j(\theta) - A_j = 0$ for some j, but there exists only a finite number of such values of θ. They form the set $\Upsilon \setminus D(\boldsymbol{U})$. For a fixed θ, on the other hand, the derivative exists w.p.1. When it does not exist, we can just define $f'(\theta, \boldsymbol{U}) = 0$. We then have

$$f'(\theta, \boldsymbol{U}) \le \frac{1}{t} \sum_{j=1}^{t} \sum_{\ell=1}^{j-1} -\ln(1 - U_\ell) \le \sum_{j=1}^{t} -\ln(1 - U_j) \stackrel{\text{def}}{=} \Gamma,$$

where Γ is an $\mathsf{Erlang}(t, 1)$ random variable, whose mean and variance are both equal to t. Therefore $\mathbb{E}[\Gamma^2] < \infty$, so Corollary 6.6 applies and $\mathrm{Var}[\Delta] \in O(\delta^2)$.

We performed some experiments with $t = 50$, $\theta_1 = 0.5$, $\theta_2 = \theta_1 + \delta$ for $\delta = 0.01, 0.001, 0.0001$, and sample size $n = 10000$. Our variance estimates $\widehat{\mathrm{Var}}[\Delta] = S_n^2$ (the sample variance of the n i.i.d. copies of Δ) are given in Table 6.3, first for independent random numbers (IRN), then for common random numbers (CRN) for $\bar{W}_t(\theta_1)$ and $\bar{W}_t(\theta_2)$, with proper synchronization to make sure that the interarrival times are the same and that the same U_j is used to generate $S_j(\theta)$ for each j, for both values of θ. The results agree with the theory: The variance is approximately $0.22 \in O(1)$ for IRN and $4.8\delta^2 \in O(\delta^2)$ for CRN. That is, derivative estimator Δ/δ with CRNs has a bounded variance of about 4.8 when $\delta \to 0$. The sample derivative $f'(\theta, \boldsymbol{U})$, also called the IPA estimator, has the same variance, and no bias.

Similar results hold for more general interarrival time distributions and service time distributions G_θ, under mild conditions on these distributions. Suppose that the service time of customer j is generated by inversion for each j, in the sense that $S_j(\theta) = G_\theta^{-1}(U_j)$ for $U_j \sim U(0,1)$, and that proper synchronization is maintained so that the A_j's and U_j's remain the same for all values of θ. Then, $S_j'(\theta) = \partial G_\theta^{-1}(U)/\partial\theta$, under the assumption that this derivative exists. If there is a random variable $\tilde{\Gamma} = \tilde{\Gamma}(U)$, with finite variance, such that $\sup_{\theta \in \Upsilon} |\partial G_\theta^{-1}(U)/\partial\theta| \le \tilde{\Gamma}$, where $U \sim U(0,1)$, then Corollary 6.6 applies. □

Table 6.3. Mean and variance estimates for the difference in average waiting times, for the $M/M/1$ queue with $\lambda = 1$ and $\theta_1 = 0.5$.

Method	$\delta = 0.01$		$\delta = 0.001$		$\delta = 0.0001$	
	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta]$	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta]$	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta]$
IRN	1.9E-1	2.3E-1	-4.7E-3	2.2E-1	-7.1E-3	2.2E-1
CRN	2.6E-2	5.3E-4	2.6E-3	4.8E-6	2.6E-4	4.8E-8

Example 6.11 We consider the same $GI/GI/1$ queue as in the previous example, but now our cost function is

$$f(\theta, \boldsymbol{U}) = \sum_{j=1}^{t} \mathbb{I}[X_j(\theta) > K], \tag{6.12}$$

where $X_j(\theta) = W_j(\theta) + S_j(\theta)$ is the sojourn time in the system for customer j and K is a constant. This is the number of customers, among the first t, who spent more than K units of time in the system. This performance measure $f(\theta, \boldsymbol{U})$ can only take integer values, so it cannot be continuous with probability 1 as a function of θ (except in the uninteresting case where its expectation is a constant), and we cannot apply Corollary 6.6. In fact, the sample derivative is 0 w.p.1. On the other hand, f being bounded by t, we can apply Theorem 6.5(iv) as follows.

When θ is changed from θ_1 to $\theta_1 + \delta$, the random variable $\mathbb{I}[X_j(\theta) > K]$ changes its value if and only if

$$W_j(\theta_1) + S_j(\theta_1) \le K < W_j(\theta_2) + S_j(\theta_2),$$

if and only if

$$G_{\theta_2}(K - W_j(\theta_2)) < G_{\theta_2}(S_j(\theta_2)) = U_j = G_{\theta_1}(S_j(\theta_1)) \le G_{\theta_1}(K - W_j(\theta_1)).$$

The probability that this event happens, conditional on $W_j(\theta_1)$ and $W_j(\theta_2)$, is

$$P_j(\delta) = G_{\theta_1}(K - W_j(\theta_1)) - G_{\theta_2}(K - W_j(\theta_2)).$$

Let $p_j(\delta) = \mathbb{E}[P_j(\delta)]$, the unconditional probability that this event happens. The probability that at least one of the indicators changes its value is bounded as follows:

$$\mathbb{P}[\Delta > 0] \le \sum_{j=1}^{t} p_j(\delta).$$

If we can show that $p_j(\delta) \in O(\delta^\beta)$ for some constant β, we will obtain $p(\delta, \alpha) \le \mathbb{P}[\Delta > 0] \in O(\delta^\beta)$ for any α and, because Δ is bounded by the constant t, it will follow from Theorem 6.5(iv) that $\text{Var}[\Delta] \in O(\delta^\beta)$.

We now show that $p_j(\delta) \in O(\delta)$ in the case of exponential service times with mean θ, i.e., if $S_j(\theta) = -\theta \ln(1 - U_j)$. Denote $Z_{j,1} = \max(0,\, K - W_j(\theta_1))$ and $Z_{j,2} = \max(0,\, K - W_j(\theta_2))$. Then

$$\begin{aligned} P_j(\delta) &= G_{\theta_1}(Z_{j,1}) - G_{\theta_2}(Z_{j,2}) \\ &= 1 - e^{-Z_{j,1}/\theta_1} - \left(1 - e^{-Z_{j,2}/\theta_2}\right) \\ &= e^{-Z_{j,1}/\theta_1}\left(e^{-Z_{j,2}/\theta_2 + Z_{j,1}/\theta_1} - 1\right) \\ &\leq e^{-Z_{j,2}/\theta_2 + Z_{j,1}/\theta_1} - 1. \end{aligned}$$

On the other hand,

$$Z_{j,1} - Z_{j,2} \leq W_j(\theta_2) - W_j(\theta_1) \leq \sum_{\ell=1}^{j-1} [S_\ell(\theta_1 + \delta) - S_\ell(\theta_1)]$$

and thus

$$\begin{aligned} -\frac{Z_{j,2}}{\theta_2} + \frac{Z_{j,1}}{\theta_1} &= \frac{Z_{j,1} - Z_{j,2}\theta_1/(\theta_1+\delta)}{\theta_1} \\ &= \frac{Z_{j,1} - Z_{j,2} + Z_{j,2}\delta/(\theta_1+\delta)}{\theta_1} \\ &\leq \frac{1}{\theta_1}\sum_{\ell=1}^{j-1}[S_\ell(\theta_1+\delta) - S_\ell(\theta_1)] + \frac{\delta Z_{j,2}}{\theta_1(\theta_1+\delta)} \\ &\leq \frac{1}{\theta_1}\sum_{\ell=1}^{j-1}(-\delta\ln(1-U_\ell)) + \delta K' \end{aligned}$$

where $K' = K/[\theta_1(\theta_1+\delta)]$. Then,

$$\begin{aligned} p_j(\delta) &\leq \mathbb{E}\left[\exp\left(-Z_{j,2}/\theta_2 + Z_{j,1}/\theta_1\right)\right] - 1 \\ &\leq \mathbb{E}\left[\exp\left(-\frac{\delta}{\theta_1}\sum_{\ell=1}^{j-1}\ln(1-U_\ell) + \delta K'\right)\right] - 1 \\ &\leq \mathbb{E}\left[\prod_{\ell=1}^{j-1}(1-U_\ell)^{-\delta/\theta_1}\right] e^{\delta K'} - 1 \\ &\leq \left(\mathbb{E}\left[(1-U_\ell)^{-\delta/\theta_1}\right]\right)^{j-1} e^{\delta K'} - 1 \\ &\leq [\theta_1/(\theta_1-\delta)]^{j-1} e^{\delta K'} - 1 \\ &= \delta K' + O(\delta^2). \end{aligned}$$

This implies that $\mathrm{Var}[\Delta] \in O(\delta)$ by Theorem 6.5(iv).

This argument can be adapted to other service time distributions than the exponential and more general queueing systems.

We performed the same experiments as in Example 6.10 for this new performance measure, with $K = 2$. The results are in Table 6.4, where we see that $\mathrm{Var}[\Delta]$ is approximately proportional to δ. This agrees with our theoretical bound.

□

Example 6.12 In Section 6.2.2, we examined the impact of a slight change of the mean service time θ in the call center example. In this case, $f(\theta, \boldsymbol{U})$ represents the number of calls

Table 6.4. Mean and variance estimates for Example 6.11, for the $M/M/1$ queue with $t = 50$, $\lambda = 1$, $\theta_1 = 0.5$, and $K = 2$.

Method	$\delta = 0.01$		$\delta = 0.001$		$\delta = 0.0001$	
	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta_i]$	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta_i]$	$\bar{\Delta}_n$	$\widehat{\text{Var}}[\Delta_i]$
IRN	0.40	66	-0.05	63	-0.09	63
CRN	4.9E-1	6.3E-1	4.9E-2	5.2E-2	5.4E-3	5.3E-3

answered within s seconds when the service time is exponential with mean θ. The service times are generated by inversion. When changing θ while $\boldsymbol{U}$ is fixed (i.e., using CRNs), the service times change continuously, but $f(\theta, \boldsymbol{U})$ can change only by jumps of integer sizes. Therefore, the continuity condition of Corollary 6.6 does not hold.

We now show that condition (ii) of Theorem 6.5 holds with $\delta^\alpha = 0$, $\nu = 0$, $\beta = 1$, and any $q > 1$. This will imply that $\text{Var}[\Delta] = O(\delta^{1-\epsilon})$ for any $\epsilon > 0$. We do this by allowing the business factor B, the patience times, and the service times, to have more general distributions than those specified earlier. We assume that θ is a scale parameter of the service time distribution, that the service time has finite expectation, that we use inversion and CRNs, and that we generate a service time for every customer, including those who abandon. The argument could be adapted to other types of parameters, under additional conditions, but it may become more complicated. We also assume that the business factor B has bounded moments of all orders. The main ingredient in our proof is a lemma that provides an $O(\delta)$ bound on $\mathbb{P}[\Delta \neq 0] = \mathbb{P}[X(\delta) \neq X(0)]$, as a function of δ.

Let T_j denote the arrival time of call j, P_j its patience time, and W_j the time at which it is answered. Under our assumptions, when we change θ from θ_1 to $\theta_1 - \delta$, the service time S_j of the jth customer (by order of arrival) becomes $S_j(\delta) = (1 - \delta/\theta_1)S_j$. It waiting time changes to $W_j(\delta)$, say. Let $D_j = |W_j - W_j(\delta)|$. Call j has *good service* (receives an answer within the time limit s) in the original model if and only if $W_j \leq V_j$, where $V_j = T_j + \min(P_j, s)$ is its *virtual threshold time*. Note that the V_j's are independent of the S_j's and of δ. Call j switches from bad to good service with θ_2 if and only if

$$W_j(\delta) \leq V_j < W_j, \tag{6.13}$$

and it switches from good to bad service if and only if

$$W_j \leq V_j < W_j(\delta). \tag{6.14}$$

In general, the status of call j changes if and only if

$$\min\{W_j, W_j(\delta)\} \leq V_j < \max\{W_j, W_j(\delta)\}. \tag{6.15}$$

Let $\bar{\lambda}(b)$ be the maximum arrival rate during the day conditional on $B = b$. Let $\mathbb{P}_b$ and $\mathbb{E}_b$ denote the corresponding conditional probability and conditional expectation. Recall that A is the total number of arrivals during the day.

Lemma 6.7 *Conditional on $B = b$, we have*

$$\mathbb{P}_b[X(\delta) \neq X(0)] \;\leq\; \bar{\lambda}(b)(\delta/\theta_1)\mathbb{E}[S_1]\mathbb{E}_b[A^2] \;=\; O(\delta b^3).$$

Proof. For a small $\epsilon > 0$, a given time interval $[t, t+\epsilon)$ can contain one (or more) of the V_j's only if a call arrives in the time interval $[t-s, t-s+\epsilon)$ and reaches its virtual threshold time before abandoning, or if a call arrives at time x for $t - s \leq x \leq t + \epsilon$ and abandons during the interval $[t, t+\epsilon)$. The probability that one of these two events occurs is bounded by

$$\lambda(t-s)\,\epsilon\,(1 - F_{\mathrm{P}}(s)) + \int_{t-s}^{t+\epsilon} \lambda(x)\,\epsilon dF_{\mathrm{P}}(x) + o(\epsilon) \;\leq\; \epsilon\,\bar{\lambda}(u) + o(\epsilon)$$

where F_{P} is the distribution function of the patience time. This gives an upper bound on the probability that $[t, t+\epsilon]$ contains V_j for any fixed j. By integrating this with respect to t over $[\min\{W_j, W_j(\delta)\}, \max\{W_j, W_j(\delta)\}]$, and taking $\epsilon \to 0$, we find that the probability that (6.15) occurs cannot exceed $\bar{\lambda}(b)D_j$.

Let J^* be the smallest integer $j > 0$ for which (6.15) holds, i.e, the index of the first call that switches status. If there is none, put $J^* = \infty$. For $j \leq J^*$, we have

$$D_j = |W_j - W_j(\delta)| \leq \sum_{\ell=1}^{j-1}(S_\ell - S_\ell(\delta)) = \frac{\delta}{\theta_1}\sum_{\ell=1}^{j-1} S_\ell.$$

Combining the last two bounds, we obtain that

$$\begin{aligned}
& \mathbb{P}_b[J^* = j \mid W_j, W_j(\delta)] \\
\leq\; & \mathbb{P}_b[\min\{W_j, W_j(\delta)\} \leq V_j < \max\{W_j, W_j(\delta)\} \mid W_j, W_j(\delta)] \\
\leq\; & \bar{\lambda}(b)D_j \\
\leq\; & \frac{\bar{\lambda}(b)\delta}{\theta_1}\sum_{\ell=1}^{j-1} S_\ell.
\end{aligned}$$

By summing on j, and exploiting the fact that the S_ℓ's are independent of A, we obtain

$$\begin{aligned}
\mathbb{P}_b[X(\delta) \neq X(0)] &\leq \mathbb{P}_b[J^* < \infty] \\
&= \sum_{j=1}^{\infty} \mathbb{E}_b\left[\mathbb{I}[j \leq A]\mathbb{P}_b[J^* = j \mid W_j, W_j(\delta)]\right] \\
&\leq \sum_{j=1}^{\infty} \mathbb{E}_b\left[\mathbb{I}[j \leq A]\bar{\lambda}(b)(\delta/\theta_1)\sum_{\ell=1}^{j-1} S_\ell\right] \\
&= \sum_{j=1}^{\infty} \mathbb{P}_b[j \leq A]\bar{\lambda}(b)\mathbb{E}_b\left[(\delta/\theta_1)\sum_{\ell=1}^{j-1} S_\ell\right] \\
&= \sum_{j=1}^{\infty} \mathbb{P}_b[j \leq A](j-1)\bar{\lambda}(b)(\delta/\theta_1)\mathbb{E}[S_j] \\
&= \bar{\lambda}(b)(\delta/\theta_1)\mathbb{E}[S_1]\sum_{j=1}^{\infty} \mathbb{P}_b[A = j]j(j-1)/2 \\
&\leq \bar{\lambda}(b)(\delta/\theta_1)\mathbb{E}[S_1]\mathbb{E}_b[A^2] \\
&= O(\delta b^3).
\end{aligned}$$

This completes the proof of the lemma.

Since B has a bounded moments, we can conclude from this lemma that

$$\mathbb{P}[\Delta \neq 0] = \mathbb{E}[\mathbb{P}[\Delta \neq 0 \mid B]] = O(\delta)\mathbb{E}[B^3] = O(\delta),$$

so (6.9) is satisfied with $\beta = 1$ and $\alpha = \infty$. The crude bound

$$\mathbb{E}[\Delta^{2q}] \leq \mathbb{E}[A^{2q}] = \mathbb{E}[\mathbb{E}[A^{2q} \mid B]] = \mathbb{E}[B^{2q}]\mathbb{E}[A^{2q} \mid B = 1] < \infty,$$

which is finite because both B, and A conditional on B (which is Poisson), have bounded moments of all orders, implies that (6.8) holds with $\nu = 1$ for any $q \geq 1$. It then follows from Part (ii) of Theorem 6.5 that $\mathrm{Var}[\Delta] = O(\delta^{(q-1)/q}) = O(\delta^{1-\epsilon})$ for any $\epsilon > 0$, by taking $(q-1)/q < \epsilon$, with a hidden constant that may increase with q. This is almost $O(\delta)$. The empirical results of Section 6.2.2 indicate that $\mathrm{Var}[\Delta] \in O(\delta)$ (and perhaps better), at least for the (a + c) and (b + c) synchronization strategies and for the specific distributions used in our numerical experiment.

If $\Delta = X_2 - X_1$ was bounded by a constant, then it would follow from Theorem 6.5(iv) that $\mathrm{Var}[\Delta] = O(\delta)$. Formally speaking, Δ is not bounded, but when δ is small, the probability than Δ exceeds a few units is so small that Δ can be considered as bounded, from a practical viewpoint.

□

♣ Add other examples.

6.4.3 Comparing regenerative models

Suppose we want to estimate the difference $\mu_2 - \mu_1$ between the steady-state performance measures of two regenerative systems. Under mild conditions, the regenerative approach

(Section 5.12) can be used to compute a confidence interval for μ_1, and similarly for μ_2. To apply regenerative output analysis to $\mu_2 - \mu_1$ when the two systems are simulated with CRNs, we need the paired process induced by the parallel evolution of the two systems to be positive recurrent regenerative. Heidelberger and Iglehart (1979) give details and provide sufficient (stochastic monotonicity) conditions for a positive covariance when the two processes are Markov chains (see also Example 6.4). Glynn (1985) shows that the joint regeneration property always hold if the two systems are positive recurrent Markov chains with countable state spaces, but can fail to hold if they have more general state spaces.

6.4.4 Generalizations and related techniques

CRNs are not only useful for estimating a difference such as $\mu_2 - \mu_1$, they could be effective more generally for estimating a continuously differentiable function of several means, say $g(\mu_1, \ldots, \mu_d)$. If $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ are i.i.d. random vectors with mean $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_d)$ and positive definite covariance matrix $\boldsymbol{\Sigma}_{\mathrm{x}}$, then $\sqrt{n}(\bar{\boldsymbol{X}}_n - \boldsymbol{\mu}) \Rightarrow N(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathrm{x}})$ and $\sqrt{n}(g(\bar{\boldsymbol{X}}_n) - g(\boldsymbol{\mu}))/\sigma_{\mathrm{g}} \Rightarrow N(0,1)$ as $n \to \infty$, from the delta theorem, where $\sigma_{\mathrm{g}}^2 = (\nabla g(\boldsymbol{\mu}))^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{x}} \nabla g(\boldsymbol{\mu})$. This means that $n\mathrm{Var}[g(\bar{\boldsymbol{X}}_n)] \Rightarrow \sigma_{\mathrm{g}}^2$ when $n \to \infty$.

If the coordinates of $\boldsymbol{X}_i$ are independent, then $\boldsymbol{\Sigma}_{\mathrm{x}}$ is a diagonal matrix $\boldsymbol{\Sigma}_{\mathrm{x}}^{(d)}$ whose jth diagonal element is the variance of the jth coordinate of $\boldsymbol{X}_i$. Inducing correlations between the components of each $\boldsymbol{X}_i$ (e.g., by using CRNs) reduces the variance of the estimator $g(\bar{\boldsymbol{X}}_n)$, asymptotically, if and only if σ_{g}^2 becomes smaller than $(\sigma_{\mathrm{g}}^{(d)})^2 = (\nabla g(\boldsymbol{\mu}))^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{x}}^{(d)} \nabla g(\boldsymbol{\mu})$, i.e., if and only if

$$(\nabla g(\boldsymbol{\mu}))^{\mathsf{t}} (\boldsymbol{\Sigma}_{\mathrm{x}} - \boldsymbol{\Sigma}_{\mathrm{x}}^{(d)}) \nabla g(\boldsymbol{\mu}) < 0. \tag{6.16}$$

Example 6.13 Consider the estimation of a ratio of expectations, $\nu = g(\mu_1, \mu_2) = \mu_1/\mu_2$, by the empirical ratio $\hat{\nu}_n = \bar{X}_n/\bar{Y}_n$, where $(X_1, Y_1), \ldots, (X_n, Y_n)$ are i.i.d. vectors with mean (μ_1, μ_2). This was studied in Section 5.4.2. Asymptotically, $\sqrt{n}(\hat{\nu}_n - \nu)$ has mean zero and variance

$$\sigma_{\mathrm{g}}^2 = \frac{\mathrm{Var}[X_i] + \nu^2 \mathrm{Var}[Y_i] - 2\nu \mathrm{Cov}[X_i, Y_i]}{\mu_2^2}.$$

Thus, if $\nu > 0$, inducing a positive correlation between X_i and Y_i without changing their variance will reduce the variance of the ratio estimator (for large enough n). If $\nu < 0$, the induced correlation must be negative.

When we estimate a steady-state average using the classical regenerative approach (for a single system), we effectively use the same simulations to estimate (simultaneously) both the numerator and the denominator of such a ratio. This is exactly equivalent to using CRNs with perfect synchronization. It reduces the variance if and only if the performance measure X_i is *positively* correlated with the cycle length Y_i if $\nu > 0$, and *negatively* correlated if $\nu < 0$. If there is a slight correlation of the wrong sign (of a different sign than ν), using the same simulations increases the variance but might still be more efficient than doing independent simulations, because the reduction in computational costs can make up for the variance increase. If there is a strong correlation of the wrong sign between X_i and Y_i when they come from the same simulation (or realization), then it might be better to simulate them independently, or perhaps with antithetic variates (see Section 6.9). □

Besides the variance reduction, there are situations where the CRNs also make the computations less costly. The idea is that the random numbers need to be generated only once. When comparing similar related systems, the lower-level transformations (e.g., the generation of interarrival and service times in a queue) are sometimes exactly (or almost) the same for all systems of interest, and the systems differ only at a higher level. Then, a significant amount of computation may be common to all systems and could be performed only once. L'Ecuyer and Vázquez-Abad (1997) show how this idea can be exploited to efficiently estimate an entire function of a univariate continuous parameter. On the other hand, it may also happen that CRNs make the computations more costly, for example if all random numbers need to be generated twice, because doing otherwise would be too complicated, and if the need for synchronization brings additional overhead.

.

6.5 Control Variables

6.5.1 Setting and optimal coefficients

The control variables (CV) idea exploits auxiliary information to figure out whether the random events have been more favorable or less favorable than usual in influencing the sample performance, and to make an appropriate correction to the estimator. Here, we consider *linear corrections*, i.e., linear CVs. Let X be the default performance estimator and $\boldsymbol{C} = (C^{(1)}, \ldots, C^{(q)})^{\mathsf{t}}$ be a vector of q other random variables, presumably correlated with X, with known expectation $\mathbb{E}[\boldsymbol{C}] = \boldsymbol{\nu} = (\nu^{(1)}, \ldots, \nu^{(q)})^{\mathsf{t}}$. This $\boldsymbol{C}$ is called the CV vector. Define the *controlled estimator*

$$X_{\mathrm{c}} = X - \boldsymbol{\beta}^{\mathsf{t}}(\boldsymbol{C} - \boldsymbol{\nu}) = X - \sum_{\ell=1}^{q} \beta_\ell (C^{(\ell)} - \nu^{(\ell)}),$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_q)^{\mathsf{t}}$ is a vector of constants. Then, $\mathbb{E}[X_{\mathrm{c}}] = \mathbb{E}[X] = \mu$.

We now find an expression for the optimal vector $\boldsymbol{\beta}$. Later, we deal with the fact that this optimal $\boldsymbol{\beta}$ depends on unknown covariances that must be estimated. Let $\boldsymbol{\Sigma}_{\mathrm{C}} = \mathrm{Cov}[\boldsymbol{C}]$, a matrix whose element (i, j) is the value of $\mathrm{Cov}[C^{(i)}, C^{(j)}]$, and let $\boldsymbol{\Sigma}_{\mathrm{CX}} = (\mathrm{Cov}(X, C^{(1)}), \ldots, \mathrm{Cov}(X, C^{(q)}))^{\mathsf{t}}$ be the covariance (vector) between $\boldsymbol{C}$ and X. Throughout this section, we make the following assumption:

Assumption 6.1 $\mathrm{Var}[X] = \sigma^2 < \infty$, $\boldsymbol{\Sigma}_{\mathrm{C}}$ and $\boldsymbol{\Sigma}_{\mathrm{CX}}$ are finite, and $\boldsymbol{\Sigma}_{\mathrm{C}}$ is positive definite. □

Under Assumption 6.1,

$$\mathrm{Var}[X_{\mathrm{c}}] = \mathrm{Var}[X] + \boldsymbol{\beta}^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{C}} \boldsymbol{\beta} - 2\boldsymbol{\beta}^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{CX}}.$$

This variance is minimized by taking

$$\boldsymbol{\beta} = \boldsymbol{\beta}^* = \boldsymbol{\Sigma}_{\mathrm{C}}^{-1} \boldsymbol{\Sigma}_{\mathrm{CX}},$$

in which case

$$\mathrm{Var}[X_c] \;=\; (1 - R^2_{CX})\mathrm{Var}[X] \stackrel{\mathrm{def}}{=} \sigma_c^2, \tag{6.17}$$

where

$$R^2_{CX} = \frac{\boldsymbol{\Sigma}_{CX}^t \boldsymbol{\Sigma}_C^{-1} \boldsymbol{\Sigma}_{CX}}{\mathrm{Var}[X]}$$

is the *coefficient of determination* (the square of the multiple correlation coefficient) between $\boldsymbol{C}$ and X. If $d = 1$ and $C^{(1)} = C$, then $R^2_{CX} = \rho^2(C, X)$, the square of the usual Pearson correlation coefficient. The variance can be reduced by either positive or negative correlation, and R^2_{CX} indicates the fraction of the variance that is reduced with the optimal $\boldsymbol{\beta}$. The factor $1 - R^2_{CX} = \sigma_c^2/\sigma^2$ is called the *minimum variance ratio*. It represents the variance reduction factor with the optimal $\boldsymbol{\beta}$. In the best possible case, if the multiple correlation is ± 1, the variance is reduced to zero. In the worst case, there is no correlation and the variance is unchanged (with the optimal $\boldsymbol{\beta}$). With an arbitrary (wrong) $\boldsymbol{\beta}$, the variance may increase without bounds.

6.5.2 Types of control variables

Among the control variates frequently used, we may distinguish the three following types (Bratley, Fox, and Schrage 1987):

(a) internal CVs,

(b) external CVs, and

(c) CVs obtained by weighted averages.

The *internal CVs* are those derived from quantities generated during the simulation. Examples include average interarrival times or average service times in a queue with i.i.d. interarrival and service times with known means, or average actual lifetimes of components in a reliability system whose lifetimes are i.i.d. with known means, and so on. These CVs come almost for free: there is no need to perform additional simulations to compute their values.

External CVs are obtained by performing additional simulations on the side, such as simulating, with CRNs, a similar (simpler) system than the one of interest and for which the expected performance is known (say, $= \nu^{(\ell)}$). The observed performance of the similar system is the CV. Particularly in this case, variance reduction is not equivalent to efficiency improvement, because of the added work for simulating the other system. For example, if we want to estimate the average waiting time in a queue with complicated service-time distribution and if we know how to compute the exact average for i.i.d. exponential service times, we may use the latter as the simpler system. Here the synchronization (for the CRNs) is crucial.

CVs are obtained via *weighted averages* as follows. Suppose we have $q + 1$ unbiased estimators for μ, say $X^{(0)}, \ldots, X^{(q)}$. We can construct a weighted average of those estimators, yielding the estimator

$$X_{\mathrm{c}} = \sum_{\ell=0}^{q} \beta_\ell X^{(\ell)} = X^{(0)} - \sum_{\ell=1}^{q} \beta_\ell (X^{(0)} - X^{(\ell)})$$

where $\sum_{\ell=0}^{q} \beta_\ell = 1$. Interpret $C^{(\ell)} = X^{(0)} - X^{(\ell)}$, $\ell = 1, \ldots, q$, as CVs for $X^{(0)}$. Then, we are back to our linear CV model and we can use the standard CV machinery to estimate the optimal weights β_ℓ. The classical setting of antithetic variates (Section 6.9) is a special case of this.

6.5.3 Estimating the optimal coefficients: asymptotic theory

Typically, we cannot use $\boldsymbol{\beta}^*$ in X_{c} because it is unknown (sometimes $\boldsymbol{\Sigma}_{\mathrm{C}}$ may be known, but practically never $\boldsymbol{\Sigma}_{\mathrm{CX}}$). Suppose a weakly consistent estimator $\hat{\boldsymbol{\beta}}_n$ of $\boldsymbol{\beta}^*$ is computed from the sample $(X_1, \boldsymbol{C}_1), \ldots, (X_n, \boldsymbol{C}_n)$. Define

$$X_{\mathrm{ce},i} = X_i - \hat{\boldsymbol{\beta}}_n^{\mathsf{t}} (\boldsymbol{C}_i - \boldsymbol{\nu})$$

and replace $\bar{X}_{\mathrm{c},n}$ by the estimator

$$\bar{X}_{\mathrm{ce},n} = \bar{X}_n - \hat{\boldsymbol{\beta}}_n^{\mathsf{t}} (\bar{\boldsymbol{C}}_n - \boldsymbol{\nu}). \tag{6.18}$$

Glynn and Szechtman (2002) prove the following, which generalizes a result of Nelson (1990):

Theorem 6.8 *Suppose Assumption 6.1 holds. When $n \to \infty$, if $\hat{\boldsymbol{\beta}}_n \Rightarrow \boldsymbol{\beta}^*$, then*

$$\sqrt{n}(\bar{X}_{\mathrm{c},n} - \bar{X}_{\mathrm{ce},n}) \quad \Rightarrow \quad 0, \tag{6.19}$$

$$S^2_{\mathrm{ce},n} \stackrel{\mathrm{def}}{=} \frac{1}{n} \sum_{i=1}^{n} (X_{\mathrm{ce},i} - \bar{X}_{\mathrm{ce},n})^2 \quad \Rightarrow \quad \sigma_{\mathrm{c}}^2, \tag{6.20}$$

$$\frac{\sqrt{n}(\bar{X}_{\mathrm{ce},n} - \mu)}{S_{\mathrm{ce},n}} \quad \Rightarrow \quad \frac{\sqrt{n}(\bar{X}_{\mathrm{c},n} - \mu)}{\sigma_{\mathrm{c}}} \quad \Rightarrow \quad N(0,1), \tag{6.21}$$

where σ_{c}^2 is defined in Eq. (6.17).

Furthermore, if $\hat{\boldsymbol{\beta}}_n \to \boldsymbol{\beta}^*$ with probability 1, we also have that $S^2_{\mathrm{ce},n} \to \sigma_{\mathrm{c}}^2$ and $\bar{X}_{\mathrm{ce},n} \to \mu$ with probability 1, by the strong law of large numbers and a continuity argument.

A confidence interval for μ can be constructed in the usual way by assuming that $\sqrt{n}(\bar{X}_{\mathrm{ce},n} - \mu)/S_{\mathrm{ce},n}$ has the $N(0,1)$ distribution. Theorem 6.8 asserts that *asymptotically*, as $n \to \infty$, such a confidence interval is valid and there is no loss in having to estimate $\boldsymbol{\beta}^*$, in the sense that $\bar{X}_{\mathrm{ce},n}$ and $\bar{X}_{\mathrm{c},n}$ have the same asymptotic variance.

There are often many possibilities for $\hat{\boldsymbol{\beta}}_n$, and $S^2_{\mathrm{ce},n}$ can also be replaced by any other consistent variance estimator (e.g., we can replace $1/n$ by $1/(n-q-1)$ in front of the sum in (6.20) or we can adopt (6.22)).

Perhaps the most natural way of estimating $\boldsymbol{\beta}^*$ is by $\hat{\boldsymbol{\beta}}_n = \hat{\boldsymbol{\Sigma}}_{\mathrm{C}}^{-1} \hat{\boldsymbol{\Sigma}}_{\mathrm{CX}}$, where $\hat{\boldsymbol{\Sigma}}_{\mathrm{C}}$ and $\hat{\boldsymbol{\Sigma}}_{\mathrm{CX}}$ are the sample counterparts of $\boldsymbol{\Sigma}_{\mathrm{C}}$ and $\boldsymbol{\Sigma}_{\mathrm{CX}}$. That is, the element (ℓ, k) of $\hat{\boldsymbol{\Sigma}}_{\mathrm{C}}$ is the sample covariance $\hat{\sigma}_{\mathrm{C}}^{(\ell,k)}$ between $C^{(\ell)}$ and $C^{(k)}$, defined by

$$\hat{\sigma}_{\mathrm{C}}^{(\ell,k)} = \frac{1}{n-1}\sum_{i=1}^{n}(C_i^{(\ell)} - \bar{C}_n^{(\ell)})(C_i^{(k)} - \bar{C}_n^{(k)}).$$

However, $\bar{C}_n^{(\ell)}$ can be replaced by $\nu^{(\ell)}$ in this estimator, or $\bar{C}_n^{(k)}$ can be replaced by $\nu^{(k)}$, or both (perhaps replacing $1/(n-1)$ by $1/n$ in the latter case), and the estimator remains consistent. This holds for every pair (ℓ, k), giving rise to a whole bunch of variants for $\hat{\boldsymbol{\Sigma}}_{\mathrm{C}}$. It is often the case that some (or all) elements of $\boldsymbol{\Sigma}_{\mathrm{C}}$ are known. We could then use their exact values instead of the estimators $\hat{\sigma}_{\mathrm{C}}^{(\ell,k)}$, although this is not necessarily better; see Section 6.5.7. Similarly, the ℓth element of $\boldsymbol{\Sigma}_{\mathrm{CX}}$ is usually estimated by

$$\hat{\sigma}_{\mathrm{CX}}^{(\ell)} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}_n)(C_i^{(\ell)} - \bar{C}_n^{(\ell)}),$$

in which $\bar{C}_n^{(\ell)}$ could also be replaced by $\nu^{(\ell)}$. In all these cases, we have $\hat{\boldsymbol{\beta}}_n \to \boldsymbol{\beta}^*$ with probability 1 and Theorem 6.8 applies.

6.5.4 A multinormal setting

Theorem 6.8 holds under very broad generality, but gives only asymptotic results. The next theorem, based on standard *linear regression theory*, provides exact expressions for finite n, for the case where the vectors $(X_i, \boldsymbol{C}_i^{\mathsf{t}})^{\mathsf{t}}$ are normally distributed. It follows from the fact that estimating μ and $\boldsymbol{\beta}^*$ simultaneously is equivalent to fitting a least-squares linear regression model of the form

$$X = \mu + \boldsymbol{\beta}^{\mathsf{t}}(\boldsymbol{C} - \boldsymbol{\nu}) + \epsilon$$

to the simulation data, where ϵ is normally distributed with mean zero. It uses the following refined estimator of $n\mathrm{Var}[\bar{X}_{\mathrm{ce},n}]$:

$$\tilde{S}^2_{\mathrm{ce},n} = \frac{n}{n-q-1}\left(\frac{1}{n} + \frac{(\bar{\boldsymbol{C}}_n - \boldsymbol{\nu})^{\mathsf{t}}\hat{\boldsymbol{\Sigma}}_{\mathrm{C}}^{-1}(\bar{\boldsymbol{C}}_n - \boldsymbol{\nu})}{n-1}\right)\sum_{i=1}^{n}(X_{\mathrm{ce},i} - \bar{X}_{\mathrm{ce},n})^2, \tag{6.22}$$

and the estimator $\hat{\boldsymbol{\beta}}_n$ uses the standard sample covariances. Clearly, $\tilde{S}^2_{\mathrm{ce},n} \to \sigma_{\mathrm{c}}^2$ with probability 1 when $n \to \infty$.

Theorem 6.9 *(Lavenberg and Welch 1981) Suppose that the* $(q+1)$*-dimensional vectors* $(X_i, \boldsymbol{C}_i^{\mathsf{t}})^{\mathsf{t}}$ *are* i.i.d. *normal. Then,*

$$\begin{aligned} \mathbb{E}[\bar{X}_{\mathrm{ce},n}] &= \mu, \\ \mathbb{E}[\tilde{S}^2_{\mathrm{ce},n}/n] &= \mathrm{Var}[\bar{X}_{\mathrm{ce},n}] = \frac{n-2}{n-q-2}(1 - R^2_{\mathrm{CX}})\mathrm{Var}[\bar{X}_n], \end{aligned} \tag{6.23}$$

and $\sqrt{n}(\bar{X}_{\mathrm{ce},n} - \mu)/\tilde{S}_{\mathrm{ce},n}$ *has a* Student-t$(n-q-1)$ *distribution.*

Hence, under the normality assumption, the CV estimator is unbiased and we obtain a perfectly valid confidence interval for μ from the Student distribution (it is easily constructed

because only μ is unknown in $\sqrt{n}(\bar{X}_{\mathrm{ce},n} - \mu)/\tilde{S}_{\mathrm{ce},n})$ The factor $(n-2)/(n-q-2)$ in (6.23) is the *loss factor* due to the estimation of $\boldsymbol{\beta}^*$ by the method of least squares. We have $\mathrm{Var}[\bar{X}_{\mathrm{ce},n}] < \mathrm{Var}[\bar{X}_n]$ if and only if $q < (n-2)R^2_{\mathrm{CX}}$. This indicates that the number q of control variables must remain small relative to n and that adding control variates is worthwhile only as long as it increases R^2_{CX} significantly. This is in analogy with classical regression, where the CVs $C^{(\ell)}$ are the analogs of the regression variables. More precisely, if we currently have q control variables, adding a new one will reduce the overall variance if and only if it reduces the current value of $(1-R^2_{\mathrm{CX}})$ by a proportion larger than $1/(n-q-2)$.

Alternative CV estimators can be compared via the mean or mean square of their corresponding confidence-interval half-width (Nelson 1989). Minimizing the mean-square half-width gives a different criterion for when to stop adding control variates than minimizing $\mathrm{Var}[\bar{X}_{\mathrm{ce},n}]$, because the mean and (especially) the variance of the half-width (for a given level) depends also on the number of degrees of freedom $n-q-1$ of the Student distribution.

The multinormality assumption is not always realistic. Without that assumption, the CV estimator is generally biased, because $\hat{\boldsymbol{\beta}}_n$ is correlated with $\bar{\boldsymbol{C}}_n$, and may have a variance larger than the expression given in (6.23). But for large enough n, even if there is significant departure from normality, we may rely on Theorem 6.8 to compute an asymptotically valid confidence interval.

When n is small or q is not very small relative to n and (in addition) there is significant departure from normality, something else must be done to control the bias and construct more reliable confidence intervals. Among possible remedies, we mention *batching*, *jackknifing*, *bootstrapping*, *splitting*, and making a preliminary (pilot) sample to estimate β^* (see Avramidis and Wilson 1993, Bratley, Fox, and Schrage 1987, Nelson 1990). According to Nelson (1990), the jackknife estimator appears dominated by the splitting one and the bootstrap may be computationally too costly in this context. We examine two splitting schemes in Section 6.5.5.

To improve the normality by taking advantage of the central-limit effect, Nelson (1989, 1990) recommends batching the observations in, say, close to 60 batches when $n > 60$ and $q \le 5$ (more batches for larger q). For example, if $n = 300$, regroup the observations into 60 batches of 5 observations each, take the average within each batch, and consider these "batch means" as i.i.d. normal observations for which Theorem 6.9 can be applied. If the observations are already normal, there is a small penalty in terms of increasing the loss factor $(n-2)/(n-q-2)$ and loosing degrees of freedom (thereby inflating the size of the confidence interval). Otherwise, batching tends to improve normality and thus to reduce the bias of both the mean and variance estimators $\bar{X}_{\mathrm{ce},n}$ and $\tilde{S}^2_{\mathrm{ce},n}$.

♣ Add examples and exercises.

6.5.5 Splitting for control variates

Here we divide the sample into two or more groups, and use, within each group, an estimator of $\boldsymbol{\beta}^*$ based only on the observations outside of that group. This gives an unbiased estimator, but there remains the difficulty of estimating the variance. A special case often considered is to split into n groups: Let $\hat{\boldsymbol{\beta}}_n^{-i}$ be the estimator of $\boldsymbol{\beta}^*$ based on $n-1$ observations, with observation i removed; let

$$\begin{aligned} X_{\text{cs},i} &= X_i - (\hat{\boldsymbol{\beta}}_n^{-i})^{\mathsf{t}}(\boldsymbol{C}_i - \boldsymbol{\nu}), \\ \bar{X}_{\text{cs},n} &= \frac{1}{n}\sum_{i=1}^{n} X_{\text{cs},i}, \\ S^2_{\text{cs},n} &= \frac{1}{(n-1)}\sum_{i=1}^{n}(X_{\text{cs},i} - \bar{X}_{\text{cs},n})^2. \end{aligned} \tag{6.24}$$

Then $\mathbb{E}[\bar{X}_{\text{cs},n}] = \mu$ (because $\hat{\boldsymbol{\beta}}_n^{-i}$ is independent of $\boldsymbol{C}_i$) and $S^2_{\text{cs},n}/n$ is a (biased) estimator of $\text{Var}[\bar{X}_{\text{cs},n}]$. The latter estimator would be unbiased if the $X_{\text{cs},i}$'s where independent, but they are not. Nelson (1990) shows that the properties stated in Theorem 6.8 apply to this splitting estimator. He then shows, under conditions slightly more general than those of Theorem 6.9, that splitting inflates the variance: $0 \le \text{Var}[\bar{X}_{\text{cs},n}] - \text{Var}[\bar{X}_{\text{ce},n}] \in O(n^{-3})$. Jackknifing has the same properties, but empirical studies suggest that splitting is usually better for small sample sizes. Under normality assumptions, $S^2_{\text{cs},n}/n$ tends to *underestimate* $\text{Var}[\bar{X}_{\text{cs},n}]$ when n is small. For small n (say near 30) Nelson (1990) nevertheless prefers the splitting estimator $\bar{X}_{\text{cs},n}$ over $\bar{X}_{\text{ce},n}$, because it is unbiased. Note that $\bar{X}_{\text{cs},n}$ is more costly to compute than $\bar{X}_{\text{ce},n}$, because a different coefficient must be computed for each $\boldsymbol{C}_i$.

Avramidis and Wilson (1993) devised a different splitting scheme that provides an unbiased CV estimator $\bar{X}_{\text{ca},n}$ as well as an unbiased estimator $S^2_{\text{ca},n}/n$ for the variance of $\bar{X}_{\text{ca},n}$. It operates as follows. Split the observations into m groups of $k = n/m$ observations each. For each group j, let $\hat{\boldsymbol{\beta}}_{n,j}$ be the estimator of $\boldsymbol{\beta}^*$ (the analog of $\hat{\boldsymbol{\beta}}_n$) computed solely from the k observations of group j. For each observation i, define

$$\tau(i) = \lceil i/k \rceil + 1 = \begin{cases} 2 & \text{if } 0 < i \le k; \\ 3 & \text{if } k < i \le 2k; \\ \quad\vdots & \\ m & \text{if } n - 2k < i \le n - k; \\ 1 & \text{if } n - k < i \le n, \end{cases}$$

which represents the group that follows the one to which observation i belongs, and use $\hat{\boldsymbol{\beta}}_{n,\tau(i)}$ as a CV coefficient. This gives the ith controlled observation:

$$X_{\text{ca},i} = X_i - \hat{\boldsymbol{\beta}}^{\mathsf{t}}_{n,\tau(i)}(\boldsymbol{C}_i - \boldsymbol{\nu}).$$

The *split-CV* estimator of μ is then

$$\bar{X}_{\text{ca},n} = \frac{1}{n}\sum_{i=1}^{n} X_{\text{ca},i} \tag{6.25}$$

and the associated variance estimator is

$$S^2_{\text{ca},n} = \frac{1}{(n-1)}\sum_{i=1}^{n}(X_{\text{ca},i} - \bar{X}_{\text{ca},n})^2.$$

Note that $\hat{\boldsymbol{\beta}}_{n,\tau(i)} \overset{\text{w.p.1}}{\to} \boldsymbol{\beta}^*$ when $n \to \infty$ for fixed m, so Theorem 6.8 applies to this estimator. Avramidis and Wilson (1993) prove more:

Theorem 6.10

(i) For $m \geq 2$ (m fixed), $\mathbb{E}[\bar{X}_{\mathrm{ca},n}] = \mu$. Also, as $n \to \infty$, $\bar{X}_{\mathrm{ca},n} \xrightarrow{\text{w.p.1}} \mu$, $S^2_{\mathrm{ca},n} \xrightarrow{\text{w.p.1}} \sigma_{\mathrm{c}}^2$, and

$$\frac{\sqrt{n}(\bar{X}_{\mathrm{ca},n} - \mu)}{S_{\mathrm{ca},n}} \;\Rightarrow\; \frac{\sqrt{n}(\bar{X}_{\mathrm{ca},n} - \mu)}{\sigma_{\mathrm{c}}^2} \;\Rightarrow\; N(0,1).$$

(ii) For $m \geq 3$ (m fixed), the $X_{\mathrm{ca},i}$'s are pairwise uncorrelated and $\mathbb{E}[S^2_{\mathrm{ca},n}/n] = \mathrm{Var}[\bar{X}_{\mathrm{ca},n}]$.

(iii) If $m \geq 1$ and the vector $(X_i, \boldsymbol{C}_i^{\mathsf{t}})^{\mathsf{t}}$ is multivariate normal,

$$\mathrm{Var}[\bar{X}_{\mathrm{ca},n}] \;=\; \frac{n-2m}{n-qm-2m}(1 - R^2_{\mathrm{CX}})\mathrm{Var}[\bar{X}_n]. \tag{6.26}$$

The loss factor $(n-2m)/(n-qm-2m) = (k-2)/(k-q-2)$ in (6.26) is minimized by taking m as small as possible, but the variance estimator is unbiased without the normality assumption only for $m \geq 3$. The estimator $\bar{X}_{\mathrm{ce},n}$ in Theorem 6.9 corresponds to $m = 1$, which is the best choice when we have normality. Based on these results and supported by some experimental evaluation, Avramidis and Wilson (1993) recommend $m = 3$. They also propose an approximate confidence interval based on the Student distribution for moderate n. This scheme provides unbiased estimators for both the mean and the variance. On the other hand, the estimator is more costly to compute than the straightforward $\bar{X}_{\mathrm{ce},n}$.

♣ Add examples.

6.5.6 Pilot runs are often inefficient

Making a pilot sample is a simple way of getting unbiased CV estimators. Suppose n_0 pilot observations are used to compute an estimator $\hat{\boldsymbol{\beta}}_0$ of $\boldsymbol{\beta}^*$ and that $\hat{\boldsymbol{\beta}}_0$ is used as a coefficient of the CVs for the remaining (independent) $n - n_0$ observations, yielding the mean and variance estimators:

$$\bar{X}_{\mathrm{cp},n} \;=\; \frac{1}{n-n_0}\sum_{i=n_0+1}^{n}(X_i - \hat{\boldsymbol{\beta}}_0^{\mathsf{t}}(\boldsymbol{C}_i - \boldsymbol{\nu})) \;\text{ and}$$

$$S^2_{\mathrm{cp},n} \;=\; \frac{1}{(n-n_0-1)}\sum_{i=n_0+1}^{n}(X_i - \hat{\boldsymbol{\beta}}_0^{\mathsf{t}}(\boldsymbol{C}_i - \boldsymbol{\nu}) - \bar{X}_{\mathrm{cp},n})^2.$$

Here, $\mathbb{E}[\bar{X}_{\mathrm{cp},n}] = \mu$ and $\mathbb{E}[S^2_{\mathrm{cp},n}/(n-n_0)] = \mathrm{Var}[\bar{X}_{\mathrm{cp},n}]$. Under the assumptions of Theorem 6.9,

$$\frac{\mathrm{Var}[\bar{X}_{\mathrm{cp},n}]}{\mathrm{Var}[\bar{X}_{\mathrm{ce},n}]} = \frac{n(n-q-2)(n_0-2)}{(n-n_0)(n-2)(n_0-q-2)} > 1$$

whenever $n > n_0 > q+2$ (see Section 5.3 of Ripley 1987); that is, $\bar{X}_{\mathrm{cp},n}$ has a loss factor worst than $\bar{X}_{\mathrm{ce},n}$. If there is significant departure from normality, pilot runs yield unbiased estimators, as does the splitting scheme of Avramidis and Wilson (1993), but confidence intervals based on the normal or Student distribution are still invalid for small n.

6.5.7 Known variance of the controls

Sometimes, $\boldsymbol{\Sigma}_{\mathrm{C}}$ (or part of it) is known. Suppose $\boldsymbol{\Sigma}_{\mathrm{C}}$ is known entirely and that the estimator $\hat{\boldsymbol{\beta}}_n$ is replaced in (6.18) by $\ddot{\boldsymbol{\beta}}_n = \boldsymbol{\Sigma}_{\mathrm{C}}^{-1}\hat{\boldsymbol{\Sigma}}_{\mathrm{CX}}$, leading to an estimator of μ that we denote by $\bar{X}_{\mathrm{ck},n}$. Bauer Jr. (1987) gives an estimator of $\mathrm{Var}[\bar{X}_{\mathrm{ck},n}]$. Under the assumptions of Theorem 6.9, the latter estimator is unbiased and it turns out that $\mathrm{Var}[\bar{X}_{\mathrm{ce},n}] < \mathrm{Var}[\bar{X}_{\mathrm{ck},n}] < \mathrm{Var}[\bar{X}_n]$ whenever $\mathrm{Var}[\bar{X}_{\mathrm{ce},n}] < \mathrm{Var}[\bar{X}_n]$ (see Nelson 1990). So, perhaps counter-intuitively, replacing $\hat{\boldsymbol{\beta}}_n$ by $\ddot{\boldsymbol{\beta}}_n$ is not recommended when we believe in normality. The explanation is that $\hat{\boldsymbol{\Sigma}}_{\mathrm{C}}$ acts as a nonlinear control variate (see Section 6.5.10) that improves the estimator of $\boldsymbol{\beta}^*$. In absence of normality, however, $\bar{X}_{\mathrm{ck},n}$ could be less biased and the associate confidence interval could have a better coverage than for $\bar{X}_{\mathrm{ce},n}$.

6.5.8 Multiresponse estimation

What we have discussed can be generalized to the *multiresponse* case, where μ and X become (say) p-dimensional vectors $\boldsymbol{\mu}$ and $\boldsymbol{X}$. The variance can then be replaced by the generalized variance (the determinant of the covariance matrix) or by the trace of the covariance matrix (Rubinstein and Marcus 1985, Venkatraman and Wilson 1986, Yang and Nelson 1992). Both the determinant and the trace of the covariance matrix of $\boldsymbol{X}_{\mathrm{c}}$ in this case are minimized by setting $\boldsymbol{\beta}$ equal to the matrix $\boldsymbol{\beta}^* = \boldsymbol{\Sigma}_{\mathrm{CX}}\boldsymbol{\Sigma}_{\mathrm{C}}^{-1}$, where $\boldsymbol{\Sigma}_{\mathrm{CX}}$ is the covariance matrix between $\boldsymbol{C}$ and $\boldsymbol{X}$. The minimal generalized variance is then

$$\det(\mathrm{Cov}[\boldsymbol{X}_{\mathrm{c}}]) \;=\; (1 - R^2_{\mathrm{CX}}) \cdot \det(\mathrm{Cov}[\boldsymbol{X}]),$$

where

$$R^2_{\mathrm{CX}} = \det(\boldsymbol{\Sigma}_{\mathrm{CX}}^{\mathsf{t}}\boldsymbol{\Sigma}_{\mathrm{C}}^{-1}\boldsymbol{\Sigma}_{\mathrm{CX}})/\det(\boldsymbol{\Sigma}_{\mathrm{X}}). \tag{6.27}$$

In practice, $\boldsymbol{\beta}^*$ could be replaced by its sample estimator $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\Sigma}}_{\mathrm{CX}}\hat{\boldsymbol{\Sigma}}_{\mathrm{C}}^{-1}$ as in the single-response case, and a similar analysis can be made. Under the assumption that $(\boldsymbol{X}^{\mathsf{t}}, \boldsymbol{C}^{\mathsf{t}})^{\mathsf{t}}$ is multivariate normal, a generalization of Theorem 6.9 says that $\mathbb{E}[\bar{\boldsymbol{X}}_{\mathrm{ce},n}] = \boldsymbol{\mu}$,

$$\frac{\det(\mathrm{Cov}[\bar{\boldsymbol{X}}_{\mathrm{ce},n}])}{\det(\mathrm{Cov}[\bar{\boldsymbol{X}}_n])} = \left(\frac{n-2}{n-q-2}\right)^p (1 - R^2_{\mathrm{CX}}),$$

so the loss factor is now $[(n-2)/(n-q-2)]^p$, and the random variable

$$\frac{(\bar{\boldsymbol{X}}_{\mathrm{ce},n} - \boldsymbol{\mu})^{\mathsf{t}} G^{-1} (\bar{\boldsymbol{X}}_{\mathrm{ce},n} - \boldsymbol{\mu})(n-p-q)}{(1 + (\bar{\boldsymbol{C}} - \boldsymbol{\nu})^{\mathsf{t}}\hat{\boldsymbol{\Sigma}}_{\mathrm{C}}^{-1}(\bar{\boldsymbol{C}} - \boldsymbol{\nu})n/(n-1))p}$$

has the $\mathsf{F}(p, n-p-q)$ distribution, where $G = (\hat{\boldsymbol{\Sigma}}_{\mathrm{X}} - \hat{\boldsymbol{\Sigma}}_{\mathrm{C}}^{-1}\hat{\boldsymbol{\Sigma}}_{\mathrm{CX}}\hat{\boldsymbol{\Sigma}}_{\mathrm{CX}})(n-1)/n$. The latter permits us to construct a valid confidence region for $\boldsymbol{\mu}$. Yang and Nelson (1992) study the effect of batching to improve normality, and the penalty incurred by batching when not necessary, in the multiresponse context. Roughly, if the observations are batched into k groups and the batch means are i.i.d. multinormal, the above holds with n replaced by k in the loss factor and in the F statistic. It is then easily concluded that, as a general principle, the larger are q and p, the larger the number of batches should be for the loss factor not to

be significantly affected. For $p \le 5$ and $q \le 5$, Yang and Nelson (1992) recommend $k \le 100$. They also analyze the behavior of the mean and mean square error of the confidence region volume as functions of p, q, k, and the confidence level.

In the multiresponse context, Bauer Jr. and Wilson (1992) have proposed a procedure for selecting a "good" subset of controls from a pool of candidates, with the objective of minimizing the mean-square volume of the confidence region. The procedure makes a tradeoff between the variance inflation due to the estimation of $\boldsymbol{\beta}^*$ with more controls, and the variance reduction due to a larger R^2_{CX}. Bauer Jr. and Wilson (1992) also empirically observed a better coverage for the confidence regions based on minimizing the mean-square volume with the estimator $\bar{\boldsymbol{X}}_{\mathrm{ck},n}$ rather than with $\bar{\boldsymbol{X}}_{\mathrm{ce},n}$.

6.5.9 Linear metamodel

An even more general framework is that of a CV scheme for estimating a linear multiresponse simulation metamodel of the form

$$\boldsymbol{X} = \boldsymbol{\Theta}\boldsymbol{Z} + \boldsymbol{\xi}$$

where $\boldsymbol{Z}$ is an m-dimensional vector of design variables, $\boldsymbol{\Theta}$ is a $p \times m$ matrix of (unknown) metamodel coefficients, $\boldsymbol{X}$ is the p-dimensional response, and $\boldsymbol{\xi}$ is a vector of random noise. The idea is to "explain" part of the noise by a q-dimensional control vector $\boldsymbol{C}$ of known mean $\boldsymbol{\nu}$. This yields the metamodel

$$\boldsymbol{X} = \boldsymbol{\Theta}\boldsymbol{Z} + \boldsymbol{\beta}^{\mathsf{t}}(\boldsymbol{C} - \boldsymbol{\nu}) + \boldsymbol{\epsilon}.$$

Porta Nova and Wilson (1989, 1993) analyze such a model; they provide least square estimators and expressions for their generalized variance, and they study special cases. Under appropriate normality assumptions, the minimum variance ratio and loss factor with total sample size n (i.e., a total of n design points, counting all replications) are

$$(1 - R^2_{\mathrm{CX}})^m \quad \text{and} \quad \left(\frac{n-m-1}{n-m-q-1}\right)^{mp},$$

respectively. The case of estimating the mean $\boldsymbol{\mu}$ of a single multiresponse (previously discussed) corresponds to $m = 1$.

6.5.10 Nonlinear functions of means and nonlinear controls

So far, we used control variates to apply linear corrections to mean estimators. Following Glynn and Whitt (1989) and Glynn (1994), we now consider a more general setting where the goal is to estimate a nonlinear function of a mean vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_d)^{\mathsf{t}}$, say $g(\boldsymbol{\mu})$, where $\boldsymbol{\mu}$ is estimated by $\bar{\boldsymbol{X}}_n$, as in Section 5.4.1.

6.5.10.1 Linear controls for a function of means Adding a linear control variate in this case gives the estimator

$$h(\bar{\boldsymbol{X}}_n, \bar{\boldsymbol{C}}_n) = g(\bar{\boldsymbol{X}}_n) - \boldsymbol{\beta}^{\mathsf{t}}(\bar{\boldsymbol{C}}_n - \boldsymbol{\nu}), \tag{6.28}$$

where $\bar{\boldsymbol{C}}_n$) and $\boldsymbol{\nu}$ have the same interpretation as before. By applying the delta theorem, we obtain that

$$g(\bar{\boldsymbol{X}}_n) - \boldsymbol{\beta}^{\mathsf{t}}(\bar{\boldsymbol{C}}_n - \boldsymbol{\nu}) \;\Rightarrow\; N(0, \sigma_{\mathrm{h}}^2)$$

when $n \to \infty$, with the asymptotic variance constant

$$\sigma_{\mathrm{h}}^2 = \nabla g(\boldsymbol{\mu})^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{X}} \nabla g(\boldsymbol{\mu}) - 2\nabla g(\boldsymbol{\mu})^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{CX}}^{\mathsf{t}} \boldsymbol{\beta} + \boldsymbol{\beta}^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{C}} \boldsymbol{\beta}. \tag{6.29}$$

This result is actually a special case of Theorem 6.11 below. It can be used to construct an asymptotically valid confidence interval for $g(\boldsymbol{\mu})$, based on the normal distribution.

The variance constant σ_{h}^2 is minimized by taking

$$\boldsymbol{\beta} = \boldsymbol{\beta}^* = \boldsymbol{\Sigma}_{\mathrm{C}}^{-1} \boldsymbol{\Sigma}_{\mathrm{CX}} \nabla g(\boldsymbol{\mu}) \tag{6.30}$$

and the minimal variance σ_{c}^2 in (6.17) becomes

$$\sigma_{\mathrm{c}}^2 = \nabla g(\boldsymbol{\mu})^{\mathsf{t}} (\boldsymbol{\Sigma}_{\mathrm{X}} - \boldsymbol{\Sigma}_{\mathrm{CX}}^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{C}}^{-1} \boldsymbol{\Sigma}_{\mathrm{CX}}) \nabla g(\boldsymbol{\mu}) = (1 - R_{\mathrm{CX}}^2) \sigma_{\mathrm{g}}^2 \tag{6.31}$$

where R_{CX}^2 is the same as in Eq. (6.27), and$\sigma_{\mathrm{g}}^2 = \nabla g(\boldsymbol{\mu})^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{X}} \nabla g(\boldsymbol{\mu})$ is the variance constant without the CV, exactly as in Corollary 5.6. The nonlinear CV reduces the variance if and only if $\sigma_{\mathrm{h}}^2 < \sigma_{\mathrm{g}}^2$, if and only if

$$[2\nabla g(\boldsymbol{\mu})^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{CX}}^{\mathsf{t}} + \nabla_{\boldsymbol{\nu}} h(\boldsymbol{\mu}, \boldsymbol{\nu})^{\mathsf{t}} \boldsymbol{\Sigma}_{\mathrm{C}}] \nabla_{\boldsymbol{\nu}} h(\boldsymbol{\mu}, \boldsymbol{\nu}) < 0.$$

In practice, $\boldsymbol{\beta}^*$ can be replaced by any consistent estimator $\hat{\boldsymbol{\beta}}_n$ and the analogue of Theorem 6.8 remains valid, so there is no loss of asymptotic efficiency by having to estimate $\boldsymbol{\beta}^*$. For instance, we can take

$$\hat{\boldsymbol{\beta}}_n = \hat{\boldsymbol{\Sigma}}_{\mathrm{C}}^{-1} \hat{\boldsymbol{\Sigma}}_{\mathrm{CX}} \nabla g(\bar{\boldsymbol{X}}_n).$$

To compute a confidence interval, σ_{h}^2 also needs to be replaced by an estimator.

Example 6.14 For a ratio estimation, we have $g(\boldsymbol{\mu}) = \mu_1/\mu_2$ where $\boldsymbol{\mu} = (\mu_1, \mu_2)^{\mathsf{t}}$ and $\nabla g(\boldsymbol{\mu}) = (1/\mu_2, -\mu_1/\mu_2^2)^{\mathsf{t}}$. An estimator of μ_1/μ_2 with linear control variables can be defined as

$$\frac{\hat{\mu}_1}{\hat{\mu}_2} - \hat{\boldsymbol{\beta}}_n^{\mathsf{t}}(\bar{\boldsymbol{C}}_n - \boldsymbol{\nu}) = \frac{\hat{\mu}_1}{\hat{\mu}_2} - (1/\hat{\mu}_2, -\hat{\mu}_1/\hat{\mu}_2^2)^{\mathsf{t}} \hat{\boldsymbol{\Sigma}}_{\mathrm{CX}}^{\mathsf{t}} \hat{\boldsymbol{\Sigma}}_{\mathrm{C}}^{-1} (\bar{\boldsymbol{C}}_n - \boldsymbol{\nu})$$

where $\hat{\boldsymbol{\Sigma}}_{\mathrm{CX}}$ is the matrix of empirical covariances between the components of $\boldsymbol{C}$ and $(\hat{\mu}_1, \hat{\mu}_2)$. □

6.5.10.2 A more general setting with nonlinear controls We now extend the previous setting to a framework where the estimator has the general form $h(\bar{\boldsymbol{X}}_n, \bar{\boldsymbol{C}}_n)$ for some function h, where $\bar{\boldsymbol{X}}_n$ can be interpreted as a natural estimator of $\boldsymbol{\mu}$ (usually an average), and $\bar{\boldsymbol{C}}_n$ can be interpreted as an "average" of control variate vectors, both for a computing budget of n simulation runs. The estimator is not necessarily a linear function of the control variates as in (6.28). We will assume that $(\bar{\boldsymbol{X}}_n, \bar{\boldsymbol{C}}_n)$ converges to $(\boldsymbol{\mu}, \boldsymbol{\nu})$ and obeys a CLT when $n \to \infty$ (see Assumption 6.2), and that the function $h : \mathbb{R}^{d+q} \to \mathbb{R}$ satisfies $h(\boldsymbol{x}, \boldsymbol{\nu}) = g(\boldsymbol{x})$ and is continuously differentiable in a neighborhood of $(\boldsymbol{\mu}, \boldsymbol{\nu})$. We do not assume that $\bar{\boldsymbol{C}}_n$ is the

average of i.i.d. random variables nor that $\mathbb{E}[\bar{\boldsymbol{C}}_n] = \boldsymbol{\nu}$ for finite n. We replace Assumption 6.1 by the following:

Assumption 6.2 We have

$$\sqrt{n}\begin{pmatrix}\bar{\boldsymbol{X}}_n - \boldsymbol{\mu} \\ \bar{\boldsymbol{C}}_n - \boldsymbol{\nu}\end{pmatrix} \;\Rightarrow\; \mathsf{N}(0,\, \boldsymbol{\Sigma})$$

where

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{\mathrm{X}} & \boldsymbol{\Sigma}_{\mathrm{CX}}^{\mathsf{t}} \\ \boldsymbol{\Sigma}_{\mathrm{CX}} & \boldsymbol{\Sigma}_{\mathrm{C}} \end{pmatrix}$$

is a finite and positive definite covariance matrix, $\boldsymbol{\Sigma}_{\mathrm{X}}$ is $d \times d$, and $\boldsymbol{\Sigma}_{\mathrm{C}}$ is $q \times q$. □

Theorem 6.11 *Under Assumption 6.2, when $n \to \infty$, we have*

$$\begin{aligned}
& \sqrt{n}[h(\bar{\boldsymbol{X}}_n,\, \bar{\boldsymbol{C}}_n) - g(\boldsymbol{\mu})] \\
\Rightarrow\quad & \sqrt{n}\nabla h(\boldsymbol{\mu},\boldsymbol{\nu})^{\mathsf{t}}\begin{pmatrix}\bar{\boldsymbol{X}}_n - \boldsymbol{\mu} \\ \bar{\boldsymbol{C}}_n - \boldsymbol{\nu}\end{pmatrix} && (6.32)\\
=\quad & \sqrt{n}[\nabla g(\boldsymbol{\mu})^{\mathsf{t}}(\bar{\boldsymbol{X}}_n - \boldsymbol{\mu}) + \nabla_{\boldsymbol{\nu}} h(\boldsymbol{\mu},\boldsymbol{\nu})^{\mathsf{t}}(\bar{\boldsymbol{C}}_n - \boldsymbol{\nu})] && (6.33)\\
\Rightarrow\quad & \mathsf{N}(0, \sigma_{\mathrm{h}}^2) && (6.34)
\end{aligned}$$

where $\nabla_{\boldsymbol{\nu}} h$ is the gradient of h with respect to its last q coordinates and

$$\begin{aligned}
\sigma_{\mathrm{h}}^2 &= \nabla h(\boldsymbol{\mu},\boldsymbol{\nu})^{\mathsf{t}}\boldsymbol{\Sigma}\nabla h(\boldsymbol{\mu},\boldsymbol{\nu}) && (6.35)\\
&= \nabla g(\boldsymbol{\mu})^{\mathsf{t}}\boldsymbol{\Sigma}_{\mathrm{X}}\nabla g(\boldsymbol{\mu}) + 2\,\nabla g(\boldsymbol{\mu})^{\mathsf{t}}\boldsymbol{\Sigma}_{\mathrm{CX}}^{\mathsf{t}}\nabla_{\boldsymbol{\nu}} h(\boldsymbol{\mu},\boldsymbol{\nu}) \\
&\quad + \nabla_{\boldsymbol{\nu}} h(\boldsymbol{\mu},\boldsymbol{\nu})^{\mathsf{t}}\boldsymbol{\Sigma}_{\mathrm{C}}\nabla_{\boldsymbol{\nu}} h(\boldsymbol{\mu},\boldsymbol{\nu}).
\end{aligned}$$

Proof. Eq. (6.32) comes by doing a first-order Taylor expansion of $h(\bar{\boldsymbol{X}}_n, \bar{\boldsymbol{C}}_n)$ around $h(\boldsymbol{\mu},\boldsymbol{\nu}) = g(\boldsymbol{\mu})$ and invoking the continuous differentiability of h, Assumption 6.2, and the continuous mapping theorem (Theorem A.10). Eq. (6.33) holds because $\nabla_{\boldsymbol{\mu}} h(\boldsymbol{\mu},\boldsymbol{\nu}) = \nabla g(\boldsymbol{\mu})$, where $\nabla_{\boldsymbol{\mu}} h$ is the gradient of h with respect to its first d components. The CLT (6.34) follows by combining (6.33) with Assumption 6.2. The result is also a direct consequence of the delta theorem together with Assumption 6.2.

From (6.33) and Corollary 5.6, we have that

$$h(\bar{\boldsymbol{X}}_n,\, \bar{\boldsymbol{C}}_n) \;=\; g(\bar{\boldsymbol{X}}_n) + \nabla_{\boldsymbol{\nu}} h(\boldsymbol{\mu},\boldsymbol{\nu})^{\mathsf{t}}(\bar{\boldsymbol{C}}_n - \boldsymbol{\nu}) + o_p(n^{-1/2}), \tag{6.36}$$

which means that the nonlinear CV scheme is asymptotically equivalent (for large n) to the linear CV scheme with $\boldsymbol{\beta} = -\nabla_{\boldsymbol{\nu}} h(\boldsymbol{\mu},\boldsymbol{\nu})$. This choice of coefficient $\boldsymbol{\beta}$ is not necessarily optimal for the linear CV (and can be far from it). In general, we can do better with $\boldsymbol{\beta}^*$. Thus, every nonlinear CV scheme that satisfies Assumption 6.2 is asymptotically dominated by a linear scheme that uses its optimal coefficient $\boldsymbol{\beta}^*$. This shows that from the asymptotic standpoint, there is no loss in restricting ourselves to linear CVs. On the other hand, asymptotics tell only part of the story. For finite n, *nonlinear controls* may sometimes beat

the corresponding linear ones in terms of MSE and efficiency. This is further discussed in Section 6.5.12.

In the present setting, $\bar{\boldsymbol{C}}_n$ is not necessarily the average of i.i.d. random variables and $\mathbb{E}[\bar{\boldsymbol{C}}_n]$ is not necessarily equal to $\boldsymbol{\nu}$; we require only the weaker condition that $\sqrt{n}(\bar{\boldsymbol{C}}_n - \boldsymbol{\nu}) \Rightarrow N(0, \boldsymbol{\Sigma}_c)$. This additional freedom can be useful. For example, while estimating a ratio of expectations by a ratio of sample means, Fleming, Schaeffer, and Simon (1995) use other ratios of samples means as linear CVs for the first ratio. These CVs are consistent but biased. In this context, the components of the vector $\bar{\boldsymbol{C}}_n$ are the ratios used as CVs and the components of $\boldsymbol{\nu}$ are the corresponding ratios of expectations. For an application involving the estimation of blocking probabilities in a digital cellular phone system, these authors obtain large efficiency improvements with those CVs, and even more so when combining their CV approach with importance sampling, when the blocking probabilities are small. By using the individual sample means as linear CVs instead of the ratios, the variance of the CV estimator would be approximately the same (for large enough n), but there would be twice as many CVs, so twice as many parameters to estimate.

Examples of estimators with a *one-dimensional nonlinear CV* include

$$h(\bar{X}_n, \bar{C}_n) = g(\bar{X}_n)\bar{C}_n/\nu \tag{6.37}$$

$$h(\bar{X}_n, \bar{C}_n) = g(\bar{X}_n)\nu/\bar{C}_n \tag{6.38}$$

$$h(\bar{X}_n, \bar{C}_n) = g(\bar{X}_n)^{\bar{C}_n/\nu} \tag{6.39}$$

$$h(\bar{X}_n, \bar{C}_n) = g(\bar{X}_n)^{\nu/\bar{C}_n}, \tag{6.40}$$

where $\mathbb{E}[C_n] = \nu$. The *ratio CV* estimators (6.37) and (6.38) were studied by Cochran (1977) and Kleijnen (1974), while the *power CV* estimators (6.39) and (6.40) have been suggested by Nelson (1987). Those nonlinear CV estimator are clearly biased, but in some special situations they may decrease the variance enough to reduce the MSE.

♣ Give concrete numerical illustrations.

Example 6.15 Replacing $\boldsymbol{\beta}^*$ by $\hat{\boldsymbol{\beta}}_n$ as in Eq. (6.18) and replacing $\boldsymbol{\Sigma}_C$ by $\hat{\boldsymbol{\Sigma}}_C$ (or vice-versa) as discussed in Section 6.5.7 can be interpreted as special cases of nonlinear control variates. We already saw in Theorem 6.8 that this makes no difference in the asymptotic variance constant, but there can be a significant difference for finite n. □

The framework of Glynn (1994) is slightly more general than what we have discussed so far; the process $\{(\bar{\boldsymbol{X}}_n, \bar{\boldsymbol{C}}_n), n \geq 1\}$ is replaced by a continuous-time process $\{(\boldsymbol{X}(t), \boldsymbol{C}(t)), t \geq 0\}$ there, and it covers the case where $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ are limits of continuous-time averages. In that context, if $\boldsymbol{\mu}$ is to be estimated by a single (long) simulation run, different methods could be used to estimate the variance, including batch means and a regenerative approach. The latter transforms the problem into one of ratio estimation, while the former is studied in Yang and Nelson (1992).

6.5.11 Moments matching

The *moment matching* methods surveyed by Boyle, Broadie, and Glasserman (1997b), Section 2.4, are strongly related to the use of nonlinear control variates. One such method was

proposed by Barraquand (1995) under the name of *quadratic resampling*, in the context of financial simulations. These methods are biased and it is generally difficult to control the bias and assess the error. They are also asymptotically dominated by using the corresponding control variates, as explained below.

Suppose that i.i.d. random variables $Z_1, \ldots, Z_n$ with known first moments (e.g., known mean $\mu_z = \mathbb{E}[Z_i]$, known variance $\sigma_z^2 = \mathrm{Var}[Z_i]$, etc.) are generated during the simulation. The idea of moment matching is to modify the Z_i slightly so that the empirical moments match the theoretical ones. For example, to match the first moment, it suffices to replace each Z_i by $\tilde{Z}_i = Z_i + \mu_z - \bar{Z}_n$. The average of the $\tilde{Z}_i$'s is then equal to the theoretical average μ_z. To match the first two moments (mean and variance), we would replace each Z_i by

$$\tilde{Z}_i = \mu_z + (Z_i - \bar{Z}_n)\sigma_z / S_{z,n} \tag{6.41}$$

where $S_{z,n}^2$ is the sample variance of the Z_i's (Exercise 6.12). We can also match the covariances when more than one type of random variable is considered, etc.

Sometimes, we can choose the level at which the moments are matched. For example, if n i.i.d. replicates of the value $S(T)$ of a geometric Brownian motion at time T are generated as

$$Y_i = S(0)\exp\left[(r - \sigma^2/2)T + \sigma\sqrt{T}Z_i\right] \qquad i = 1, \ldots, n,$$

where the Z_i are i.i.d. $N(0,1)$, then we can apply moment matching either to the Z_i's or to the Y_i's. Boyle, Broadie, and Glasserman (1997b), page 1279, report numerical experiments for an option pricing example involving this particular setting.

[1]

Suppose that the quantity of interest is $\mu = \mathbb{E}[h(Z_i)]$ where h is twice continuously differentiable at Z_i with probability 1. Let $X_i = h(Z_i)$ and $X_{\mathrm{mm},i} = h(\tilde{Z}_i)$, where $\tilde{Z}_i$ is defined by (6.41) (the first two moments are matched). The *moment matching estimator*

$$\bar{X}_{\mathrm{mm},n} \stackrel{\mathrm{def}}{=} \frac{1}{n}\sum_{i=1}^{n} X_{\mathrm{mm},i} \tag{6.42}$$

does not fit the nonlinear CV framework of Section 6.5.10, because it is not a function of $(\bar{X}_n, \bar{Z}_n, \sigma_z/S_{z,n})$ only. However, we can show directly (see Exercise 6.12 and Boyle, Broadie, and Glasserman 1997b) that

$$\sqrt{n}(\bar{X}_{\mathrm{mm},n} - \mu) \quad \Rightarrow \quad \sqrt{n}(\bar{X}_n - \beta_1 C^{(1)} - \beta_2 C^{(2)} - \mu) \tag{6.43}$$

as $n \to \infty$, where $C^{(1)} = 1 - \sigma_z/S_{z,n}$, $C^{(2)} = \bar{Z}_n\sigma_z/S_{z,n} - \mu_z$, $\beta_1 = \mathbb{E}[Z_i h'(Z_i)]$, $\beta_2 = \mathbb{E}[h'(Z_i)]$, and $h'(Z_i)$ is the derivative of h evaluated at Z_i. This is asymptotically equivalent to using $C^{(1)}$ and $C^{(2)}$ as control variates with coefficients β_1 and β_2. But these coefficients are not necessarily optimal. Using these control variates with their optimal coefficients is at least as good (and usually better) than matching moments, at least for large n. Furthermore, in general, $\mathbb{E}[C^{(1)}] \neq 0$, $\mathbb{E}[C^{(1)}] \neq 0$, and these expectations are unknown. So even with constant coefficients, these control variates are biased. It would seem more natural to use the control variates $\bar{Z}_n - \mu_z$ and $S_{z,n}^2 - \sigma_z^2$ instead; their mean is zero. It is also unclear how to estimate

[1] From Pierre: We should compare the MSE's for some numerical examples.

$\mathrm{Var}[\bar{X}_{\mathrm{mm},n}]$ for computing confidence intervals. An easy way is to replicate the entire scheme, say, m times, compute the empirical variance of the m i.i.d. copies of $\bar{X}_{\mathrm{mm},n}$, and compute a confidence interval by assuming that $\bar{X}_{\mathrm{mm},n}$ is normally distributed and using the Student distribution with $m-1$ degrees of freedom.

6.5.12 Biased control variates

The nonlinear control variates studied in Section 6.5.10 are examples of biased control variates. More generally, allowing bias sometimes permit one to consider control variates C whose expectation is known only approximately, but whose correlation with X is so large that using them reduces the MSE significantly (and improves the efficiency) despite the introduction of some bias. Schmeiser, Taafe, and Wang (2001) study this type of situation and provide examples.

6.5.13 Examples

Example 6.16 ♣ Try all that CV stuff on a call center example. □

Example 6.17 For the Asian call option model of Example 1.11, we can derive an explicit pricing formula for the option if the arithmetic average is replaced by a *geometric average*, i.e., if the payoff discounted to time 0 is

$$C = e^{-rT}\max(0,\, Y - K)\,. \tag{6.44}$$

where $Y = \prod_{j=1}^{d} S(t_j)^{1/d}$. Here,

$$\begin{aligned}\ln Y &= \frac{1}{d}\sum_{j=1}^{d}\ln(S(t_j)) \\ &= \ln(S(0)) + \frac{1}{d}\sum_{j=1}^{d}\left((r-\sigma^2/2)t_j + \sigma B(t_j)\right)\end{aligned}$$

has the normal distribution with mean

$$m_{\mathrm{y}} = \ln S(0) + \frac{(r-\sigma^2/2)}{d}\sum_{j=1}^{d} t_j$$

and variance

$$\begin{aligned}s_{\mathrm{y}}^2 &= \frac{\sigma^2}{d^2}\mathrm{Var}\left[\sum_{j=1}^{d} B(t_j)\right] \\ &= \frac{\sigma^2}{d^2}\mathrm{Var}\left[\sum_{j=1}^{d}(d-j+1)(B(t_j)-B(t_{j-1}))\right] \\ &= \frac{\sigma^2}{d^2}\sum_{j=1}^{d}(t_j - t_{j-1})(d-j+1)^2,\end{aligned}$$

where $t_0 = 0$. By exploiting the fact that Y has the lognormal distribution and using the same approach as for deriving the Black-Scholes formula (see, e.g., Hull 1997, page 466), we obtain that

$$\mathbb{E}[C] = e^{-rT}\left[\exp(m_{\mathrm{y}} + s_{\mathrm{y}}^2/2)\Phi(z_0 + s_{\mathrm{y}}) - K\Phi(z_0)\right],$$

where $z_0 = (-\ln K + m_{\mathrm{y}})/s_{\mathrm{y}}$. [2]

Knowing the expectation of C, we can use it as a control variable for estimating $\mathbb{E}[X]$, where X is the discounted payoff of the option based on the arithmetic average, given by

$$X = e^{-rT}\max\left(0,\ \frac{1}{d}\sum_{j=1}^{d} S(t_j) - K\right). \tag{6.45}$$

This was suggested by Kemna and Vorst (1990). When C and X are computed from the same simulation of the process $\{S(t),\ t \geq 0\}$, they are highly correlated, especially if $\sigma^2(t_d - t_1)$ is relatively small (roughly). The resulting estimator is $X_{\mathrm{c}} = X - \beta(C - \mathbb{E}[C])$ for an appropriate constant β. With this method, Lemieux and L'Ecuyer (1998) have observed variance reductions by factors of up to 10^6 for some examples. A similar approach can be used for pricing Asian put options.

Another potential control variate for this example is $\sum_{j=1}^{d} S(t_j)$. Recall that each $S(t_j)/S(0)$ is lognormal with parameters $((r - \sigma^2/2)t_j,\ \sigma^2 t_j)$, so its mean is $\mathbb{E}[S(t_j)/S(0)] = e^{rt_j}$ and the CV has expectation $S(0)\sum_{j=1}^{d} e^{rt_j}$.

You are asked to experiment with these control variates in Exercise 6.9. Other control variates for this problem, based on Laplace transforms, have been explored by Fu, Madan, and Wang (1998). □

Example 6.18 In a *down-and-out call option* with *barrier* ℓ effective in discrete time, the discounted payoff is

$$X = e^{-rT}\max\left(0,\ S(T) - K\right)\mathbb{I}[\min(S(t_1), \ldots, S(t_d)) > \ell], \tag{6.46}$$

for some observation dates $0 \leq t_1 < \cdots < t_d \leq T$, where ℓ is a constant (the barrier). In other words, the payoff is the same as in the standard Black-Scholes model but the option is *knocked out* whenever $S(t) \leq \ell$ at one of the observation dates t_j. Boyle, Broadie, and Glasserman (1997b), page 1282, suggest using as a control variate the payoff of the standard European option, $C = e^{-rT}\max\left(0,\ S(T) - K\right)$ (i.e., with $\ell = -\infty$), whose expectation is given by Black-Scholes formula (1.9). This CV is especially effective when the knock-out probability is small.

A closed-form formula is available for $\mathbb{E}[X]$ in the case where the asset is monitored in continuous time, i.e., if $\min(S(t_1), \ldots, S(t_d))$ is replaced by $\min_{0 \leq t \leq T} S(t)$. This suggests that the payoff of the continuously monitored option could be used as another control variate. □

Example 6.19 Hsu and Nelson (1990) introduce CV estimators for quantiles of a distribution and study their properties without the multinormality assumption. See also Avramidis and Wilson (1998). □

[2] From Pierre: The ordinary Black-Scholes formula is only a special case of this. Explain.

Example 6.20 (Glynn and Szechtman 2002) Let $C_1, C_2, \cdots$ be i.i.d. random variables with known mean ν. If τ is a random variable, in general it is *not true* that $(1/\tau)\sum_{j=1}^{\tau} C_j$ has expectation ν, so we cannot use this average as a control variate.

However, if τ is a stopping time with respect to some filtration $\{\mathcal{F}_j,\, j \geq 1\}$ to which $\{C_j,\, j \geq 1\}$ is adapted (i.e., each C_j is $\mathcal{F}_j$-measurable), then Wald's first-order identity ensures that $C = \sum_{j=1}^{\tau}(C_j - \nu)$ has expectation $\mathbb{E}[C] = \mathbb{E}[\tau]\mathbb{E}[C_j - \nu] = 0$. This C can then be used as a control variate. Wald's second-order identity also tells us that $\mathrm{Var}[C] = \mathbb{E}[\tau]\mathrm{Var}[C_j]$. This can be easily generalized to the case where the C_j's are random vectors. □

6.6 Conditional Monte Carlo

The idea of *conditional Monte Carlo (CMC)* is to replace the estimator X by its conditional expectation given another random variable Z, or more generally a collection of information $\mathcal{G}$ (a σ-field, to be mathematically precise), that contains too little information to compute the value of X (i.e., such that X is not $\mathcal{G}$-measurable), but enough to compute (or approximate) its conditional expectation. Here, $\mathcal{G}$ can be interpreted as *partial information* from the sample path of the simulation model. (Some writers reserve the expression *conditional Monte Carlo* for the situation where Monte Carlo simulation is used to estimate a conditional expectation for its own sake, but its use to refer to the efficiency improvement techniques described in this section is now well spread.) The CMC estimator can be written as

$$X_{\mathrm{e}} \stackrel{\mathrm{def}}{=} \mathbb{E}[X \mid \mathcal{G}]. \tag{6.47}$$

Clearly, $\mathbb{E}[X_{\mathrm{e}}] = \mathbb{E}[\mathbb{E}[X \mid \mathcal{G}]] = \mathbb{E}[X]$. Using the well-known general variance decomposition (Proposition A.5 in the Appendix)

$$\mathrm{Var}[X] = \mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}]] + \mathrm{Var}[\mathbb{E}[X \mid \mathcal{G}]], \tag{6.48}$$

we find that

$$\mathrm{Var}[X_{\mathrm{e}}] \;=\; \mathrm{Var}[X] - \mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}]] \;\leq\; \mathrm{Var}[X]. \tag{6.49}$$

This inequality is a special case of the Rao-Blackwell theorem (see, e.g., Lehmann and Casella 1998, page 47), in which the variance can be replaced by a more general convex *loss* (or *penalty*) function of X.

We can interpret the first term on the right side of (6.48) as the (expected) residual variability once the information in $\mathcal{G}$ is known, and the second term as the variability due to the randomness of $\mathcal{G}$. Replacing X by X_{e} erases the first term. In contrast, stratification with deterministic allocation (Section 6.8) zeros out the second term by determining $\mathcal{G}$ a priori (in that context, $\mathcal{G}$ represents the choice of stratum).

If a σ-field $\mathcal{G}_1$ contains less pertinent information to guess the value of X than another σ-field $\mathcal{G}_2$ (e.g., if $\mathcal{G}_1 \subset \mathcal{G}_2$), then $\mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}_1]] \geq \mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}_2]]$ (Exercise 6.14) and therefore, by (6.49), $\mathbb{E}[X \mid \mathcal{G}_1]$ has less variance than $\mathbb{E}[X \mid \mathcal{G}_2]$. Generally speaking, the lesser the information in $\mathcal{G}$, the smaller the variance of $X_{\mathrm{e}} = \mathbb{E}[X \mid \mathcal{G}]$, but the harder it is to compute this conditional expectation, because it requires integrating out more. If $\mathcal{G}$ contains too little information, X_{e} may become too expensive or impossible to compute. There is a trade-off in terms of efficiency. It may also happen that X_{e} is *less expensive* to compute than X, a

win-win situation. Two extreme cases: If $\mathcal{G}$ gives *no information* at all on X, the variance of X_e is reduced to 0, because $X_e = \mathbb{E}[X \mid \mathcal{G}] = \mathbb{E}[X]$, but computing X_e amounts to computing $\mathbb{E}[X]$ analytically. On the other hand, if X is $\mathcal{G}$-measurable (i.e., it can be computed when the information in $\mathcal{G}$ is know), $\mathrm{Var}[X_e] = \mathrm{Var}[X]$.

Remark. Since $X - X_e$ has zero expectation, it may come to mind to use it as a control variate jointly with the CMC estimator. However, the fact that the difference $X - X_e$ is uncorrelated with X_e, i.e.,

$$\mathrm{Cov}[X - X_e,\, X_e] = \mathbb{E}[(X - X_e)X_e] = 0 \tag{6.50}$$

(see Exercise 6.13), knocks off this idea. In other words, this difference can be seen as *independent noise* added to X_e.

Example 6.21 We return to the *stochastic activity network* model of Example 1.4, where we want to estimate $\mu = \mathbb{P}[T > x]$, the probability that the project completion time T exceeds x. The naive estimator is the indicator $\mathbb{I}[T > x]$. A CMC estimator can be defined as follows (Avramidis and Wilson 1996 and references therein). Select a set of activities $\mathcal{L} \subseteq \mathcal{A}$ such that each directed path from the source to the sink contains exactly one activity from $\mathcal{L}$. This $\mathcal{L}$ is called a *uniformly directed cutset*. Now let $\mathcal{G}$ represent the durations of the activities in $\mathcal{B} = \mathcal{A} \setminus \mathcal{L}$. The CMC estimator is then

$$X_e = \mathbb{P}[T > x \mid \{Y_j,\, j \in \mathcal{B}\}].$$

This estimator can be computed as follows. Each $l \in \mathcal{L}$ is an arc that goes, say, from node a_l to node b_l. Given the values of Y_j for $j \in \mathcal{B}$, let α_l be the length of the longest path from the source to a_l, and β_l the length of the longest path from b_l to the sink. There is *no* path longer than x that passes through l if and only if $\alpha_l + Y_l + \beta_l \le x$. Conditional on $\{Y_j,\, j \in \mathcal{B}\}$, the probability that this happens is $\mathbb{P}[Y_l \le x - \alpha_l - \beta_l] = F_l[x - \alpha_l - \beta_l]$. Since the Y_l's are independent, the conditional probability that $T \le x$ is the product of these probabilities for all $l \in \mathcal{L}$, and therefore

$$X_e = 1 - \prod_{l \in \mathcal{L}} F_l[x - \alpha_l - \beta_l].$$

In addition to reducing the variance, this estimator reduces the number of random variates Y_k that need to be generated.

We performed experiments with the network shown in Figure 1.3, with $x = 90$ and the same cdfs F_j as in Example 1.4, and the same set $\mathcal{L}$ as in Avramidis and Wilson (1998), Section 4.1. The set $\mathcal{L}$ contains the 5 arcs that separate the nodes $\{0, 1, 2, 3, 4\}$ from the nodes $\{5, 6, 7, 8\}$. We observed that the variance of the CMC estimator was approximately 0.25 times that of the naive estimator. □

Example 6.22 Consider a simplified version of the call center example where $\nu = 0$, i.e., an arriving call abandons immediately with probability p if all agents are busy and otherwise has infinite patience. Suppose we want to estimate the expected number of abandonments (lost calls) in a day. The naive estimator just counts the number L of abandonments.

Now think of an observer who has information only on the calls handled by the agents and cannot see the abandonments. His available information is the number of busy agents

and the number of available agents at each time of the day, and the arrival times, answering times, and service times of all answered calls. Let $\mathcal{G}$ represent this information (formally, $\mathcal{G}$ is the σ-field generated by this information), and consider the CMC estimator $L_e = \mathbb{E}[L \mid \mathcal{G}]$.

To compute this conditional expectation, notice that if the arrival rate is $\lambda(t)$ at time t, then the (conditional) arrival process of lost calls is a hidden Poisson process with rate $\tilde{\lambda}(t) = P(t)\lambda(t)$, where $P(t) = p$ if all agents are busy at time t and $P(t) = 0$ otherwise. Thus, conditional on $\mathcal{G}$, the total number of abandonments L is a Poisson random variable with mean equal to the integral of $\tilde{\lambda}(t)$ over the entire day. This conditional mean is an unbiased CMC estimator which can be written as

$$L_e = \mathbb{E}[L \mid \mathcal{G}] = \int_0^m P(t)\lambda(t)dt \tag{6.51}$$

if time is in hours and the center operates for m hours. This estimator is easy to compute during the simulation and we have the guarantee that $\mathrm{Var}[L_e] < \mathrm{Var}[L]$. Exercise 6.15 asks you to experiment with L_e and its extensions. □

Example 6.23 In Example 1.11, suppose that the constant volatility parameter σ is replaced by a stochastic process $\{\sigma(t),\, t \geq 0\}$. The underlying asset is then said to evolve according to a model with *stochastic volatility*.

To price an European call or put option under such a model, in the case where σ and B are two independent processes, we can simulate the volatility process, and then compute the expectation of the price conditional on the trajectory of the volatility process, via the Black-Scholes formula.

♣ See Ben Ameur, Breton, and L'Ecuyer (1999), Boyle, Broadie, and Glasserman (1997b), Willard (1997) for specific volatility models, more details, and results. □

Example 6.24 (Boyle, Broadie, and Glasserman 1997b, page 1288.) The price of a *down-and-in call option* with discretely monitored barrier can be written as the mathematical expectation of

$$X = e^{-rT} \max(0,\, S(T) - K) \cdot \mathbb{I}[T_\ell \leq T],$$

where ℓ is a constant, $0 = t_0 < t_1 < \cdots < t_d = T$ are the observation dates, and $T_\ell = \inf\{t_j : S(t_j) < \ell\}$ ($T_\ell = \infty$ if this set is empty). A CMC estimator observes the sample path of the process $\{S(t),\, t \geq 0\}$ only up to time $\min(T_\ell, T)$, and estimates $\mathbb{E}[X]$ by the expectation of X conditional on this information, i.e., by

$$X_e = \mathbb{E}[X \mid T_\ell, S(T_\ell)] \cdot \mathbb{I}[T_\ell \leq T].$$

For $T_\ell \leq T$, if we assume that the asset price follows a geometric Brownian motion as in Example 1.11, X_e can be computed by the Black-Scholes formula for a model over a time horizon $T - T_\ell$ and with initial value $S(T_\ell)$ (see Exercise 6.16). Exercise 6.17 discusses an alternative CMC estimator. □

Example 6.25 Consider a continuous-time Markov chain for which the response X can be expressed as the integral, from time 0 to time T, of a cost rate whose value is a function of the current state of the chain (only), and that T is the hitting time of a particular set of

states. Then, we can condition on the sequence of states visited by the chain, i.e., replace the exponential holding times in the different states by their conditional expectations. This can also be generalized to semi-Markov processes. Fox and Glynn (1986, 1990) study this and call it *discrete-time conversion*.

♣ Detail a specific example. □

♣ See Bratley, Fox, and Schrage (1983), Problems 4.9.9–4.9.11 and 3.6.17–3.6.20.

Example 6.26 A situation where control variates are guaranteed to work cooperatively with CMC is when the control variates are $\mathcal{G}$-measurable (Glynn and Szechtman 2002). Suppose for instance that the conditioning is on some random variable Z, i.e., $X_e = \mathbb{E}[X \mid Z]$, that $\mathbb{E}[|Z|^j] < \infty$ for $j \geq 1$, that the moments $\mathbb{E}[Z]$, $\mathbb{E}[Z^2]$, ..., $\mathbb{E}[Z^q]$ are known, and that we use them as control variates with X. The CV estimator is then $X_c = X - \boldsymbol{\beta}^t(\boldsymbol{C} - \boldsymbol{\nu})$ where $\boldsymbol{C} - \boldsymbol{\nu} = (Z - \mathbb{E}[Z], \ldots, Z^q - \mathbb{E}[Z^q])^t$. Let $\boldsymbol{\beta}^* = \boldsymbol{\Sigma}_C^{-1}\boldsymbol{\Sigma}_{CX}$ be the optimal value of $\boldsymbol{\beta}$. Consider also the combined CMC-CV estimator $X_{e,c} = X_e - \boldsymbol{\beta}^t(\boldsymbol{C} - \boldsymbol{\nu})$. It is easy to see that $\mathrm{Cov}[\boldsymbol{C}, X_e] = \mathrm{Cov}[\boldsymbol{C}, X] = \boldsymbol{\Sigma}_{CX}$ because $\mathbb{E}[\mathbb{E}[X \mid Z]Z^j] = \mathbb{E}[\mathbb{E}[X \mid Z]\mathbb{E}[Z^j \mid Z]] = \mathbb{E}[XZ^j]$, so $\mathrm{Cov}[\mathbb{E}[X \mid Z], Z_j] = \mathrm{Cov}[X, Z^j]$ for each j. Therefore, the optimal $\boldsymbol{\beta}$ is the same for both X_c and $X_{e,c}$. Glynn and Szechtman (2002), prove that with this optimal $\boldsymbol{\beta}$, $\mathrm{Var}[X_c] \to \mathbb{E}[\mathrm{Var}[X \mid Z]]$ and $\mathrm{Var}[X_{e,c}] \to 0$ when $q \to \infty$.

As an illustration, in Example 6.21, we could use $\{Y_j,\ j \in \mathcal{B}\}$ (and perhaps some of their higher moments) as control variates, jointly with the CMC scheme proposed there. Since these CVs are obviously $\mathcal{G}$-measurable in this case, the combination is guaranteed to reduce the variance at least as much as the CMC scheme alone. □

6.6.1 Extended CMC and filtered Monte Carlo

Suppose the naive estimator X can be written as

$$X = C_1 + \cdots + C_T \tag{6.52}$$

where T can be a random variable. The standard CMC estimator has the form $X_e = \mathbb{E}[C_1 + \cdots + C_T \mid \mathcal{G}]$ for some σ-field $\mathcal{G}$. An *extended conditional Monte Carlo* (ECMC) estimator has the form

$$X_{ee} = \mathbb{E}[C_1 \mid \mathcal{G}_1] + \cdots + \mathbb{E}[C_T \mid \mathcal{G}_T], \tag{6.53}$$

where $\mathcal{G}_1, \ldots, \mathcal{G}_T$ are *different σ-fields*. Such an estimator can be simpler to compute than X_e.

Assuming for the moment that $T = t$ (a constant), we have

$$\mathbb{E}[X_{ee}] \;=\; \mathbb{E}\left[\sum_{j=1}^{t} \mathbb{E}[C_j \mid \mathcal{G}_j]\right] \;=\; \sum_{j=1}^{t} \mathbb{E}[C_j] \;=\; \mathbb{E}\left[\sum_{j=1}^{t} C_j\right] \;=\; \mathbb{E}[X]$$

and

$$\mathrm{Var}[X_{ee}] = \sum_{j=1}^{t} \mathrm{Var}[\mathbb{E}[C_j \mid \mathcal{G}_j]] + 2\sum_{j=1}^{t}\sum_{i=1}^{j-1} \mathrm{Cov}[\mathbb{E}[C_j \mid \mathcal{G}_j], \mathbb{E}[C_i \mid \mathcal{G}_i]] \tag{6.54}$$

whereas

$$\mathrm{Var}[X] = \sum_{j=1}^{t} \mathrm{Var}[C_j] + 2\sum_{j=1}^{t}\sum_{i=1}^{j-1} \mathrm{Cov}[C_j, C_i]. \tag{6.55}$$

Clearly, the first sum in (6.54) is smaller or equal to the first sum in (6.55), because $\mathrm{Var}[\mathbb{E}[C_t \mid \mathcal{G}_t]] \le \mathrm{Var}[C_t]$. However, the sum of covariances can be either smaller or larger in (6.54) than in (6.55). For this reason, X_{ee} does not always have less variance than X (in fact, one can construct examples where X_{ee} has more variance). There are situations where one can *prove* that $\mathrm{Var}[X_{\mathrm{ee}}] < \mathrm{Var}[X]$; see, e.g., Glasserman (1993a, 1993b) and Glynn and Iglehart (1988). Otherwise, the variances can be compared empirically.

When T is a random variable possibly correlated with the C_j's, things get even more complicated, because we cannot always interchange the sums and expectations as we did above.

The σ-fields in (6.53) are often *nested*: $\mathcal{G}_j \subset \mathcal{G}_{j+1}$ for $j \ge 1$; i.e., the sequence $\{\mathcal{G}_j,\ j \ge 1\}$ is a *filtration*. Glasserman (1993) calls this *filtered Monte Carlo (FMC)* and he also considers continuous-time analogs. A natural way this nesting occurs is when C_j is an $\mathcal{F}_j$-measurable cost incurred at the time t_j of the jth simulation event and $\mathcal{F}_j$ represents the information available up to time t_j. For each j, let $\mathcal{G}_j \subset \mathcal{F}_j$ be a σ-field that contains too little information to compute C_j, but enough so we can easily compute $\mathbb{E}[C_j \mid \mathcal{G}_j]$. Glasserman (1993) provides *sufficient conditions* for having $\mathrm{Var}[X_{\mathrm{ee}}] \le \mathrm{Var}[X]$ in this context.

Example 6.27 In the call center example, when we pick a call from the queue to start its service, suppose we look at its waiting time but not its patience time. Conditional on the waiting time, we can easily compute the probability that this call was lost by abandonment. So instead of counting the number L of abandonments during the day, we can add up these conditional probabilities over all calls having a nonzero waiting time. Since we condition on different information for each call, this is an ECMC estimator. There is no guarantee that the variance will be reduced. If the variance is not reduced, a tentative (partial) explanation could be that when a given call is lost by abandonment, this *reduces* the chance that the next one does, and this induces a negative component to the covariance between the successive abandonment indicators. □

6.6.2 Filtered Monte Carlo for Poisson input

Suppose that we have a (possibly time-inhomogeneous) Poisson input to our model, with rate function $\{\lambda(t),\ t \ge 0\}$. Let $T_1 \le T_2 \le \cdots$ be the times of the Poisson arrivals. Suppose that costs are incurred only at those arrival times. Let $\{\mathcal{G}(t),\ t \ge 0\}$ be a filtration that contains less information than $\{\mathcal{F}(t),\ t \ge 0\}$ (the one generated by the model evolution; see Chapter 2), i.e., $\mathcal{G}(t) \subseteq \mathcal{F}(t)$ for all t. Let $C(t) \ge 0$ be the cost that would be incurred by a Poisson arrival occurring at time t. We assume that $C(t)$ is $\mathcal{F}(t)$-measurable, but not $\mathcal{G}(t)$-measurable. The naive estimator just adds the costs up to some time horizon T:

$$X = \sum_{j=1}^{\infty} \mathbb{I}[T_j \le T] C(T_j). \tag{6.56}$$

An ECMC estimator is

$$X_{\text{ee}} = \sum_{j=1}^{\infty} \mathbb{I}[T_j \le T]\mathbb{E}[C(T_j) \mid \mathcal{G}_j]. \tag{6.57}$$

A *continuously-filtered* Monte Carlo estimator has the form

$$X_{\text{ep}} = \int_0^T \lambda(t)\mathbb{E}[C(t) \mid \mathcal{G}(t)]\, dt. \tag{6.58}$$

♣ This would need more elaboration. See "Filtered Monte Carlo" by Glasserman. Should make links with PASTA, event vs time averages, etc.

Example 6.28 The ECMC scheme of Example 6.27 is an instance of filtered Monte Carlo as in (6.57), where $\mathcal{G}_j$ contains all the information known up to the instant T_j when call j is answered, except for the patience time of call j (so we do not know if it is still waiting). However, the times T_j here are *not* the arrival times of a Poisson process. On the other hand, the scheme of Example 6.22 is an instance of a continuously-filtered estimator with Poisson input, where $\mathbb{E}[C(t) \mid \mathcal{G}(t)]$ in (6.58) is the conditional probability that a call arriving at time t would abandon immediately, given the number of customers in the system at time t. □

♣ See Ross (1993), Probability Models, Example 11.17, and the reference given there.

6.6.3 Conditional expectation for smoothing small differences

An important application of conditional Monte Carlo is for smoothing the estimator when estimating small differences, as in Section 6.4.2. In the context of Theorem 6.5, it happens frequently that the conditions in part (iii) of the theorem, or the continuity condition in Corollary 6.6, are not satisfied for the original estimator, but they are satisfied if we replace the estimator by an appropriate conditional expectation. The book of Fu and Hu (1997) is devoted to this topic and its use for gradient estimation and optimization.

Example 6.29 (L'Ecuyer and Perron 1994, Section 5.2.) In Example 6.11, we studied

$$Y^{(A)} = \sum_{j=1}^{t} (\mathbb{I}[X_j(\theta+\delta) > K] - \mathbb{I}[X_j(\theta) > K]). \tag{6.59}$$

as an estimator of the difference $\mu(\theta+\delta) - \mu(\theta)$, where $\mu(\theta)$ is the expected number of customers, among the first t, whose sojourn time exceeds K. We saw that the indicator $\mathbb{I}[X_j(\theta) > K]$ is a discontinuous function of θ for a fixed $\boldsymbol{U}$. To construct an estimator that obeys Corollary 6.6, we now replace the indicators $\mathbb{I}[X_j(\theta) > K]$ in (6.12) by conditional expectations.

A first candidate is the conditional expectation

$$\begin{aligned}
\mathbb{E}[\mathbb{I}[X_j(\theta) > K] \mid W_j(\theta)] &= \mathbb{P}[X_j(\theta) > K \mid W_j(\theta)] \\
&= \mathbb{P}[S_j(\theta) > K - W_j(\theta) \mid W_j(\theta)] \\
&= 1 - G_\theta(K - W_j(\theta)).
\end{aligned}$$

The corresponding difference estimator is

$$Y^{(B)} = \sum_{j=1}^{t} [G_\theta(K - W_j(\theta)) - G_{\theta+\delta}(K - W_j(\theta+\delta))]. \tag{6.60}$$

Under reasonable conditions on the interarrival and service time distributions, one can show that the conditions of Corollary 6.6 are satisfied (Exercise 6.20). Then, $\mathrm{Var}[Y^{(B)}/\delta]$ remains bounded when $\delta \to 0$. [3]

We could be more ambitious and condition on less. In Exercise 6.21, the reader is asked to construct an estimator $Y^{(C)}$ that replaces the conditioning on $W_j(\theta)$ by a conditioning on $X_{j-1}(\theta)$, the sojourn time of the previous customer.

Table 6.5. Sample MSE for the estimators of Example 6.29, with $t = 20$, $K = 2$, and $\theta_0 = 1/2$.

	$\delta = 10^{-3}$	$\delta = 10^{-4}$	$\delta = 10^{-5}$	$\delta = 10^{-6}$
$Y^{(A)}$	1.47×10^{-2}	2.09×10^{-3}	1.99×10^{-4}	1.50×10^{-5}
$Y^{(B)}$	2.96×10^{-4}	3.05×10^{-6}	2.94×10^{-8}	2.92×10^{-10}
$Y^{(C)}$	1.75×10^{-4}	1.66×10^{-6}	1.65×10^{-8}	1.63×10^{-10}

We performed numerical experiments for this example by taking F and G_θ as the exponential distributions with means 1 and $\theta = 1/2$, respectively, $K = 2$, and $t = 20$. We estimated the variances of the three difference estimators $Y^{(A)}$, $Y^{(B)}$, and $Y^{(C)}$, based on $n = 10000$ i.i.d. copies of each of them, for $\delta = 10^{-3}, 10^{-4}, 10^{-5}$, and 10^{-6}. We used the same n pairs of runs for all three estimators. Table 6.5 reports the variance estimates that we obtained. They are accurate to within (approximately) 3% with 95% confidence. The MSE is approximately 18δ for $Y^{(A)}$, $300\delta^2$ for $Y^{(B)}$, and $165\delta^2$ for $Y^{(C)}$. Thus, for this particular example, smoothing the estimator by conditioning changes the *convergence rate* of the MSE from $O(\delta)$ to $O(\delta^2)$, and $Y^{(C)}$ improves upon $Y^{(B)}$ only by a small factor. □

6.7 Indirect Estimation

♣ Should start with a simpler special case.

Suppose we want to estimate $h(\mu, \nu)$, where $\mu \in \mathbb{R}^d$ and $\nu \in \mathbb{R}^q$ are vectors of constants for which consistent estimators $(\bar{X}_n, \bar{C}_n)$ are available; that is, $(\bar{X}_n, \bar{C}_n) \Rightarrow (\mu, \nu)$ when $n \to \infty$. These estimators are not necessarily averages of i.i.d. observations and could be biased for finite n. The *direct estimator* of $h(\mu, \nu)$ is $h(\bar{X}_n, \bar{C}_n)$. In the case where ν is known, we can replace $\bar{C}_n$ by ν in this estimator. This yields the *indirect estimator*

$$g(\bar{X}_n) \stackrel{\text{def}}{=} h(\bar{X}_n, \nu). \tag{6.61}$$

As pointed out by Glynn and Whitt (1989), these two estimators correspond exactly to the estimators studied in Section 6.5.10, with and without the nonlinear CV, respectively, so they can be analyzed by a direct application of the results developed in that section.

[3] From Pierre: Should add details here.

Specifically, if $h : \mathbb{R}^{d+q} \to \mathbb{R}$ is continuously differentiable and Assumption 6.2 holds, Theorem 6.11 tells us that

$$\sqrt{n}(h(\bar{X}_n, \bar{C}_n) - g(\mu)) \quad \Rightarrow \quad N(0, \sigma_{\mathrm{h}}^2) \tag{6.62}$$

and

$$\sqrt{n}(h(\bar{X}_n, \nu) - g(\mu)) \quad \Rightarrow \quad N(0, \sigma_{\mathrm{g}}^2) \tag{6.63}$$

when $n \to \infty$, where σ_{h}^2 and σ_{g}^2 are defined as in Section 6.5.10 and correspond to the asymptotic variance constants of the direct and indirect estimators, respectively. The *direct estimator* has smaller asymptotic variance if and only if $\sigma_{\mathrm{h}}^2 < \sigma_{\mathrm{g}}^2$, i.e., if and only if the nonlinear CV $\bar{C}_n$ reduces the asymptotic variance.

Example 6.30 Consider an arbitrary queue where customer j has waiting time W_j, service time S_j, and sojourn time $X_j = W_j + S_j$. Suppose that these quantities obey the CLT $\sqrt{n}(\bar{W}_n - w, \bar{S}_n - \nu) \Rightarrow N(0, \boldsymbol{\Sigma})$ where w and ν are finite constants and $\boldsymbol{\Sigma}$ is a 2×2 matrix with elements σ_{ij}. We do not assume that the S_j's are i.i.d., but if they are, then $\nu = \mathbb{E}[S_j]$ and $\sigma_{22} = \mathrm{Var}[S_j]$. We want to estimate the mean sojourn time $\mu = w + \nu$ from a simulation (or observation) of the first n customers. If ν is known, an indirect estimator of μ is

$$g(\bar{W}_n) = h(\bar{W}_n, \nu) = \bar{W}_n + \nu \tag{6.64}$$

whereas the corresponding direct estimator is

$$h(\bar{W}_n, \bar{S}_n) = \bar{X}_n = \bar{W}_n + \bar{S}_n = g(\bar{W}_n) + (\bar{S}_n - \nu). \tag{6.65}$$

We have

$$n(\mathrm{Var}[h(\bar{W}_n, \bar{S}_n)] - \mathrm{Var}[g(\bar{W}_n)]) = n(\mathrm{Var}[\bar{S}_n] + 2\mathrm{Cov}[\bar{W}_n, \bar{S}_n]) \Rightarrow \sigma_{22} + 2\sigma_{12}$$

as $n \to \infty$, so the indirect estimator has smaller asymptotic variance if and only if $\sigma_{22} + 2\sigma_{12} > 0$, which is typical of most queuing systems (the contrary may occur only if the average waiting time is *negatively correlated* with the average service time).

However, both the direct and indirect estimators are generally beaten by the estimator $g(\bar{W}_n) - \beta(\bar{S}_n - \nu)$, which uses $\bar{S}_n$ as a control variate, with asymptotically optimal coefficient $\beta = \beta^* = \sigma_{12}/\sigma_{22}$. The direct and indirect estimators correspond to taking $\beta = -1$ and $\beta = 0$, respectively, rather than β^*.

For the case of a $GI/GI/s$ queue, $\nu = \mathbb{E}[S_j]$ so (6.64) is an unbiased estimator of $\mathbb{E}[\bar{X}_n]$, and Law (1974) has shown that this indirect estimator has less variance than (6.65) for any n. □

Example 6.31 Consider a queuing system as in the previous example, where A_j is the jth interarrival time, W_j is the waiting time of the jth customer, and $Q(t)$ the queue length at time t. We want to estimate the infinite-horizon average waiting time w and average queue length q. Following Glynn and Whitt (1989), we make the following functional CLT assumption, which is a two-dimensional version of Assumption 5.1.

Define the family of processes $\boldsymbol{Y}_n(t) = \sqrt{n}(\bar{W}_{\lfloor nt \rfloor} - wt,\ \bar{A}_{\lfloor nt \rfloor} - t/\lambda)$ for $0 \le t \le 1$ and $n = 1, 2, \ldots$. We assume that $\boldsymbol{Y}_n \ \Rightarrow\ (B_1, B_2)\boldsymbol{L}^{\mathsf{t}}$ in $D[0,1] \times D[0,1]$, where $\boldsymbol{L}$ is a 2×2 matrix, B_1 and B_2 are two independent standard Brownian motions, and $D[0,1]$ is the space of right-continuous real-valued functions over $[0,1]$ with left-hand limits (as in Section 5.10.1). Putting $t = 1$ in this assumption yields the CLT:

$$\sqrt{n}(\bar{W}_n - w,\ \bar{A}_n - 1/\lambda)^{\mathsf{t}} \ \Rightarrow\ N(0, \boldsymbol{\Sigma}) \tag{6.66}$$

where $\boldsymbol{\Sigma} = \boldsymbol{L}^{\mathsf{t}}\boldsymbol{L}$ is a covariance matrix with elements σ_{ij}. In the special case where the A_j's are i.i.d., we have $\lambda = 1/\mathbb{E}[A_j]$ and $\sigma_{22} = \mathrm{Var}[A_j]$. This CLT implies in turn that

$$\sqrt{n}\left(\frac{N(t)}{t} - \lambda,\ \frac{1}{t}\int_0^t Q(\zeta)\mathrm{d}\zeta - q\right)^{\mathsf{t}} \ \Rightarrow\ N(0, \tilde{\boldsymbol{\Sigma}}) \tag{6.67}$$

for some covariance matrix $\tilde{\boldsymbol{\Sigma}}$, where $N(t)$ is the number of arrivals during the time interval $(0, t]$ (see Glynn and Whitt 1989, page 87).

To estimate q, a *natural estimator* is $(1/t)\int_0^t Q(\zeta)d\zeta$, the sample average queue length over some time horizon t, as explained in Section 1.11.5. An alternative *indirect estimator* can be based on *Little's law*

$$q = \lambda w,$$

where λ is the arrival rate: if λ is known, take

$$g(\bar{W}_n) = \lambda \bar{W}_n$$

as an indirect estimator of q, for some large n. The corresponding *direct estimator* of q is $h(\bar{W}_n, \bar{A}_n) = \bar{W}_n/\bar{A}_n$. We can easily verify (see Glynn and Whitt 1989, Theorem 2) that under (6.66), $\sqrt{n}(h(\bar{W}_n, \bar{A}_n) - q) \ \Rightarrow\ N(0, \sigma_{\mathrm{h}}^2)$ where $\sigma_{\mathrm{h}}^2 = \lambda^2(\sigma_{11} - 2q\sigma_{12} + q^2\sigma_{22})$, whereas $\sqrt{n}(g(\bar{W}_n) - q) \ \Rightarrow\ N(0, \sigma_{\mathrm{g}}^2)$ where $\sigma_{\mathrm{g}}^2 = \lambda^2\sigma_{11}$. Therefore, we have

Proposition 6.12 *The indirect estimator $\lambda\bar{W}_n$ has smaller asymptotic variance than the direct estimator $\bar{W}_n/\bar{A}_n$ if and only if*

$$2\sigma_{12} < q\sigma_{22}. \tag{6.68}$$

In most queues, the average waiting times and interarrival times are negatively correlated, which is sufficient for (6.68) to hold. The indirect estimator is therefore preferable. This was shown by Carson and Law (1980) for $GI/GI/s$ queues. Glynn and Whitt (1989) prove that a *sufficient condition* for such a negative correlation is that $\mathbb{E}[W_i \mid A_j = a]$ is a nondecreasing function of a for each (i, j), plus some mild additional uniform integrability condition.

Glynn and Whitt (1989) also show that under the functional CLT assumption, the direct estimator $\bar{W}_n/\bar{A}_n$ and the *natural estimator* $(1/T_n)\int_0^{T_n} Q(\zeta)d\zeta$, where T_n is the nth departure time, converge at the same rate and with the same variance constant σ_{h}^2 when $n \to \infty$. Likewise, $\lambda\sqrt{t}((1/t)\int_0^t Q(\zeta)d\zeta - q) \ \Rightarrow\ N(0, \sigma_{\mathrm{h}}^2)$, where the additional factor λ is to take into account the change in time scale between counting in number of customers n and

counting in intrinsic time t. These three estimators have the same asymptotic variance and are all dominated by the indirect estimator when (6.68) holds.

Suppose now that the aim is to estimate $w = q/\lambda$ rather than q. Under the same conditions, it can be proved that the natural estimator $\bar{W}_n$ has smaller asymptotic variance than the direct and natural estimators of q defined in the previous paragraph, divided by λ. So if λ is known and (6.68) holds, it is better to estimate both w and q via $\bar{W}_n$ rather than via the natural or direct estimators of q. However, interpreting the indirect estimator as a nonlinear CV scheme also shows that we can do even better by using $\bar{A}_n$ as a control variate with its optimal coefficient. For example, to estimate q, we can use the CV estimator $\lambda\bar{W}_n + \beta(\bar{A}_n - 1/\lambda)$. The optimal value of β is $\beta^* = -\lambda\sigma_{12}/\sigma_{22}$ and the corresponding asymptotic variance constant is $\lambda^2(\sigma_{11} - \sigma_{12}/\sigma_{22}) < \sigma_g^2$. [4]

On the other hand, if λ is unknown and estimated by $1/\bar{A}_n$, the natural, direct, and indirect estimators all have the same asymptotic variance, for both w and q. See Glynn and Whitt (1989) for the details.

The same analysis applies to the general case where w is the average sojourn time per customer in some system and q is the time-average of the number of customers in that system, in the long run. If the average service time ν is known, this can be combined with the indirect estimator (6.64). This would yield $\lambda(\bar{W}_n + \nu)$ as an estimator of the average number in the system, where $\bar{W}_n$ is the average waiting time of the first n customers in the queue. □

Example 6.32 As a special case of Example 6.31, consider an $M/M/1$ queue with arrival rate λ and service rate μ. Let $\rho = \lambda/\mu$, the traffic intensity. We saw in Example 5.10 that

$$\sigma_{11} = \frac{\rho(\rho^3 - 4\rho^2 + 5\rho + 2)}{\mu^2(1-\rho)^4}$$

We also have $\sigma_{22} = \mathrm{Var}[A_j] = 1/\lambda^2$ and it can be shown that $\sigma_{12} = -1/(\mu(1-\rho))^2 < 0$. Thus, Condition (6.68) is satisfied. The asymptotic variance constants are $\sigma_h^2 = 2\rho^3(\rho^3 - 4\rho^2 + 4\rho + 1)/(1-\rho)^4$ and $\sigma_g^2 = \rho^3(\rho^3 - 4\rho^2 + 5\rho + 2)/(1-\rho)^4$. The variance reduction factor σ_g^2/σ_h^2 converges to 1 (no variance reduction) when $\rho \to 0$ or $\rho \to 1$, and is approximately 0.85 and 0.92 (modest reductions of 15% and 8%) for $\rho = 0.5$ and $\rho = 0.8$, respectively.

To estimate q via the CV estimator $\lambda\bar{W}_n - \beta(\bar{A}_n - 1/\lambda)$, the optimal β is $\beta^* = -\lambda\rho^2/(1-\rho)^2$ and the corresponding asymptotic variance constant is $\sigma_c^2 = \lambda^2(\sigma_{11} - \sigma_{12}/\sigma_{22}) = \rho^3(-\rho^3 - 4\rho^2 + 4\rho + 2)/(1-\rho)^4$. The variance reduction ratio σ_c^2/σ_h^2 converges to 1 (no reduction) when $\rho \to 0$, to 1/4 when $\rho \to 1$, and equals 23/34 when $\rho = 0.5$. This is better than the indirect estimator alone, especially when ρ is large (heavy-traffic).

The method tends to be more effective when there is high variability in the interarrival times compared with the variability of the service times, as illustrated by the next example. □

Example 6.33 A $GI/D/1$ queue with very "bursty" arrival process and constant service times.... See Srikant and Whitt (1999). Consider a single-server queue with ON/OFF Markov-modulated arrival process and constant service time $\nu = 1$. .. □

[4] From Pierre: Add an exercise on this. E.g., the call center with fixed B.

6.8 Stratification

Stratification, in its standard form, partitions a *population* into S *strata*, ideally so that the variance within the individual strata tends to be smaller than the overall population variance, and allocates a fixed fraction of the sample to each stratum (Cochran 1977 is the classical reference). The call center example in Section 6.2.1 illustrates the concept. To be more specific, suppose the stratification is based on a random variable S taking its values in the finite set $\{1, \ldots, k\}$. We can decompose:

$$\mu = \mathbb{E}[X] = \sum_{s=1}^{k} p_s \mu_s \tag{6.69}$$

where $p_s = \mathbb{P}\{S = s\}$ and $\mu_s = \mathbb{E}[X \mid S = s]$. If the p_s are known, μ can be estimated via (6.69) by estimating each μ_s separately. We distinguish the following cases:

(a) We can decide the value of S (as for B_i in the call center example of Section 6.2.1);

(b) we have no direct control over S.

In case (a), the number of observations in each strata can be decided either deterministically (fixed in advance) or dynamically, trying to increase the efficiency by readjusting the allocation as new information is collected. Case (b) occurs when it is too hard to generate a random variable from the conditional distribution of X given that $S = s$.

6.8.1 Deterministic allocation to strata

Suppose we can choose the value of S and then generate X according to its distribution conditional on S. Let n_s be the number of observations (or runs) with $S = s$, so $n = n_1 + \cdots + n_k$ is the total sample size, and let $X_{s,1}, \ldots, X_{s,n_s}$ denote the n_s i.i.d. observations within stratum s. We assume here that each n_s is positive and deterministic. A *stratified* (unbiased) estimator of μ is

$$\bar{X}_{\mathrm{s},n} = \sum_{s=1}^{k} p_s \hat{\mu}_s \tag{6.70}$$

where

$$\hat{\mu}_s = \frac{1}{n_s} \sum_{i=1}^{n_s} X_{s,i} \tag{6.71}$$

is the sample mean within stratum s. Let $\sigma_s^2 = \mathrm{Var}[X \mid S = s]$, the conditional variance of X given that we are in stratum s. Then,

$$\mathrm{Var}[\bar{X}_{\mathrm{s},n}] = \sum_{s=1}^{k} p_s^2 \sigma_s^2 / n_s. \tag{6.72}$$

An unbiased estimator of $\mathrm{Var}[\bar{X}_{\mathrm{s},n}]$ is

$$S^2_{\mathrm{s},n} = \sum_{s=1}^{k} p_s^2 \hat{\sigma}_s^2 / n_s, \tag{6.73}$$

where $\hat{\sigma}_s^2$ is the sample variance of $X_{s,1}, \ldots, X_{s,n_s}$, assuming that $n_s \geq 2$.

Proportional allocation allocates the observations to strata in proportion to their expected frequencies p_s; that is, $n_s = np_s$ (if we neglect the integrality condition on n_s). Call $\bar{X}_{\mathrm{sp},n}$ the corresponding version of $\bar{X}_{\mathrm{s},n}$. Then, (6.72) simplifies to

$$\mathrm{Var}[\bar{X}_{\mathrm{sp},n}] = \frac{1}{n} \sum_{s=1}^{k} p_s \sigma_s^2. \tag{6.74}$$

The so-called *optimal allocation*, which minimizes the variance (6.72) for a given sample size n, can be found by solving a nonlinear optimization problem with variables $n_1, \ldots, n_k$, to minimize (6.72) under the constraints that $n_s > 0$ for each s and $n_1 + \cdots + n_k = n$. Using a Lagrange multiplier, it is easily shown (Exercise 6.22) that the solution is to take n_s proportional to $p_s\sigma_s$; that is

$$n_s^* = np_s\sigma_s / \bar{\sigma}$$

where

$$\bar{\sigma} = \sum_{s=1}^{k} p_s \sigma_s$$

is the weighted average of the standard deviations within strata. Again, we neglect the rounding of n_s^*. Note that we must have each $n_s \geq 1$ for (6.70) to be well defined and each $n_s \geq 2$ for (6.73) to be defined. If $\bar{X}_{\mathrm{so},n}$ denotes $\bar{X}_{\mathrm{s},n}$ with optimal allocation,

$$\mathrm{Var}[\bar{X}_{\mathrm{so},n}] = \bar{\sigma}^2 / n.$$

We now compare the variances of $\bar{X}_n$, $\bar{X}_{\mathrm{sp},n}$, and $\bar{X}_{\mathrm{so},n}$. From variance decomposition (6.48), we obtain

$$\begin{aligned}
\mathrm{Var}[\bar{X}_n] &= \frac{1}{n}\left(\mathbb{E}[\mathrm{Var}[X|S]] + \mathrm{Var}[\mathbb{E}[X|S]]\right) \\
&= \frac{1}{n}\left(\sum_{s=1}^{k} p_s\sigma_s^2 + \sum_{s=1}^{k} p_s(\mu_s - \mu)^2\right) \\
&= \mathrm{Var}[\bar{X}_{\mathrm{sp},n}] + \frac{1}{n}\sum_{s=1}^{k} p_s(\mu_s - \mu)^2 \qquad (6.75) \\
&= \mathrm{Var}[\bar{X}_{\mathrm{so},n}] + \frac{1}{n}\sum_{s=1}^{k} p_s(\sigma_s - \bar{\sigma})^2 + \frac{1}{n}\sum_{s=1}^{k} p_s(\mu_s - \mu)^2. \qquad (6.76)
\end{aligned}$$

The two sums in (6.76) represent the variability due to the different standard deviations among strata and the variability due to the differences between stratum means, respectively. Proportional allocation eliminates the last sum while optimal allocation also eliminates the first. So, the improvement of stratification with proportional allocation over straightforward random sampling is significant if the μ_s's differ greatly, while that of optimal allocation over

proportional allocation is significant if the σ_s's differ greatly. Equation (6.76) also tells how the variance reduction with respect to the naive estimator (the sum of the last two terms) depends on the design of the strata. Ideally, we should design the strata to maximize the sum of the last two terms. Thus, there are two things to optimize: first, the design of the strata and, second, the allocation among the strata. To maximize the last term in (6.76), we maximize inhomogeneity across strata (or, equivalently, minimize the weighted sum of the internal variances σ_s^2). The middle term in (6.76) shows that optimal allocation improves over proportional allocation to the extent that the internal variances differ across strata.

The optimal allocation discussed above is optimal only under the assumption that sampling costs the same from all strata. More generally, if the expected sampling cost from stratum s is c_s, it is easily shown (again using a Lagrange multiplier) that we should take n_s proportional to $p_s\sigma_s/\sqrt{c_s}$ to maximize the efficiency. From this relationship, we can find the optimal allocation of observations to strata to minimize the variance for a fixed computing budget, or to minimize the computing cost for a fixed target variance.

This analysis of optimal allocation is idealized. Usually, σ_s is unknown and must be replaced by an estimate $\hat{\sigma}_s^{(0)}$ in the expression for n_s^*. We address this issue in the next subsection.

Note that a CMC estimator that conditions on S would erase the term $\mathbb{E}[\mathrm{Var}[X \mid S]] = \mathrm{Var}[\bar{X}_{\mathrm{sp},n}]$ in (6.75). The expectation of X conditional on $S = s$ is μ_s. Usually, if we can compute this expectation exactly, then we do not know the p_s's, otherwise we could compute μ exactly via (6.69).

Example 6.34 For any simulation problem, stratification can be applied directly to one (or more) of the underlying uniforms. To illustrate how to do this, we start with the (simplistic) one-dimensional case: Suppose we want to estimate the integral $\mu = \mathbb{E}[f(U)] = \int_0^1 f(u)du$ by Monte Carlo.

An obvious way to stratify is to partition the interval $(0, 1)$ into k subintervals of length $1/k$, to generate U_s uniformly over the sth subinterval for each s, and to estimate μ by

$$X_{\mathrm{s}} = \frac{1}{k}\sum_{s=1}^{k} f(U_s).$$

This can be repeated m times, independently. The total number of function evaluations (or simulation runs) is then $n = mk$. An unbiased estimator of μ is the sample mean $\bar{X}_{\mathrm{s},n}$ of the m i.i.d. copies of X_{s}, while an unbiased estimator of $\mathrm{Var}[X_{\mathrm{s}}]$ is given by (6.73), which is this case becomes $(1/n)\sum_{s=1}^{k}\hat{\sigma}_s^2$. Another unbiased variance estimator, but with fewer degrees of freedom, is given by the sample variance of the m i.i.d. copies of X_{s}. The latter estimator does not exploit the knowledge that the observations are independent across the different strata, whereas the estimator based on (6.73) exploits this information.

By (6.75), we have

$$\mathrm{Var}[\bar{X}_n] = \mathrm{Var}[\bar{X}_{\mathrm{s},n}] + \frac{1}{nk}\sum_{s=1}^{k}(\mu_s - \mu)^2$$

where $\mu_s = k \int_{(s-1)/k}^{s/k} f(u)du$, the average of $f(u)$ over the sth subinterval. So, $\bar{X}_{\mathrm{s},n}$ has less variance than the usual Monte Carlo estimator based on n independent runs, $\bar{X}_n$. The more the average value of f differs between the subintervals, the more the variance is reduced.

In principle, this stratification scheme works in any number of dimensions: In t dimensions, partition the unit hypercube into k^t cubic boxes by partitioning the interval $(0, 1)$ into k subintervals, generate a point uniformly over each of the k^t boxes, and repeat the entire process m times, independently. This type of partitioning quickly becomes impractical when t increases, because k^t becomes too large. But for large t, the unit hypercube $[0, 1)^t$ can also be partitioned in a different way into k non-cubic pieces of equal volume. One way of doing that, for instance, is by taking the k parallelepipeds determined by the points of an integration lattice of density k, modulo 1 (L'Ecuyer and Lemieux 2000).

Even for high-dimensional problems, applying this type of stratification to only a few of the uniforms (e.g., the one or two coordinates deemed the most important) can be effective in some situations. The call center example is a vivid illustration: the stratification scheme applied to this example in Section 6.2.1, with proportional allocation, is equivalent to applying the scheme considered here to the single uniform used for generating the number of tellers in the morning.

Section 6.9.3 generalizes this scheme to t dimensions in a different way. □

6.8.2 Dynamic allocation

We now discuss situations where the allocation of observations to strata is made dynamically during the experiment, e.g., in attempt to minimize the variance or maximize the efficiency. Thus, n_s is now the value of a random variable N_s. We still suppose here that the stratum of each run can be decided before the run. In Section 6.8.3, we relax this assumption and consider the situation in which the stratum of a run is a random variable generated during the run.

Two-stage sampling and sequential estimation. Assuming that the σ_s are unknown, we want to replace them, in the expression for n_s^*, by estimates $\hat{\sigma}_s^{(0)}$. Suppose these estimates $\hat{\sigma}_s^{(0)}$ are obtained from independent simulations (e.g., pilot runs) used only for the purpose of estimating n_s^* in a first stage sampling of size n_0, and the latter estimates determine the allocation for a second stage sampling of size n. Suppose for now that the mean and variance estimators $\bar{X}_{\mathrm{s},n}$ and $S^2_{\mathrm{s},n}$ are computed *from the second stage only*. Assuming $n_s^* \geq 2$ for all s, this $\bar{X}_{\mathrm{s},n}$ is unbiased and has larger variance than $\bar{X}_{\mathrm{so},n}$ for the same n (due to the variability of the $\hat{\sigma}_s^{(0)}$), while $S^2_{\mathrm{s},n}$ is an unbiased estimator of the conditional variance of $\bar{X}_{\mathrm{s},n}$ given the allocation selected for stage 2.

To improve efficiency, we could *re-use the observations* from the first stage in computing $\bar{X}_{\mathrm{s},n}$ in the second stage. This yields the following *two-stage sampling scheme*:

(1) Perform $n_s^{(0)}$ i.i.d. runs in stratum s for each s, for some predetermined values of $n_s^{(0)}$, and compute the empirical standard deviation $\hat{\sigma}_s^{(0)}$ in each stratum.

(2) Use these $\hat{\sigma}_s^{(0)}$'s to estimate the optimal n_s, for each s, by:

$$\hat{n}_s^* = np_s\hat{\sigma}_s^{(0)}/\hat{\bar{\sigma}}$$

where

$$\hat{\bar{\sigma}} = \sum_{s=1}^{k} p_s\hat{\sigma}_s^{(0)}.$$

For each s, if $\hat{n}_s^* > n_s^{(0)}$, perform an additional $\hat{n}_s^* - n_s^{(0)}$ i.i.d. runs in stratum s, and use all the runs from both stages to compute $\bar{X}_{\mathrm{s},n}$ and its variance estimator $S^2_{\mathrm{s},n}$ from (6.70) and (6.73), with n_s replaced by the random variable $N_s = \max(n_s^{(0)}, \hat{n}_s^*)$.

Of course, the $\hat{n}_s^*$'s must be adjusted to integers (e.g., by rounding). The total number of runs with this scheme may exceed n if $\hat{n}_s^* < n_s^{(0)}$ for some strata.

If we insists on not exceeding n, we can redefine the number of *additional runs* to perform in stratum s as

$$n_s^{(1)} = \max\left(0,\ \hat{n}_s^*/\kappa - n_s^{(0)}\right) \tag{6.77}$$

where κ is the unique normalization constant for which $\sum_{s=1}^{k}(n_s^{(0)} + n_s^{(1)}) = n$. Exercise 6.23 justifies this formula. The constant κ can be found via binary search or other root finding algorithms (see Section 4.1.2).

An alternative to two-stage sampling is a *sequential estimation* scheme: Perform stage one as above, but in the second stage, update the variance estimators $\hat{\sigma}_s^2$ and the $\hat{n}_s^*$ after each run and use them to decide the stratum of the next run. After any given run, if $\tilde{N}_s$ is the number of observations already taken from stratum s, the next stratum is the one with the largest current value of $\hat{n}_s^* - \tilde{N}_s$. At the end, compute $\bar{X}_{\mathrm{s},n}$ and $S^2_{\mathrm{s},n}$ as before, where N_s is the final value of $\tilde{N}_s$.

With both sampling schemes, the mean and variance estimators $\bar{X}_{\mathrm{s},n}$ and $S^2_{\mathrm{s},n}$ are generally biased, because $N_s = \hat{n}_s^*$ is a random variable correlated with the $\hat{\sigma}_s^2$'s and (generally) with the $\hat{\mu}_s$'s. This is essentially the same problem as for sequential estimation (Section 5.2.5).

For instance, with the two-stage sampling, we have

$$\begin{aligned}
\mathbb{E}[\hat{\mu}_s] &= \mathbb{E}\left[\frac{1}{N_s}\sum_{i=1}^{N_s} X_{s,i}\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\frac{1}{N_s}\sum_{i=1}^{N_s} X_{s,i} \mid N_s\right]\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\frac{1}{N_s}\sum_{i=1}^{n_s^{(0)}} X_{s,i} + \frac{1}{N_s}\sum_{i=n_s^{(0)}+1}^{N_s} X_{s,i}\,\middle|\, N_s\right]\right] \\
&= \mathbb{E}\left[\frac{n_s^{(0)}}{N_s}\mathbb{E}[X_{s,1} \mid N_s]\right] + \mathbb{E}\left[\frac{N_s - n_s^{(0)}}{N_s}\right]\mu_s
\end{aligned}$$

because $\mathbb{E}[X_{s,i}|N_s] = \mathbb{E}[X_{s,i}] = \mu_s$ for $i > n_s^{(0)}$. However, $\mathbb{E}[X_{s,i}|N_s] \neq \mathbb{E}[X_{s,i}] = \mu_s$ for $i \le n_s^{(0)}$ in general, because the value of N_s depends on the variance of the $X_{s,i}$'s for $i \le n_s^{(0)}$. So the estimator is generally biased. However, if the sample mean and sample variance of the $X_{s,i}$'s are uncorrelated, then $\mathbb{E}[X_{s,i}|N_s] = \mathbb{E}[X_{s,i}]$, so $\mathbb{E}[\hat{\mu}_s] = \mu_s$ and $\bar{X}_{\mathrm{s},n}$ is an unbiased

estimator of μ. This happens (in particular) if the $X_{s,i}$ are normally distributed. We should then expect the bias on $\bar{X}_{\mathrm{s},n}$ to be small if the distribution of the $X_{s,i}$ is not too far from normal. This is likely when the observations $X_{s,i}$ are themselves averages.

The sample variance $S^2_{\mathrm{s},n}$ generally underestimates the variance of the two-stage or sequential estimation estimator, because N_s being an increasing function of $\hat{\sigma}^2_s$, we have that (roughly speaking) when $\hat{\sigma}^2_s$ is smaller than usual, N_s is also smaller than usual, and this overemphasizes the contribution of $\hat{\sigma}^2_s$ to the variance estimator, whereas when $\hat{\sigma}^2_s$ is larger than usual, its contribution tends to be reduced. Variance underestimation is likely to be more severe in the sequential alternative than in the two-stage case.

Estimated proportions. Sometimes, the p_s's are unknown but estimates $\hat{p}_s$'s from previous experiments or other data are available. Suppose the estimators $\hat{p}_s$'s are sample averages from a first stage with n_0 (non-stratified) observations. The estimators $\hat{\mu}_s$ are computed from a second stage with sample size n, with n_s observations in stratum s. Then (see Cochran 1977):

$$\bar{X}_{\mathrm{sep},n} = \sum_{s=1}^{k} \hat{p}_s \hat{\mu}_s$$

is an unbiased estimator for μ, with variance

$$\mathrm{Var}[\bar{X}_{\mathrm{sep},n}] = \sum_{s=1}^{k} \frac{p_s^2 \sigma_s^2}{n_s} + \frac{1}{n_0} \sum_{s=1}^{k} p_s \left[\frac{(1-p_s)\sigma_s^2}{n_s} + (\mu_s - \mu)^2 \right]$$

if the n_s are fixed independently of the $\hat{p}_s$'s. If $N_s = n_s$ is proportional to $\lambda_s \hat{p}_s$ for fixed constants λ_s, $s = 1, \ldots, S$, then

$$\mathrm{Var}[\bar{X}_{\mathrm{sep},n}] = \sum_{s=1}^{k} p_s \left[\frac{\sigma_s^2}{n} \left(K_s + \frac{1}{n_0}(1 - K_s) \right) + \frac{p_s(\mu_s - \mu)^2}{n_0} \right]$$

where $K_s = (1/\lambda_s) \sum_{j=1}^{k} p_j \lambda_j$. This covers proportional allocation. If $\hat{p}_s$ and $\hat{\mu}_s$ are sample means both from the same set of observations, then $\bar{X}_{\mathrm{sep},n} = \bar{X}_n$. Stratified sampling with estimated proportions achieves nothing in this case.

6.8.3 Random allocation with poststratification

Any allocation of fixed numbers of observations to respective strata is *invasive*, as Glynn (1994) points out. It may make the simulation harder to carry out, or may be just impossible to implement because the distribution conditional on S is too complicated. In contrast, we may put observations into strata as they occur naturally during the simulation, and then compute (6.70), where n_s is the value of a random variable N_s. This is *poststratification*.

If C is a random variable correlated with X and with known distribution, we can poststratify according to the value of C. Wilson and Pritsker (1984a, 1984b) compare that with using C as a control variable. In their numerical experiments, they obtain lower variance when using C as a control than when stratifying on it, but the opposite can certainly occur, for example if the relationship between X and C is strongly nonlinear (as for for the CV B_i in the call center example).

Using the random allocations as control variates. The fraction of observations that random allocation gives to stratum s (when the observations are allocated to strata as they occur naturally) is N_s/n and has expectation p_s. So, $(N_s/n) - p_s$, $s = 1, \ldots, k-1$, can be used as control variables as in Section 6.5 (we discard $(N_k/n) - p_k$ since it is a linear combination of the others). Glynn and Szechtman (2002) prove that the optimal coefficients on $(N_s/n) - p_s$ are $\beta_s^* = \mu_s$, for all s. Therefore, taking $\hat{\beta}_s = \hat{\mu}_s$ is asymptotically optimal in the sense of Theorem 6.8, for $n \to \infty$. Using these coefficients is actually equivalent to poststratification.

Empty strata. With poststratification, we must make sure that either $N_s > 0$ for all s, or that $\hat{\mu}_s$ is well-defined when $N_s = 0$. Most authors, including Cochran (1977), simply ignore the latter possibility. It may be practically neglected, for example, if N is fixed at n and $\mathbb{P}\{N_s = 0 \mid N = n\} = (1 - p_s)^n$ is negligibly small for all s. Otherwise, one possibility may be to continue sampling until $N_s \geq 1$ for all s, or until $N_s \geq 2$ for all s if we want to use (6.73) as a variance estimator. For the case where this is impractical (e.g., because it may take an excessive number of runs), we now consider two estimators that allow $N_s = 0$. These estimators also allow poststratification with an infinite number of strata. They both perform a kind of *a posteriori aggregation*. The first candidate (estimator A) estimates the means μ_s within strata using the local sample mean for the non-empty strata, and the general sample mean for the empty strata. Exercise 6.24 show that this estimator is biased, but that the bias vanishes exponentially with n. The second candidate (estimator B) is

$$\bar{X}_{\mathrm{sr},n} = \sum_{s=1}^{k} \tilde{p}_{s,n} \hat{\mu}_s, \tag{6.78}$$

where $\hat{\mu}_s = 0$ when $N_s = 0$, and

$$\tilde{p}_{s,n} = \frac{p_s}{\mathbb{P}\{N_s > 0 \mid N = n\}} = \frac{p_s}{1 - (1 - p_s)^n}.$$

Since $\mathbb{E}[\hat{\mu}_s \mid N_s > 0, N = n] = \mu_s$, we get $\mathbb{E}[\bar{X}_{\mathrm{sr},n}] = \mu$ (unbiasedness). In Exercise 6.25, we analyze the variance of this estimator and show that we may have $\mathrm{Var}[\bar{X}_{\mathrm{sr},n}] > \mathrm{Var}[\bar{X}_n]$ if, for some s, $np_s \ll 1$ and $p_s(\mu_s - \mu)^2$ is large.

Example 6.35 ♣ Consider again the call center example.... Give numerical results with different variants of stratification... □

6.9 Antithetic Variates

6.9.1 A General Antithetic Variates Framework

The idea of *antithetic variates (AV)* resembles that of CRNs. Now, we want to estimate a single mathematical expectation μ, using the *average* of a set of k unbiased estimators $(X^{(1)}, \ldots, X^{(k)})$. Assuming that $\mathrm{Var}[X^{(1)}] = \cdots = \mathrm{Var}[X^{(k)}]$, the average

$$X_{\mathrm{a}} = \frac{1}{k} \sum_{j=1}^{k} X^{(j)}$$

is an unbiased estimator of μ with variance

$$\begin{aligned}\mathrm{Var}[X_{\mathrm{a}}] &= \frac{1}{k^2}\sum_{j=1}^{k}\sum_{\ell=1}^{k}\mathrm{Cov}[X^{(j)},X^{(\ell)}] \\ &= \frac{\mathrm{Var}[X^{(1)}]}{k}+\frac{2}{k^2}\sum_{j<\ell}\mathrm{Cov}[X^{(j)},X^{(\ell)}]. \end{aligned} \tag{6.79}$$

If the $X^{(\ell)}$'s are independent, the last sum is zero. The variance is reduced if and only if the sum of covariances in (6.79) is negative, and increased if and only if this sum is positive. Since we are doing k simulations, we compare $\mathrm{Var}[X_{\mathrm{a}}]$ with the variance of the average from k independent simulation runs, $\mathrm{Var}[X^{(1)}]/k$.

If $n = mk$ for some integer m, an AV estimator of μ based on n runs is obtained by getting m independent copies of $(X^{(1)},\ldots,X^{(k)})$, say $(X_1^{(1)},\ldots,X_1^{(k)}),\ldots,(X_m^{(1)},\ldots,X_m^{(k)})$, whose respective averages are $X_{\mathrm{a},1},\ldots,X_{\mathrm{a},m}$. The AV estimator is then defined by

$$\bar{X}_{\mathrm{a},n} = \frac{1}{m}\sum_{i=1}^{m}X_{\mathrm{a},i} = \frac{1}{n}\sum_{i=1}^{m}\sum_{\ell=1}^{k}X_i^{(\ell)}. \tag{6.80}$$

An unbiased estimator of $\mathrm{Var}[X_{\mathrm{a}}]$ is given by the empirical variance of these m independent copies of X_{a}:

$$S^2_{\mathrm{a},m} = \frac{1}{m-1}\sum_{i=1}^{m}\left(X_{\mathrm{a},i}-\bar{X}_{\mathrm{a},n}\right)^2. \tag{6.81}$$

In fact, for this variance estimator to be unbiased, it suffices that the $X_{\mathrm{a},i}$ be pairwise uncorrelated.

We can now compute a confidence interval for μ in a standard way, if we assume that the $X_{\mathrm{a},i}$ are approximately normally distributed. Under this assumption, $\sqrt{m}(\bar{X}_{\mathrm{a},m}-\mu)/S_{\mathrm{a},m}$ has approximately the Student distribution with $m-1$ degrees of freedom, and a $100(1-\alpha)\%$ confidence interval is given by $(\bar{X}_{\mathrm{a},n}\pm t_{m-1,1-\alpha/2}S_{\mathrm{a},m}/\sqrt{m})$, where $t_{m-1,1-\alpha/2}$ is the $(1-\alpha/2)$-quantile of the Student$(m-1)$ distribution (Section 5.2.1).

The variance estimator (6.81) is generally more noisy than the regular variance estimator S_n^2, because it has $m-1$ degrees of freedom instead of $n-1$. In particular, if we assume that X and X_{a} are normally distributed, then $(m-1)S^2_{\mathrm{a},m}/\mathrm{Var}[X_{\mathrm{a}}]$ and $(n-1)S_n^2/\mathrm{Var}[X]$ have chi-square distributions with $m-1$ and $n-1$ degrees of freedom, respectively, and

$$\frac{\mathrm{Var}[S^2_{\mathrm{a},m}]}{\mathrm{Var}[X_{\mathrm{a}}]} \approx \frac{n}{m}\frac{\mathrm{Var}[S_n^2]}{\mathrm{Var}[X]} \tag{6.82}$$

for large n (see Section 5.4.3). Then the variance estimator $S^2_{\mathrm{a},m}$ has approximately n/m times more relative variance than S_n^2, so a confidence interval on the variance will be roughly $\sqrt{n/m}$ times wider *relative to the exact variance*.

In general, even if X and X_{a} are *not* normally distributed, if we assume that sufficient uniform integrability is present, Eq. (5.21) in Section 5.4.3 implies that the factor n/m in (6.82) must be multiplied by the ratio of fourth moments of the standardized versions of X_{a} and X:

$$\frac{\mathrm{Var}[S^2_{\mathrm{a},m}]}{\mathrm{Var}[X_\mathrm{a}]} \approx \frac{n}{m}\frac{\mathbb{E}\left[((X_\mathrm{a}-\mu)/\mathrm{Var}[X_\mathrm{a}])^4\right]}{\mathbb{E}\left[((X-\mu)/\mathrm{Var}[X])^4\right]}\frac{\mathrm{Var}[S^2_n]}{\mathrm{Var}[X]}. \tag{6.83}$$

for large n. In the normal case, the fourth moments cancel one another.

Thus, confidence intervals for μ based on the AV estimator usually have *more variable widths* than confidence intervals based on the regular estimator $\bar{X}_n$. They have a *smaller average width* if AV reduces the variance.

Most methods induce the negative dependence directly between the streams of uniforms that drive the simulation runs and fit the following *general antithetic variates (GAV)* framework (Wilson 1983). We start from an unbiased estimator of μ written as $X = f(\boldsymbol{U})$, where $\boldsymbol{U} = (U_1, U_2, \dots)$ is a sequence of independent $U(0,1)$ random variables. For $\ell = 1, \dots, k$, we define $X^{(\ell)} = f(\boldsymbol{U}_\ell)$ where each $\boldsymbol{U}_\ell = (U_{\ell,1}, U_{\ell,2}, \dots)$ is a sequence of i.i.d. uniforms. The goal of GAV is to induce a dependence structure across the $\boldsymbol{U}_\ell$'s so that the sum of covariances in (6.79) becomes negative. How do we construct those $\boldsymbol{U}_\ell$'s? We will see various methods for doing that, among which we find Latin hypercube sampling and randomized quasi-Monte Carlo.

Under what conditions can we have a guarantee that this works? The success certainly depends on the function f. The following theorem, adapted from Avramidis and Wilson (1996), gives *sufficient* conditions for each covariance in the sum to be non-positive. These are rather strong conditions. It is obviously *not necessary* that each covariance be negative for the sum to be negative.

Let $\Psi \subseteq \{1, 2, \dots\}$ be a subset of arguments such that $f(U_1, U_2, \dots)$ is monotone with respect to U_j for each $j \in \Psi$. A sampling scheme such that each $U_{\ell,j}$ is $U(0,1)$, each pair $(U_{\ell,j}, U_{s,j})$ is negatively quadrant dependent when $j \in \Psi$, and the $U_{\ell,j}$'s for $j \notin \Psi$ are generated independently as in the standard MC method, is called *GAV-concordant*, and *completely GAV-concordant* if Ψ contains all the arguments of f. [5]

Theorem 6.13 *If $\boldsymbol{U}_1, \dots, \boldsymbol{U}_k$ are generated from a GAV-concordant sampling scheme, then $\mathrm{Var}[X_\mathrm{a}] \le \mathrm{Var}[\bar{X}_k] = \mathrm{Var}[X^{(1)}]/k$. Moreover, one can have $\mathrm{Var}[X_\mathrm{a}] = \mathrm{Var}[\bar{X}_k]$ only if the $X^{(\ell)}$'s are pairwise independent.*

Proof. GAV-concordant is equivalent to CRN-discordant for all pairs $(X^{(\ell)}, X^{(s)})$ with $\ell \ne s$ if we take $f_1 = f_2 = f$ in the definition of CRN-discordant. It then follows from Theorem 6.3 that $\mathrm{Cov}[X^{(\ell)}, X^{(s)}] \le 0$ for all $\ell \ne s$, and that this covariance can be zero only if $X^{(\ell)}$ and $X^{(s)}$ are independent. This implies the result.

6.9.2 Antithetic pairs.

In the *classical AV* method, we have $k = 2$, $\boldsymbol{U}_1 = \boldsymbol{U} = (U_1, U_2, \dots)$, and we define the *antithetic sequence* $\boldsymbol{U}_2 = \mathbf{1} - \boldsymbol{U} = (1 - U_1, 1 - U_2, \dots)$. In this case, $X_\mathrm{a} = [X^{(1)} + X^{(2)}]/2$,

$$\begin{aligned}\mathrm{Var}[X_\mathrm{a}] &= (\mathrm{Var}[X^{(1)}] + \mathrm{Var}[X^{(2)}] + 2\,\mathrm{Cov}[X^{(1)}, X^{(2)}])/4 \\ &= (\mathrm{Var}[X^{(1)}] + \mathrm{Cov}[X^{(1)}, X^{(2)}])/2,\end{aligned}$$

[5] From Pierre: It should be sufficient to have $(U_{\ell,j}, U_{\zeta,j})$ negatively quadrant dependent where ζ is a random index in $\{1, \dots, k\} \setminus \{\ell\}$. To be verified. Also add examples where this is useful. Can we exploit this for RQMC?

and $\mathrm{Var}[X_a] < \mathrm{Var}[X]/2$ if and only if $\mathrm{Cov}[X^{(1)}, X^{(2)}] < 0$. The rationale is that by choosing the antithetic sequence for $X^{(2)}$, disastrous events occurring in the first simulation are likely to be compensated by "antithetic" lucky events in the second one, thus reducing the variance of the average. For example, in a queuing system, if the service times are generated by inversion and if a particular service time is very long in the simulation for $X^{(1)}$, this means that the uniform U_j used to generate this service time is close to 1. Then, $1 - U_j$ is close to 0, so this service time will be very short in the simulation for $X^{(2)}$.

Noting that the pair $(U, 1-U)$ is always negatively quadrant dependent, we get a GAV-concordant sampling scheme by taking $U_{2,j} = 1 - U_{1,j}$ for $j \in \Psi$, and $U_{2,j}$ independent of $U_{1,j}$ otherwise. We call it an *AV-concordant* sampling scheme, and *completely AV-concordant* when Ψ contains all the arguments of f. We immediately get the following corollary of Theorem 6.13. It provides sufficient variance-reduction conditions which are *folklore* in the case where Ψ contains all arguments.

Corollary 6.14 *For any AV-concordant sampling scheme,* $\mathrm{Cov}[X^{(1)}, X^{(2)}] \leq 0$ *and the covariance can be zero only if* $X^{(1)}$ *and* $X^{(2)}$ *are independent.*

Theorem 6.4 also implies that a completely AV-concordant sampling scheme minimizes the covariance if f is fixed and if correlation induction is allowed only between its corresponding arguments. As with CRNs, the AVs could "backfire" (increase the variance) if they are used for arguments for which f is not monotone. Proper synchronization is again important. The best possible situation occurs when the response is a linear function of all underlying uniforms. Then, $\rho(X^{(1)}, X^{(2)}) = -1$ and the variance is reduced to zero. The worst case is when $X^{(1)}$ and $X^{(2)}$ are perfectly correlated, i.e., $\rho(X^{(1)}, X^{(2)}) = 1$. The AV method then doubles the variance. For an example of this, take $f(\boldsymbol{U}) = |U_1 - 1/2|$.

To minimize the covariance, we could *in theory* use Theorem 2.6: If F is the distribution function of $X^{(1)}$, the covariance is minimized by taking $X^{(1)} = F^{-1}(U)$ and $X^{(2)} = F^{-1}(1 - U)$, where $U \sim U(0,1)$ (or any equivalent sampling scheme, as indicated by the last part of Theorem 2.6). However, the latter is usually impractical because F is too complicated. As with CRNs, inversion is typically used at a *lower level*.

Other variants of the AV method are analyzed in Cheng (1982), Cheng (1984), Fishman and Wang (1983), Wilson (1983), Wilson (1984). For applications to finance models, see Boyle (1977), Boyle, Broadie, and Glasserman (1997b), Clewlow and Carverhill (1994), Hull and White (1987).

Example 6.36 In the stochastic activity network model of Example 1.4, we want to estimate $\mu = \mathbb{P}[T > x]$, where T is the project completion time and x is a constant. Suppose we generate each activity duration Y_j by inversion: $Y_j = F_j^{-1}(U_j)$, where the U_j's are i.i.d. $U(0,1)$. To compute an AV estimator based on *one pair* of runs, we need one vector of i.i.d. uniforms $(U_1, \ldots, U_t)$, where t is the number of activities with random duration in the project. Let $X^{(1)}$ be the value of the indicator $\mathbb{I}[T > x]$ when $Y_j = F_j^{-1}(U_j)$ for each j, and $X^{(2)}$ be the value of $\mathbb{I}[T > x]$ when $Y_j = F_j^{-1}(1-U_j)$ for each j. This can be repeated m times, independently, to obtain the pairs $(X_1^{(1)}, X_1^{(2)}), \ldots, (X_m^{(1)}, X_m^{(2)})$. Then, the AV estimator of μ and the corresponding variance estimator are computed by (6.80) and (6.81).

Here, T is an increasing function of each Y_j, which is in turn an increasing function of U_j when $Y_j = F_j^{-1}(U_j)$. So, the indicator $\mathbb{I}[T > x]$ is a monotone (increasing) function of each U_j. Therefore, the AV sampling scheme that we just described is *completely AV-concordant* and provides a guaranteed variance reduction: $\mathrm{Var}[\bar{X}_{\mathrm{a},n}] \le \mathrm{Var}[\bar{X}_n]$ by Corollary 6.14.

We performed experiments with the same network as in Example 1.4, with $n = 10000$. The empirical variance of the AV estimator was approximately 0.46 times that of the naive estimator. □

Example 6.37 We return to the Asian option pricing model considered in Example 1.11. The price of the underlying asset at time t satisfies Eq. (1.8). Suppose that the values of the standard Brownian motion $B(\cdot)$ in (1.8), at the observation dates t_j, are $B(t_j) = B(t_{j-1}) + \sqrt{t_j - t_{j-1}}Z_j$, where $t_0 = 0$ and $Z_1, \ldots, Z_d$ are i.i.d. $N(0,1)$. Here, it is clear that $S(t_j)$ is nondecreasing with respect to Z_ℓ for each j and ℓ, so the payoff (1.10) is a nondecreasing function of the Z_ℓ's. If the Z_ℓ's are generated by inversion, Corollary 6.14 guarantees a variance reduction if we take antithetic variates for all t coordinates (i.e., $\Psi = \{1, \ldots, t\}$).

Again, "inversion" here can be interpreted in a wide sense, as in Example 6.1. We can generate the Z_j's by any method, and take $-Z_j$ as the antithetic variate to Z_j. If we define $U_j = \Phi(Z_j)$, then $U_j \sim U(0,1)$ and $1 - U_j = 1 - \Phi(Z_j) = \Phi(-Z_j)$, so by using $(Z_j, -Z_j)$ as an antithetic pair of normals, we obtain the same result as if we had used inversion with the antithetic pair of uniforms $(U_j, 1 - U_j)$.

If the Brownian motion is simulated using the Brownian bridge approach instead, then the payoff remains nondecreasing, and the antithetic variates are also guaranteed to reduce the variance (Exercise 6.28). □

♣ Stratification with locally antithetic pairs: Haber 1967.

6.9.3 Latin Hypercube Sampling

Latin hypercube sampling (LHS) is a special case of GAV that can also be interpreted as a form of stratified sampling with proportional allocation. It was introduced by McKay, Beckman, and Conover (1979) and has been studied and extended by Stein (1987), Owen (1992), and Avramidis and Wilson (1996).

In Example 6.34, we saw how to stratify on a single uniform and argued that a straightforward t-dimensional generalization that stratifies the hypercube into k^t cubic boxes and generates one point in each of these boxes quickly becomes impractical when t increases.

LHS works with the same partition of the unit hypercube, but instead of generating one point in each of the k^t boxes, it generates a total of k points in a way that for each coordinate (dimension) j in a selected set Ψ, and for each of the k subintervals $[(i-1)/k, i/k)$, for $i = 1, \ldots, k$, there is exactly one point whose jth coordinate takes its value in the subinterval. Figure 6.2 gives an illustration. For the coordinates $j \notin \Psi$, the uniforms are generated independently (no LHS). The k points

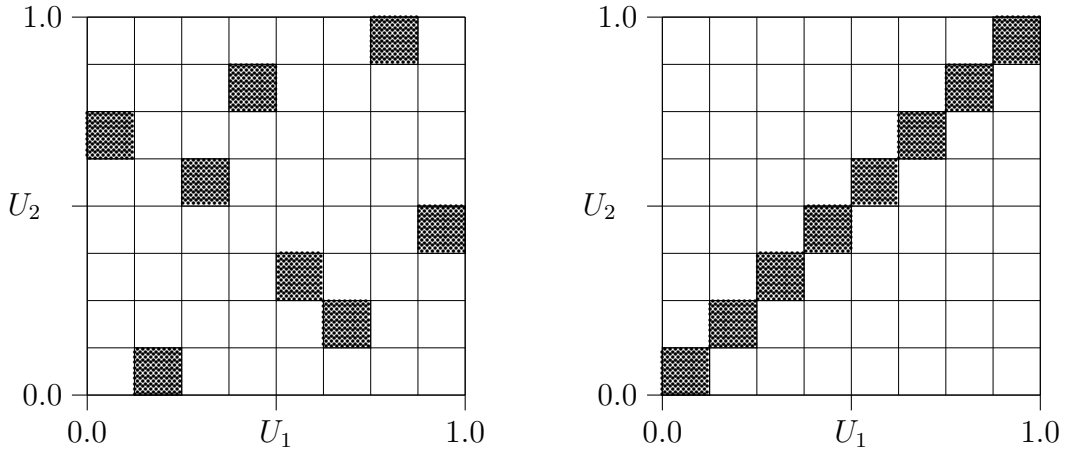


Fig. 6.2. An illustration of LHS for $t = 2$ and $k = 8$, for two different realizations of the permutations. In each case, one point will be generated at random in each of the eight darkly-shaded cubic boxes. The choice of those boxes depends on the randomly generated permutations. There is always one box per row and one box per column.

$$\begin{aligned} \boldsymbol{U}_1 &= (U_{1,1}, \ldots, U_{1,j}, \ldots, U_{1,t}), \\ \boldsymbol{U}_2 &= (U_{2,1}, \ldots, U_{2,j}, \ldots, U_{2,t}), \\ &\vdots \\ \boldsymbol{U}_k &= (U_{k,1}, \ldots, U_{k,j}, \ldots, U_{k,t}), \end{aligned}$$

are generated as follows:

1. Select a subset of coordinates $\Psi \subseteq \{1, \ldots, t\}$. LHS will be applied only for the coordinates in Ψ.

2. For each coordinate $j \in \Psi$, generate a *random permutation* $(\pi_{1,j}, \ldots, \pi_{k,j})$ of the integers $\{1, \ldots, k\}$

3. For each (s, j), generate $U_{s,j}$, the jth coordinate of the sth point, uniformly over the interval $((\pi_{s,j} - 1)/k,\ \pi_{s,j}/k)$ if $j \in \Psi$, and generate $U_{s,j} \sim U(0, 1)$ independently of the other uniforms if $j \notin \Psi$.

4. Compute $X_{\text{lh}} = [f(\boldsymbol{U}_1) + \cdots + f(\boldsymbol{U}_k)]/k$.

It is easily seen that each vector $\boldsymbol{U}_s$ produced by this scheme is uniformly distributed over $[0, 1]^t$. On the other hand, for each $j \in \Psi$, $U_{1,j}, \ldots, U_{k,j}$ are a stratified sample over the interval $(0, 1)$, as in Example 6.34.

To estimate the variance and compute a confidence interval, we can replicate the algorithm m times to generate m i.i.d. copies of X_{lh}, estimate μ by their sample mean $\bar{X}_{\text{lh},n}$ where $n = mk$, and estimate $\text{Var}[X_{\text{lh}}]$ by their sample variance, as explained in Section 6.9.1.

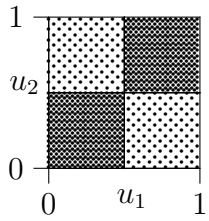


Fig. 6.3. The function f in Example 6.38 takes the value 1 in the darkly-shaded area and 0 in the lightly-shaded area.

The next example shows that LHS does not guarantee a variance reduction in general. However, the theorem that follows gives conditions under which LHS cannot increase the variance.

Example 6.38 Let $t = 2$ and $k = 2$. Take $f(u_1, u_2) = 1$ if $u_1 < 1/2$ and $u_2 < 1/2$, or if $u_1 > 1/2$ and $u_2 > 1/2$, and $f(u_1, u_2) = 0$ otherwise (Figure 6.3). Then, LHS will always choose two points where the value of f is the same, i.e., X_{lh} is either 0 or 1. As a result, $\text{Var}[X_{\text{lh}}] = \text{Var}[X_1] = 2\text{Var}[\bar{X}_2]$. LHS doubles the variance compared with crude Monte Carlo. □

Theorem 6.15 *If $f(U_1, \ldots, U_t)$ is monotone with respect to U_j for each $j \in \Psi$, then* $\text{Var}[X_{\text{lh}}] \leq \text{Var}[\bar{X}_k]$.

Proof. It is easily seen that for $j \in \Psi$, each pair $(U_{s,j}, U_{\ell,j})$ is negatively quadrant dependent. Then the result follows from Theorem 6.13.

Example 6.39 (Avramidis and Wilson 1996) The stochastic activity network model considered in Example 1.4 satisfies the condition of Theorem 6.15 (see Example 6.3), so LHS is guaranteed to reduce the variance for this example. □

6.10 Quasi-Monte Carlo Point Sets and Sequences

Classical QMC methods replace the uniform random points of the MC method by carefully designed *deterministic point sets*, as explained briefly in Section 1.5.3. In particular, these point sets can be the first n points of an infinite sequence, where the points are enumerated in a specific order. In this section, we explain the main construction methods for these highly-uniform (or QMC) point sets and sequences. Complementary coverages can be found in Niederreiter (1992), Sloan and Joe (1994), L'Ecuyer (2009), for example. One limitation of these deterministic QMC methods is that reliable error estimates are difficult to obtain. Asymptotic error bounds can be obtained under certain assumptions on f, but these bounds are generally too hard to compute explicitly (see Section 1.5.3). In Section 6.11, we will see how the deterministic point sets can be randomized to provide unbiased mean estimators

that fit the GAV framework of Section 6.9.1, with k equal to the cardinality of the QMC point set. This is randomized QMC (RQMC).

We already said in Section 1.5.3 that filling the unit hypercube very evenly with a reasonable number of points becomes practically impossible when the dimension is large. But many high-dimensional integrands can be well-approximated by a sum of low-dimensional functions, and the integration error can be made very small if these low-dimensional functions are well integrated by the QMC or RQMC method. Recall that for any point set P_n and any set of indices $I = \{i_1, \ldots, i_d\} \subseteq \{1, \ldots, t\}$, $P_n(I)$ denotes the projection of P_n over the d-dimensional subspace determined by I. A key idea underlying the construction of a QMC point set P_n is, roughly speaking, to make sure that the low-dimensional projections $P_n(I)$ over the sets of coordinates I deemed important are very uniform in the corresponding lower-dimensional unit hypercubes.

One natural requirement in this context is that all projections $P_n(I)$ contain as many distinct points as the original point set P_n. This rules out rectangular grids, for example. A point set P_n in $[0,1)^s$ is called *fully projection-regular* (Sloan and Joe 1994, L'Ecuyer and Lemieux 2000) if for each non-empty $I \subseteq \{1, \ldots, s\}$, $P_n(I)$ has n distinct points.

Constructions for which several projections are identical also make the analysis easier. P_n is called *dimension-stationary* (Lemieux and L'Ecuyer 2001) if whenever $1 \le i_1 < \ldots < i_d < s$ and $1 \le j \le s - i_d$, $P_n(\{i_1, \ldots, i_d\}) = P_n(\{i_1 + j, \ldots, i_d + j\})$. This means that $P_n(I)$ depends only on the *spacings* between the indices in I.

The main classes of construction methods are digital nets and sequences, digital point sets with a different basis for each coordinate, and integration lattices.

In this section and the next, the points of P_n are enumerated from 0 to $n-1$, for consistency with the usual notation in the QMC literature and with the software implementations. Note that in the GAV setting, they are enumerated from 1 to k.

6.10.1 Digital nets and sequences: definitions

In Section 1.5.3, we considered the point set $P_n = \mathbb{Z}_n/n = \{0, 1/n, \ldots, (n-1)/n\}$ as a reasonable possibility when $s = 1$ and n is fixed. For $s > 1$, we suggested point sets P_n for which each one-dimensional projection is $\mathbb{Z}_n/n$. That is to say, for each dimension j, the jth coordinate of the points $\boldsymbol{u}_i$ will visit all n values in $\mathbb{Z}_n/n$, but in a different order for the different coordinates. Such a point set is defined by selecting a different permutation of $\mathbb{Z}_n/n$ (or equivalently, of the integers in $\mathbb{Z}_n$) for each coordinate j. The permutations must be different, because otherwise all the points would lie on a diagonal line in the unit hypercube. We want to select these permutations so that P_n is highly uniform over $[0,1)^s$.

The Latin hypercube sampling of Section 6.9.3 is a first step in that direction: It selects the permutations at random, independently for the different coordinates. It then adds a random $\boldsymbol{V}_i$ to each point $\boldsymbol{u}_i$, where the $\boldsymbol{V}_i$'s are independent and uniformly distributed over $[0, 1/n)^s$. A good uniformity is guaranteed for each one-dimensional projection of the point set, but not necessarily for its projections in more than one dimension. If the permutations are selected carelessly or just generated at random, it may happen for example that two of them are the same (or are very similar), in which case the projection of P_n on the corresponding two coordinates will be aligned (or nearly aligned) on the main diagonal of the unit square.

Example 6.40 Let $s = 3$ and $n = 8$. To construct P_n, we need three permutations of the integers $\{0, 1, \ldots, 8\}$, say $\boldsymbol{\pi}_j = (\pi_{0,j}, \ldots, \pi_{7,j})$ for $j = 1, 2, 3$. After applying the permutations, suppose we place the 8 pairs $(U_{i,1}, U_{i,3})$ (the projection of P_n on its first and third coordinates) on the unit square. If it turns out that $\boldsymbol{\pi}_3 = \boldsymbol{\pi}_1$, then all these pairs are on the main diagonal, as in the right panel of Figure 6.2. If we have $\boldsymbol{\pi}_3 = (\pi_{7,1}, \ldots, \pi_{0,1})$ ($\boldsymbol{\pi}_1$ in reverse order), then all the points are on the diagonal line between $(0, 1)$ and $(1, 0)$. □

Digital nets (Niederreiter 1992) are general construction methods that permit one to select the permutations so that P_n itself, as well as its projections in more than one dimension, are highly uniform in a sense that can be measured, e.g., by the equidistribution criteria introduced in Section 3.3.4.

We start with a simplified version of their definition, which covers the most popular constructions. Let $b \geq 2$ be an arbitrary integer, usually a prime, called the *base*. To define a net of $n = b^k$ points in s dimensions, we select s *generator matrices* $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_s$, which are (in theory) $\infty \times k$ matrices whose elements are in $\mathbb{Z}_b = \{0, \ldots, b-1\}$. The matrix $\boldsymbol{C}_j$ determines coordinate j of all the points, for $j \geq 1$. To define the ith point $\boldsymbol{u}_i$, for $i = 0, \ldots, b^k - 1$, write the digital expansion of i in base b and multiply the vector of its digits by $\boldsymbol{C}_j$, modulo b, to obtain the digits of the expansion of $u_{i,j}$, the jth coordinate of $\boldsymbol{u}_i$. That is,

$$
\begin{aligned}
i &= \sum_{\ell=0}^{k-1} a_{i,\ell} b^{\ell}, \\
\begin{pmatrix} u_{i,j,1} \\ u_{i,j,2} \\ \vdots \end{pmatrix} &= \boldsymbol{C}_j \begin{pmatrix} a_{i,0} \\ a_{i,1} \\ \vdots \\ a_{i,k-1} \end{pmatrix} \bmod b, \qquad (6.84) \\
u_{i,j} &= \sum_{\ell=1}^{\infty} u_{i,j,\ell} b^{-\ell}, \qquad (6.85) \\
\boldsymbol{u}_i &= (u_{i,1}, \ldots, u_{i,s}).
\end{aligned}
$$

The point set thus obtained is a *digital net in base b*. In practice, the expansion (6.85) is truncated to r digits for some r, so each $\boldsymbol{C}_j$ becomes a $r \times k$ matrix. Typically, r is equal to k, or is slightly larger, or is selected so that b^r is near 2^{31}. If the first k rows of each $\boldsymbol{C}_j$ form a nonsingular $k \times k$ matrix (which we shall assume henceforth), then the n output values for coordinate j, truncated to their first k fractional digits in base b, are a permutation of $\mathbb{Z}_n/n = \{0, 1/n, \ldots, (n-1)/n\}$.

The setting of Niederreiter (1992) is more general: one can apply bijections (or permutations) to the digits of $\mathbb{Z}_b$ before and after the multiplication by $\boldsymbol{C}_j$. To do this, we take an arbitrary ring R of cardinality b, then define bijections $\psi_\ell : \mathbb{Z}_b \to R$ for $\ell = 0, \ldots, k-1$, and $\eta_{j,\ell} : R \to \mathbb{Z}_b$ for $\ell = 1, \ldots, r$ and $j = 1, \ldots, s$. In Eq. (6.84), each digit $a_{i,\ell}$ is replaced by $\psi_\ell(a_{i,\ell})$, and the multiplications by $\boldsymbol{C}_j$ are done in the ring R. After that, in Eq. (6.85), each $u_{i,j,\ell}$ is replaced by $\eta_{j,\ell}(u_{i,j,\ell})$. These bijections provide additional flexibility for improving the uniformity. If $R = \mathbb{Z}_b$, they are equivalent to permuting the digits of $\mathbb{Z}_b$. If b is a power of a prime, then R can be taken as a finite field, so the multiplications by $\boldsymbol{C}_j$ are performed in the finite field.

An *infinite sequence* of points can be obtained by defining an infinite number of columns for each $\boldsymbol{C}_j$. This gives a *digital sequence in base b*. Only the first k columns are needed to get the first b^k points, for any k. Likewise, an infinite-dimensional point set is obtained with an infinite sequence of matrices $\boldsymbol{C}_j$. In both cases, the sequence of columns and matrices can be defined via recurrences (each column and matrix being a function of the previous ones). Concrete constructions will be explained in forthcoming subsections; they include the sequences of Sobol' (1967), in base 2, of Faure (1982), in prime base b, of Niederreiter (1987), and of Niederreiter and Xing (1998).

Definition 6.4 The expression *digital net in base b* refers to a construction defined as above, potentially with the bijections, and also to the point set P_n produced by such a construction. When the matrices $\boldsymbol{C}_j$ have an infinite number of columns, we have a *digital sequence in base b*. □

If P_n is a digital net, then its *projection* $P_n(I)$ over the subspace determined by a subset of indices $I = \{i_1, \ldots, i_d\} \subseteq \{1, \ldots, s\}$ is a d-dimensional digital net, and similarly for a digital sequence. The matrix $\boldsymbol{C}_{i_j}$ becomes the matrix $\boldsymbol{C}_j$ for the new net.

6.10.2 Equidistribution

The uniformity of P_n for digital nets is most often measured by the equidistribution criteria defined in Section 3.3.4. We recall that P_n is $(q_1, \ldots, q_s)$-equidistributed in base b if each of the b^{k-t} rectangular boxes determined by the first q_j b-ary digits for each j, where $k - t = q_1 + \cdots + q_s$, contains exactly b^{k-t} points. Note that $(q_1, \ldots, q_s)$-equidistribution implies $(q'_1, \ldots, q'_s)$-equidistribution whenever $(q'_1, \ldots, q'_s) \leq (q_1, \ldots, q_s)$. The following definitions are equivalent to those of Niederreiter (1992).

Definition 6.5 P_n is a *(t, k, s)-net in base b* if it is $(q_1, \ldots, q_t)$-equidistributed in base b for all non-negative integers $q_1, \ldots, q_t$ whose sum does not exceed $k - t$ (or equivalently, whose sum equals $k - t$). The smallest such t is the *t-value* of P_n. [6] □

Definition 6.6 An infinite sequence of s-dimensional points $\{\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \}$ is called a *(t, s)-sequence in base b* if for all integers $k > 0$ and $\nu \geq 0$, the point set $Q(k, \nu) = \{\boldsymbol{u}_i : i = \nu b^k, \ldots, (\nu + 1)b^k - 1\}$, of cardinality b^k, is a (t, k, s)-net in base b. □

Ideally, we want the t-value to be as small as possible, but there are theoretical bounds on the best that can be achieved. These bounds, and the best values achieved so far by known constructions, have been tabulated by Schmid and Schürer (2005). There cannot exist a $(0, k, s)$ unless $b \geq s - 1$, and a *$(0, s)$-sequence in base b* is not possible unless $b \geq s$.

♣ Should add a small table here, giving some bounds on t-values for $b = 2$.

The next result shows how we can verify the equidistribution properties directly from the matrices $\boldsymbol{C}_j$. These properties do not depend on the bijections. Note that this equidistribution is possible only if $q_1 + \cdots + q_s \leq k$.

[6] From Pierre: The (t, k, s)-net is often called (t, m, s)-net using a different notation (Niederreiter 1992).

Theorem 6.16 *A digital net is $(q_1, \ldots, q_s)$-equidistributed if and only if the set of $k - t = q_1 + \cdots + q_s$ rows that comprise the first q_1 rows of $\boldsymbol{C}_1$, the first q_2 rows of $\boldsymbol{C}_2$, ..., and the first q_s rows of $\boldsymbol{C}_s$, is linearly independent in the finite ring R (i.e., there is no nontrivial linear combination of these rows that gives zero). This holds regardless of the bijections ψ_ℓ and $\eta_{j,\ell}$. In particular, we have a (t, k, s)-net if and only if this set of rows is linearly independent over R whenever $q_1 + \cdots + q_s = k - t$.*

Proof. The proof is a direct adaptation of that of Theorem 4.26 of Niederreiter (1992). The matrix $\boldsymbol{M}$ defines a linear mapping from R^k to R^{k-t}, which maps the same number of points to each element of R^{k-t} if and only if the dimension of its kernel is zero. This gives the result when all the bijections are the identity. But the ψ_ℓ's define a bijection between $\mathbb{Z}_b^k$ and R^k on one side (changing them only change the order in which the points of R^k are enumerated), and the bijections $\eta_{j,\ell}$ define bijections between R^{k-t} and $\mathbb{Z}_b^{k-t}$ on the other side (they permute the b^{k-t} boxes of the equidissection). Thus, the bijections preserve the equidistribution. (When $R = \mathbb{Z}_b$, these bijections just reshuffle the elements of $\mathbb{Z}_b^k$ and of $\mathbb{Z}_b^{k-t}$). The last part follows directly from the definition of (t, k, s)-net.

If R is a field, then the condition of Theorem 6.16 holds if and only if the matrix $\boldsymbol{M}$ formed by these rows has rank $k - t = q_1 + \cdots + q_s$. But this is not necessarily true if R is not a field.

Example 6.41 Suppose $s = k = 1$, $b = 6$, $R = \mathbb{Z}_6$, and $\boldsymbol{C}_1$ is the 1×1 matrix whose single element is 2. Then this matrix has rank 1, but its single row vector is not linearly independent over $\mathbb{Z}_6$. The corresponding one-dimensional net contains only the points $\{0, 2/6, 4/6\}$, each repeated twice. When we partition the interval $[0, 1)$ into $b = 6$ equal subintervals, half of the subintervals contain two (identical) points, and the other half are empty. □

6.10.3 Digital shift and matrix scramble

A useful type of transformation often applied to digital nets is a digital shift. We will use it in Section 6.11 to obtain unbiased RQMC estimators. It is defined as follows, for an arbitrary point set P_n. Select a single vector $\boldsymbol{v} = (v_1, \ldots, v_s)$ in $[0, 1)^s$, and write the digital expansion in base b of each of its coordinates, say $v_j = \sum_{\ell=1}^{\infty} v_{j,\ell} b^{-\ell}$. Add $v_{j,\ell}$ modulo b to the ℓth digit of the digital expansion in base b of the jth coordinate of each point $\boldsymbol{u}_i \in P_n$, for each j and ℓ. For $b = 2$, the digit-wise addition modulo b becomes a bitwise exclusive-or, which is fast to perform on a computer.

Definition 6.7 This transformation of P_n is called a *digital shift in base b*. □

If P_n is a digital net, the digital shift corresponds to replacing $\eta_{j,\ell}(u_{i,j,\ell})$ by $\tilde{\eta}_{j,\ell}(u_{i,j,\ell}) = (\eta_{j,\ell}(u_{i,j,\ell}) + v_{j,\ell}) \bmod b$, which defines a new bijection $\tilde{\eta}_{j,\ell}$. That is, the digital shift just changes the bijections $\eta_{j,\ell}$. As a consequence, we have:

Corollary 6.17 *Any digital shift modulo b preserves all the equidistribution properties of a digital net in base b.*

Example 6.42 As a baby example, consider the two-dimensional Hammersley net with $k = 4$ and $r = 4$, for which $n = 16$, $\boldsymbol{C}_1$ is the reflected identity, and $\boldsymbol{C}_2$ is the identity. We apply a digital shift in base 2 with the vector $\boldsymbol{v} = (3/16,\ 5/16) = (0.0011_2,\ 0.0110_2)$, where the last representation is in base 2. Table 6.6 gives the 16 points before and after the shift. Those points are illustrated in Figure 6.4. They form a $(0, 4, 2)$-net in base 2 both before and after the shift. A similar example is given in Figure 1.15. □

Table 6.6. The 16 points of a two-dimensional Hammersley net (left) and the same points after a digital shift by $\boldsymbol{v} = (0.0011_2,\ 0.0110_2)$ (right). The coordinates are in base 2.

i	$(u_{i,1},\ u_{i,1})$	$(u_{i,1},\ u_{i,1}) \oplus \boldsymbol{v}$
0	(.0000, .0000)	(.0011, .0110)
1	(.0001, .1000)	(.0010, .1110)
2	(.0010, .0100)	(.0001, .0010)
3	(.0011, .1100)	(.0000, .1010)
4	(.0100, .0010)	(.0111, .0100)
5	(.0101, .1010)	(.0110, .1100)
6	(.0110, .0110)	(.0101, .0000)
7	(.0111, .1110)	(.0100, .1000)
8	(.1000, .0001)	(.1011, .0111)
9	(.1001, .1001)	(.1010, .1111)
10	(.1010, .0101)	(.1001, .0011)
11	(.1011, .1101)	(.1000, .1011)
12	(.1100, .0011)	(.1111, .0101)
13	(.1101, .1011)	(.1110, .1101)
14	(.1110, .0111)	(.1101, .0001)
15	(.1111, .1111)	(.1100, .1001)

The matrices $\boldsymbol{C}_j$ in popular digital net constructions are typically defined by some general rule that might not take into account the uniformity of all the important projections of P_n. So one could try to further improve (or perhaps optimize) their behavior by modifying the $\boldsymbol{C}_j$'s. One type of transformation multiplies $\boldsymbol{C}_j$ on the left (in R) by a $k \times k$ lower-triangular matrix $\boldsymbol{M}_j$ with elements in R, and whose diagonal elements are invertible in R, for each j. If R is a field, then any lower-triangular matrix with nonzero diagonal elements satisfies this property. Replacing $\boldsymbol{C}_j$ by $\boldsymbol{M}_j\boldsymbol{C}_j$ for each j is called a *matrix scramble* or *linear scramble* of the net (Matoušek 1999, Owen 2003).

The $\boldsymbol{M}_j$'s can be carefully constructed to optimize some uniformity criterion, or may be just generated at random, e.g., if the average uniformity measure with random $\boldsymbol{M}_j$'s is likely to be better than for the original $\boldsymbol{C}_j$'s. Of course, $\boldsymbol{M}_j$ can be taken as the identity (no scramble) for some of the coordinates. This linear scramble has the following important property.

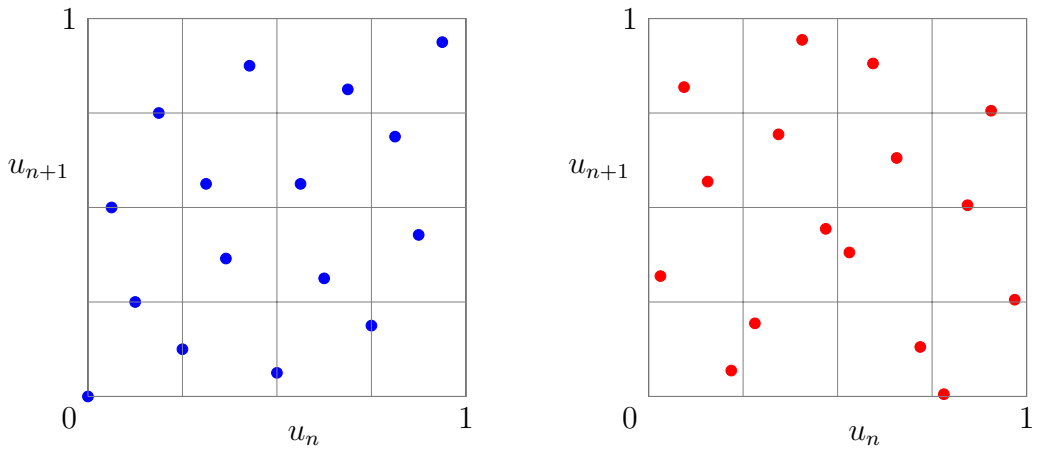


Fig. 6.4. The 16 points of an Hammersley net (left) and the same points after a digital shift by $\boldsymbol{v} = (3/16,\, 5/16) = (0.0011_2,\, 0.0110_2)$ (right).

Theorem 6.18 *If a digital net in base b is $(q_1, \ldots, q_s)$-equidistributed, then applying a matrix scramble with invertible lower-triangular matrices $\boldsymbol{M}_j$, as defined above, preserves the $(q_1, \ldots, q_s)$-equidistribution.*

Proof. Let $\boldsymbol{M}$ be the matrix defined in Theorem 6.16, before applying the scramble. We suppose that this matrix has full rank $t = q_1 + \cdots + q_s$. The impact of the matrix scramble on $\boldsymbol{M}$ is as follows. The first q_1 rows of $\boldsymbol{M}$ are replaced by q_1 nonzero linear combinations of the original q_1 first rows. These new q_1 rows are linearly independent, because for each $\ell \le q_1$, the new row ℓ is a nonzero multiple of the old row ℓ, plus a linear combination of the previous rows, which are linearly independent from row ℓ (because $\boldsymbol{M}$ is lower-triangular and its diagonal elements are invertible). The next q_2 rows are replaced by q_2 nonzero linear combinations of the original next q_2 rows, which are also linearly independent for the same reason. Moreover, these new q_2 rows are linearly independent from the first q_1 rows, because the original ones had that property. Then the next q_3 rows are replaced in the same way by q_3 new rows so that the first $q_1 + q_2 + q_3$ new rows form a linearly independent set over R, and so on. In the end, all the new rows of $\boldsymbol{M}$ are linearly independent, and the result follows from Theorem 6.16.

6.10.4 The van der Corput sequence and $(0, k, 2)$ nets

Let $b \ge 2$ be an arbitrary integer and $R = \mathbb{Z}_b$. If the first k rows of $\boldsymbol{C}_j$ form the *identity matrix* and the other rows are zero, then the first n output values are the first n elements of the *van der Corput sequence in base b*, defined as $\psi_b(0), \psi_b(1), \psi_b(2), \ldots$, where $\psi_b : \mathbb{N} \to [0, 1)$ is the *radical inverse function* in base b, defined as follows: if i is an integer with digital b-ary expansion

$$i = a_0 + a_1 b + \cdots + a_{k-1} b^{k-1},$$

then

$$\psi_b(i) = a_0 b^{-1} + a_1 b^{-2} + \cdots + a_{k-1} b^{-k}.$$

The first n numbers of a van der Corput sequence fill up the unit interval quite uniformly for any large enough n. For $n = b^k$, the first n numbers are $\{0, 1/n, \ldots, (n-1)/n\}$.

Example 6.43 ♣ Tabulate the van der Corput sequence in base 3. □

If the first k rows of $\boldsymbol{C}_j$ form the *reflected identity* matrix (that is, the identity matrix but with its columns taken in reverse order):

$$\boldsymbol{C}_j = \begin{pmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{pmatrix},$$

then coordinate j runs through the values $0, 1/n, \ldots, (n-1)/n$ in this order.

Example 6.44 Suppose the generator matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ are the reflected identity and the identity, respectively, so the first coordinate of each point i is i/n and the second coordinate follows the van der Corput sequence. For $b = 2$, this is the two-dimensional *Hammersley net*. If $q_1 + q_2 = k$, then the matrix $\boldsymbol{M}$ in Theorem 6.16 contains the last q_1 rows of the $k \times k$ identity matrix, followed by the first $q_2 = k - q_1$ rows of this identity matrix. Obviously, this matrix has full rank. Therefore, this digital net is always a $(0, k, 2)$-net in base b.

In Section 1.5.3, we saw an example of this type of net, in base $b = 2$, with $k = 8$ (so $n = 256$). The corresponding point set is a $(0, 8, 2)$-net in base 2.

♣ Prove that this is also a $(0, 2)$-sequence? □

For digital sequences, it is customary to take $\boldsymbol{C}_1$ as the identity, so the first coordinate follows the van der Corput sequence. But when the number of points n is fixed in advance, we can always enumerate them by order of their first coordinate, which corresponds to taking $\boldsymbol{C}_1$ as the reflected identity. Then one can take the van der Corput sequence for the second coordinate, and for the other coordinates, we may use the same construction as for the infinite sequence, shifted by one coordinate. That is, the matrix $\boldsymbol{C}_j$ of the digital sequence becomes $\boldsymbol{C}_{j+1}$ for $1 \le j < s$. In this sense, we "gain" one dimension compared with the infinite sequence.

♣ Using Gray code to enumerate the points. Without changing first coordinate, equiv. to applying a permutation.

6.10.5 Sobol' sequences and nets

Sobol' (1967) introduced digital sequences in base $b = 2$, defined as follows. The generator matrices $\boldsymbol{C}_j$ are upper triangular binary matrices with 1's on the diagonal:

$$\boldsymbol{C}_j = \begin{pmatrix} 1 & v_{j,1,2} & \dots & v_{j,1,c} & \dots \\ 0 & 1 & \dots & v_{j,2,c} & \dots \\ \vdots & 0 & \ddots & \vdots & \\ & \vdots & & 1 & \end{pmatrix}.$$

For each integer $k > 0$, the first k rows and k columns of each $\boldsymbol{C}_j$ form a non-singular matrix. The bits in column c of $\boldsymbol{C}_j$ can be represented as an odd integer smaller than 2^c:

$$\begin{aligned} m_{j,1} &= 1, \\ m_{j,2} &= 2\,v_{j,1,2} + 1, \\ &\vdots \\ m_{j,c} &= 2^{c-1}v_{j,1,c} + \cdots + 2v_{j,c-1,c} + 1 = \sum_{l=1}^{c} 2^{c-1}v_{j,l,c}, \end{aligned}$$

where $v_{j,c,c} = 1$. Column c thus contain the c digits of the binary expansion of $m_{j,c}$, from the most to the least significant, followed by zeros. Sobol' calls the real numbers $v_{j,c} = 2^{-c}m_{j,c}$, for $c = 1, \dots, k$ and $j = 1, \dots, s$, the *direction numbers*. These numbers completely define the matrices $\boldsymbol{C}_j$.

To determine the integers $m_{j,c}$, for each j, we first select a primitive polynomial over $\mathbb{F}_2$, say

$$f_j(z) = z^{d_j} + a_{j,1}z^{d_j-1} + \cdots + a_{j,d_j},$$

of degree d_j, and choose the first d_j integers $m_{j,1}, \dots, m_{j,d_j}$. The integers $m_{j,d_j}, m_{j,d_j+1}, \dots$ are then determined by the recurrence

$$m_{j,c} = 2a_{j,1}m_{j,c-1} \oplus \cdots \oplus 2^{d_j-1}a_{j,d_j-1}m_{j,c-d_j+1} \oplus 2^{d_j}m_{j,c-d_j} \oplus m_{j,c-d_j}$$

for $c \geq d_j$, or equivalently,

$$v_{j,l,c} = a_{j,1}v_{j,l,c-1} \oplus \cdots \oplus a_{j,d_j-1}v_{j,l,c-d_j+1} \oplus v_{j,l,c-d_j} \oplus v_{j,l+d_j,c-d_j}$$

for $l \geq 1$, where $\oplus$ means bitwise exclusive-or (i.e., bitwise addition modulo 2).

♣ Should add a small example of this, eventually.

Sobol' (1967) has shown that with this construction, if the primitive polynomials $f_j(z)$ are all distinct, one obtains a (t, s)-sequence for some $t \leq d_1 + \cdots + d_s + 1 - s$. He then suggested to list the set of all primitive polynomials over $\mathbb{F}_2$ by increasing order of degree, starting with $f_1(z) \equiv 1$ (whose corresponding matrix $\boldsymbol{C}_1$ is the identity), and take $f_j(z)$ as the jth polynomial in the list, for $j \geq 1$. Such lists of primitive polynomials are easily available, e.g., at `http://fchabaud.free.fr/`.

For the initial direction numbers $m_{j,1}, \dots, m_{j,d_j}$, there are several possibilities, based on different selection criteria. The original values proposed by Sobol' were selected in terms of the $(1, \dots, 1)$- and $(2, \dots, 2)$-equidistribution only (i.e., by looking only at the first two bits of each coordinate). Lemieux, Cieslak, and Luttmer (2004) suggest different ones, based on stronger equidistribution properties.

If $n = 2^k$ is fixed in advance, we can gain one dimension as explained earlier: set the first coordinate of point i to i/n, so $\boldsymbol{C}_1$ becomes the reflected identity, and move all previous coordinates by one position to the right. In two dimensions, this gives the Hammersley net (see Example 6.44).

6.10.6 Faure sequences and nets in prime base b.

Faure (1982) proposed digital sequences with generator matrices

$$\mathbf{C}_j = \mathbf{P}^{j-1} \mod b \tag{6.86}$$

for $j = 1, \ldots, s$, where b is prime and $\mathbf{P}$ is a $k \times k$ upper triangular matrix whose entry (l, c) is

$$\binom{c-1}{l-1} = \frac{(c-1)!}{(l-1)!(c-l)!}$$

for $l \le c$ and is 0 for $l > c$. This gives $\mathbf{C}_1 = \boldsymbol{I}$ (the identity) and the other matrices $\mathbf{C}_j$ are defined recursively via $\mathbf{C}_j = \mathbf{P}\mathbf{C}_{j-1} \mod b$. We thus have an infinite sequence of points (because we can increase k as much as we want) and an unbounded number of dimensions.

Faure (1982) proved that if b is prime and $b \ge s$, this sequence is a $(0, s)$-sequence in base b. That is, for any integers $k > 0$ and $\nu \ge 0$, the point set $Q(k, \nu) = \{\boldsymbol{u}_i : i = \nu b^k, \ldots, (\nu+1)b^k - 1\}$, is a $(0, k, s)$-net in base b. If we pick a coordinate j, truncate its expansion to its first k digits, and go through the $n = b^k$ points of the set $Q(k, \nu)$, we obtain a permutation of $\{0, 1/n, \ldots, (n-1)/n\}$. By taking $\nu = 0$, this says that the first $n = b^k$ points of the sequence form a $(0, k, s)$-net in base b. These point sets $Q(k, \nu)$ are thus projection-regular. The point sets $Q(k, 0)$ are also dimension-stationary.

Again, if $n = b^k$ is fixed in advance, we can gain one dimension by setting the first coordinate of point i to i/n and moving all other coordinates by one position, as for the Sobol' nets. Faure (1982) has shown that for $b \ge s - 1$, the s-dimensional point set thus obtained is also a $(0, k, s)$-net in base b.

Unfortunately, the condition $b \ge s$ (or $b \ge s-1$ when n is fixed) is a practical limitation for the use of Faure sequences when the dimension s is large.

Example 6.45 If $s = 100$, the minimal prime base b that gives a $(0, k, s)$-net is $b = 101$, so $n = b^k$ must be a power of 101, which leaves little freedom. With $k = 1$, the $(0, 1, s)$-net property only guarantees that each of the 100 one-dimensional projections has exactly one point in each interval of the form $[i/101, (i+1)/101)$ for $i = 0, \ldots, 100$. It says nothing about the two-dimensional (or more) projections. For $k = 2$, we already have $n = 101^2 = 10201$ points and the $(0, 2, s)$-net property means that each one-dimensional projection has one point in each interval $[i/10201, (i+1)/10201)$ for $i = 0, \ldots, 10200$, and that each two-dimensional projection has one point in each square box of the form $[i/101, (i+1)/101) \times [j/101, (j+1)/101)$ for $i, j = 0, \ldots, 100$. □

♣ To motivate the following: plot some projections.

In the *generalized Faure sequence* (Tezuka 1995, Faure and Tezuka 2002), one applies a matrix scramble (see Section 6.10.3) to improve the uniformity without loosing the $(0, s)$-sequence and $(0, m, s)$-net properties.

6.10.7 Niederreiter and Niederreiter-Xing sequences.

♣ Niederreiter's seq. are defined for any prime power b. Can choose $b < s$ and still get reasonably good t-values. In base 2, the Niederreiter-Xing sequences have a better t-value than Sobol' sequences.

6.10.8 Hammersley Point Sets and Halton Sequence

Digital nets and sequences can be "generalized" by allowing different bases for the different coordinates, say b_j for coordinate j. For example, the point sets introduced long ago by Hammersley (1960) have

$$\boldsymbol{u}_i = (i/n, \psi_{b_1}(i), \psi_{b_2}(i), \ldots, \psi_{b_{s-1}}(i)),$$

for $i = 0, \ldots, n-1$, where the basis b_j used for coordinate $j+1$ is the jth smallest prime number. Here, $\boldsymbol{C}_1$ is the reflected identity and $\boldsymbol{C}_j$ is the identity for all $j > 1$. The corresponding infinite sequence, proposed by Halton (1960), takes

$$\boldsymbol{u}_i = (\psi_{b_1}(i), \psi_{b_2}(i), \ldots, \psi_{b_s}(i))$$

for all $i \geq 0$, where b_j is again the jth smallest prime. One drawback is that b_j becomes quite large for large j. In any case, the identity matrices $\boldsymbol{C}_j$ could also be replaced by more general generating matrices, which may give room to further improvement.

♣ Permutations and scrambles for improvement: See Faure and Lemieux (2009).

6.10.9 Integration Lattices

An interesting special case of a digital net is when $k = r = 1$. Then, $n = b$, all coordinates are multiples of $1/n$, each matrix $\boldsymbol{C}_j$ contains a single integer $a_j \in \mathbb{Z}_b \equiv \mathbb{Z}_n$, and the ith point can be written as

$$\boldsymbol{u}_i = i\boldsymbol{v} \bmod 1 = (i\boldsymbol{a} \bmod n)/n,$$

where $\boldsymbol{a} = (a_1, a_2, \ldots, a_s)$ and $\boldsymbol{v} = \boldsymbol{a}/n$. We recover the s-dimensional point set P_n used by a *lattice rule of rank 1*, introduced in Section 1.5.3. The vector $\boldsymbol{a}$ is called the *generating vector*. To make sure that each coordinate goes through all multiples of $1/n$, we will suppose that $\gcd(a_j, n) = 1$ for all j. In this case, P_n is fully projection-regular, and there is no loss of generality in assuming that $a_1 = 1$.

As an important special case, if $a_j = a^{j-1} \bmod n$ for all j, so $\boldsymbol{a} = (1,\ a,\ a^2,\ \ldots,\ a^{s-1})$, for some $a \in \mathbb{Z}_n$, we have a *Korobov rule*, introduced by Korobov (1959). The corresponding point set can also be written as

$$P_n = \{(x_0/n, \ldots, x_{s-1}/n) : x_0 \in \mathbb{Z}_n \text{ and } x_j = ax_{j-1} \bmod n \text{ for all } j > 0\}.$$

This is the set of all vectors of successive values produced by a LCG with modulus n and multiplier a, from all possible initial states (including 0), i.e., exactly the same as the set Ψ_t defined in Section 3.2.7. A Korobov rule is fully projection-regular and dimension-stationary if and only if $\gcd(n, a) = 1$ (L'Ecuyer and Lemieux 2000).

The LCG recurrence offers a convenient way to enumerate the points: for point i, start with $x_0 = i$ and apply the recurrence to get the successive coordinates, then divide by n. As an alternative, one can start with $u_0 = i/n$ and apply the recurrence $u_j = au_{j-1} \bmod 1$ to obtain the successive coordinates of the ith point $\boldsymbol{u}_i$. Korobov point sets are a special case of recurrence-based point sets, discussed in Section 6.10.11, where we describe another implementation method that precomputes and stores the output values u_j over all cycles of the LCG. These types of point sets have the advantage of being infinite-dimensional. one can generate as many coordinates as needed, from the recurrence. It is true that the coordinates of each point are periodic (with period at most $n - 1$), but the random shift discussed in Section 6.11 can easily remove this periodicity. On the other hand, having a single parameter a (for a given n) has the disadvantage of offering less room to optimize the uniformity compared with the more general rank-1 lattice.

In general, an *integration lattice* is a lattice of the form

$$L_s = \left\{ \boldsymbol{v} = \sum_{j=1}^{t} h_j \boldsymbol{v}_j \text{ such that each } h_j \in \mathbb{Z} \right\},$$

as in (3.19), where the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s \in \mathbb{R}^s$ are linearly independent over $\mathbb{R}$, and such that L_s contains $\mathbb{Z}^s$, the set of all integer vectors. The QMC approximation of μ obtained by taking $P_n = L_s \cap [0, 1)^s$ as a point set is a *lattice rule* (Sloan and Joe 1994). The dual lattice L_s^*, the generator matrix $\boldsymbol{V}$, and its inverse $\boldsymbol{W} = \boldsymbol{V}^{-1}$, are defined as in Sections 3.2.7 and 3.2.8. The lattice L_s contains $\mathbb{Z}^s$ if and only if $L_s^* \subseteq \mathbb{Z}^s$, if and only if all entries of $\boldsymbol{W}$ are integers. When this holds, $n = \det(\boldsymbol{W})$ and all entries of $\boldsymbol{V}$ are multiples of $1/n$. The *projection* of L_s over the subspace determined by the set of coordinates $I = \{i_1, \ldots, i_\eta\}$ is also a lattice, with point set $P_n(I)$.

The *rank* of L_s is defined as the smallest r such that one can find a basis of the form $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r, \boldsymbol{e}_{r+1}, \cdots, \boldsymbol{e}_s$, where $\boldsymbol{e}_j$ is the jth unit vector in s-dimensions. The point set of a lattice rule of rank r can be written as

$$P_n = \{i_1\boldsymbol{v}_1 + \cdots + i_r\boldsymbol{v}_r \ : 0 \le i_j < n_j \text{ for } 1 \le j \le r\},$$

where $n = n_1 \cdots n_r$, the n_j's are (unique) integers such that n_{j+1} divides n_j for $j = 2, \ldots, r$, $\boldsymbol{v}_j = \boldsymbol{a}_j/n$, and the $\boldsymbol{a}_j$ are linearly independent integer vectors (Sloan and Joe 1994). When $r > 1$, the point set P_n is not fully projection-regular. This is one of the reasons why lattices rules of rank 1 are by far the most popular.

Lattice rules normally have a fixed number of points, n. What if we want to add new points, to improve the accuracy, after having seen the evaluations at these n points, and without throwing them away? It is actually possible to construct a sequence of embedded lattices $L_s^1 \subset L_s^2 \subset L_s^3 \subset \ldots$, so that each lattice contains the previous one (Cranley and Patterson 1976, Joe and Sloan 1992, Hickernell et al. 2001). This type of construction

permit one to increase the cardinality of P_n sequentially, until the desired accuracy has been achieved. If $L_s^\xi \cap [0,1)^s$ contains n_ξ points, then $n_{\xi-1}$ must divide n_ξ, for each ξ. For example, the ξth rule can be a Korobov rule with $n = n_\xi$ points and multiplier $a = a_\xi$, where a_ξ mod $n_{\xi-1} = a_{\xi-1}$, for each ξ. The simplest (and usual) case is to take $n_\xi = 2^\xi$. Then, for each ξ, we have $a_\xi = a_{\xi-1}$ or $a_\xi = a_{\xi-1} + n_{\xi-1}$.

Example 6.46 ♣ Take an example from Hickernell et al. (2001). Give a table of a vs 2^ξ. □

An explicit expression for the integration error with a lattice rule is available when the integrand f has a Fourier series expansion that converges absolutely. That is, if f can be expanded as

$$f(\boldsymbol{u}) = \sum_{\boldsymbol{h}\in\mathbb{Z}^s} \hat{f}(\boldsymbol{h}) \exp(2\pi\sqrt{-1}\,\boldsymbol{h}\cdot\boldsymbol{u}), \tag{6.87}$$

with *Fourier coefficients*

$$\hat{f}(\boldsymbol{h}) = \int_{[0,1)^s} f(\boldsymbol{u}) \exp(-2\pi\sqrt{-1}\,\boldsymbol{h}\cdot\boldsymbol{u}) d\boldsymbol{u}$$

such that $\sum_{\boldsymbol{h}\in\mathbb{Z}^s} |\hat{f}(\boldsymbol{h})| < \infty$, then the integration error is (Sloan and Joe 1994):

$$E_n = \sum_{\mathbf{0}\neq\boldsymbol{h}\in L_s^*} \hat{f}(\boldsymbol{h}). \tag{6.88}$$

Each term in (6.87) represents a periodic function whose frequency is determined by the vector $\boldsymbol{h}$. For a smooth function, the coefficients $|\hat{f}(\boldsymbol{h})|$ tend to decrease quickly with the size of $\boldsymbol{h}$, because the large [small] $\boldsymbol{h}$'s correspond to high [low] frequency oscillations. In view of (6.88), this suggests that ideally, the dual lattice L_s^* should not contain short nonzero vectors $\boldsymbol{h}$.

The assumption that the series converges absolutely is very restrictive, however. It can hold only if f is continuous in the unit torus $[0,1)^s$, i.e., the function $f : \mathbb{R}^s \to \mathbb{R}$ defined by $f(\boldsymbol{x}) = f(\boldsymbol{x} \bmod 1)$ must be continuous. This holds in a minority of situations in simulation settings. But we will see in Section 6.11.3 that when the lattice is randomly shifted, the variance of the RQMC estimator can be written as in (6.88), but with the Fourier coefficients squared, under the only condition that the variance is finite. This confirms the need to avoid short nonzero vectors in the dual lattice. This immediately suggests the following simple selection criterion: maximize the length of the shortest nonzero vector in the dual lattice. We recover the same criterion as we had in Section 3.2.7 for MRGs! We can also define a criterion that takes into account the lower-dimensional projections, as in Equations (3.28) and (3.29).

Based on these observations, a general class of figures of merit can be defined by

$$\mathcal{M}_w(P_n) = \sum_{\mathbf{0}\neq\boldsymbol{h}\in L_s^*} w(\boldsymbol{h}) \quad \text{or} \quad \mathcal{M}'_w(P_n) = \sup_{\mathbf{0}\neq\boldsymbol{h}\in L_s^*} w(\boldsymbol{h}) \tag{6.89}$$

where $w : \mathbb{Z}^s \to \mathbb{R}$ is a *weight function* such that $w(\boldsymbol{h})$ decreases with $\|\boldsymbol{h}\|$ in a way that may try to mimic how we anticipate $|\hat{f}(\boldsymbol{h})|$ to decrease with $\|\boldsymbol{h}\|$, where $\|\cdot\|$ is an arbitrary norm.

In practice, the norm and the weights are chosen somewhat arbitrarily. These expressions measure (in a different way) the discrepancy between the empirical distribution of the point set and the uniform distribution, and we want them to be as small as possible. These figures of merit are equivalent to the *generalized spectral test* and the *weighted spectral test* defined by Hellekalek (1998), if we assume that $w(\boldsymbol{h}) > 0$ for each $\boldsymbol{h}$ and $\sum_{\boldsymbol{h}\in\mathbb{Z}^t} w(\boldsymbol{h}) < \infty$. Here, we allow $w(\boldsymbol{h}) = 0$.

Let $\|\boldsymbol{h}\|_2$ denote the Euclidean norm of $\boldsymbol{h}$. If we take $\mathcal{M}_w$ in (6.89), with $w(\boldsymbol{h}) = \max\{\ell^*_{|I|} : I \in \mathcal{J}, \mathbf{0} \neq \boldsymbol{h} \in L^*_I\}/\|\boldsymbol{h}\|_2$ when this set is nonempty, and $w(\boldsymbol{h}) = 0$ otherwise, we recover the criterion defined in (3.28), inverted (so we want to minimize it, instead of maximize). L'Ecuyer and Lemieux (2000) provide a table of parameters selected on the basis of this criterion.

If we put $w(\boldsymbol{h}) = \beta^2_{I(\boldsymbol{h})}\|\boldsymbol{h}\|^{-\alpha}$ for some integer $\alpha \geq 2$ and some coefficients β_I for each $I \subseteq \{1,\ldots,s\}$, where $I(\boldsymbol{h}) = \{j : h_j \neq 0\}$, and take $\mathcal{M}'_w$ (the sum), we obtain the *weighted p-alpha* defined by Hickernell (1998a). In the case where α an even integer, this $\mathcal{M}'_w$ can be computed in $O(ns)$ time via the formula

$$\tilde{\mathcal{P}}_\alpha = \beta_0^2\left[-1 + \frac{1}{n}\sum_{\boldsymbol{u}\in P_n}\prod_{j=1}^{s}\left(1 - \frac{(-1)^{\alpha/2}(2\pi\beta_j)^\alpha}{\alpha!}B_\alpha(u_j)\right)\right], \tag{6.90}$$

where $\boldsymbol{u} = (u_1,\ldots,u_s)$ and B_α is the Bernoulli polynomial of degree α. This criterion is not practical for large LCGs and MRGs, because it would be too long to compute (n is too large), but it is appropriate for QMC point sets. The criterion (3.28), in contrast, has a computing cost that does not depend much on n, but increases much faster than linearly with s. For the special case where $\beta_j = 1$ for all j, this criterion is known as $\mathcal{P}_\alpha$ (Sloan and Joe 1994). Other criteria can be found in Sloan and Joe (1994), Hickernell (1998b), Heinrich, Hickernell, and Yue (2004), for example.

Under a variety of settings, classes of functions can be defined so that the worst-case integration error for functions in the class is bounded by the variation of the function (in some sense) multiplied by the discrepancy. The weighted p-alpha comes up as the appropriate discrepancy when we consider classes of periodic smooth functions (with period 1 with respect to each coordinate) whose mixed partial derivatives up to order α are all square integrable. In that case, it is known that there exist integration lattices for which the discrepancy (and the worst-case integration error) are $O(n^{-\alpha+\delta})$ for an arbitrarily small $\delta > 0$. See L'Ecuyer (2009) for the details.

♣ To be detailed later.

♣ Give tables of parameters later on.

It is interesting to note that the oldest QMC method for an arbitrary dimension (to our knowledge), proposed by Niedermeyer (1951), was based on a rank-1 lattice whose generating vector $\boldsymbol{v}$ has all *irrational coordinates*. This construction provides an infinite sequence of points, whose ith point is $\boldsymbol{u}_i = i\boldsymbol{v} \bmod 1$. For any fixed n, one can take $P_n = \{\boldsymbol{u}_\nu,\ldots,\boldsymbol{u}_{\nu+n-1}\}$ for an arbitrary ν (usually $\nu = 0$). The error and variance expressions given earlier for integration lattices are not valid for this type of point set.

6.10.10 Polynomial Integration Lattices

See Lemieux and L'Ecuyer (2003), L'Ecuyer (2004).

6.10.11 Recurrence-based point sets

The usual way of defining an infinite-dimensional QMC point set is via a recurrence over the coordinates, in exactly the same way as we did in Section 3.1.2 to define an RNG. We have a finite state space $\mathcal{S}$, a transition function $f : \mathcal{S} \to \mathcal{S}$, an output function $g : \mathcal{S} \to [0, 1)$, and we define

$$P_n = \{\boldsymbol{u} = (u_0, u_1, \dots) : s_0 \in \mathcal{S},\ s_j = f(s_{j-1}),\ \text{and } u_j = g(s_j) \text{ for all } j\}. \tag{6.91}$$

This is the set of all vectors of successive output values produced by the RNG, from all possible initial states. The recurrence is assumed to be purely periodic, i.e., for each $s_0 \in \mathcal{S}$, there is a positive integer ν such that $s_\nu = f^\nu(s_0) = s_0$. Then the successive coordinates of each point are purely periodic. In general, there could be several disjoint cycles. The sum of the lengths of all cycles equals n, which is also the cardinality of $\mathcal{S}$ (duplicate points are counted as many times as they appear). The dimension t is infinite. The points are easy to enumerate simply by using the recurrence, even if the required dimension is random and unbounded. All recurrence-based point sets are fully projection-regular and dimension-stationary.

If the generator is an LCG, we obtain a Korobov lattice point set. In the case where the generator is linear over $\mathbb{F}_2$, as in Section 3.3, then it is equivalent to an infinite dimensional digital net in base 2, which can be truncated to the number of dimensions we want. This means, for example, that we can use small LFSR generators to construct digital nets.

One way of implementing such a point set is to precompute all cycles of the recurrence and store them in a list of cycles. This precomputation makes sense in the (frequent) case where the same point set is reused several times with independent randomizations, or if the required dimension is large. It implies the storage of n numbers only, compared with ns numbers for a general QMC point set. The points can be enumerated easily by going through all the cycles, using each value of each cycle as a staring point. In the case where the randomization is a random shift modulo 1, one can store the (non-randomized) real numbers u_j's directly, store the coordinates of the random shift in a separate vector, and apply the randomization (by addition modulo 1) on-the-fly when generating the points. If the randomization involves manipulations of the digits, as in digital random shifts or random matrix scrambles, it is better to store the blocks of digits in base b instead. For example, in the case of a digital net in base 2 defined via an LFRS generator, one would store the successive k-bit blocks over all the cycles, store the random shift in the same format, in a separate vector, and transform the coordinates into real numbers (on-the-fly) only after applying the digital shift.

♣ Example...

6.11 Randomized Quasi-Monte Carlo

6.11.1 General Setting

QMC methods give deterministic approximations for which error estimation is difficult. RQMC, which is actually a special type of GAV technique, provides one way of estimating the error (Owen 1998, L'Ecuyer and Lemieux 2002). RQMC starts with a QMC point set $P_n = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\}$, that cover the unit hypercube $[0,1)^s$ in a very uniform way, and randomizes P_n so that after the randomization:

(R1) it retains its high uniformity when taken as a set and
(R2) each individual point has the uniform distribution over $[0,1)^s$.

(The n here corresponds to the k of Section 6.9; we use n for compatibility with standard QMC notation.) Let $\tilde{P}_n = \{\boldsymbol{U}_0, \ldots, \boldsymbol{U}_{n-1}\}$ denote the randomized points. The estimator of $\mu = \mathbb{E}[f(\boldsymbol{U})]$ based on one copy of the randomization is

$$X_{\text{rqmc}} = Q_n = \frac{1}{n}\sum_{i=0}^{n-1} f(\boldsymbol{U}_i). \tag{6.92}$$

This randomization is repeated m times, independently, for some positive integer m, with the same P_n. Assume that any points taken from different randomizations are pairwise independent, in the sense that if $\boldsymbol{U}_i^{(j)}$ is the ith point from the jth randomization, then

(R3) for every i_1, i_2 and $j_1 \neq j_2$, $\boldsymbol{U}_{i_1}^{(j_1)}$ and $\boldsymbol{U}_{i_2}^{(j_2)}$ are independent.

Let $X_{\text{rqmc},1}, \ldots, X_{\text{rqmc},m}$ denote the m copies of X_{rqmc} obtained by these randomizations, and let $\bar{X}_{\text{rqmc},mn}$ and $S^2_{\text{rqmc},m}$ be their sample mean and sample variance.

Proposition 6.19 *Under Conditions (R1) to (R3),* $\mathbb{E}[\bar{X}_{\text{rqmc},mn}] = \mu$ *and*

$$\mathbb{E}[S^2_{\text{rqmc},m}] = \text{Var}[X_{\text{rqmc}}] = m\text{Var}[\bar{X}_{\text{rqmc},mn}].$$

Proof. [7] Property (R2) implies that $\mathbb{E}[f(\boldsymbol{U}_i)] = \mu$ for each i. Therefore, $\mathbb{E}[X_{\text{rqmc}}] = \mu$ and this implies unbiasedness of the average (the first part). For the second part, it suffices to show that the m copies of X_{rqmc} are pairwise uncorrelated. Condition (R3) implies that for $j_1 \neq j_2$, $f(\boldsymbol{U}_{i_1}^{(j_1)})$ and $f(\boldsymbol{U}_{i_2}^{(j_2)})$ are independent, so $\text{Cov}[f(\boldsymbol{U}_i^{(j_1)}),\ f(\boldsymbol{U}_\ell^{(j_2)})] = 0$. Then,

$$\begin{aligned}
\text{Cov}[X_{\text{rqmc},j_1},\ X_{\text{rqmc},j_2}] &= \text{Cov}\left[\frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{U}_i^{(j_1)}),\ \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{U}_i^{(j_2)})\right] \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{\ell=1}^{n} \text{Cov}[f(\boldsymbol{U}_i^{(j_1)}),\ f(\boldsymbol{U}_\ell^{(j_2)})] \\
&= 0.
\end{aligned}$$

[7] From Pierre: The proof is the same as parts (c) and (d) of Exercise 1.26. This proposition should probably be stated as a corollary of a general result on GAV.

With this result in hand, we can compute a confidence interval for μ exactly as in Section 6.9.1, if we assume that X_{rqmc} is approximately normally distributed. For a given computing budget mn, a larger m gives a more accurate variance estimator, but a larger n is likely to provide a more accurate estimator of μ because of a better uniformity of the point set. If the main goal is to estimate μ, we would usually take m somewhere from 5 to 25. But if our aim is really to estimate the variance (e.g., to compare the efficiencies of RQMC and MC), we should take a larger m.

6.11.2 Randomizations

6.11.2.1 Random shifts. Two simple randomizations that satisfy Conditions (R1) to (R3) of Section 6.11.1 with a small amount of change in the point set, *for an arbitrary point set* P_n, are the random shift modulo 1 and the random digital shift. They are defined as follows.

For the *random shift modulo 1*, we simply generate a *single point* $\boldsymbol{U}$ uniformly over $[0,1)^s$ and add it to each point of P_n, coordinate-wise, modulo 1. In the case of a lattice rule, the lattice structure of the points is preserved by the shift.

♣ Should add figure.

♣ Applying a random shift to a digital net does not give a net, but a shifted net. This is equivalent to randomly shifting the function f.

The *random digital shift in base b* is a digital version of this method. We generate again a single vector $\boldsymbol{U} = (U_1, \ldots, U_s)$ uniformly over $[0,1)^s$, write the digital expansion in base b of each of its coordinates, say $U_j = \sum_{\ell=1}^{\infty} D_{j,\ell} b^{-\ell}$, then add $D_{j,\ell}$ modulo b to the ℓth digit of the digital expansion in base b of the jth coordinate of each point $\boldsymbol{u}_i \in P_n$. For $b = 2$, the digit-wise addition modulo b becomes a bitwise exclusive-or, which is fast to perform on a computer. We already saw in Section 6.10.3 that for digital nets in base b, the digital shift in base b preserves the $\boldsymbol{q}$-equidistribution properties for all vectors $\boldsymbol{q}$. [8] In particular, it preserves the (t, k, s)-net properties of the point set.

For well-designed point sets, with good uniformity properties, these random shifts are usually sufficient. But for certain types of point sets, we can obtain a better upper bound on the variance with a deeper scramble (more randomization). This usually happens because a point set taken at random from the class in which we randomize can have better uniformity *on average* in the larger class (with more randomization) than in the restricted class (with a random shift only). These more involved randomizations are examined next.

6.11.2.2 Random matrix scrambles. For digital nets, additional randomization can be obtained by applying a *random matrix scramble* before the digital shift (Matoušek 1999, Faure and Tezuka 2002, Hong and Hickernell 2003, Owen 2003). That is, we left-multiply each matrix $\boldsymbol{C}_j$ by a random $r \times r$ lower triangular matrix $\boldsymbol{M}_j$, in the ring R. In the usual case where $R = \mathbb{Z}_b$, the standard approach is to generate the diagonal entries uniformly over $\{1, \ldots, b-1\}$, and the entries below the diagonal uniformly over $\{0, \ldots, b-1\}$, all independently. Just by itself, this random scramble does not satisfy (R1) (for example, the point $\mathbf{0}$ remains unchanged), so it must be followed by some type of random shift. Its

[8] From Pierre: This is actually true for *any* point set. Prove it.

combination with a digital random shift gives an *affine matrix scramble*. This was proposed by Matoušek (1999) and implemented by Hong and Hickernell (2003).

Some variants further restrict the class of matrices from which the $\boldsymbol{M}_j$'s are generated. The *i-binomial matrix scramble* of Tezuka and Faure (2002) adds the constraint that all entries on any given diagonal or subdiagonal of $\boldsymbol{M}_j$ are identical. The *striped matrix scramble* proposed by Owen (2003) adds the constraint that in any given column, all entries below the diagonal are equal to the diagonal entry, which is generated randomly over $\{1, \ldots, b-1\}$.

♣ Local properties of striped scramble...

6.11.2.3 A deeper scramble. Owen (1995) proposed a *nested uniform scramble*, as randomization method for digital nets. It randomly permutes the values $\{0, \ldots, b-1\}$ used for the digits $u_{i,j,\ell}$, independently across the coordinates and across the digits. For each coordinate j, this randomization acts as follows: Partition the interval $[0, 1)$ into b intervals of length $1/b$, and permute them randomly, moving the points along with the intervals. Then partition each interval $[(\ell-1)/b, \ell/b)$ into b subintervals of length $1/b^2$ and permute them randomly, using b independent permutations for the b different intervals. Repeat this recursively *ad infinitum*. Do it for each coordinate j, independently across coordinates. Each level of recursivity scrambles one digit in the base-b expansion of the points $\boldsymbol{U}_i$. In practice, the recursivity must be stopped at some level ℓ, which means that only the first ℓ output digits are scrambled. For example, if $b = 2$, $\ell = 31$ or so is usually sufficient. But still, we need $(1 + b + \cdots + b^{\ell-1})s$ independent permutations to scramble the first ℓ digits. This method is therefore very time-consuming.

– CLT with this scramble. See Loh (1996).

6.11.2.4 Asymptotic variance bounds. Suppose P_n is a (t, k, s)-net in base b, randomized by either a nested uniform scramble or a linear matrix scramble. Let $\sigma^2 = \mathrm{Var}[X]$ and X_{rqmc} be the RQMC estimator based on averaging over the n randomized points. In this setting, we have (Owen 1997, Owen 1998, Hong and Hickernell 2003, Owen 2003):

Proposition 6.20 *(a) If f is square integrable, then*

$$\mathrm{Var}[X_{\mathrm{rqmc}}] \le b^q \left(\frac{b+1}{b-1}\right)^s \frac{\sigma^2}{n}.$$

(b) If all the mixed partial derivatives of f satisfy a Lipschitz condition (see Owen 1997, Owen 1998) [9] *, then*

$$\mathrm{Var}[X_{\mathrm{rqmc}}] = O(n^{-3}(\log n)^s).$$

Part (a) says that the variance can never be much worse than with standard MC, for which the variance is σ^2/n; it cannot be worse by more than a constant factor that does not depend on n. Part (b) says that when f is sufficiently smooth, the variance converges to 0

[9] From Pierre: Add details

at a much faster rate with this type of RQMC estimator than with MC. Unfortunately, such smoothness holds only in a minority of typical simulation applications.

♣ Counter-example showing that these properties do not hold if we are only using the random shift?

6.11.3 Randomly-shifted lattice rules

6.11.3.1 Variance expression. Applying a random shift modulo 1 to an integration lattice was first proposed by Cranley and Patterson (1976). For a randomly-shifted lattice rule, we have an exact expression for the variance in terms of the square Fourier coefficients of f:

Proposition 6.21 (L'Ecuyer and Lemieux 2000) *If* $f : [0,1)^s \to \mathbb{R}$ *and* $\mathrm{Var}[f(\boldsymbol{U})] = \sigma^2 < \infty$*, then the variance with a randomly-shifted lattice rule is*

$$\mathrm{Var}[X_{\mathrm{rqmc}}] = \sum_{\mathbf{0} \neq \boldsymbol{h} \in L_s^*} |\hat{f}(\boldsymbol{h})|^2, \tag{6.93}$$

where L_s^* *is the dual lattice.*

Observe that the standard MC estimator is equivalent to using a randomly-shifted lattice rule that contains a single point. In that case, we have $L_s^* = \mathbb{Z}^s$, i.e., the dual lattice contains all integer vectors. Therefore, with n independent replications, we obtain

$$\mathrm{Var}[\bar{X}_n] = \frac{\sigma^2}{n} = \frac{1}{n} \sum_{\mathbf{0} \neq \boldsymbol{h} \in \mathbb{Z}^s} |\hat{f}(\boldsymbol{h})|^2.$$

Comparing this expression with (6.93), and noticing that L_s^* in (6.93) contains a fraction $1/n$ of the points of $\mathbb{Z}^s$, we find that the RQMC method does better than MC if and only if the squared Fourier coefficients are smaller on the average over the dual lattice than over $\mathbb{Z}^s$. In other words, the lattice should be constructed so that the vectors $\boldsymbol{h}$ for which $|\hat{f}(\boldsymbol{h})|^2$ is important do not belong to L_s^*. The most important vectors $\boldsymbol{h}$ are typically among the small ones, which correspond to the low-frequency terms in the Fourier expansion (the main trends of f). This agrees with our discussion of Section 6.10.9, leading to the criteria in (6.89).

6.11.3.2 Adding a baker transformation. For a randomly-shifted lattice rule, a simple technique that often reduces the variance significantly is the *baker transformation*, which replaces each coordinate u (after the shift) by $2u$ if $u \leq 1/2$ and by $2(1-u)$ if $u > 1/2$. This corresponds to stretching everything by a factor of two and folding back, along each coordinate. Hickernell (2002) has shown that this technique reduces the variance to $O(n^{-4+\epsilon})$ for non-periodic smooth functions.

6.11.4 Digital nets with a random digital shift

For a digital net, the counterpart of a random shift modulo 1 turns out to be a random digital shift in base b. It preserves all equidistribution properties. We also have an analogue

of Proposition 6.21, where the variance expression is in terms of a Walsh expansion of f instead of a Fourier expansion. This Walsh expansion has been used earlier by Larcher, Niederreiter, and Schmid (1996), Larcher et al. (1996), and Larcher and Pirsic (1999) to study the integration error for non-randomized digital nets, under the (strong) condition that the sum of Walsh coefficients converges absolutely.

Suppose b is a prime, so $\mathbb{Z}_b$ is the finite field $\mathbb{F}_b$. For vectors $\boldsymbol{h} = (h_1, \ldots, h_s) \in \mathbb{N}_0^s \stackrel{\text{def}}{=} \{0, 1, 2, \ldots\}$ and $\boldsymbol{u} = (u_1, \ldots, u_s) \in [0,1)^s$, let $h_j = \sum_{\ell=1}^{\infty} h_{j,\ell} b^{\ell-1}$ and $u_j = \sum_{\ell=1}^{\infty} u_{j,\ell} b^{-\ell}$ for each j, where all the digits $h_{j,\ell}$ and $u_{j,\ell}$ are in $\mathbb{Z}_b$, and define

$$\boldsymbol{h} \cdot \boldsymbol{u} = \sum_{j=1}^{s} h_j \cdot u_j = \sum_{j=1}^{s} \sum_{l=1}^{\infty} h_{j,l} u_{j,l}$$

The Walsh basis function that corresponds to $\boldsymbol{h}$, evaluated at $\boldsymbol{u}$, is

$$\phi_{\boldsymbol{h}}(\boldsymbol{u}) = e^{2\pi\iota \boldsymbol{h}\cdot\boldsymbol{u}/b}.$$

We can write the Walsh expansion of f as

$$f(\boldsymbol{u}) = \sum_{\boldsymbol{h}\in\mathbb{N}_0^s} \tilde{f}(\boldsymbol{h}) \phi_{\boldsymbol{h}}(\boldsymbol{u}),$$

where

$$\tilde{f}(\boldsymbol{h}) = \int_{[0,1)^s} f(\boldsymbol{u}) e^{-2\pi\iota \boldsymbol{h}\cdot\boldsymbol{u}/b} d\boldsymbol{u}. \tag{6.94}$$

is the Walsh coefficient of f in $\boldsymbol{h}$.

Define the *dual space* of the net P_n as

$$\mathcal{C}_s^* = \{\boldsymbol{h} \in \mathbb{N}_0^s : \boldsymbol{h} \cdot \boldsymbol{u} = 0 \text{ for all points } \boldsymbol{u} \in P_n\}.$$

♣ Explain the meaning of this. See L'Ecuyer (2004).

Proposition 6.22 (L'Ecuyer and Lemieux 2002) *If $f : [0,1)^s \to \mathbb{R}$ and $\mathrm{Var}[f(\boldsymbol{U})] = \sigma^2 < \infty$, then the RQMC variance for a digital net with a random digital shift is*

$$\mathrm{Var}[X_{\mathrm{rqmc}}] = \sum_{\boldsymbol{0}\neq\boldsymbol{h}\in\mathcal{C}_s^*} |\tilde{f}(\boldsymbol{h})|^2, \tag{6.95}$$

where $\mathcal{C}_s^*$ is the dual space of the net.

This result has an analogous interpretation as Proposition 6.21. It suggests that we should avoid small vectors $\boldsymbol{h}$ in the dual space $\mathcal{C}_s^*$, where the length of $\boldsymbol{h}$ can be defined (loosely) as the total number of digits $h_{j,\ell}$ required to write its components.

6.11.5 Randomizing the Halton sequence

Wang and Hickernell (2000) randomize the Halton sequence simply by selecting the starting point $\boldsymbol{u}_0$ randomly over $[0,1)^s$, and exploiting the fact that there is a simple way of getting $\psi_b(i+1)$ directly from $\psi_b(i)$, so the successive points can be generated without knowing their indices i in the original sequence. They show that this method satisfies conditions (R1) and (R2) of Section 6.11. In their numerical experiments, it performs better than randomly shifting (modulo 1) the Halton sequence.

♣ Nice recent overview by Faure and Lemieux (2009).

6.11.6 Variance Decomposition and Effective Dimension

We now describe a general way of decomposing the variance of $f(\boldsymbol{U})$, independently of any point set. This decomposition gives an heuristic justification of why RQMC often works well even in a large number of dimensions.

As we pointed out in Section 1.5.3, filling up the unit hypercube $[0,1)^s$ very uniformly is practically impossible when the dimension s is large. However, in many practical settings, the s-dimensional function f can be well approximated by a sum of lower-dimensional function, that depend only on a small number of coordinates of $\boldsymbol{u}$. Then, a *sufficient condition* for QMC to be efficient is that these low-dimensional functions are integrated with small error. For example, if $s = 1000$, a direct application of Eq. (**??**) is useless, but if f can be written approximately as a sum of three-dimensional functions (i.e., where each function depends only on three coordinates of $\boldsymbol{u}$ or less), then (**??**), with P_n replaced by its appropriate three-dimensional projection in each case, can be used to show that the integration error for each of these three-dimensional functions converges rapidly, provided that all important three-dimensional projections of P_n have fast-converging discrepancy.

For RQMC, this can be studied generally and rigorously using a *functional ANOVA decomposition* of f (Hoeffding 1948, Owen 1998, Liu and Owen 2006). The idea is to write f as

$$f(\boldsymbol{u}) = \mu + \sum_{I \subseteq \{1,\ldots,s\},\, I \neq \phi} f_I(\boldsymbol{u})$$

where each $f_I : (0,1)^s \to \mathbb{R}$ depends only on $\{u_i,\, i \in I\}$, the f_I's integrate to zero and are orthogonal, and the variance decomposes as $\sigma^2 = \sum_{I \subseteq \{1,\ldots,s\}} \sigma_I^2$ where $\sigma_I^2 = \mathrm{Var}[f_I(\boldsymbol{U})]$ for $\boldsymbol{U}$ uniformly distributed over $[0,1)^s$. The f_I are defined recursively by $f_\phi = \mu$ (a constant) and

$$f_I(\boldsymbol{u}) = \int_{(0,1)^{s-|I|}} f(\boldsymbol{u})\, d\boldsymbol{u}_{\bar{I}} - \sum_{I' \subset I} f_{I'}(\boldsymbol{u})$$

for $\phi \neq I \subseteq \mathcal{S}$, where the first integral is with respect to the coordinates of $\boldsymbol{u}$ whose indexes are not in I, denoted by $\boldsymbol{u}_{\bar{I}}$.

For a given function f, if $\sum_{I \in \mathcal{J}} \sigma_I^2 \approx \sigma^2$, for some "small" class $\mathcal{J}$ of subsets of $\{1,\ldots,s\}$, then it suffices to construct P_n so that the projections $P_n(I)$ are highly uniform for all $I \in \mathcal{J}$, in order to reduce the important variance terms σ_I^2.

In this context, f is said to have *effective dimension d in proportion ρ in the superposition sense* (Owen 1998) if

$$\sum_{|I|\le d} \sigma_I^2 \ge \rho\sigma^2.$$

If ρ is close to 1, this means that f is well approximated by a sum of d-dimensional (or less) functions. The following definitions reduce even further the number of projections that have to be considered.

Sometimes, $\sum_{I\in\mathcal{J}} \sigma_I^2$ is close to σ^2 if $\mathcal{J}$ contains all the sets I formed by indices that are not too far apart, and it suffices to have good uniformity for the corresponding projections. For example, if we simulate a single queue over a long time horizon, the random numbers used to generate the interarrival times and service times of customers that are close to each other in time have a much more important interaction than those that are far away. We say that f has effective dimension d in proportion ρ in the *successive-dimensions sense* (L'Ecuyer and Lemieux 2000) if

$$\sum_{I\subseteq\{i,\ldots,i+d-1\},\, 0\le i\le s-d} \sigma_I^2 \ge \rho\sigma^2$$

There are cases where the first few random numbers of the simulation are much more important than the others. If

$$\sum_{I\subseteq\{1,\ldots,d\}} \sigma_I^2 \ge \rho\sigma^2,$$

then f has effective dimension d in proportion ρ in the *truncation sense* (Caflisch, Morokoff, and Owen 1997). Low effective dimension in the truncation sense can sometimes be achieved by redesigning the simulation program (i.e., the function f) in a way that the first few uniforms account for most of the variance in f (Morokoff 1998, Caflisch, Morokoff, and Owen 1997, Fox 1999).

6.11.7 Transforming the Function f

We saw that RQMC is generally more effective when the function f is smoother (has less variability) and/or has lower effective dimension in some sense. One can often improve effectiveness by a large factor by transforming the function f so that the integral (the expectation) remains the same, but the variability and/or effective dimension are reduced. Sometimes, one can reduce the effective dimension simply by generating the random variates in a different order (Moskowitz and Caflisch 1996, Fox 1999). Several standard variance reduction techniques such as control variates, conditional Monte Carlo, and importance sampling, for example, can be applied to smooth out the integrand f before applying RQMC. Certain RQMC schemes also converge faster (with n) when the function f is periodic with period 1 in each coordinate, so it may be profitable to apply a transformation that makes the function periodic (Hickernell 2002, Sloan and Joe 1994). We now discuss different ways of transforming f.

6.11.7.1 Change of variables. The primary technique for reducing the variability of f is a change of variable. We select a differentiable one-to-one function $\varphi : [0,1]^s \to [0,1]^s$, where $\varphi(\boldsymbol{v}) = (\varphi_1(\boldsymbol{v}), \ldots, \varphi_s(\boldsymbol{v}))$ for $\boldsymbol{v} = (v_1, \ldots, v_s) \in [0,1]^s$, and write

$$\mu = \int_{(0,1)^s} f(\boldsymbol{u})\, d\boldsymbol{u} = \int_{(0,1)^s} f(\varphi(\boldsymbol{v}))J(\boldsymbol{v})\, d\boldsymbol{v} = \int_{(0,1)^s} g(\boldsymbol{v})\, d\boldsymbol{v},$$

where $J(\boldsymbol{v})$ is the Jacobian of the transformation φ at $\boldsymbol{v}$, defined as the determinant of the $s \times s$ matrix whose element (i, j) is $\partial\varphi_i(\boldsymbol{u})/\partial u_j$ (see Section 4.5.1). To estimate μ by MC or RQMC, we compute the function g (instead of f) at each sample point, and average. This is in fact equivalent to applying a change of measure as in importance sampling: the uniform density of $\boldsymbol{u}$ is replaced by the density $1/J(\boldsymbol{v})$ obtained for $\boldsymbol{u} = \varphi(\boldsymbol{v})$ when $\boldsymbol{v}$ is uniform. The purpose is also the same as in importance sampling. The aim is to select φ so that g has smaller variation than f. In the case of rare-event simulation, for example, f has a high narrow peak somewhere in the space (where the rare event occurs) and it is essential to smooth out that peak first (e.g., via a change of measure) before considering RQMC.

In theory, there is always a way of reducing the effective dimension to 1, as follows: replace f by g where $g(\boldsymbol{u}) = g(u_1) = G^{-1}(u_1)$, where G is the distribution function of the random variable $f(\boldsymbol{U})$, i.e., $G(x) = \mathbb{P}[f(\boldsymbol{U}) \le x]$ where $\boldsymbol{U}$ is uniform over $(0, 1)^s$. Finding this g is usually much too difficult, but there are special cases where it can be approximated (for example, if $f(\boldsymbol{U})$ can be approximated by a function of a linear combination of normal random variables).

6.11.7.2 Periodizing the function. For convenience, the transformations that periodize the function usually take the form of a one-dimensional change of variable applied to each coordinate. A general class of such transformations change the integrand $f(u_1, \ldots, u_s)$ into $f(\varphi(u_1), \ldots, \varphi(u_s))|\varphi'(u_1) \cdots \varphi'(u_s)|$, for some smooth one-to-one transformation $\varphi : [0, 1] \to [0, 1]$ such that $\varphi^{\ell}(0) = \varphi^{(\ell)}(1) = 0$ for $\ell = 1, \ldots, \alpha$. This transformation does not change the value of the integral. Specific choices of φ proposed in the literature include polynomial and trigonometric functions whose degrees or frequencies increase with α (Boyle, Lai, and Tan 2005, Hua and Wang 1981, Sloan and Joe 1994). A major difficulty is that while making the function periodic, this type of transformation can also increase its variation $V(f)$. In particular, we must be careful that $|\varphi'(u)|$ does not become too large.

A variant of this is to select a continuous transformation $\varphi : [0, 1] \to [0, 1]$ (not necessarily one-to-one) with the property that $\int_a^b \varphi(u)\, du = b - a$ for every interval $[a, b] \subseteq [0, 1]$, and $\varphi(0) = \varphi(1)$. Then,

$$\int_{[0,1]^s} f(\boldsymbol{u})\, d\boldsymbol{u} = \int_{[0,1]^s} f(\varphi(\boldsymbol{u}))\, d\boldsymbol{u},$$

where $\varphi(\boldsymbol{u}) = (\varphi(u_1), \ldots, \varphi(u_s))$. This can be conveniently (and equivalently) implemented by transforming the points P_n by applying φ to each coordinate, and keeping f unchanged (so the simulation program that computes f can remain the same). This applies to either deterministic or randomized points. When the points are randomized (RQMC), the transformation must be applied *after* the randomization.

A simple instance of this is the *baker's transformation*, which takes $\varphi(u) = 2u$ for $u \le 1/2$ and $\varphi(u) = 2(1 - u)$ for $u > 1/2$. It stretches each coordinate of each point $\boldsymbol{u}_i$ by a factor of two, then fold back the coordinates that become larger than 1. Equivalently, this transformation can be visualized as contracting the graph of f (for any given coordinate) horizontally by a factor of two, so the function is now defined over the interval $[0, 1/2]$ only,

and then making a mirror copy over the interval $[1/2, 1]$, so the transformed function is now symmetric with respect to $1/2$, and its periodic continuation of period 1 is continuous.

Hickernell (2002) shows that there exist lattice rules for which adding the baker's transformation to the random shift reduces the variance from $O(n^{-2+\delta})$ to $O(n^{-4+\delta})$ for non-periodic smooth functions. A similar result applies to a digital net with a random digital shift (Cristea et al. 2007). Empirical results showing significant variance reductions provided by the baker's transformation can be found in L'Ecuyer, Lécot, and Tuffin (2008), for example.

6.11.7.3 Reducing the effective dimension. The effective dimension of f can often be reduced by changing the way the random variates or the sample paths are generated. We will illustrate this by the situation where $f(\boldsymbol{U})$ can be written as a function of a multinormal vector. These ideas can be applied more generally, for example to generate a Lévy process.

Example 6.47 *Functions of a Multinormal Vector.* Suppose that our integrand of interest can be written as a function of the multinormal vector $\boldsymbol{Y} = (Y_1, \ldots, Y_s)$, say with mean zero (without loss of generality) and covariance matrix $\boldsymbol{\Sigma}$. That is, $\mu = \mathbb{E}[g(\boldsymbol{Y})]$ for a known function g, and $g(\boldsymbol{Y})$ is the estimator. For example, we may have a basket of c financial assets whose values evolve as (potentially correlated) geometric Brownian motions (GMBs). If the net payoff at time T is a function g of the c assets values at fixed observation times $0 < t_1 < \cdots < t_d = T$, then we have $s = cd$. This covers a wide range of option types.

To generate $\boldsymbol{Y}$, we decompose the covariance matrix as $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$ for some matrix $\boldsymbol{A}$, generate a vector $\boldsymbol{Z} = (Z_1, \ldots, Z_s) \sim N(\boldsymbol{0}, \boldsymbol{I})$, and return $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{Z}$ (see Section 2.10.3). There are many possibilities for the choice of $\boldsymbol{A}$. The most common method, the *Cholesky factorization*, takes $\boldsymbol{A}$ to be lower triangular.

A second possibility is an eigen-decomposition of $\boldsymbol{\Sigma}$, which takes $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}^{1/2}$ where $\boldsymbol{D}$ is a diagonal matrix that contains the eigenvalues of $\boldsymbol{\Sigma}$ in decreasing order and $\boldsymbol{P}$ is an orthogonal matrix whose columns are the corresponding unit-length eigenvectors. This is the decomposition used in *principal component analysis (PCA)*. It was suggested by Acworth, Broadie, and Glasserman (1998) as a way to reduce the effective dimension in the truncation sense, when simulating GBMs for option pricing via QMC. This decomposition selects the matrix $\boldsymbol{A}$ so that the maximum amount of variance of $\boldsymbol{Y}$ comes from Z_1, then the maximum amount of residual variance conditional on Z_1 comes from Z_2, then the maximum amount of residual variance conditional on (Z_1, Z_2) comes from Z_3, and so on. Thus, the method concentrates the variance in the first coordinates of $\boldsymbol{Z}$ as much as possible. If $Z_1, Z_2, \ldots, Z_s$ are generated by inversion, in this order, then this method minimizes the effective dimension in the truncation sense if we consider the variance of $\boldsymbol{Y}$.

This PCA technique, on the other hand, does not take into account the function g. It may turn out that with the PCA sampling scheme, $g(\boldsymbol{Y})$ depends very little on Z_1 and very much on Z_{25}, for example, even if Z_1 has more influence on the variance of $\boldsymbol{Y}$. In such a situation, PCA will miss its target. Ideally, one would like to find a decomposition $\boldsymbol{A}\boldsymbol{A}^{\mathsf{t}}$ that minimizes the effective dimension of the integrand $f(\boldsymbol{U}) = g(\boldsymbol{Y})$ in a given sense. This minimization depends on g. The objective could be to maximize the fraction of $\mathrm{Var}[g(\boldsymbol{Y})]$ that comes from Z_1, then maximize the fraction of the residual variance of $g(\boldsymbol{Y})$ that comes from Z_2 given Z_1, and so on. For nonlinear functions g, this is difficult. Imai and Tan (2002, 2004, 2006) suggest

making a linear approximation $\tilde{g}$ of g, obtained via a first-order Taylor expansion around a representative point in the unit cube, and use it to compute the first column of $\boldsymbol{A}$ so that Z_1 accounts for the maximal amount of variance of the linear approximation. This procedure is repeated for each new column of $\boldsymbol{A}$, which is computed so that the corresponding Z_j accounts for the maximal amount of residual variance of the linear approximation, given the previous columns. A distinct representative point must be selected for each new column of $\boldsymbol{A}$. The main difficulty is to find a good linear approximation at each step. This can easily be impractical, especially if g is highly nonlinear. On the other hand, using this technique for just a few steps (a few columns of $\boldsymbol{A}$) can be worthwhile.

In the special case where $\boldsymbol{Y}$ corresponds to the realization of a c-dimensional BM $\{\boldsymbol{W}(t) = (W_1(t), \ldots, W_c(t)), t \geq 0\}$ observed at times $0 = t_0 < t_1 < \cdots < t_d$, then $s = cd$, and the vector $\boldsymbol{Y} = (W_1(t_1), \ldots, W_c(t_1), \ldots, W_1(t_d), \ldots, W_c(t_d))^\intercal$ has a covariance matrix $\boldsymbol{\Sigma}$ that can be written as a Kronecker product; this can be exploited to speed up the computations, especially for PCA (Glasserman 2004). In this special case, $\boldsymbol{Y}$ can also be generated by Brownian bridge sampling, as explained in Section 2.14.4. When $c > 1$, this method requires the decomposition of a $(c \times c)$ conditional covariance matrix at each step to generate $\boldsymbol{W}(t)$ from its conditional distribution. These decompositions are usually computed beforehand, via either Cholesky, or PCA, or another method. When the observation points are equally spaced, many of these covariance matrices are the same. Brownian bridge sampling typically reduces the effective dimension in the truncation sense. But like for PCA, its overall impact depends on the function g. It is possible to construct instances of g for which it actually increases the effective dimension and the RQMC variance (Papageorgiou 2002, Wang and Sloan 2008). □

♣ Extensions to Lévy processes.

6.11.8 Examples of applications to option pricing

The following examples are inspired from Glasserman (2004), Section 5.5.1, and from Imai and Tan (2002). They are also used in L'Ecuyer (2009). Consider a set of c GMB processes, $\{S_i(t), t \geq 0\}$, $1 \leq i \leq c$, where S_i has *drift parameter* r and *volatility parameter* σ_i. That is,

$$S_i(t) = S_i(0) \exp\left[(r - \sigma_i^2/2)t + \sigma_i B_i(t)\right]$$

where B_i is a standard Brownian motion (Section 2.14.1). Moreover, we assume that the B_i's are correlated as follows: $\mathrm{Cov}[B_i(t+\delta) - B_i(t), B_j(t+\delta) - B_j(t)] = \rho_{i,j}\delta$ for all $\delta > 0$. We have an option whose discounted payoff is $e^{-rT}\max[\bar{S} - K, 0]$, where $\bar{S}$ can be either the geometric average

$$\bar{S}^{(\mathrm{G})} = \prod_{i=1}^{c}\prod_{j=1}^{d} S_i(t_j)^{1/cd} \tag{6.96}$$

or the arithmetic average

$$\bar{S}^{(\mathrm{A})} = \frac{1}{cd}\sum_{i=1}^{c}\sum_{j=1}^{d} S_i(t_j), \tag{6.97}$$

for fixed observation times $0 < t_1 < \cdots < t_d = T$. In all our examples, we take $t_j = jT/d$. Denoting $\boldsymbol{Y} = (B_1(t_1), \ldots, B_c(t_1), B_1(t_2), \ldots, B_c(t_2), \ldots, B_1(t_d), \ldots, B_c(t_d))^{\mathsf{t}}$, the element $((i-1)c+j), (i'-1)c+j')$ of $\boldsymbol{\Sigma} = \mathrm{Cov}(\boldsymbol{Y})$ is $\rho_{i,i'} \min(t_j, t_{j'})$.

The price of the option based on the geometric average is known exactly (Glasserman 2004), so we can take the payoff $\bar{S}^{(\mathrm{G})}$ as a CV to reduce the variance.

We use the following point sets:

(a) Sobol' nets with a random digital shift only (Sob-S),
(b) Sobol' nets with a left matrix scramble followed by a digital shift (Sob-LMS-S),
(c) Korobov lattice rules with a random shift modulo 1 (Kor-S), and
(d) Korobov lattice rules with a random shift modulo 1 followed by a baker transformation (Kor-S-B).

The primitive polynomials and the direction numbers for the Sobol' sequence were taken from Lemieux, Cieslak, and Luttmer (2004). The lattice rule parameters are from L'Ecuyer and Lemieux (2000). The parameters of these point sets are definitely not optimal. All standard normal random variables were generated by inversion.

The *variance reduction factor* (VRF) is defined as the Monte Carlo variance (per observation) divided by n times the variance of Q_n for the randomized QMC method. The RQMC variance was estimated by making $m = 100$ independent replications of the randomization. These VRFs are noisy, with a standard error of about 20 percent or more. The simulations were written in Java using SSJ.

Example 6.48 For our first numerical illustration, we take c independent assets with a single observation time, with the following parameters borrowed from Glasserman (2004): $d = 1$, $\rho_{i,j} = 0$, $T = 1$, $\sigma = 0.5$, $r = 0.05$, $S_i(0) = 100$, and $K = 100$. We first take $c = 5$, then $c = 10$. We consider the payoff based on the arithmetic average (6.97). The exact values and the MC variance per observation (the values of μ and σ^2) are approximately 11.72 and 305 for $c = 5$, and 9.207 and 145 for $c = 10$.

Table 6.7. Variance Reduction Factors for Example 6.48, without correlation, for $c = 5$ (Left Number) and $c = 10$ (Right Number)

Sobol' Nets

	$n = 2^{14}$	$n = 2^{16}$	$n = 2^{18}$
Sob+S	953, 168	2363, 162	7156, 180
Sob+LMS+S	733, 112	2265, 174	7058, 253

Korobov Lattice Rules

	$n = 16381$ $a = 5693$	$n = 65521$ $a = 944$	$n = 262139$ $a = 21876$
Kor+S	178, 74	312, 21	416, 117
Kor+S+B	376, 77	440, 89	3434, 425

Table 6.7 gives the empirical variance reduction factors observed for the selected point sets. We see that the VRFs (i.e., efficiency gains) increase rapidly with n for the 5-dimensional problem, especially for the Sobol' net, but much less rapidly in the 10-dimensional case. For the Korobov rules, the baker transformation helps significantly, but the Sobol' nets are doing even better, with or without LMS. For $c = 5$ and $n = 2^{18}$, they reduce the variance by a factor of 7000 compared with ordinary MC. All methods require approximately the same CPU time for a given value of n. This means that they require (at least) 7000 less CPU time than MC to compute an estimator of comparable precision. □

Example 6.49 We modify the previous example so that $\rho_{i,j} = 0$ is now replaced by $\rho_{i,j} = 0.4$ for $i \neq j$. Table 6.8 gives the VRFs. All values are for $c = 10$, for which we have $\mu \approx 15.77$ and $\sigma^2 \approx 674$. We compare two ways of sampling the vector $\boldsymbol{Y}$ by transforming a 10-dimensional vector of independent standard normals: (a) the Cholesky factorization (left number in each table entry) and (b) PCA (right number in each table entry). PCA definitely outperforms the Cholesky factorization, and the combination of PCA with randomized Sobol' nets gives the largest VRFs. The left matrix scramble brings some variance improvement. □

Table 6.8. Variance Reduction Factors for Example 6.49, with $c = 10$ and positive correlation, for Cholesky (Left Number) and PCA (Right Number).

Sobol' Nets

	$n = 2^{14}$		$n = 2^{16}$		$n = 2^{18}$	
Sob+S	289	882	508	3567	1033	10299
Sob+LMS+S	381	4931	491	11452	593	39831

Korobov Lattice Rules

	$n = 16381$, $a = 5693$		$n = 65521$, $a = 944$		$n = 262139$, $a = 21876$	
Kor+S	106	737	30	1614	193	4218
Kor+S+B	185	6820	217	6864	684	20984

Example 6.50 For our next illustration, we modify an example from Imai and Tan (2002): we take $c = 10$, $d = 25$, $\rho_{i,j} = 0.4$ for all $i \neq j$, $T = 1$, $r = 0.04$, $\sigma_i = 0.1 + 0.4(i - 1)/9$ for all i, $S_i(0) = 100$, and $K = 100$. This gives a 250-dimensional integration problem. The exact value and the MC variance are $\mu \approx 5.818$ and $\sigma^2 \approx 72.3$ (these values are accurate up to the given digits).

The results are in Table 6.9, in the same format as for Table 6.8. They are similar. The main difference is that here we have a 250-dimensional problem instead of a 10-dimensional one, so PCA has more room to reduce the effective dimension compared with Cholesky. The VRFs are smaller than in Table 6.8 with Cholesky, but the improvement provided by PCA over Cholesky is larger. □

Example 6.51 Here we consider an Asian option on a single asset ($c = 1$) whose price follows a GBM process. The payoff is based on the arithmetic average (6.97). In this context,

Table 6.9. Variance Reduction Factors for Example 6.50 (250 Dimensions) with Cholesky (Left) and PCA (Right)

Sobol' Nets

	$n = 2^{14}$		$n = 2^{16}$		$n = 2^{18}$	
Sob+S	10	1299	17	3184	32	6046
Sob+LMS+S	6	4232	4	9219	35	16557

Korobov Lattice Rules

	$n = 16381$, $a = 5693$		$n = 65521$, $a = 944$		$n = 262139$, $a = 21876$	
Kor+S	18	878	18	1504	9	2643
Kor+S+B	50	4553	46	3657	43	7553

as seen in Example 6.17, we can use the payoff based on the geometric average (6.96) as a control variate (CV) to reduce the variance. Here, we look at the improvement of RQMC over MC with and without the CV, with sequential sampling (SEQ), Brownian bridge sampling (BBS), and PCA, for an example with $S(0) = 100$, $r = \ln(1.09)$, $\sigma_i = 0.2$, $T = 120/365$, $t_j = D_1/365 + (T - D_1/365)(j-1)/(d-1)$ for $j = 1, \ldots, d$, for combinations of values of (D_1, d, K) given in Table 6.10. This table provides estimates of the exact value μ, the MC variance σ^2 without the CV, and the VRF $\sigma^2/\sigma^2_{\text{cv}}$, where σ^2_{cv} is the MC variance with the CV. These values are accurate at least to the digit given. We see that the CV alone (without RQMC) can reduce the variance by a huge factor, especially when d is small and the observation times are close to each other. This is because the geometric and arithmetic averages are almost the same in this case.

Table 6.10. Estimates of μ, σ^2, and the VRF $\sigma^2/\sigma^2_{\text{cv}}$, for Example 6.51

d	D_1	K	μ	σ^2	VRF
10	111	90	13.008	105	1.53×10^6
10	111	100	5.863	61	1.07×10^6
10	12	90	11.367	46	5400
10	12	100	3.617	23	3950
120	1	90	11.207	41	5050
120	1	100	3.367	20	4100

Table 6.11 gives the VRFs of RQMC over MC, with and without the CV, with approximately $n = 2^{16}$ points. We recall that the optimal CV coefficient depends on the RQMC point set and on the sampling method, because it depends on the estimator's variance and its covariance with the CV, which may vary significantly across the methods (Hickernell, Lemieux, and Owen 2005). In our experiments, these variances and covariances were estimated from the same simulation runs used to compute the estimators of μ.

Table 6.11. VRFs for Example 6.51 with and without CV, for Sequential Sampling (SEQ), Brownian bridge sampling (BBS), and PCA sampling. The Sobol' point sets with a random digital shift (Sob+DS) have $2^{16} = 65536$ points, and the Korobov rules with a random shift (Kor+S) and with a random shift followed by a baker's transformation (Kor+S+B) have $n = 65521$ and $a = 944$.

d	D_1	K	P_n	without CV			with CV		
				SEQ	BBS	PCA	SEQ	BBS	PCA
10	111	90	Sob+DS	9572	12549	14279	63	183	4436
10	111	90	Kor+S	5943	6014	13751	18	29	291
10	111	90	Kor+S+B	88927	256355	563665	90	177	668
10	111	100	Sob+DS	5764	6638	10309	42	82	1913
10	111	100	Kor+S	2224	3682	8782	12	31	397
10	111	100	Kor+S+B	27214	29042	313724	29	61	635
10	12	90	Sob+DS	2205	9053	12175	27	67	434
10	12	90	Kor+S	442	1720	13790	13	50	71
10	12	90	Kor+S+B	1394	26883	446423	31	66	200
10	12	100	Sob+DS	368	2025	9506	21	42	274
10	12	100	Kor+S	63	909	5039	8	26	47
10	12	100	Kor+S+B	133	1317	123650	18	54	119
120	1	90	Sob+DS	325	7079	15101	3	48	483
120	1	90	Kor+S	192	2025	984	5	47	75
120	1	90	Kor+S+B	394	15575	474314	13	55	280
120	1	100	Sob+DS	39	1776	10244	3	48	217
120	1	100	Kor+S	24	672	5538	3	23	29
120	1	100	Kor+S+B	29	1101	162531	9	29	144

Without the CV, RQMC reduces the variance by a large factor, especially when combined with BBS or PCA. The Korobov rule with the random shift and the baker's transformation provides the largest variance reduction. With the CV, significant *additional* VRFs are obtained by the RQMC methods on top of those obtained by the CV alone. In this case, the Sobol' net with a random digital shift is the best performer. As an illustration, in the first row of Table 6.11, for PCA, the additional VRF over MC+CV is around 4436, whereas the CV alone was already providing a VRF of around 1.53×10^6. The combined VRF with both methods is approximately 6.8×10^9. The CPU times per run are about 20% larger with PCA in this case (in our implementation), so plain (naive) MC would take about 5.6×10^9 times more CPU time to yield an estimator with equivalent precision. For $d = 120$, the CPU time for PCA sampling is about three times that of SEQ. With SEQ, our implementation needs about 2.7 seconds to make one million simulation runs and compute the estimators with and without CV for $d = 10$, and about 29 seconds for $d = 120$. These timings are for an AMD Athlon 64-bit processor running at 2.4 GHz. □

Example 6.52 *An Asian Option Under a Variance Gamma Process.* We consider an asset price that evolves according to a *geometric variance-gamma (VG)* process S defined as follows (Avramidis, L'Ecuyer, and Tremblay 2003, Avramidis and L'Ecuyer 2006, Madan, Carr, and Chang 1998):

$$S(t) = S(0) \exp\left[rt + X(G(t)) + \omega t\right],$$

where X is a BM with drift and variance parameters θ and σ, G is a gamma process with mean and variance parameters 1 and ν, X and G are independent, and $\omega = \ln(1 - \theta\nu - \sigma^2\nu/2)/\nu$. The process Y defined by $Y(t) = X(G(t))$ is a VG process (see Section 2.16.6). We want to estimate the value of $\mathbb{E}[e^{-rT} \max(\bar{S} - K, 0)]$ where $\bar{S} = (1/d) \sum_{j=1}^{d} S(t_j)$ and $t_j = jT/d$ for $0 \le j \le d$. To generate $(S(t_1), \ldots, S(t_d))$, we use (and compare) the following methods described in Section 2.16.6: sequential sampling (BGSS), Brownian and gamma bridge sampling (BGBS), and the difference of gammas bridge sampling (DGBS).

We ran simulations with the following parameters, taken from Avramidis, L'Ecuyer, and Tremblay (2003): $\theta = -0.1436$, $\sigma = 0.12136$, $\nu = 0.3$, $r = 0.1$, $T = 1$, $d = 16$, $K = 101$, and $S(0) = 100$. The exact value and the MC variance are $\mu \approx 5.725$ and $\sigma^2 \approx 29.89$. Table 6.12 gives the variance reduction factors of QMC compared with MC. The gain is large and DGBS provides the best improvement.

□

Example 6.53 Boyle et al. Boyle, Lai, and Tan (2005) consider a *spread option*, where $d = 1$, $c = 2$, and the payoff is $e^{-rT} \max[S_2(T) - S_1(T) - K, 0]$, which is again a function of a bivariate normal with known covariance matrix $\boldsymbol{\Sigma}$. To generate the payoff, they use importance sampling as follows: generate $S_1(T)$ from its original distribution, then generate $S_2(T)$ from its conditional distribution given that the payoff is nonzero, and multiply the estimator by the appropriate likelihood ratio. This reduces the variability of the integrand and makes it smoother. For RQMC, they use a two-dimensional lattice rule, and they periodize the function with polynomial and sine transformations. Their best results are with the transformations $\varphi_{\text{poly},4}(u) = u^4(35 - 84u + 70u^2 - 20u^3)$ and $\varphi_{\sin,3}(u) = (12\pi u - 8\sin(2\pi u) + \sin(4\pi u))/12\pi$, which are special cases of well-known classes of transformations. We ran some experiments to compare their proposed transformation with

Table 6.12. Variance Reduction Factors for Example 6.52 with BGSS (Left), BGBS (middle), and DGBS (right)

Sobol' Nets

	$n = 2^{14}$	$n = 2^{16}$	$n = 2^{18}$
Sob+S	37 359 585	41 421 1077	75 510 1154
Sob+LMS+S	29 530 557	49 565 995	77 735 1642

Korobov Lattice Rules

	$n = 16381$, $a = 5693$	$n = 65521$, $a = 944$	$n = 262139$, $a = 21876$
Kor+S	17 54 119	24 138 263	22 285 557
Kor+S+B	52 53 57	44 44 433	92 93 1688

the baker's transformation, which also periodizes the function, and found that $\varphi_{\text{poly},4}$ and $\varphi_{\text{sin},3}$ gave much larger variance reductions than the baker's transformation, for $n \approx 2^{16}$ and with the same lattices, for this two-dimensional example. We also tried an Asian option with $d = 2$, $t_1 = 1/2$, $t_2 = 1$, $S(0) = 100$, $K = 90$, $r = \ln(1.09)$, and $\sigma_1 = 0.2$, with sequential sampling combined with importance sampling, and the proposed transformations did slightly better than the baker's transformation (approximately by a factor of 2).

However, for higher dimensional problems, we observed the opposite: these transformations give much higher variance than the baker's transformation. For Example 3 with $d = 10$, $K = 90$, and $n \approx 2^{16}$, for instance, with sequential sampling, the higher-order transformations gave a larger variance than plain Monte Carlo. In other words, they annihilate all the RQMC gain. The explanation is that the higher-order transformations also increase the variation of the function, and the impact of this higher variation increases with s. □

6.12 Importance Sampling

6.12.1 Basics

Importance sampling (IS), introduced in Section 1.6, changes the input distributions that drive the model in order to concentrate the sampling effort in the most important areas of the sample space. Its primary use as an efficiency improvement tool is for dealing with *important rare events*. In this context, the input distributions are modified so that these rare events become more likely to happen. To recover an unbiased estimator of the original quantity of interest, the original (naive) estimator is multiplied by an appropriate *likelihood ratio*. Examples of applications include estimating the reliability of highly dependable systems, the fraction of losses in a queuing system with finite buffers, the fraction of long waiting times in a queue, the probability of bankruptcy of an insurance company, etc. (see the references in Section 6.12.11). In these application settings, standard MC estimators are highly inefficient because of the huge amount of simulation time that is typically required to observe a sufficient number of these rare events (viz. Examples 1.29 and 1.44).

We recall the basic ideas of IS, already outlined in Section 1.6. If the original estimator is $X = h(\boldsymbol{Y})$, where $h : \mathbb{R}^d \to \mathbb{R}$ and $\boldsymbol{Y} \in \mathbb{R}^d$ is a random vector with density π, the IS

estimator is

$$X_{\text{is}} = h(\boldsymbol{Y})\pi(\boldsymbol{Y})/g(\boldsymbol{Y})$$

where g is another density over $\mathbb{R}^d$ such that $g(\boldsymbol{y}) > 0$ for all $\boldsymbol{y}$ for which $h(\boldsymbol{y})\pi(\boldsymbol{y}) \neq 0$. We have $\mu = E_\pi[X] = E_g[X_{\text{is}}]$.

If the function h satisfies either $h(\boldsymbol{y}) \geq 0$ for all $\boldsymbol{y}$ or $h(\boldsymbol{y}) \leq 0$ for all $\boldsymbol{y}$, then we can (in theory) choose g so that $g(\boldsymbol{y})$ is proportional to $|h(\boldsymbol{y})|\pi(\boldsymbol{y})$, i.e., $h(\boldsymbol{y})\pi(\boldsymbol{y})/g(\boldsymbol{y})$ is a constant, say $\tilde{\mu}$. Then, the estimator becomes $X_{\text{is}} = \tilde{\mu}$, a constant, so its variance is reduced to 0. The fact that the probability density function g must integrate to 1 immediately yields $\tilde{\mu} = \int_{\mathbb{R}^d} h(\boldsymbol{y})\pi(\boldsymbol{y})d\boldsymbol{y} = \mu$. In other words, the normalizing constant in the definition of the optimal g is exactly the same quantity that we want to estimate. If we knew it, there would be no need to do the simulation in the first place! As a result, this zero-variance scheme is usually impractical to implement. Nevertheless, knowing the general form of the optimal g often gives insight on how to modify the input distributions in practice to reduce the variance.

In the general case where $h(\boldsymbol{y})$ can take both positive and negative values, the optimal density is $g(\boldsymbol{y}) = |h(\boldsymbol{y})|\pi(\boldsymbol{y})/\tilde{\mu}$, where $\tilde{\mu} = \int_{\mathbb{R}^d} |h(\boldsymbol{y})|\pi(\boldsymbol{y})d\boldsymbol{y}$. Thus the goal is to inflate the density (or likelihood) $\pi(\boldsymbol{y})$ of every outcome $\boldsymbol{y}$ by a factor proportional to the absolute cost $|h(\boldsymbol{y})|$, and normalize. In particular, the IS density should be 0 in the areas where $h(\boldsymbol{y}) = 0$.

For the case of a *discrete random variable* Y, say with probability mass function given by $\mathbb{P}[Y = y_k] = p(y_k)$ for $k = 0, 1, \ldots$, the method works as follows. In fact, Y can be a random vector, or any other type of random object with discrete distribution, and h a function that assigns a real number to each possible realization of Y, and the following applies in just the same way. We can change the probabilities $p(y_k)$ into another set of probabilities $q(y_k)$ such that $q(y_k) > 0$ whenever $h(y_k)p(y_k) \neq 0$. Then,

$$\begin{aligned} \mu &= \mathbb{E}_p[h(Y)] = \sum_{k=0}^{\infty} h(y_k)p(y_k) = \sum_{k=0}^{\infty} [h(y_k)p(y_k)/q(y_k)]q(y_k) \\ &= \mathbb{E}_q[h(Y)p(Y)/q(Y)] \end{aligned} \tag{6.98}$$

where $\mathbb{E}_p$ and $\mathbb{E}_q$ are the expectations when Y has the probability mass functions p and q, respectively. The (unbiased) IS estimator of μ is $X_{\text{is}} = h(Y)p(Y)/q(Y)$ where the random variable Y satisfies $\mathbb{P}\{Y = y_k\} = q(y_k)$. Here, the likelihood ratio is $L(Y) = p(Y)/q(Y)$. The probabilities q that minimize the variance are defined by $q(y_k) = |h(y_k)|p(y_k)/\tilde{\mu}$ where $\tilde{\mu} = \sum_{k=0}^{\infty} |h(y_k)|p(y_k)$.

6.12.2 General formulation and key properties

Let ω denote the randomness that drives the simulation and suppose we want to estimate

$$\mu = \mathbb{E}_{\mathbb{P}}[X] = \mathbb{E}_{\mathbb{P}}[h(\omega)] = \int_{\Omega} h(\omega)\mathbb{P}(d\omega)$$

where $\mathbb{P}$ is a probability measure over the sample space Ω (i.e., $\mathbb{P}[\omega \in A] = \mathbb{P}[A] = \int_A \mathbb{P}(d\omega)$ for every measurable subset A of Ω) and $\mathbb{E}_{\mathbb{P}}$ is the corresponding mathematical expectation. We can replace $\mathbb{P}$ by another probability measure $\mathbb{Q}$ such that $\mathbb{Q}$ dominates $\mathbb{P}$ in the region

where $h(\omega) \neq 0$; that is, for all measurable sets $A \subseteq \Omega$, $\int_A h(\omega)\mathbb{P}(d\omega) > 0$ implies that $\mathbb{Q}(A) > 0$. Under this assumption, the *likelihood ratio* $L(\mathbb{P}, \mathbb{Q}, \omega) = (d\mathbb{P}/d\mathbb{Q})(\omega)$, which is the Radon-Nikodym derivative of $\mathbb{P}$ with respect to $\mathbb{Q}$, exists whenever $h(\omega) \neq 0$, and we can write:

$$\begin{aligned} \mu &= \mathbb{E}_{\mathbb{P}}[h(\omega)] = \int_\Omega h(\omega) d\mathbb{P}(\omega) = \int_\Omega [h(\omega)(d\mathbb{P}/d\mathbb{Q})(\omega)]\, d\mathbb{Q}(\omega) \\ &= \mathbb{E}_{\mathbb{Q}}\left[h(\omega)L(\mathbb{P}, \mathbb{Q}, \omega)\right], \end{aligned}$$

where $\mathbb{E}_{\mathbb{Q}}$ is the expectation that corresponds to $\mathbb{Q}$, and $L(\mathbb{P}, \mathbb{Q}, \omega)$ is defined as 0 when $(d\mathbb{P}/d\mathbb{Q})(\omega)$ does not exist. This means that $X_{\text{is}} = h(\omega)L(\mathbb{P}, \mathbb{Q}, \omega)$ is an unbiased IS estimator for μ when ω is generated from $\mathbb{Q}$. This setting encompasses the cases discussed in Section 6.12.1. When $\mathbb{P}$ and $\mathbb{Q}$ have densities with respect to the Lebesgue measure, then $(d\mathbb{P}/d\mathbb{Q})$ is the ratio of these densities, as we saw earlier. When they correspond to discrete distributions, then the likelihood ratio becomes a ratio of probabilities. In general, ω may represent a complicated object, such as the sample path of a stochastic process, for example.

Theorem 6.23 *Taking $\mathbb{Q} = \mathbb{Q}^*$ defined by*

$$\mathbb{Q}^*(d\omega) = \mathbb{P}(d\omega)|h(\omega)|/\tilde{\mu},$$

where $\tilde{\mu} = \int_\Omega |h(\omega)| d\mathbb{P}(\omega)$, minimizes the variance of the IS estimator. The IS estimator with this $\mathbb{Q}^*$ becomes $X^*_{\text{is}} = \tilde{\mu}$ when $h(\omega) > 0$, $X^*_{\text{is}} = -\tilde{\mu}$ when $h(\omega) < 0$, and we have $\mathbb{Q}^*[h(\omega) = 0] = 0$. If $\mathbb{P}[h(\omega) \geq 0] = 1$ or $\mathbb{P}[h(\omega) \leq 0] = 1$, then this minimal variance is zero.

Proof. The likelihood ratio in this case is $L(\mathbb{P}, \mathbb{Q}^*, \omega) = \tilde{\mu}/|h(\omega)|$ and the estimator is $X^*_{\text{is}} = \tilde{\mu}h(\omega)/|h(\omega)|$. The second and third sentences of the statement then follow immediately. For the case where $h(\omega)$ can take both positive and negative values, for an arbitrary admissible measure $\mathbb{Q}$ and corresponding estimator X_{is}, we obtain from the Cauchy-Schwartz inequality:

$$\begin{aligned} \mathbb{E}_{\mathbb{Q}^*}[(X^*_{\text{is}})^2] &= \tilde{\mu}^2 = \mathbb{E}^2_{\mathbb{P}}[|h(\omega)|] = \mathbb{E}^2_{\mathbb{Q}}[|h(\omega)|L(\mathbb{P}, \mathbb{Q}, \omega)] \\ &\leq \mathbb{E}_{\mathbb{Q}}[h^2(\omega)L^2(\mathbb{P}, \mathbb{Q}, \omega)] = \mathbb{E}_{\mathbb{Q}}[(X_{\text{is}})^2]. \end{aligned}$$

By minimizing the second moment, we minimize the variance as well.

As we already said, this $\mathbb{Q}^*$ is usually impractical to compute and use because the normalization constant $\tilde{\mu}$ is as hard to compute as μ and because sampling from $\mathbb{Q}^*$ (if it was known) would be too difficult anyway. But the knowledge of its shape can provide insight on how to select a good $\mathbb{Q}$.

We know that the likelihood ratio is never negative. It is sometimes taken for granted that its expectation under $\mathbb{Q}$ equals 1. Proposition 6.24 reminds us that this is true only if $\mathbb{P}[A] > 0$ implies $\mathbb{Q}[A] > 0$ for every measurable set A, a condition often not verified in practice. For the changes of measure used in Examples 1.43, 1.41, and 1.44, for instance, this condition does not hold and the expected value of the likelihood ratio is less than 1.

Proposition 6.24 *If the Radon-Nikodym derivative $(d\mathbb{P}/d\mathbb{Q})(\omega)$ exists on a measurable set B such that $\mathbb{Q}[B] = 1$, then $\mathbb{E}_{\mathbb{Q}}[L(\mathbb{P}, \mathbb{Q}, \omega)] = \mathbb{P}[B]$.*

Proof. We have

$$\mathbb{E}_{\mathbb{Q}}[L(\mathbb{P},\mathbb{Q},\omega)] = \int_B [(d\mathbb{P}/d\mathbb{Q})(\omega)]d\mathbb{Q}(\omega) = \int_B d\mathbb{P}(\omega) = \mathbb{P}[B].$$

The following propositions give a necessary and sufficient condition for the IS estimator to have smaller variance than the standard MC estimator. It indicates that the likelihood ratio should be small when $h^2(\omega)$ is large.

Proposition 6.25 $\mathrm{Var}[X_{\mathrm{is}}] < \mathrm{Var}[X]$ *if and only if* $\mathbb{E}_{\mathbb{P}}[h^2(\omega)(1 - L(\mathbb{P},\mathbb{Q},\omega))] \geq 0$.

Proof. Note that

$$\mathbb{E}_{\mathbb{Q}}[X_{\mathrm{is}}^2] \;=\; \mathbb{E}_{\mathbb{Q}}[h^2(\omega)L^2(\mathbb{P},\mathbb{Q},\omega)] = \mathbb{E}_{\mathbb{P}}[h^2(\omega)L(\mathbb{P},\mathbb{Q},\omega)].$$

Therefore, $\mathrm{Var}[X] - \mathrm{Var}[X_{\mathrm{is}}] = \mathbb{E}_{\mathbb{P}}[X^2] - E_{\mathbb{Q}}[X_{\mathrm{is}}^2] = \mathbb{E}_{\mathbb{P}}[h^2(\omega)(1 - L(\mathbb{P},\mathbb{Q},\omega))]$.

The next proposition provides a simple *sufficient condition* under which the variance is guaranteed to be reduced at least by a factor $\rho \leq 1$.

Proposition 6.26 *Suppose that* $L(\mathbb{P},\mathbb{Q},\omega) \leq \rho$ *whenever* $h(\omega) \neq 0$, *for some constant* $\rho \leq 1$. *Then,*

$$\mathrm{Var}_{\mathbb{Q}}[X_{\mathrm{is}}] \leq \rho\mathrm{Var}_{\mathbb{P}}[X] - (1-\rho)\mu^2 \leq \rho\mathrm{Var}_{\mathbb{P}}[X]. \tag{6.99}$$

Moreover, the equality may occur only if for any $\epsilon > 0$, $Q[h^2(\omega)(L(\mathbb{P},\mathbb{Q},\omega) - \rho) > \epsilon] = 0$,

Proof. We have

$$\begin{aligned} \mathbb{E}_{\mathbb{Q}}[X_{\mathrm{is}}^2] &= \mathbb{E}_{\mathbb{Q}}[h^2(\omega)L^2(\mathbb{P},\mathbb{Q},\omega)] \\ &\leq \rho\mathbb{E}_{\mathbb{Q}}[h^2(\omega)L(\mathbb{P},\mathbb{Q},\omega)] \\ &= \rho\mathbb{E}_{\mathbb{P}}[h^2(\omega)] \;=\; \rho\mathbb{E}_{\mathbb{P}}[X^2]. \end{aligned} \tag{6.100}$$

Therefore $\mathrm{Var}_{\mathbb{Q}}[X_{\mathrm{is}}] = \mathbb{E}_{\mathbb{Q}}[X_{\mathrm{is}}^2] - \mu^2 \leq \rho\mathbb{E}_{\mathbb{P}}[X^2] - \mu^2 \leq \rho(\mathbb{E}_{\mathbb{P}}[X^2] - \mu^2) - (1-\rho)\mu^2 = \rho\mathrm{Var}_{\mathbb{P}}[X] - (1-\rho)\mu^2$. Moreover, if the set $\{\omega : h^2(\omega)(L(\mathbb{P},\mathbb{Q},\omega) - \rho) > \epsilon\}$ has a positive probability $\mathbb{Q}$, it is easy to see that the inequality in (6.100) must be strict. Note that for $\rho > 1$, the first inequality in (6.99) holds, but not the second one.

The above results provide insight on what to do and what to avoid when applying IS. We must change the measure in a way that $L(\mathbb{P},\mathbb{Q},\omega)$ remains small when $h^2(\omega)$ is large. This means that these ω should become more likely under $\mathbb{Q}$ than under $\mathbb{P}$. To avoid disaster (huge variance), one should make sure that $L(\mathbb{P},\mathbb{Q},\omega)$ cannot take huge values when $h(\omega) \neq 0$. These huge values occur when a realization ω having a given probability [or density] under $\mathbb{P}$ has a much smaller probability [or density] under $\mathbb{Q}$. This is exactly what causes the infinite variance in Example 1.38 when λ_0 is too large: for $\lambda_0 > \lambda$ the likelihood ratio increases exponentially with Y, so it takes huge values when Y is large.

Example 6.54 To generalize Example 1.43, we can replace the random variable Y by the sample point ω, and write $p = \mathbb{P}[A] = \mathbb{E}_{\mathbb{P}}[h(\omega)]$ where $h(\omega) = \mathbb{I}[\omega \in A]$. The zero-variance measure $\mathbb{Q}^*$ is defined by $\mathbb{Q}^*(B) = \mathbb{I}[A]\mathbb{P}[B]/\mathbb{P}[A] = \mathbb{P}[B \mid A]$. This $\mathbb{Q}^*$ reallocates all of the sampling effort to the area where A occurs. In practice, we should seek a $\mathbb{Q}$ that is easy to sample from and that resembles $\mathbb{Q}^*$, or at least for which the likelihood ratio tends to be small when A occurs. □

Example 6.55 Suppose our unbiased estimator of μ is $h(Y_1, \ldots, Y_T)$, where T is a stopping time with respect to the sigma-field generated by $\{Y_j,\, j \geq 1\}$, and $\mathbb{P}[T < \infty] = 1$. Suppose also that Y_1 has density π_1, and the density of Y_j conditional on $(Y_1, \ldots, Y_{j-1}) = (y_1, \ldots, y_{j-1})$ is $\pi_j(\cdot \mid y_1, \ldots, y_{j-1})$. If we replace these conditional densities π_j by other conditional densities g_j, the IS estimator becomes $X_{\text{is}} = h(Y_1, \ldots, Y_T)L(Y_1, \ldots, Y_T)$ where

$$L(y_1, \ldots, y_T) = \frac{\pi_1(y_1)\pi_2(y_2 \mid y_1) \cdots \pi_T(y_T \mid y_1, \ldots, y_{T-1})}{g_1(y_1)g_2(y_2 \mid y_1) \cdots g_T(y_T \mid y_1, \ldots, y_{T-1})}$$

when $T < \infty$. The densities g_j must satisfy the requirement that this likelihood ratio is well-defined (finite) whenever $h(y_1, \ldots, y_T)\pi_1(y_1)\pi_2(y_2 \mid y_1) \cdots \pi_T(y_T \mid y_1, \ldots, y_{T-1}) \neq 0$. Then we can verify that X_{is} is unbiased:

$$\begin{aligned}
\mu &= \mathbb{E}_\pi[h(Y_1, \ldots, Y_T)] \\
&= \sum_{n=1}^{\infty} \mathbb{E}_\pi[\mathbb{I}[T = n]\, h(Y_1, \ldots, Y_n)] \\
&= \sum_{n=1}^{\infty} \mathbb{E}_g[\mathbb{I}[T = n]\, h(Y_1, \ldots, Y_n)L(Y_1, \ldots, Y_n)] \\
&= \mathbb{E}_g[h(Y_1, \ldots, Y_T)L(Y_1, \ldots, Y_T)].
\end{aligned}$$

Suppose that in addition to the above assumptions, we can also write

$$h(Y_1, \ldots, Y_T) = \sum_{j=1}^{T} c_j(Y_j)$$

for some real-valued functions c_j. In this case, since $c_j(Y_j)$ depends only on $Y_1, \ldots, Y_j$, it suffices to multiply it by the partial likelihood ratio $L(Y_1, \ldots, Y_j)$. This gives the (unbiased) IS estimator

$$X_{\text{is}} = \sum_{j=1}^{T} c_j(Y_j)L(y_1, \ldots, y_j).$$

□

Example 6.56 Suppose $Y_1, \ldots, Y_d$ are i.i.d. with density π, and we change π to a new density g such that $g(y) > 0$ whenever $\pi(y) > 0$. We then apply IS by generating $Y_1, \ldots, Y_d$ i.i.d. from this new density. Suppose also that $\mathbb{E}_g\left[|\ln(\pi(Y_1)/g(Y_1))|\right] < \infty$ (a mild condition). The likelihood ratio is

$$L(Y_1, \ldots, Y_d) = \prod_{j=1}^{d} \pi(Y_j)/g(Y_j).$$

In this situation, it is shown in Exercise 6.35 that when $d \to \infty$, $L(Y_1, \ldots, Y_d) \to 0$ with probability 1 and $\mathrm{Var}[L(Y_1, \ldots, Y_d)] \to \infty$, despite the fact that $\mathbb{E}[L(Y_1, \ldots, Y_d)] = 1$ for all d. This suggests that we should keep d reasonably small, or/and choose $\mathbb{Q}$ closer to $\mathbb{P}$ when d increases. On the other hand, many IS schemes in practice are designed so that the random variates that are generated are not i.i.d. and several terms of the likelihood ratio cancel one another when d is large. We will see many examples of that. □

6.12.3 Zero-variance simulation of a Markov chain

♣ This has been moved from the intro.

A zero-variance IS sampling scheme also exists in the more general situation where the model is a discrete-time Markov chain (DTMC). To explain how it works, we take a DTMC $\{Y_j,\, j \geq 0\}$ with denumerable state space $\mathcal{Y} = \{0, 1, 2, \ldots\}$, transition probabilities $p(y, y') = \mathbb{P}[Y_j = y' \mid Y_{j-1} = y]$, and nonnegative one-step cost function $c : \mathcal{Y}^2 \to [0, \infty)$, so that a cost $c(Y_{j-1}, Y_j)$ is incurred at step j. We assume that the chain stops whenever it reaches a given set of absorbing states $\Delta \subset \mathcal{Y}$. We may put $p(y, y) = 1$ and $c(y, y) = 0$ for all $y \in \Delta$. Let $T = \inf\{j : Y_j \in \Delta\}$,

$$X = \sum_{j=1}^{T} c(Y_{j-1}, Y_j),$$

the total cost until absorption, and

$$v(y) = \mathbb{E}[X \mid Y_0 = y],$$

the expected total cost when starting in state y. We assume that $\mathbb{E}[T \mid Y_0 = y] < \infty$ and $v(y) < \infty$ for all $y \in \mathcal{Y}$. Theorem A.22 tells us that the function $v : \mathcal{Y} \to [0, \infty)$ satisfies the recurrence

$$v(y) = \sum_{y'=0}^{\infty} p(y, y')[c(y, y') + \mathbb{I}(y' \notin \Delta)\, v(y')], \qquad y = 0, 1, \ldots$$

Suppose we change the transition probabilities $p(y, y')$ to the new probabilities

$$q(y, y') = p(y, y') \frac{c(y, y') + \mathbb{I}(y' \notin \Delta)\, v(y')}{v(y)}.$$

The IS estimator of $v(y)$ is then

$$\begin{aligned} X_{\text{is}} &= \sum_{j=1}^{T} c(Y_{j-1}, Y_j) \prod_{i=1}^{j} L(Y_{i-1}, Y_i) \\ &= \sum_{j=1}^{T} c(Y_{j-1}, Y_j) \prod_{i=1}^{j} \frac{v(Y_{i-1})}{c(Y_{i-1}, Y_i) + \mathbb{I}(Y_i \notin \Delta)\, v(Y_i)} \\ &= v(Y_0), \end{aligned} \tag{6.101}$$

where the last equality is obtained by expanding the sum and products and simplifying the terms (see Exercise 1.29 and Kollman et al. 1999). This IS estimator is a constant, so it has zero variance. This zero-variance scheme applies in principle to Markov chains with more general (non-denumerable and high-dimensional) state spaces as well. Again, we cannot implement it exactly, because it requires knowledge of the entire function v, but a crude approximation may already bring a substantial variance reduction.

Example 6.57 A gambler (or an investor, as you prefer) starts with $y > 0$ dollars (an integer) and at each step, he gains one dollar win probability p and looses one dollar with probability $1-p$. We want to estimate his *ruin probability*, which is the probability that the amount he owns ever reaches 0. For $p \leq 1/2$, this probability is 1, so let us assume that $p > 1/2$. To simulate this process with IS, suppose we change p to $q = 1-p$. Then, under the new probabilities, ruin occurs with probability 1, because $q < 1/2$. Moreover, the likelihood ratio is a product of terms where each term is $p/q = p/(1-p)$ when the gambler wins one dollar and $(1-p)/(1-q) = (1-p)/p$ when the gambler looses one dollar. Note that the product of two terms that correspond to one win and one loss is 1, so two such terms cancel out in the likelihood ratio. As a result, when ruin occurs, the estimator X_{is} (which is equal to the likelihood ratio at the end) is always equal to $[(1-p)/p]^y$, because there must have been y more losses than gains. Thus, we have found a zero-variance estimator for this process. □

6.12.4 Zero-variance for a general Markov chain

In the previous subsection, we saw how to define a zero-variance sampling for a finite-state DTMC with a non-negative cost function. We now generalize it to a Markov chain with a general state space. The setting is taken from L'Ecuyer and Tuffin (2007).

We have DTMC $\{Y_j,\, j \geq 0\}$ with general state space $\mathcal{Y}$, transition kernel $\mathbb{P}$, and non-negative one-step cost function $c : \mathcal{Y}^2 \to [0, \infty)$. When the chain is in state $Y_{j-1} = y \in \mathcal{Y}$, the next state Y_j obeys a probability law defined by $\mathbb{P}(B \mid y) = \mathbb{P}[Y_j \in B \mid Y_{j-1} = y)$ for all (measurable) $B \subseteq \mathcal{Y}$, and a transition cost $c(y, Y_j)$ is incurred. The state space contains a set of absorbing states $\Delta \subset \mathcal{Y}$ from which the transition cost is zero: $\mathbb{P}(\{y\} \mid y) = 1$ and $c(y, y) = 0$ for all $y \in \Delta$. Let $\tau = \inf\{j : Y_j \in \Delta\}$, the number of steps until absorption,

$$X = \sum_{j=1}^{\tau} c(Y_{j-1}, Y_j),$$

the total cost until absorption, and

$$\mu(y) = \mathbb{E}[X \mid Y_0 = y],$$

the expected total cost when starting in state y. We assume that $\mathbb{E}[\tau \mid Y_0 = y] < \infty$ and $\mu(y) < \infty$ for all $y \in \mathcal{Y}$.

The function $\mu : \mathcal{Y} \to [0, \infty)$ satisfies the recurrence

$$\begin{aligned} \mu(y) &= \mathbb{E}[c(y, Y_1) + \mu(Y_1) \mid Y_0 = y] \\ &= \int_{\mathcal{Y}} [c(y, y_1) + \mu(y_1)] d\mathbb{P}(y_1 \mid y) \end{aligned}$$

for all $y \in \mathcal{Y}$.

We consider changing the transition kernel $\mathbb{P}$ for another kernel $\mathbb{Q}$ such that $\mathbb{Q}(B \mid y) > 0$ whenever $\int_B [c(y, y_1) + \mu(y_1)] d\mathbb{P}(y_1 \mid y) > 0$. The estimator X is then replaced by

$$X_{\text{is}} = \sum_{j=1}^{\tau} c(Y_{j-1}, Y_j) \prod_{i=1}^{j} L(Y_{i-1}, Y_i), \tag{6.102}$$

where $L(Y_{i-1}, Y_i) = (d\mathbb{P}/d\mathbb{Q})(Y_i \mid Y_{i-1})$. Let $\mathbb{E}_{\mathbb{Q},y}$ and $\text{Var}_{\mathbb{Q},y}$ denote the expectation and variance operators under $\mathbb{Q}$, from initial state $Y_0 = y$. We have $\mathbb{E}_{\mathbb{Q},y}[X_{\text{is}}] = \mu(y)$.

Suppose that we choose $\mathbb{Q} = \mathbb{Q}^*$ defined by

$$\mathbb{Q}^*(dy_1 \mid y) = \mathbb{P}(dy_1 \mid y) \frac{c(y, y_1) + \mu(y_1)}{\mu(y)} \tag{6.103}$$

when $\mu(y) > 0$ (this measure integrates to 1), and $\mathbb{Q}^*(\cdot \mid y) = \mathbb{P}(\cdot \mid y)$ when $\mu(y) = 0$. This measure gives zero variance:

Proposition 6.27

$$\text{Var}_{\mathbb{Q}^*,y}[X_{\text{is}}] = 0.$$

Proof. See L'Ecuyer and Tuffin (2007).

♣ Example: Suppose $\mu(y)$ is the probability of hitting a given set of absorbing states. Then Q^* becomes the h-transform of Doob.

♣ Note that this zero-variance strategy does not take the work into account.

♣ Proposition: Balance equations: under the zero-variance IS, over any cycle of positive probability, the LR must be equal to 1.

♣ Show how we can obtain variance bounds via Lyapunov functions, or sub-solutions to the Bellman equations.

6.12.5 Examples

Example 6.58 *The Asian option.* We consider again the Asian call option as in Examples 1.11 and 1.21, under the GMB model, with observation times $t_1, \ldots, t_d = T$. We want to estimate $v(s_0, T) = \mathbb{E}[Y(s_0)]$, where

$$\begin{aligned} Y(s_0) &= e^{-rT} \max(0,\, s_0 W - K), \\ W &= \frac{1}{d} \sum_{i=1}^{d} \exp\left[(r - \sigma^2/2) t_i + \sigma \sum_{j=1}^{i} \sqrt{t_j - t_{j-1}} Z_j \right] \end{aligned}$$

and $Z_1, \ldots, Z_d$ are i.i.d. $N(0,1)$. When K is large compared with $s_0 \exp[r - \sigma^2/2]$, we may want to increase the mean of the Z_j's to make $Y(s_0)$ positive more frequently.

Suppose we change the mean of Z_j from 0 to ν_j for each j. This adds the constant $\delta_i = \sigma \sum_{j=1}^{i} \nu_j \sqrt{t_j - t_{j-1}}$ to the exponent of the exponential. We may want to select the ν_j's so that this exponent becomes equal to $\ln(K/s_0)$, for example. The likelihood ratio that corresponds to this change of means is

$$L(\omega) = \prod_{j=1}^{d} \frac{\exp\left(-Z_j^2/2\right)}{\exp\left(-(Z_j-\nu_j)^2/2\right)} = \exp\left(\sum_{j=1}^{d}(\nu_j^2/2 - \nu_j Z_j)\right).$$

One simple way of applying this type of IS is to generate the Z_j's $\mathsf{N}(0,1)$ as usual, add the constant δ_i to the exponent of the exponential for each i, then multiply $Y(s_0)$ by $L(\omega)$ to obtain an unbiased IS estimator. □

Example 6.59 *A discrete-time Markov chain.* We consider a discrete-time stationary Markov chain $\{Y_j,\, j \geq 0\}$ evolving over the finite state space $\{0, 1, \ldots, K\}$, where $K \geq 2$. The initial state is $Y_0 = 0$. If the chain is in state y at any given step, the probability that it jumps to state y' for the next step is $p_{y,y'} = \mathbb{P}[Y_j = y' \mid Y_{j-1} = y]$. We want to estimate, by simulation, the probability μ that the chain visits the state K before returning to state 0 for the first time. Here the absorbing set is $\Delta = \{0, K\}$.

Define $T = \inf\{j \geq 1 : Y_j \in \{0, K\}\}$. The naive estimator of μ is the indicator $\mathbb{I}[Y_T = K]$. To apply IS while still having a stationary Markov chain, we replace the probabilities $p_{y,y'}$ by another set of probabilities $q_{y,y'}$, selected so as to increase the chances of reaching K before returning to 0.

A simple (naive) idea is to simply cut off all the paths that return to 0, by setting $q_{y,0} = 0$ for all $y > 0$, and then rescale by inflating the other probabilities appropriately: $q_{y,y'} = p_{y,y'}/(1 - p_{y,0})$ for $y, y' > 0$, and $q_{0,y'} = p_{0,y'}$ for all y'. Under these new probabilities, $Y_T = K$ with probability 1 and the likelihood ratio is

$$L(\omega) = L(Y_1, \ldots, Y_T) = \prod_{j=1}^{T} \frac{p_{Y_{j-1},Y_j}}{q_{Y_{j-1},Y_j}} = \prod_{j=2}^{T}(1 - p_{Y_{j-1},0}) \leq 1.$$

By Proposition 6.26, the variance per run is guaranteed to be reduced. On the other hand, it is unclear if the efficiency is improved, because this IS scheme may increase the average computing time per run, especially if the chain has a tendency to drift toward 0 and away from K. In any case, this change of measure is far from optimal in general.

The zero-variance change of measure for this example can be written as follows. Let $\mu(y) = \mathbb{P}[Y_T = K \mid Y_0 = y]$ under the original probabilities. These $\mu(y)$ satisfy the system of linear equations

$$\mu(y) = \sum_{y'=1}^{K-1} p_{y,y'}\mu(y')$$

for $y < K$, and $\mu(K) = 1$. The zero-variance IS changes each $p_{y,y'}$ to

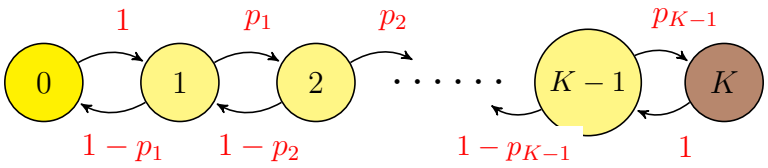
$$q_{y,y'} = p_{y,y'}\frac{\mu(y')}{\mu(y)}. \tag{6.104}$$

When K is not too large, it can be computed exactly by solving the linear system, but for applications that require simulation, K is too large to do that, and we must rely on some

form of approximation or heuristic to select the $q_{y,y'}$'s. The next example examines a special case with more structure. □

Example 6.60 *A discrete birth-and-death process over* $\{0, 1, \ldots, K\}$. As a special case of the preceding example, suppose that when in state y, the chain can only move to one of the two neighboring states $y-1$ and $y+1$. We assume that $p_{y,y+1} = p_y$ and $p_{y,y-1} = 1 - p_y$ where $0 < p_y < 1$, for $1 \le y \le K-1$, and $p_{0,1} = p_{K,K-1} = 1$. If the p_y's are small and K is large, the chain is drifted toward 0 and μ can be very small. A good IS scheme should turn this drift around, toward K.

Fig. 6.5. A random walk over $\{0, 1, \ldots, K\}$



Here we have $\mu(1) = p_1\mu(2)$, $\mu(y) = p_y\mu(y+1) + (1-p_y)\mu(y-1)$ for $y = 2, \ldots, K-1$, and $\mu(K) = 1$. In theory, we can compute the zero-variance IS by solving these equations. Alternatively, we can derive the zero-variance IS from the following observations. First, we note that under zero-variance IS, any sample path for which $Y_T = 0$ must have zero probability, while any sample path for which $Y_T = K$ must have positive probability, and the likelihood ratio must take the same value for any of these paths with $Y_T = K$. Since a path leading to K may have a random number of cycles of the form $y \to y+1 \to y$, for $1 \le y \le K-2$, the likelihood ratio over each such cycle must therefore be 1. This leads to the equations:

$$\frac{p_y(1-p_{y+1})}{q_y(1-q_{y+1})} = 1,$$

i.e.,

$$q_{y+1} = 1 - p_y(1-p_{y+1})/q_y,$$

for $y = 1, \ldots, K-2$, with $q_1 = 1$. Using the fact that $p_y(1-p_y) \le 1/4$, it is easily seen by induction on y that the solution satisfies $q_y > 1/2$ for all y. Interestingly, these new probabilities do not depend on K. They are also unchanged if we replace p_y by $1-p_y$ for all y. Of course, solving the equations for the $\mu(y)$ and using (6.104) yields the same solution.

In the special case where $p_y = p$ for all y, we obtain $q_2 = 1 - p(1-p)$ and $q_{y+1} = 1 - p(1-p)/q_y$ for $y \ge 2$. For $p = 1/2$, this gives $q_y = (y+1)/(2y)$ for $y \ge 2$. For $p = 1/4$, this gives $q_2 = 13/16$, $q_3 = 10/13$, $q_4 = 121/160$, and so on. Since $(1-q_{y+1})q_y = p(1-p)$ for all $y \ge 2$, and $q_1 = 1$, we find by induction on y that $1 > q_y > q_{y+1} > 1/2$ for all y, so q_y must converge to a limit $q_* = \lim_{y\to\infty} q_y \ge 1/2$. Then, $(1-q_*)q_* = \lim_{y\to\infty}(1-q_{y+1})q_y = p(1-p)$, which implies that $q_* = \max(p, 1-p)$. This means that when we get far from 0, it becomes approximately optimal to replace the original probabilities p by $q = 1-p$ if $p < 1/2$, and to keep the original probabilities otherwise.

The optimal q_y does not depend on K but it depends on the current state y, even if the original probabilities do not depend on y. Suppose now that we insist on using the same probability $q_y = q$ for all states $y \geq 1$. This is sometimes called *state-independent IS*. If K is large, the obvious choice of $q = q_*$ should do well, even though it is not optimal.

We now show how to arrive at this choice from different argument. Suppose that $p < 1/2$ (the interesting case). We replace p by some q. Observe that on any sample path that leads to K before returning to 0, there must be a single transition from 0 to 1, no transition from 1 to 0, and the total number of transitions to the right from states $y \geq 1$ must exceed the total number of transitions to the left by exactly $K - 1$. The likelihood ratio is then

$$L(\omega) \;=\; \prod_{j=1}^{T} \frac{p_{Y_{j-1},Y_j}}{q_{Y_{j-1},Y_j}} \;=\; \left(\frac{p}{q}\right)^{K-1} \left(\frac{p(1-p)}{q(1-q)}\right)^{(T-K)/2}.$$

We have $L(\omega) < 1$ for any sample path ω if and only if $q > p$ and $q(1-q) \geq p(1-p)$, i.e., $p < q \leq 1 - p$ (because $q(1-q)$ is a concave function of q which equals $p(1-p)$ when $q = p$ or $q = 1 - p$). Maximizing q under this constraint, i.e., taking $q = q_* = 1 - p$, maximizes the drift to the right and the probability of the event $\{Y_T = K\}$. Conditional on this event, $q = 1 - p$ also minimizes the average simulation time and simplifies the likelihood ratio to the constant

$$L(\omega) = (p/q)^{K-1} = [p/(1-p)]^{K-1}.$$

Proposition 6.26 then implies that the variance per run is divided at least by the factor $[(1-p)/p]^{K-1}$. This variance reduction factor can be arbitrarily large if p is small and/or K is large. For example, if $p = 1/3$ and $K = 101$, the variance is divided by $2^{100} \approx 1.2 \times 10^{30}$. □

Example 6.61 Suppose we have an $M/M/s$ queue that starts empty, and we want to estimate the probability that queue length builds up to size K before the system returns to the empty state. This system can be modeled as a Markov chain whose transitions are a customer arrival or departure. Here, p_y represents the probability that the next event is an arrival when there are y customers in the system. If λ is the arrival rate and μ is the service rate, we have $p_y = \lambda/(\lambda + \min(s, y)\mu)$. We are exactly in the setting of Example 6.60. In the case of an $M/M/1$ queue, we have $p_y = p = \lambda/(\lambda + \mu)$ for all states $y > 0$. In this case, the state-independent strategy that replaces p by $q = 1 - p$ is equivalent to permuting λ and μ when simulating the $M/M/1$ queue. □

♣ Other variants:
– the process does not stop at zero but can go toward $-\infty$ and is drifted in that direction. Want to estimate the probability of ever reaching K.
– Fixed number of steps, n. Want to estimate the probability that $Y_n \geq K$.
– Fixed number of steps. Want to estimate the probability of reaching either 0 or K in n steps or less, starting from y such that $0 < y < K$.

Example 6.62 *Maximum of a random walk with negative drift over the real line.* This simple model generalizes Example 6.60. It has several applications to be discussed later. The

state of a random walk at step n is [10]

$$D_n = D_0 + \sum_{j=1}^{n} Y_j, \qquad \text{for } n \geq 0, \tag{6.105}$$

where D_0 is fixed and the Y_j's are i.i.d. with density π and with $\mathbb{E}[Y_j] < 0$. By the strong law of large numbers, $\lim_{n\to\infty} D_n \to -\infty$ with probability 1. So, the walk starts at D_0 and wanders around for a while before eventually drifting toward $-\infty$. For a given constant $\ell > 0$, let $T_\ell = \inf\{n \geq 0 : D_n \geq \ell\}$, the time when the walk first crosses level ℓ, with $T_\ell = \infty$ if it never crosses. Define

$$\mu_\ell(x) = \mathbb{P}[T_\ell < \infty \mid D_0 = x] = \mathbb{P}[\max\{D_n,\, n > 0\} \geq \ell \mid D_0 = x]$$

and note that $\mu_\ell(x) = 1$ for $x \geq \ell$. We want to estimate $\mu_\ell(0)$ by simulation. The naive estimator $\mathbb{I}[T_\ell < \infty]$ is basically useless, because we can never be sure that $T_\ell = \infty$ unless we simulate the system for an infinite number of steps. Moreover, $\{T_\ell < \infty\}$ is a rare event when ℓ is large.

To apply IS, we can replace the density π by another density $g = g(\cdot \mid x)$ that may depend on the current state x. That is, when in state $D_n = x$, we generate Y_{n+1} from $g(\cdot \mid x)$. The zero-variance change of measure takes $g = g_*$, where

$$g_*(y \mid x) = \pi(y)\mu_\ell(x + y)/\mu_\ell(x). \tag{6.106}$$

One approach to approximate this g_* is to find a reasonable approximation for the function μ_ℓ and plug it into (6.106).

In the case where ℓ is large and Y_j has a finite and well-behaved moment generating function, it turns out that

$$\mu_\ell(x) \approx e^{-\theta^*(\ell - x)} \tag{6.107}$$

for some constant θ^* defined as follows. The *moment generating function* of Y_j is a function $M(\theta)$ defined by

$$M(\theta) = \mathbb{E}\left[e^{\theta Y_j}\right] = \int_{-\infty}^{\infty} e^{\theta y}\pi(y)dy.$$

Here, we *assume* that $M(\theta) < \infty$ in some neighborhood of $\theta = 0$. This is equivalent to assuming that Y_j has finite moments of all orders. The function M is convex, with $M(0) = 1$ and $M'(0) = \mathbb{E}[Y_j] < 0$. Let $\theta^* \stackrel{\text{def}}{=} \sup\{\theta > 0 : M(\theta) \leq 1\}$ and *assume* that $\theta^* < \infty$ (typically, we have $M(\theta) \to \infty$ when $\theta \to \infty$, so $\theta^* < \infty$). The function $M(\theta)$ then behaves as in Figure 6.6: It first decreases below 1, then increases, and we have $M(\theta^*) = 1$.

In risk theory, θ^* is called the *Lundberg parameter* and Eq. (6.107) is the *Lundberg approximation* (Asmussen 1987, Chapter XII).

Plugging this approximation of μ_ℓ in (6.106) gives

$$g(y \mid x) = \pi(y)e^{-\theta^* y}. \tag{6.108}$$

The integral of this density is $M(\theta^*)$, which equals 1 by construction. This g does not depend on x, so it gives a state-independent IS scheme. Under this new density, we have

[10] From Pierre: Should change D for Y, and n for j, and Y for ...

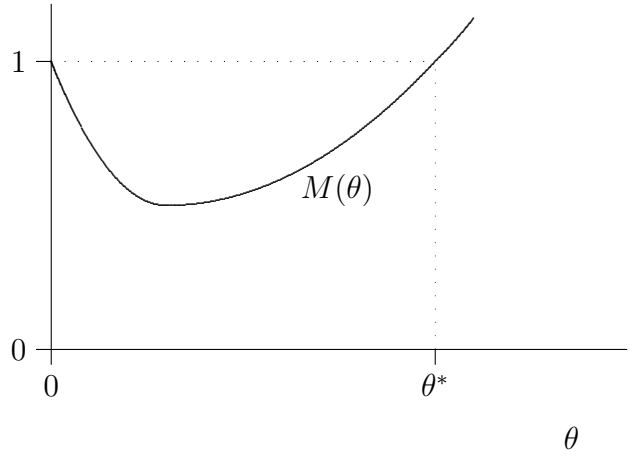


Fig. 6.6. The moment-generating function $M(\theta)$

$\mathbb{E}_{\theta^*}[Y_j] = M'(\theta^*) > 0$, so the walk drifts to the right and the event $\{T_\ell < \infty\}$ occurs with probability 1. We also have

$$L(\omega) \;=\; e^{-\theta^*\ell} e^{-\theta^*(D_{T_\ell}-\ell)} \;\le\; e^{-\theta^*\ell}, \tag{6.109}$$

which implies that the variance of the IS estimator is no more than $e^{-\theta^*\ell}$ that of the naive estimator. Thus, the desired probability can be written as

$$\mu_\ell = e^{-\theta^*\ell}\mathbb{E}_{\theta^*}[\exp(-\theta^*(D_{T_\ell} - \ell))]$$

and this IS scheme estimates the latter expectation instead of estimating μ_ℓ directly. When ℓ gets large, under IS, the variance of $\exp(-\theta^*(D_{T_\ell} - \ell))$ does not blow up and its expectation does not converge to 0, so its relative error remains "reasonable."

To view this from a different angle, suppose we consider replacing θ^* in (6.108) by an arbitrary parameter $\theta > 0$, and normalizing. That is, suppose we replace π by the *exponentially twisted* density

$$\pi_\theta(y) = e^{\theta y}\pi(y)/M(\theta) = e^{\theta y - \Psi(\theta)}\pi(y), \qquad y \in \mathbb{R},$$

where $M(\theta)$ turns out to be the normalization constant required to ensure that π_θ integrates to 1, and $\Psi(\theta) = \ln M(\theta)$ is the *cumulant generating function* of Y_j. This function has the property that $\mathbb{E}_\theta[Y_j] = \Psi'(\theta) = M'(\theta)/M(\theta)$, $\mathrm{Var}_\theta[Y_j] = \Psi''(\theta)$, and similarly for higher moments, where $\mathbb{E}_\theta$ and Var_θ are the expectation and the variance associated with the new density π_θ.

On the event $\{T_\ell < \infty\}$, the likelihood ratio is

$$L(\omega) \;=\; \prod_{j=1}^{T_\ell} \frac{\pi(Y_j)}{\pi_\theta(Y_j)} \;=\; [M(\theta)]^{T_\ell} \exp\left(-\theta \sum_{j=1}^{T_\ell} Y_j\right) \;=\; [M(\theta)]^{T_\ell} \exp\left(-\theta D_{T_\ell}\right).$$

So with exponential twisting, the likelihood ratio depends on the Y_j's only through T_ℓ and D_{T_ℓ}. To control this likelihood ratio and keep it as small as possible, we would like to take θ as large as possible, but keep $M(\theta) \le 1$, because otherwise $L(\omega)$ could blow up due to large values of T_ℓ. This immediately suggests taking $\theta = \theta^*$.

This example generalizes to the situations where the distributions of the Y_j's are not necessarily continuous and may differ for different values of j. If Y_j has distribution F_j, we replace this distribution by the twisted distribution $F_{\theta,j}$ such that

$$\frac{dF_{\theta,j}}{dF_j}(y) = \frac{e^{\theta y}}{M_j(\theta)}, \tag{6.110}$$

where M_j is the moment generating function of Y_j. The family of distributions $\{F_{\theta,j},\, \theta \in \Theta\}$, where Θ is the set of values of θ for which $M(\theta) < \infty$ and $F_{\theta,j}$ exists, is called a *conjugate family* of distributions.

♣ Exercise: Fill up the details for the case where the Y_j's have different densities π_j, with MGF $M_j(\theta)$.

□

Example 6.63 *Probability of large waiting times in a $GI/GI/1$ queue.* Consider a $GI/GI/1$ queue as in Section 1.11, where A_j is the interarrival time between customers j and $j+1$, while S_j and W_j are the service time and waiting time of customer j, respectively. We suppose that the A_j and S_j are mutually independent continuous random variables with densities h and g, respectively, and that $\mathbb{E}[S_j] < \mathbb{E}[A_j]$, so the queue is stable. The first customer arrives to an empty queue at time 0. We are interested in estimating

$$\mu_\ell = \mathbb{P}[W > \ell],$$

where ℓ is a constant and W is the waiting time of a customer in the queue, in steady-state. It is known from random walk theory (e.g., Asmussen 1987, Gut 1988) that W has the same distribution as $\sup\{D_n,\, n \geq 0\}$ where $D_n = \sum_{j=1}^{n}(S_j - A_j)$. Defining $Y_j = S_j - A_j$, we are in the framework of Example 6.62.

The density of Y_j is a convolution of g and h, namely

$$\pi(y) = \int_0^\infty g(x)h(x-y)dx, \qquad y \in \mathbb{R},$$

and the exponentially twisted density

$$e^{\theta y}\pi(y) = \int_0^\infty e^{\theta x} g(x) e^{-\theta(x-y)} h(x-y)dx, \qquad y \in \mathbb{R}$$

is obtained by applying an exponential twist to the densities h and g as follows:

$$h_\theta(x) = e^{-\theta x}h(x)/M_h(-\theta)$$

and

$$g_\theta(x) = e^{\theta x}g(x)/M_g(\theta)$$

where

$$M_h(-\theta) = \int_0^\infty e^{-\theta x}h(x)dx = \mathbb{E}\left[e^{-\theta A_j}\right] < 1$$

and

$$M_g(\theta) = \int_0^\infty e^{\theta x} g(x)dx = \mathbb{E}\left[e^{\theta S_j}\right] > 1.$$

Here, M_h and M_g are the moment generating functions of A_j and S_j, and the moment generating function of Y_j is $M(\theta) = M_g(\theta)M_h(-\theta)$. We *assume* that these moment generating functions are finite in some neighborhood of $\theta = 0$ (i.e., that S_j and A_j have finite moments of all orders). The change of measure inflates the service times density $g(x)$ by a factor that increases exponentially with x, and deflates the interarrival times density $h(x)$ by a factor that decreases exponentially with x. This makes the arrivals more frequent and the service times longer, thus increasing the likelihood that some D_n exceeds ℓ.

Sadowsky (1991) shows that $\theta^* < \infty$ if $\mathbb{P}[S_j > A_j] > 0$, which we *assume* (otherwise, no queue can build up and the problem becomes trivial). The function $M(\theta)$ then behaves as shown in Figure 6.6: It first decreases below 1, then increases, and we have $M(\theta^*) = 1$. As in Example 6.62, by taking $\theta = \theta^*$, we obtain $\mathbb{P}[T_\ell < \infty] = 1$ and (6.109) holds. We also have $\mathbb{E}_{\theta^*}[S_j - A_j] = M'(\theta^*) > 0$, which means that the queue is unstable under IS with θ^*.

□

Example 6.64 *Time to overflow in a $GI/GI/1/K$ queue.* (Parekh and Walrand 1989, Sadowsky 1991) We consider again a stable $GI/GI/1$ queue as in Example 6.63, except that we are now interested in estimating $\mathbb{E}[T_K]$, where T_K is the first time when there are K customers in the system. This may be interpreted as *buffer overflow* if the queue is maintained in a finite-size buffer. The customers could be packets in a telecommunication network, for instance, and the number $K - 1$ to exceed could be the size of a buffer used to store the packets waiting to be transfered at a communication switch, viewed as the queueing system.

We will use the regenerative property of this system: The regeneration times are the instants at which a new customer arrives while the system is empty. Define T_0 as the first positive time when this happens, and $T = \min(T_0, T_K)$. We have $T_K = T + (T_K - T)\mathbb{I}[T = T_0]$ and $\mathbb{E}[T_K - T \mid T = T_0] = \mathbb{E}[T_K]$, because the system regenerates at T_0. This gives the recurrence

$$\mathbb{E}[T_K] \;=\; \mathbb{E}[T] + \mathbb{E}[T_K]\mathbb{P}[T = T_0],$$

which implies

$$\mathbb{E}[T_K] \;=\; \mathbb{E}[T]/(1 - \mathbb{P}[T = T_0]) \;=\; \mathbb{E}[T]/\mathbb{P}[T = T_K].$$

Our problem has been turned into a ratio estimation problem. Typically, $\mathbb{E}[T]$ is easy to estimate, but $p_K = \mathbb{P}[T = T_K]$ is hard to estimate if the queue rarely builds up to size K. The idea then is to estimate $\mathbb{E}[T]$ by standard simulation and to estimate p_K by an independent set of simulation runs using IS. In what follows, we concentrate on the latter.

We will estimate p_K with the same change of measure as in Example 6.63, with the same θ^*. We also make the same assumptions ($M(\theta) < \infty$ around $\theta = 0$ and $\mathbb{P}[S_j > A_j] > 0$). A simulation run with IS stops at time T. Let N_A and N_S be the numbers of interarrival times A_j and service times S_j, respectively, that are generated up to time T. If $T = T_0$, the IS estimator takes the value 0. If $T = T_K$, then at time T_K, the $(N_A + 1)$th customer arrives, $N_S - 1$ customers have left so far, and there are K customers in the system, so we must have

$N_A = N_S + K - 2$, $T_K = \sum_{j=1}^{N_A} A_j \le \sum_{j=1}^{N_S} S_j$, and the IS estimator is equal to the likelihood ratio

$$\begin{aligned} L(\omega) &= \prod_{j=1}^{N_A} \frac{h(A_j)}{h_\theta(A_j)} \prod_{j=1}^{N_S} \frac{g(S_j)}{g_\theta(S_j)} \\ &= [M_h(-\theta)]^{K-2} [M_h(-\theta) M_g(\theta)]^{N_S} \exp\left(\theta \sum_{j=1}^{N_A} A_j - \theta \sum_{j=1}^{N_S} S_j\right) \\ &\le [M_h(-\theta)]^{K-2} [M(\theta)]^{N_S}. \end{aligned}$$

With $\theta = \theta^*$, this likelihood ratio becomes

$$L(\omega) = M_h(-\theta^*)^{K-2} \exp\left(\theta \sum_{j=1}^{N_A} A_j - \theta \sum_{j=1}^{N_S} S_j\right) \le M_h(-\theta^*)^{K-2} < 1$$

and we have a guaranteed variance reduction. The larger is θ, the smaller is $M_h(-\theta)$, and θ^* is the largest θ for which $M(\theta) \le 1$ (which guarantees that $L(\omega) \le 1$). The change of measure is the same as in Example 6.63, but the likelihood ratio differs because the number of interarrival times and service times that are generated are not the same in the two cases.

— Exercise: Try it.

— Exercise (Sadowsky 1991): Generalize to the $GI/GI/m$ queue.

□

Example 6.65 As a special case of Examples 6.63 and 6.64, consider an $M/M/1$ queue with arrival rate λ and service rate μ, where $\lambda < \mu$. The interarrival times A_j have density $h(y) = \lambda \exp(-\lambda y)$, for $y > 0$, and moment generating function

$$M_h(-\theta) = \int_0^\infty e^{-\theta y} \lambda e^{-\lambda y} dy = \frac{\lambda}{\lambda + \theta}$$

for $\theta > -\lambda$. Likewise, the moment generating function of the service times is $M_g(\theta) = \mu/(\mu - \theta)$ for $\theta < \mu$. By solving $M_h(-\theta) M_g(\theta) = 1$, i.e., $\lambda\mu = (\lambda + \theta)(\mu - \theta)$, we find $\theta^* = \mu - \lambda$. The twisted densities with $\theta = \theta^*$ are

$$h_\theta(x) = e^{-\theta^* x} h(x)/M_h(-\theta^*) = e^{-(\mu-\lambda)x} \lambda e^{-\lambda x}/M_h(-\theta^*) = \mu e^{-\mu x}$$

and

$$g_\theta(x) = e^{\theta x} g(x)/M_g(\theta^*) = e^{(\mu-\lambda)x} \lambda e^{-\mu x}/M_g(\theta^*) = \lambda e^{-\lambda x}.$$

Thus, this IS strategy simply permutes the arrival rate and the service rate. Recall that we arrived at this same change of measure from a different direction in Example 6.61. Since $M_h(-\theta^*) = \lambda/\mu$, this strategy divides the variance by at least $e^{(\mu-\lambda)\ell}$ in Example 6.63 and $(\mu/\lambda)^{K-2}$ in Example 6.64. □

Example 6.66 *Ruin probability of an insurance firm.* See Asmussen (1985), Asmussen (1988), Asmussen and Rubinstein (1995), Asmussen and Rubinstein (2000), Juneja and Shahabuddin (1999). We return to Example 1.44. Suppose that the claims to an insurance firm

arrive according to a stationary Poisson process $\{N(t),\, t \geq 0\}$ with rate λ, that the claim sizes are random variables $C_1, C_2, \ldots$, and that the premiums arrive at constant rate $c > 0$. The *reserve* (i.e., amount of money available) at time t, $R(t)$, obeys

$$R(t) = R(0) + ct - \sum_{j=1}^{N(t)} C_j, \tag{6.111}$$

where $R(0)$ is the initial reserve. We want to estimate the *ruin probability*, i.e., the probability that $R(t)$ eventually becomes negative, by simulation. Typically (hopefully) this probability is very small. Moreover, the first time when $R(t)$ becomes negative (if it ever does) must be at the occurrence of a claim. If t is the time of a claim, then we can write

$$R(0) - R(t) = \sum_{j=1}^{N(t)} (C_j - A_j c) = \sum_{j=1}^{N(t)} Y_j = D_{N(t)}$$

where $Y_j = C_j - A_j c$ and A_j is the time between claims $j-1$ and j. This puts the problem in the framework of Example 6.62. The ruin probability is the probability that $D_{N(t)}$ ever exceeds $\ell = R(0)$, i.e., the same as $\mu_\ell = \mathbb{P}[T_\ell < \infty]$ in Example 6.62.

To apply the IS scheme of Example 6.62, assume that the C_j's are i.i.d. with density f and finite moment generating function $M_f(\theta)$ in a neighborhood of 0. The moment generating function of A_j is

$$M_a(\theta) = \int_0^\infty e^{\theta x} \lambda e^{-\lambda x} dx = \frac{\lambda}{\lambda - \theta},$$

so the moment generating function of Y_j is

$$M(\theta) = \mathbb{E}\left[e^{\theta(C_j - cA_j)}\right] = M_f(\theta) M_a(-\theta c) = M_f(\theta) \frac{\lambda}{\lambda + \theta c}.$$

Here, IS replaces the density $f(x)$ by $f_\theta(x) = f(x)e^{\theta x}/M_f(\theta)$ and the exponential density $\lambda e^{-\lambda x}$ of A_j by $(\lambda+\theta c)e^{-(\lambda+\theta c)x}$, i.e., the rate of the Poisson process is increased to $\lambda_\theta = \lambda + \theta c$.

The Lundberg equation $M(\theta) = 1$ can be rewritten as $M_f(\theta) = (\lambda + \theta c)/\lambda$, and the Lundberg parameter $\theta^* > 0$ is the largest solution to that equation. With $\theta = \theta^*$, the Poisson rate becomes $\lambda_{\theta^*} = \lambda M_f(\theta^*) = \lambda + \theta^* c$. We also have $\mathbb{E}_{\theta^*}[Y_j] = M'(\theta^*) > 0$, so $D_{N(t)} \to \infty$ and therefore $R(t) \to -\infty$ when $t \to \infty$. That is, ruin occurs with probability 1 under IS. The likelihood ratio at ruin time T_ℓ is

$$L(\omega) = e^{\theta^*(R(T_\ell) - R(0))} \leq e^{-\theta^* R(0)}.$$

This IS scheme reduces the variance at least by the factor $e^{\theta^* R(0)}$. □

Example 6.67 *Estimating the cell loss ratio in an ATM switch.* (Chang et al. 1994, L'Ecuyer and Champoux 2001.) We consider a single queue fed by m_0 sources of arrivals. These sources are time-synchronized but otherwise independent discrete-time Markov modulated processes. Each source is OFF for a certain number of steps, producing nothing, then ON for a certain number of steps, producing exactly one customer per step, then OFF again,

and so on. The successive OFF and ON periods have mutually independent geometric durations with means κ_0 and κ_1, respectively. In other words, the state of a source evolves as a discrete-time Markov chain with the two states 0 = OFF and 1 = ON, and with the transition probability matrix

$$R = \begin{pmatrix} r_{00} & r_{01} \\ r_{10} & r_{11} \end{pmatrix} = \begin{pmatrix} 1 - 1/\kappa_0 & 1/\kappa_0 \\ 1/\kappa_1 & 1 - 1/\kappa_1 \end{pmatrix}. \tag{6.112}$$

where r_{ij} is the probability that the chain goes to state j at the next step if it is in state i at the current step. The arrival rate per source, which is the fraction of the time where the source is ON in the long run, is thus $\rho = \kappa_1/(\kappa_0 + \kappa_1)$, and the total arrival rate is $m_0\rho$.

At each time step, the server serves exactly c customers (if there are less than c in the system at that time, it serves them all) before the new arrivals join the queue. The system has a finite buffer size B, so that whenever there is B customers already in the system, any new arriving customer is lost and disappears. The aim is to estimate the fraction μ of customers that are lost in the long run. If $m_0\rho < c$ and B is large, μ can be quite small and hard to estimate. This model is a simplified representation of a single node at the first layer of an ATM switch. In that context, the customers are *cells of information* and μ, called the *cell loss ratio (CLR)*, is typically somewhere between 10^{-12} and 10^{-7}.

⋮

See L'Ecuyer and Champoux (2001) for the rest....

□

Example 6.68 *Pricing a down-and-in call option.* (Boyle, Broadie, and Glasserman 1997b) We return to the *down-and-in call option* model of Example 6.24, with low barrier ℓ, strike price K, time horizon T, and observation times $0 = t_0 < t_1 < \cdots < t_d = T$. The value of the option is the mathematical expectation of $X = e^{-rT}\max(0, S(T) - K)\mathbb{I}[T_\ell \le T]$, where $T_\ell = \inf\{t_j : S(t_j) < \ell\}$, and $S(\cdot)$ is assumed to follow a geometric Brownian motion as in Example 1.11. If ℓ is small and/or K is large compared with $S(0)$, a naive simulation will yield $X = 0$ most of the time, so we might consider importance sampling.

In order for X to be nonzero, the price of the asset must first go down below ℓ, then up above K. Note that conditional on $(t_j, S(t_j), S(T))$, the process $\{Y(t) = \ln S(t),\, t \ge 0\}$ is a Brownian bridge over $[0, t_j]$ and another Brownian bridge over $[t_j, T]$, so its conditional probability law does not depend on the original drift parameter μ. For the rare event to happen, $Y(t)$ must decrease by $\ln S(0) - \ln \ell$, then increase by $\ln K - \ln \ell$, i.e., cover a vertical distance of $\ln K + \ln S(0) - 2\ln \ell$ during T units of time. This gives an average speed (or slope) of $\tilde{\mu} = [\ln K + \ln S(0) - 2\ln \ell]/T$. As a heuristic, Boyle, Broadie, and Glasserman (1997b) propose an IS scheme that changes the drift of the Brownian motion $Y(t)$ to $\mu_1 = -\tilde{\mu}$ until the first time t_j when $S(t_j) < \ell$, then reverses the drift to $\mu_2 = \tilde{\mu}$ until time T. The volatility is left unchanged. In their setting, the observation times t_j are assumed equidistant and very frequent. They give numerical examples where the variance is reduced by factors ranging from 7 to 1124. With this heuristic, there is no guarantee that the likelihood ratio is less than 1 when the payoff is nonzero, and the probability that the barrier is reached is not 1, but the method works nicely in most cases.

This can be generalized to a process S defined by

$$S(t_j) = S(0)\exp(Y_1 + \cdots + Y_j)$$

where the Y_j's are i.i.d. random variables with some density π. Exponential twisting can be applied to the Y_j's with parameter $\theta = \theta_1$ until time T_ℓ, and $\theta = \theta_2$ thereafter. See Exercise 6.36 and Glasserman (2004), pages 264–266.

— Exercise: We might consider changing the volatility as well?

□

6.12.6 When things can go wrong

♣ The IS schemes discussed so far inflate the probability of the most likely sample paths, conditional on the occurrence of the rare event of interest. There are situations, however, where this strategy backfires. By inflating the probability of the (conditionally) most likely sample path and normalizing, we may decrease by a large factor the probability of another sample path whose contribution to the mean is not negligible, and this may inflate the variance considerably.

Example 6.69 Consider a random walk over the real line as in Example 6.62, but suppose we want to estimate the probability $\mu_{\ell,n} = \mathbb{P}[|D_n| > n\ell]$ for some fixed values of n and ℓ. If we increase the drift to one side, that would decrease the probability of hitting the rare event on the other side, and thus inflate the variance of the corresponding part of the likelihood ratio. See Asmussen (2002), Section 3, and Glasserman and Wang (1997) for details, examples, and other references. □

6.12.7 Asymptotics

[11] In this section, we parameterize the general model of Section 6.12.2 by a *rarity parameter* $\epsilon > 0$, so that h, $\mathbb{P}$, μ, X, and X_{is} now depend on ϵ. We suppose that as $\epsilon \to 0$, the costly events become rarer and the relative error of the naive estimator blows up: $\lim_{\epsilon\to 0} \text{RE}[X(\epsilon)] = \infty$.

An alternative unbiased estimator $Y(\epsilon)$ for μ is said to have *bounded relative error* if $\text{RE}[Y(\epsilon)]$ remains bounded as $\epsilon \to 0$. Obtaining an IS estimator with relative error bounded by some constant K is equivalent to finding a probability measure $\mathbb{Q}(\epsilon)$ such that

$$\text{Var}_{\mathbb{Q}(\epsilon)}[X_{\text{is}}(\epsilon)] \le K^2\mu^2(\epsilon).$$

In the special case where $X = \mathbb{I}[A]$, the indicator of some rare event A, a sufficient condition for this to happen is that the likelihood ratio satisfies $L(\mathbb{P}(\epsilon), \mathbb{Q}(\epsilon), \omega) \le K\mu(\epsilon)$ when the event A occurs. The latter condition implies in turn that $\mathbb{E}_{\mathbb{Q}(\epsilon)}[\mathbb{I}[A]] = \mathbb{E}_{\mathbb{P}(\epsilon)}[\mathbb{I}[A]/L(\mathbb{P}(\epsilon), \mathbb{Q}(\epsilon), \omega)] \ge 1/K$, i.e., that under the measure $\mathbb{Q}(\epsilon)$ the probability of A no longer converges to 0 as $\epsilon \to 0$.

Define $\gamma_0(1/\epsilon) = -2\ln\mu(\epsilon)$ and $\gamma_1(1/\epsilon) = -\ln \mathbb{E}_{\mathbb{Q}(\epsilon)}[X_{\text{is}}^2(\epsilon)]$. It is common place in rare event contexts that $\mu(\epsilon)$ decreases exponentially as a function of $1/\epsilon$, and γ_0 is often asymptotically linear. Since $\text{Var}_{\mathbb{Q}(\epsilon)}[X_{\text{is}}(\epsilon)] = \mathbb{E}_{\mathbb{Q}(\epsilon)}[X_{\text{is}}^2(\epsilon)] - \mu^2(\epsilon)$ cannot be negative, we have

$$-\gamma_1(1/\epsilon) = \ln \mathbb{E}_{\mathbb{Q}(\epsilon)}[X_{\text{is}}^2(\epsilon)] \ge 2\ln\mu(\epsilon) = -\gamma_0(1/\epsilon).$$

[11] From Pierre: To be updated in view of L'Ecuyer, Blanchet, Tuffin, and Glynn (2008).

The squared relative error of the IS estimator can be written as

$$\mathrm{RE}^2[X_{\mathrm{is}}(\epsilon)] = \frac{\mathrm{Var}_{\mathbb{Q}(\epsilon)}[X_{\mathrm{is}}(\epsilon)]}{\mu^2(\epsilon)} = \exp[\gamma_0(1/\epsilon) - \gamma_1(1/\epsilon)] - 1,$$

so the bounded relative error requirement is equivalent to having $|\gamma_0(1/\epsilon) - \gamma_1(1/\epsilon)|$ bounded.

A *weaker requirement* is that

$$\lim_{\epsilon \to 0} \frac{\ln \mathrm{RE}[X_{\mathrm{is}}(\epsilon)]}{-\ln \mu(\epsilon)} \le 0, \tag{6.113}$$

which means that $\ln \mathrm{RE}[X_{\mathrm{is}}(\epsilon)]$ grows at a slower rate than $\ln(1/\mu(\epsilon))$ when $\epsilon \to 0$. When (6.113) holds, the IS estimator is said to be *asymptotically optimal*, or *asymptotically efficient*, or *logarithmically efficient* (Heidelberger 1995, Asmussen and Rubinstein 2000). Some authors (e.g., Glasserman and Kou 1995) call an estimator asymptotically efficient only when the slightly stronger condition $\lim_{\epsilon \to 0} \gamma_1(1/\epsilon)/\gamma_0(1/\epsilon) = 1$ is satisfied. This condition is sufficient for (6.113) to hold, because it implies that

$$\lim_{\epsilon \to 0} \frac{\ln \mathrm{RE}[X_{\mathrm{is}}(\epsilon)]}{-\ln \mu(\epsilon)} \le \lim_{\epsilon \to 0} \frac{\gamma_0(1/\epsilon) - \gamma_1(1/\epsilon)}{\gamma_0(1/\epsilon)} = 0. \tag{6.114}$$

If $\mu(\epsilon)$ decreases as $\exp(-\kappa/\epsilon - o(1/\epsilon))$ for some constant $\kappa > 0$, for example, then an asymptotically optimal IS scheme is one for which the relative error grows slower than exponentially as a function of $1/\epsilon$ (e.g., it is allowed to grow polynomially in $1/\epsilon$, or even grow as $\exp(\epsilon^{-0.999})$, for example). If the relative error is bounded by a polynomial function of $\gamma_0(1/\epsilon)$ for ϵ small enough, Asmussen and Rubinstein (1995) say that we have a *polynomial-time estimator*. This property is weaker than having bounded relative error, but stronger than being asymptotically optimal.

An IS estimator that is asymptotically optimal, or that has bounded relative error, is not necessarily a minimal variance estimator for a given ϵ, neither asymptotically when $\epsilon \to 0$, but it is certainly a good step in the right direction.

Example 6.70 For the model described in Example 6.64, with $\epsilon = 1/K$, Sadowsky (1991) has shown that the IS scheme we have described there, with $\theta = \theta^*$, is the *unique* asymptotically optimal change of measure for estimating p_K among all changes of measure under which the queue remains $GI/GI/1$. Under very mild conditions, p_K decreases exponentially with K, and this asymptotically optimal IS scheme is the only one for which the required sample size to keep the relative error below a given threshold does not grow exponentially fast with K. Sadowsky (1991) has also generalized these results to the case of a $GI/GI/m$ queue.

For Example 6.62, which covers Example 6.63 in particular, Lehtonen and Nyrhinen (1992b) have shown that the IS scheme that we have described, with $\theta = \theta^*$, is the unique asymptotically optimal change of measure among essentially all the changes of measures for which the Y_j's remain i.i.d.. The results of Lehtonen and Nyrhinen (1992a, 1992b) also imply that the IS estimator of Example 6.66 is asymptotically optimal if we take $\epsilon = 1/R(0)$. □

Example 6.71 Let $Y_1, Y_2, \ldots$ be i.i.d. random variables with density π and finite moment generating function $M(\theta)$ in a neighborhood of 0. Let N be a Poisson random variable with mean λ, independent of the Y_j's. We want to estimate

$$\mu(\epsilon) = \mathbb{P}[Y_1 + \cdots + Y_N > 1/\epsilon]$$

by simulation. Asmussen and Rubinstein (2000) give a set of conditions on the density π (e.g., if π is bounded by a constant times a gamma density, or if π is log-concave, etc.) under which they show that the following IS estimator of $\mu(\epsilon)$ is asymptotically optimal. Replace $\pi(y)$ by $\pi_\theta(y) = \pi(y)e^{\theta y}/M(\theta)$ and λ by $\lambda_\theta = \lambda M(\theta)$. Take θ so that $E_\theta[Y_1 + \cdots + Y_N] = 1/\epsilon$, i.e., as the solution θ^* of $\lambda M'(\theta) = 1/\epsilon$. □

6.12.8 Links with large deviations theory

Reference: Bucklew (1990), Bucklew (2004).

6.12.9 IS for heavy-tailed distributions

♣ What do we do when the generating function is infinite? This is frequent: Pareto, Cauchy, Student, etc.

6.12.10 Adaptive IS

♣ The idea is to approximate the zero-variance IS by "learning" the value function $\gamma(y)$ in some way, or by learning some optimal parameter.

6.12.11 To probe further

Excellent surveys on IS are given by Glynn and Iglehart (1989), Glynn (1994), Heidelberger (1995), Juneja and Shahabuddin (2006), Asmussen and Glynn (2007).

6.13 Splitting

6.13.1 A rare-event setting

Consider a discrete-time Markov chain $\{X_j,\ j \geq 0\}$ with arbitrary state space $\mathcal{X}$. Let A and B be two disjoint subsets of $\mathcal{X}$. The chain starts in the initial state $X_0 = x_0 \in A$, eventually leaves the set A, and then may eventually reach B or return to A. For simplicity, suppose step 1 is when the chain exits A for the first time, i.e., $X_1 \notin A$. Define $\tau_A = \inf\{j > 0 : X_j \in A\}$, the first time when the chain returns to A after leaving it, and $\tau_B = \inf\{j > 0 : X_j \in B\}$, the first time when the chain reaches the set B. The goal is to estimate μ, the probability that the chain reaches B before it returns to A, i.e. $\mu = \mathbb{P}[\tau_B < \tau_A]$. This probability is assumed to be very small, e.g., 10^{-8} or even less.

This problem occurs in many practical situations; see, e.g., Nicola et al. (1991), Goyal et al. (1992), Heidelberger (1995). We saw one instance in Example 6.64, where we estimated the expected time until the number of customers in the queue reaches a given number K.

In that case, the sets A and B are the sets of states where $X_j = 0$ and where $X_j \geq K$, respectively. The random variables τ_A and τ_B correspond to T_0 and T_K in that example, so $\mathbb{E}[\tau_B]$ is the expected time until the first buffer overflow and μ is the probability that the buffer overflows before returning to empty, assuming that we start afresh with an empty system. If the system evolves indefinitely, $\mathbb{E}[\tau_B]$ also represents approximately the average time between buffer overflows, which is often the quantity of interest. We saw in Example 6.64 that

$$\mathbb{E}[\tau_B] = \mathbb{E}[\min(\tau_A, \tau_B)]/\mu. \tag{6.115}$$

In this expression, $\mathbb{E}[\min(\tau_A, \tau_B)]$ is easy to estimate, but μ is very inefficient to estimate by standard MC when it is small (Example 1.29).

As another example, we may want to estimate the expected time until failure for a complex multicomponent system whose initial state is "new". Components fail once in a while and are replaced by new ones after some random delay. When the set of working components satisfies certain conditions, the system is operational, otherwise it is in the *failed state*. Let $A = \{x_0\}$, the set that contains only the "new" state, and let B be the set of failed states. We may be interested in estimating $\mathbb{E}[\tau_B]$, the expected time until failure for a new system. Again, this quantity can be written as in (6.115), where the denominator μ is the difficult piece to estimate.

We already saw in Section 6.12 how to attack these problems via IS, by changing the probability laws so that the chain has a stronger attraction toward B. An important difficulty with that method is to find an appropriate change of the probability laws.

In the splitting method, the probability laws of the system remain unchanged, but an artificial drift toward B is created by terminating the trajectories that seem to get away from it and cloning (*splitting*) those that are moving toward B. To recover an unbiased estimator, we multiply the original estimator by an appropriate factor. This technique was proposed by Kahn and Harris (1951) and has been studied by several authors, including Bayes (1972), Villén-Altamirano and Villén-Altamirano (1991), Villén-Altamirano and Villén-Altamirano (1994), Garvels and Kroese (1998), Glasserman et al. (1998), Glasserman et al. (1999), Garvels (2000), and Demers, L'Ecuyer, and Tuffin (2005).

6.13.2 Multilevel splitting

A key ingredient for the splitting algorithm is the definition of a function $h : \mathcal{X} \to \mathbb{R}$ that assigns a real number to each state of the chain. This function h is called the *importance function* (Garvels 2000, Garvels, Kroese, and Van Ommeren 2002). Define the real-valued process $\{Z_j = h(X_j), j \geq 0\}$. We assume that $A = \{x \in \mathcal{X} : h(x) \leq 0\}$ and $B = \{x \in \mathcal{X} : h(x) \geq L\}$ for some constant $L > 0$. In the *multilevel splitting* method, we partition the interval $[0, L)$ into m subintervals with boundaries $0 = L_0 < L_1 < \cdots < L_m = L$. For $k = 1, \ldots, m$, define $T_k = \inf\{j > 0 : Z_j \geq L_k\}$, let $D_k = \{T_k < \tau_A\}$ denote the event that the process Z reaches level L_k before returning to level 0, and define the conditional probabilities $p_k = \mathbb{P}[D_k \mid D_{k-1}]$ for $k > 1$, and $p_1 = \mathbb{P}[D_1]$. Since $D_m \subset D_{m-1} \subset \cdots \subset D_1$, we see immediately that

$$\mu = \mathbb{P}[D_m] = \prod_{k=1}^{m} p_k.$$

The basic idea of splitting is to estimate each probability p_k "separately", by starting a large number of chains in states that are generated from the distribution of $X_{T_{k-1}}$ conditional on the event D_{k-1}. This conditional distribution is called the *entrance distribution at threshold* L_k and we shall denote it by G_k.

This is done in successive *stages*, as follows. In the first stage, we start N_0 independent chains from the initial state x_0 and simulate each of them until time $\min(\tau_A, T_1)$. Let R_1 be the number of those chains for which D_1 occurs. Then $\hat{p}_1 = R_1/N_0$ is an obvious unbiased estimator of p_1. The empirical distribution of these R_1 entrance states X_{T_1} can be viewed as an estimate of the conditional distribution G_1.

In the second stage, we start N_1 chains from these R_1 entrance states, by cloning (splitting) some chains if $N_1 > R_1$, and continue the simulation of these chains independently up to time $\min(\tau_A, T_2)$. Then $\hat{p}_2 = R_2/N_1$ is an unbiased estimator of p_2, where R_2 is the number of those chains for which D_2 occurs. This procedure is repeated at each stage. In stage k, we pick N_{k-1} states out of the R_{k-1} that are available (by cloning if necessary), simulate independently from these states up to time $\min(\tau_A, T_k)$, and estimate p_k by $\hat{p}_k = R_k/N_{k-1}$ where R_k is the number of chains for which D_k occurs.

Even tough the $\hat{p}_k$'s are not independent, it turns out that the product $\hat{p}_1 \cdots \hat{p}_m = (R_1/N_0)(R_2/N_1)\cdots(R_m/N_{m-1})$ is an unbiased estimator of μ (Garvels 2000).

♣ Should add proof.

There are many ways of doing the splitting (Garvels 2000). For example, one may clone each of the R_k chains that reached level k in c_k copies for a fixed integer c_k, in which case $N_k = c_k R_k$ is random. This is called *fixed splitting*. In contrast, in the *fixed effort* method, we fix a priori each value of N_k and make just the right amount of splitting to reach this target value. One way of doing this is by sampling the N_k starting states at random, with replacement, from the R_k available states. This is called *random assignment* and is equivalent to sampling from the empirical distribution of the states. In a *fixed assignment*, on the other hand, we would split each of the R_k states the same number of times (or approximately the same number of times, in the case where N_k is not a multiple of R_k). The fixed effort method tends to perform better, because it reduces the variance of the number of chains that are simulated at any given stage, and we prefer a fixed assignment strategy to a random assignment because it amounts to using stratified sampling over the empirical distribution, and thus typically reduces the variance.

Under a number of simplifying assumptions (e.g., that $\mathbb{P}[D_k \mid D_{k-1}, X_{T_{k-1}} = x]$ does not depend on x) and for the fixed splitting setting, it has been shown (Villén-Altamirano et al. 1994, Garvels and Kroese 1998) that the efficiency of the splitting method is maximized by selecting the thresholds so that $p_k \approx e^{-2} \approx 0.135$ and $\mathbb{E}[N_k] = N_0$ for each k. This gives $m \approx -(\ln \mu)/2$ stages. However, these simplifying assumption typically do not hold, so these results only give guidelines, and more importantly the p_k's are unknown in practice and selecting the appropriate threshold may be difficult. Moreover, the choice of the importance function h may have a large impact on the performance of the method and is not trivial (Garvels, Kroese, and Van Ommeren 2002). Nevertheless, the method has been shown empirically to be very effective in some situations that fit the above framework.

Table 6.13. Results with the splitting estimators for the tandem queue

N	m	h	$\hat{\mu}$	$\hat{\sigma}^2$	RE
4096	10	h_1	1.2E-9	4.7E-20	0.18
4096	10	h_2	1.2E-9	3.9E-20	0.16
4096	20	h_2	1.2E-9	2.1E-20	0.12
16384	20	h_2	1.2E-9	3.1E-21	0.046

6.13.3 Splitting: examples

Example 6.72 Consider an open Jackson queueing network with two FIFO queues in tandem. All external arrivals are at the first queue. After being served there, customers go to the second queue and then exit the system. Suppose the arrival rate is $\lambda = 1$ and the mean service time is ρ_i at queue i, for $i = 1, 2$. The events are the arrivals and service completions (at any queue) and the state $X_j = (X_{1,j}, X_{2,j})$ gives the number of customers in each of the two queues immediately after the jth event. The set A contains only the empty state $(0, 0)$ and $B = \{(x_1, x_2) : x_2 \geq \ell\}$ for some fixed threshold ℓ, i.e., B is the set of states for which the length of the second queue is at least ℓ.

Garvels (2000) and Garvels, Kroese, and Van Ommeren (2002) study the application of splitting to this model. A simple choice of importance function h here is $h_1(x_1, x_2) = x_2$, the number of customers in the second queue. A weakness of this definition is that it ignores the state of the first queue, neglecting its impact on the chances of reaching B. This neglect can have an important impact especially when the bottleneck is at the first queue. Garvels (2000) observes this and proposes alternative choices of h that depend on both x_1 and x_2.

Demers, L'Ecuyer, and Tuffin (2005) considered the importance function

$$h_2(x_1, x_2) = 2\ell - d(x_1, x_2) = x_2 + \min(0, x_2 + x_1 - \ell), \tag{6.116}$$

where $d(x_1, x_2)$ is the minimal number of steps required to reach B from the current state (x_1, x_2). (To reach B, we need at least $\ell - \min(0, x_2 + x_1 - \ell)$ arrivals at the first queue and $\ell - x_2$ transfers from the first queue to the second queue.)

For a numerical illustration, let $\rho_1 = 4$, $\rho_2 = 2$, and $L = \ell = 30$. Table 6.13 gives empirical results for the two importance functions h_1 and h_2, for $m = 10$ and 20 equidistant levels, with $N = 2^{12}$ and 2^{14} chains. In this table $\hat{\sigma}^2$ is an estimator of the variance of the estimator of μ based on the N copies of the chain, and RE $= \hat{\sigma}/\hat{\mu}$ is the estimated relative error of that estimator. These estimations are based on 30 independent replications. These results are taken from Demers, L'Ecuyer, and Tuffin (2005).

□

6.14 Exercises

♣ Some of these exercises will have to be revised or replaced.

6.1 For the example of Section 6.2.2, compare CRNs with the (a + c) synchronization, CRNs without synchronization, and IRNs, for the case where B_i is fixed at 1. Try the three values of δ.

6.2 For the example of Section 6.2.2, implement a combination of the CRN methodology with the CV A_i and with stratification based on B_i, with proportional allocation.

6.3 (Adapted from Bratley, Fox, and Schrage 1987.) In a $GI/GI/1$ queue, we want to estimate $\mu_2(t) - \mu_1(t)$, where $\mu_k(t) = \mathbb{E}[N_k(t)]$, and $N_k(t)$ is the number of customers who *complete their service* during the time interval $(0, t]$ if the service time distribution is G_k, $k = 1, 2$. In both cases, the system is initially empty and the interarrival time distribution is the same.

(a) Suppose we simulate the two systems with common random numbers, using inversion and good synchronization, and compute the estimator $N_2(t) - N_1(t)$. Is this a CRN-concordant scheme?

(b) Suppose now that we are interested, instead, in $q_2(t) - q_1(t)$, where $q_k(t) = \mathbb{E}[Q_k(t)]$, and $Q_k(t)$ is the number of customers in the system at time t. We simulate with CRNs as before and compute the estimator $Q_2(t) - Q_1(t)$. Why is it not a CRN-concordant scheme in the strict sense of the definition?

(c) Argue that $N_1(t) - N_2(t) = Q_2(t) - Q_1(t)$, and use it to prove that the variance of $Q_2(t) - Q_1(t)$ is necessarily reduced by taking common random numbers. (Hint: $N_k(t) = A(t) - Q_k(t)$, where $A(t)$ is the number of arrivals during $(0, t]$.)

(d) Let $\bar{W}_k(t)$ be the average sojourn time per customer for the $N_k(t)$ customers who have left by time t. We want to estimate $w_2(t) - w_1(t)$, for very large t, where $w_k(t) = \mathbb{E}[\bar{W}_k(t)]$. Construct a CRN-concordant estimator for this difference, after arguing (via Little's law; see Example 6.31) that $q_k(t) \approx w_k(t)\mathbb{E}[A(t)/t]$ for large t.

6.4 In Theorem 6.5(iii), explain why the continuity of the function f with respect to θ for every fixed $\boldsymbol{U}$ is sufficient to obtain the result (give a counterexample).

6.5 In Example 6.10, for the $M/M/1$ queue, we showed that the conditions of Corollary 6.6 are satisfied by bounding the derivative $f'(\theta, \boldsymbol{U})$. Verify the conditions of Theorem 6.5(iii) for this example, directly with $f(\theta, \boldsymbol{U})$, without using the derivative.

6.6 In the call center example of Section 6.2.2, suppose the service times are $\mathsf{Erlang}(k, \gamma)$ instead of exponential and that we generate each of them as follows: generate k independent exponentials with mean $1/\gamma$ by inversion and add them up. This requires k independant uniform random variable for each service time. The two systems that we compare have the same value of k and slightly different values of γ, and we use the same set of uniforms for both.

Prove that this method is equivalent to direct inversion of the Erlang (or gamma) distribution function for a single uniform random number U that we do not observe (i.e., this is

implicit inversion), and consequently it maximizes the correlation between the service times of corresponding calls in the two systems. What is this maximal correlation?

6.7 As mentioned at the end of Section 6.5.2, averaging an antithetic pair $(X^{(1)}, X^{(2)})$ (hoping that they are negatively correlated) is equivalent to using $X^{(1)}$ as the main estimator and $X^{(1)} - X^{(2)}$ as a CV with coefficient $\beta_1 = 1/2$. Show that if $X^{(1)} = h(\boldsymbol{U})$ and $X^{(2)} = h(1 - \boldsymbol{U})$ where $\boldsymbol{U}$ is a sequence of i.i.d. $U(0,1)$, as in Section 6.9, then the value of the CV coefficient that minimizes the variance is $\beta_1 = 1/2$, regardless of the sign of the correlation between $X^{(1)}$ and $X^{(2)}$.

6.8 Let $X_1, \ldots, X_q$ be independent unbiased estimators of μ, with $\mathrm{Var}[X_j] = \sigma_j^2$. Consider the linear combination $X = \sum_{j=1}^q \beta_j X_j$, where $\sum_{j=1}^q \beta_j = 1$. Show that $\mathbb{E}[X] = \mu$ and that $\mathrm{Var}[X]$ is minimized by taking $1/\beta_j = \sigma_j^2 \sum_{\ell=1}^q (1/\sigma_\ell^2)$.

6.9 You will experiment with the two control variates suggested in Example 6.17, for pricing an Asian option, and their combination with antithetic variates. The idea is to see how effective are these methods depending on the parameters of the option. Time is measured in years. Fix $\sigma = 0.2$, $r = 0.08$, $S(0) = 100$, $T = 120/365$. For the observation times, consider three cases: (i) $d = 10$ and $t_j = (110 + j)/365$ for $j = 1, \ldots, d$; (ii) $d = 10$ and $t_j = (12j)/365$ for $j = 1, \ldots, d$; (iii) $d = 120$ and $t_j = j/365$ for $j = 1, \ldots, d$. The strike price K can take the following four values: $K = 80$, 90, 100, and 110.

(a) For each of the 12 parameter combinations, perform $n = 10000$ simulation runs (1) without any VRT, (2) with only the payoff based on the geometric mean as a CV, (3) with only the sum $\sum_{j=1}^d S(t_j)$ as a CV, (4) with both CVs together, and (5) with both CVs together with antithetic variates. In each case, give an estimate of the variance reduction factor. Discuss your results extensively.

(b) Explore how the effectiveness of the CVs in (2) and (4) of part (a) above in terms if the values of σ^2 and $(t_d - t_1)$. To answer this, you can make additional experiments to see what happens when $\sigma^2, t_1, \ldots, t_d$ are changed. Can you explain the behavior that you observe (i.e., why it occurs).

6.10 You are asked to try the control variate suggested in Example 6.18. Estimate the variance reduction factor compared with naive Monte Carlo by performing $n = 10000$ simulation runs with the following parameters: $\sigma = 0.2$, $r = 0.08$, $S(0) = 100$, $T = 1$, $K = 100$, $d = 10$, and $t_j = j/10$ for $j = 1, \ldots, d$. For the barrier ℓ, try $\ell = 80$, 90, and 95. Discuss your results. Can you suggest other ways of reducing the variance for this example?

6.11 Show that in Example 6.20, if $\mathbb{E}[C_j] = \nu_j$ for each j and $C = \sum_{j=1}^\tau (C_j - \nu_j)$, where the ν_j's can be different, then it remains true that $\mathbb{E}[C] = 0$, despite the fact that τ is random.

6.12 (Boyle, Broadie, and Glasserman 1997b, page 1317.) We want to prove Eq. (6.42). Suppose $X_i = h(Z_i)$, $\mu = \mathbb{E}[h(Z_i)]$, $\tilde{X}_i = h(\tilde{Z}_i)$ where $\tilde{Z}_i$ is defined by (6.41), $\bar{\tilde{X}}_n = (1/n)\sum_{i=1}^n \tilde{X}_i$, and h is twice continuously differentiable at Z_i with probability 1.

(a) Prove that the $\tilde{Z}_i$ defined in (6.41) effectively match the first two moments, i.e, that their mean and variance are μ_z and σ_z^2, respectively.

(b) Show, via Taylor's expansion, that

$$h(\tilde{Z}_i) = h(Z_i) + h'(Z_i)(\tilde{Z}_i - Z_i) + O_p(1/n).$$

(c) Use (a) to show that

$$\begin{aligned} \bar{\tilde{X}}_n - \bar{X}_n \quad &= \quad -\left(\frac{1}{n}\sum_{i=1}^{n} Z_i h'(Z_i)\right)\left(\frac{1-\sigma_{\mathrm{z}}}{S_{\mathrm{z},n}}\right) \\ &\quad -\left(\frac{1}{n}\sum_{i=1}^{n} h'(Z_i)\right)\left(\frac{\bar{Z}_n\sigma_{\mathrm{z}}}{S_{\mathrm{z},n}} - \mu_{\mathrm{z}}\right) + O_p(1/n) \\ &\Rightarrow \quad -\beta_1 C^{(1)} - \beta_2 C^{(2)} \quad \text{when } n \to \infty. \end{aligned}$$

(d) Explain why $\mathbb{E}[C^{(1)}] \neq 0$ and $\mathbb{E}[C^{(2)}] \neq 0$ in general.

6.13 Prove Eq. (6.50).

6.14 Prove that if $\mathcal{G}_1 \subset \mathcal{G}_2$, then $\mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}_1]] \geq \mathbb{E}[\mathrm{Var}[X \mid \mathcal{G}_2]]$. Hint: Replace X by $X \mid \mathcal{G}_1$ and $\mathcal{G}$ by $\mathcal{G}_2$ in (6.48), and take the expectation.

6.15 Implement the CMC estimator described in Example 6.22. Combine it with stratification and a control variate, as in Section 6.2.1, and estimate by what factor it improves upon the estimator with the smallest variance obtained in Table 6.1. You can use a two-stage procedure, estimating the σ_s's in stage 1, and taking a final estimator that uses both stages. Make enough runs to make your variance comparison significant.

6.16 Give the explicit formula for X_{e} in Example 6.24. Do you believe that this CMC should improve the efficiency compared with X when the barrier ℓ tends to be hit early and frequently, or late and more rarely? Why? Explain. Design and make an experiment to verify it empirically. You can take for example the same model and parameters as in Example 1.11, with the same observation times (only for the barrier), and try several values of $\ell < 100$ to compare.

6.17 (Boyle, Broadie, and Glasserman 1997b.) In Example 6.24, we can define an alternative extended CMC estimator that conditions on *less information*, as follows. At each observation time $t_j < T_\ell$, compute the conditional expectation of $\mathbb{I}[T_\ell = t_{j+1}]X$ and add up these conditional expectations. Write an explicit expression for this estimator and show that it is unbiased. In their empirical experiments, Boyle, Broadie, and Glasserman (1997b) found this estimator to be less efficient than the previous one, because it is much more costly to compute while the additional variance reduction is only modest.

6.18 Give a concrete example where X_{ee} defined in (6.53) has more variance than X.

6.19 Suppose we want to estimate $\nu = g(\boldsymbol{\mu})$ where $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{X}] \in \mathbb{R}^d$ and $g : \mathbb{R}^d \to \mathbb{R}$ as in Section 5.4.1. Consider an estimator based on a conditional expectation given some $\mathcal{G}$. An obvious candidate is the estimator $\tilde{Y}_{\mathrm{e}} = \mathbb{E}[g(\boldsymbol{X}) \mid \mathcal{G}]$. However, $\tilde{Y}_{\mathrm{e}}$ can be too hard to

compute, so we may consider the alternative estimator $Y_e = g(\boldsymbol{X}_e) = g(\mathbb{E}[\boldsymbol{X} \mid \mathcal{G}])$ instead. Let n be the number of independent simulation runs, $\boldsymbol{X}_i$ and $\boldsymbol{X}_{e,i}$ the values of $\boldsymbol{X}$ and $\boldsymbol{X}_e$ on run i, and $\bar{\boldsymbol{X}}_n$ and $\bar{\boldsymbol{X}}_{e,n}$ the averages of the $\boldsymbol{X}_i$'s and $\boldsymbol{X}_{e,i}$'s, respectively.

Show that the estimator $g(\bar{\boldsymbol{X}}_{e,n})$ is (generally) *biased* but *consistent* for ν, and use the delta method to derive a central-limit theorem for it (as $n \to \infty$). Give conditions under which $\lim_{n\to\infty} \mathrm{MSE}[g(\bar{\boldsymbol{X}}_{e,n})]/\mathrm{MSE}[g(\bar{\boldsymbol{X}}_n)] < 1$ (See Fox and Glynn 1986.)

6.20 In Example 6.29, suppose that the service times are generated via $S_j = G_\theta^{-1}(U_j)$. Suppose that $G_\theta(x)$ is differentiable with respect to θ for x fixed and differentiable in x for θ fixed. This derivative with respect to x gives the service time density $g_\theta(x)$. We assume that this density is bounded by a constant C, uniformly in θ and x.

Under what additional assumptions can we prove that $\mathrm{Var}[Y^{(B)}/\delta]$ remains bounded when $\delta \to 0$ by showing that $Y^{(B)}$ satisfies the conditions of Corollary 6.6? Prove it.

Note: the conditions given in L'Ecuyer 1994a for this example are *stronger* than what we need here, because a stronger result than that of Corollary 6.6 is proved in that paper.

6.21 In Example 6.29, suppose that instead of conditioning on W_j for customer j, we condition on X_{j-1}, i.e., on the sojourn time of the previous customer. Derive an expression for this estimator (call it $Y^{(C)}$) in terms of F and G_θ. Simplify this expression (into a formula which can be programmed directly) for the case where F and G are exponential with means 1 and θ.

6.22 For stratified sampling with deterministic allocation (Section 6.8.1), show that the optimal allocation is $n_s^* = np_s\sigma_s/\bar{\sigma}$ to stratum s (neglecting the condition that n_s must be an integer).

6.23 Show that if $\hat{\sigma}_s^{(0)} = \sigma_s$, allocating $n_s = n_s^{(0)} + n_s^{(1)}$ to stratum s, where $n_s^{(1)}$ is defined in (6.77), minimizes the variance under the constraints that $n_s \geq n_s^{(0)}$ for each s and $\sum_{s=1}^k n_s = n$. Hint: Write the Kuhn-Tucker optimality conditions.

6.24 In poststratification (Section 6.8.3), when $N_s = 0$ for some s, one estimator uses the general sample mean as an estimator of μ_s within the empty strata, and the usual estimator otherwise. For example, if there are two strata and none is empty, use the stratified estimator (6.70), whereas if one is empty, use the general sample mean as an estimator. Assume that $N = n$ (a constant). Show that this estimator is biased, and give an expression for the bias. Show that the bias can be made arbitrary large (for fixed n) by changing the means within the strata, but that the bias vanishes exponentially fast if $n \to \infty$ while the other parameters are fixed.

6.25 Derive a formula for the variance of the estimator (6.78). Hint: Use the variance decomposition

$$\mathrm{Var}[\bar{X}_{\mathrm{sr},n}] = \mathbb{E}[\mathrm{Var}[\bar{X}_{\mathrm{sr},n} \mid N_1, \ldots, N_s]] + \mathrm{Var}[\mathbb{E}[\bar{X}_{\mathrm{sr},n} \mid N_1, \ldots, N_s]]$$

and the fact that (see Stephan 1945) $\mathbb{E}\left[\mathbb{I}[N_s > 0]/N_s\right] = \sum_{j=1}^\infty u_{s,j}$, where $u_{s,1} = [1 - q_s^n - \epsilon_s]/[(n+1)p_s]$, $u_{s,j} = [(j-1)u_{s,j-1} - \epsilon_s/j]/[(n+j)p_s]$ for $j > 1$, and $\epsilon_s = np_sq_s^n$. Prove that

$\mathrm{Var}[\bar{X}_{\mathrm{sr},n}] = \sum_{s=1}^{k} \psi_s$ where

$$\psi_s = \tilde{p}_s^2 \sigma_s^2 \sum_{j=1}^{\infty} u_{s,j} + \tilde{p}_s^2 \mu_s^2 q_s^n (1 - q_s^n).$$

Show that if $np_s \ll 1$, $\psi_s \approx (\sigma_s^2 + \mu_s^2) p_s / n$. Show that for the naive estimator $\bar{X}_n$, the contribution of stratum s to the variance is $(\sigma_s^2 + (\mu_s - \mu)^2) p_s / n \le \psi_s$.

6.26 (Yakowitz, Krimmel, and Szidarovszky 1978 ? (to be verified)) Consider the following post-stratification scheme for the one-dimensional integration problem of Example 6.34: Take an i.i.d. sample as usual form the $U(0,1)$ over [0,1]. Then, weight the observations $f(U_i)$ proportionally to the sum of distances to the two nearest neighbors of U_i. More specifically, the estimator of μ becomes

$$\bar{X}_{\mathrm{sw},n} = \frac{1}{2} \sum_{i=1}^{n} (U_{(i+1)} - U_{(i-1)}) f(U_{(i)}).$$

where $U_{(0)} = -U_{(1)}$ and $U_{(n+1)} = 2 - U_{(n)}$. What are the mean and variance of this estimator? How would you estimate its variance?

6.27 (a) Prove that for the stratified sampling scheme of Example 6.34 (which is equivalent to one-dimensional Latin hypercube sampling), $\mathbb{E}[\bar{X}_{\mathrm{s},n}] = \mu$ and $\mathbb{E}[S^2_{\mathrm{s},m}] = \mathrm{Var}[X_{\mathrm{s}}]$, where $S^2_{\mathrm{s},m}$ is the sample variance of the m replicates of X_{s}.

(b) Prove that this also holds for the t-dimensional Latin hypercube sampling.

(c) Explain why the variance estimator given by (6.73) is better (less noisy) than the estimator $S^2_{\mathrm{s},m}$ defined in (a) in the one-dimensional case.

6.28 (a) In Example 6.37, show that variance reduction is guaranteed if we simulate the Brownian motion $\{B(t),\, t \ge 0\}$ via the Brownian bridge approach and use antithetic variates $(Z_j, -Z_j)$ for the underlying standard normals, where the Z_j's can be generated by any method, not necessarily by inversion.

(b) Implement this AV scheme and estimate the efficiency improvement factor for an example with the following parameters: $\sigma = 0.3$, $r = 0.05$, $K = 55$, $S(0) = 50$, $T = 1$ year, $d = 64$, and $t_j = j/64$ for $j = 1, \ldots, d$.

6.29 (*Rotation sampling*; Fishman and Wang 1983.) This technique is related to antithetic variates, Latin hypercube sampling, and lattice rules. It generates a single random vector $\boldsymbol{U} = (U_1, \ldots, U_s)$, uniformly over $[0,1)^s$, adds this vector modulo 1 (coordinatewise) to each of the points $\boldsymbol{u}_i = (i/k, \ldots, i/k)$, $i = 0, \ldots, k-1$, for some positive integer k, and estimates $\mu = \int_{[0,1)^s} f(\boldsymbol{u}) d\boldsymbol{u}$ by

$$X_{\mathrm{rs}} = \frac{1}{k} \sum_{i=0}^{k-1} f((\boldsymbol{u}_i + \boldsymbol{U}) \bmod 1).$$

This is much less expensive to apply than LHS, because only s uniforms need to be generated, whereas LHS requires ks uniforms and s random permutations of the first k integers. As for

LHS, we can replicate the scheme m times, independently, and use the empirical mean and variance of the m copies of X_{rs} to compute a confidence interval on μ.

(a) Show that this method is a special case of a randomly-shifted lattice rule. Give a basis for the corresponding s-dimensional *integration lattice*. Does it look like a good integration lattice? Why?

(b) Give a *simple* concrete example of a function for which this method *increases the variance* by the factor k compared with standard Monte Carlo (for the same total number of function evaluations).

6.30 (*Weighted average over a randomly-shifted lattice rule.*) This generalizes Theorem 5 of Glynn and Szechtman (2002). See also Ben-Ameur, L'Ecuyer, and Lemieux (2004). Let $\{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}\}$ be the point set of an integration lattice. To estimate $\mu = \int_{[0,1)^s} f(\boldsymbol{u}) d\boldsymbol{u}$, consider the *weighted randomly-shifted lattice rule*

$$X_{\text{wlr}} = \frac{1}{n} \sum_{i=0}^{n-1} w_i f((\boldsymbol{u}_i + \boldsymbol{U}) \bmod 1),$$

where $w_i \in \mathbb{R}$ for each i and $w_0 + \cdots + w_{n-1} = 1$.

(a) Show that $\text{Var}[X_{\text{wlr}}]$ is minimized by taking $w_i = 1/n$ for each i.

(b) Give an counterexample showing that the result in (a) does not hold in general for an arbitrary point set (which is not a lattice).

(c) How would you optimize the weights in the general case? Hint: X_{wlr} can be viewed as a CV estimator, where the differences $C^{(i)} = f((\boldsymbol{u}_i + \boldsymbol{U}) \bmod 1) - f((\boldsymbol{u}_0 + \boldsymbol{U}) \bmod 1)$ act as control variates for the estimator $f((\boldsymbol{u}_0 + \boldsymbol{U}) \bmod 1)$.

6.31 (Adapted from Rubinstein and Shapiro 1993, Exercice 2.8.5.)

Supposons que durant la simulation, on génère les v.a. $Y_1, \ldots, Y_t$, où t est déterministe. Les Y_i sont indépendantes et Y_i a une densité de la forme (famille exponentielle):

$$f_i(\theta_i, y) = a_i(\theta_i) u_i(y) e^{b_i(\theta_i) v_i(y)} \tag{6.117}$$

où chaque θ_i est un paramètre continu, a_i et u_i sont des fonctions non-négatives, et b_i et v_i sont des fonctions à valeur réelle. La densité conjointe du vecteur de v.a. générées est donc le produit de ces densités. On veut estimer $\mu = \mu(\theta) = E_\theta[X]$, où E_θ dénote l'espérance mathématique lorsque le vecteur des paramètres est $\theta = (\theta_1, \theta_2, \ldots, \theta_t)$, $X = h(\omega) = h(Y_1, \ldots, Y_t, Z_1, Z_2, \ldots)$ et les Z_j (s'il y en a) sont d'autres v.a. indépendantes des Y_i, pour lesquelles nous n'allons pas changer les lois de probabilité (donc seulement les densités f_i vont apparaitre dans le rapport de vraisemblance). On veut effectuer nos simulations avec les valeurs des paramètres fixés à $(\theta_0 = (\theta_{1,0}, \theta_{2,0}, \ldots, \theta_{t,0})$ pour estimer $\mu(\theta)$, supposé fini. Soient $L(\theta, \theta_0, \omega)$ le rapport de vraisemblance associé à ce changement de mesure et $X_{\text{is}} = XL(\theta, \theta_0, \omega)$ l'estimateur IS.

(a) Écrivez l'expression pour $L(\theta, \theta_0, \omega)$.

(b) Montrez que pour tout entier $k \geq 2$,

$$E_{\theta_0}[X_{\text{is}}^k] \;=\; \prod_{i=1}^{t} \frac{a_i^k(\theta_i)}{a_i^{k-1}(\theta_{i,0}) \tilde{a}_{i,k}} \tilde{\mathbb{E}}[X^k],$$

où $\tilde{E}$ est l'espérance qui correspond au cas où chaque v.a. Y_i est générée selon la densité

$$\tilde{f}_i(y) = \tilde{a}_{i,k} u_i(y) e^{\tilde{b}_{i,k} v_i(y)},$$

et où $\tilde{b}_{i,k}$ et $\tilde{a}_{i,k}$ sont définis par:

$$\begin{aligned} \tilde{b}_{i,k} &= k b_i(\theta_i) - (k-1) b_i(\theta_{i,0}); \\ \tilde{a}_{i,k} &= \left(\int_0^\infty u_i(y) e^{\tilde{b}_{i,k} v_i(y)} dy \right)^{-1} \end{aligned}$$

pour chaque i.

(c) Donnez une condition nécessaire et suffisante pour que ce k-ième moment soit fini, en termes des $\tilde{b}_{i,k}$ et des fonctions u_i et v_i. (Supposez que $v_i(y)$ est un polynôme en y.) Pour simplifier, vous pouvez supposer que $\tilde{\mathbb{E}}[X^k]$ est fini lorsque chaque $\tilde{f}_i$ est une densité dont l'intégrale vaut 1.

(d) Supposons maintenant que comme dans notre example de file $M/M/1$, les v.a. Y_i sont toutes exponentielles de moyenne θ, de sorte que:

$$f_i(\theta, y) = \theta^{-1} e^{-y/\theta}, \qquad \text{pour } y > 0, \tag{6.118}$$

pour chaque i. Montrez dans ce cas que le k-ième moment de X_{is} est fini si et seulement si $\theta < k\theta_0/(k-1)$. En particulier, la variance est finie si et seulement si $\theta < 2\theta_0$. (Remarque: la variance de la variance échantillonnale de X_{is} est finie si et seulement si le quatrième moment de X_{is} est fini.)

6.32 In Example 6.60, suppose that we put $q_{1,0} = 0$ and $q_{1,2} = 1$ instead of $q_{1,0} = 1 - q$ and $q_{1,2} = q$. We still have $q_{i,i+1} = q$ for $i \geq 2$. How does this change the variance and the efficiency of IS the estimator? Show that for $K = 2$, this reduces the variance to 0.

6.33 Implement the IS scheme described in Example 6.60, with $K = 20$ and $p = 1/3$. Try several values of q in a close neighborhood of $q = 2/3$ and compare the efficiencies. Try to find the optimal q, i.e., the value of q that maximizes the efficiency. Try also the modification suggested in Exercise 6.32 and compare.

6.34 In Example 6.60, suppose that $p_{i,i+1} = p_i \leq 1/2$ and $p_{i,i-1} = 1 - p_i > 0$ for $1 \leq i \leq K - 1$, but that the p_i are no longer all the same. We want to design an IS scheme that replaces p_i by some q_i for each i, again to estimate $\mathbb{P}[X_T = K]$, the probability of reaching K before coming back to 0.

(a) Write the corresponding likelihood ratio on the event $\{X_T = K\}$.

(b) How would you select the q_i's in order to minimize an upper bound on this likelihood ratio? With your selection, the variance is reduced by *at least* what factor?

6.35 Suppose $Y_1, \ldots, Y_d$ are i.i.d. random variables with density π, and we replace this density by g to estimate the expectation of some random variable $X = h(Y_1, \ldots, Y_d)$. Suppose also that $E_g[|\ln(\pi(Y_1)/g(Y_1))|] < \infty$ (a mild condition).

(a) Use the strong law of large numbers to show that

$$\lim_{d\to\infty} \frac{\ln L(y_1,\ldots,y_d)}{d} = c$$

for some constant c. Then show that $\ln E_g[\pi(Y_1)/g(Y_1)] = 0$ use it together with Jensen's inequality to show that $c \le 0$.

(b) Show that c can be zero only if $E_g[\mathbb{I}[\pi(Y_1) = g(Y_1)]] = 1$, which would mean that π and g are essentially the same.

(c) Explain why this implies that when $d \to \infty$ under g, $L(Y_1,\ldots,Y_d) \to 0$ with probability 1 and $\mathrm{Var}[L(Y_1,\ldots,Y_d)] \to \infty$, even though $\mathbb{E}[L(Y_1,\ldots,Y_d)] = 1$ for all d.

6.36 Generalize Example 6.68 to the setting where

$$S(t_j) = S(0)\exp(Y_1 + \cdots + Y_j)$$

and the Y_j's are i.i.d. with an arbitrary density π.

(a) Explain how to apply exponential twisting to the Y_j's, with parameter $\theta = \theta_1$ until time T_ℓ, and $\theta = \theta_2$ thereafter. Propose a method for selecting θ_1 and θ_2. Note that the method proposed in Example 6.68 is a *heuristic*; there is no guarantee that the likelihood ratio is less than 1 when the payoff is nonzero, and no guarantee that the barrier is reached with probability 1. The same can be true for the method you propose.

(b) In the case where π is the normal density, compare the change of measure given by your method with that suggested in Example 6.68 for the geometric Brownian motion. More specifically, prove that the exponentially twisted normal density is still a normal density. For what values of θ_1 and θ_2 is the exponential twisting *exactly equivalent* to the method proposed in Example 6.68?

7. Optimization

References

Acworth, P., M. Broadie, and P. Glasserman. 1998. A comparison of some Monte Carlo and quasi-Monte Carlo techniques for option pricing. In *Monte Carlo and Quasi-Monte Carlo Methods 1996*, ed. P. Hellekalek, G. Larcher, H. Niederreiter, and P. Zinterhof, volume 127 of *Lecture Notes in Statistics*, 1–18. New York: Springer-Verlag.

Afflerbach, L. and H. Grothe. 1985. Calculation of Minkowski-reduced lattice bases. *Computing*, 35:269–276.

Ahmed, S. and A. Shapiro. 2008. Solving chance-constrained stochastic programs via sampling and integer programming. In *Tutorials in Operations Research*, 261–269. INFORMS.

Ahrens, J. H. and K. D. Kohrt. 1981. Computer methods for efficient sampling from largely arbitrary statistical distributions. *Computing*, 26:19–31.

Aiello, W., S. Rajagopalan, and R. Venkatesan. 1998. Design of practical and provably good random number generators. *Journal of Algorithms*, 29(2):358–389.

Akşin, O. Z., M. Armony, and V. Mehrotra. 2007. The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management*, 16(6):665–688.

Alanen, J. D. and D. E. Knuth. 1964. Tables of finite fields. *SANKHYĀ: The Indian Journal of Statistics, Series A*, 26:305–328.

Alexopoulos, C. 1998. Output data analysis. In *Handbook of Simulation*, ed. J. Banks, 225–272. Wiley. chapter 7.

Alexopoulos, C and S.-H. Kim. 2002. Output data analysis for simulations. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 85–96. IEEE Press.

Andradóttir, S. 2006. An overview of simulation optimization by random search. In *Simulation*, ed. S. G. Henderson and B. L. Nelson, Handbooks in Operations Research and Management Science, 617–632. Amsterdam, The Netherlands: Elsevier. Chapter 20.

Arkin, B. L. and L. M. Leemis. 2000. Nonparametric estimation of the cumulative intensity function for a nonhomogeneous Poisson process from overlapping realizations. *Management Science*, 46(7):989–998.

Arnold, B. C. 1983. *Pareto Distributions*. Fairland, Maryland: International Cooperative Publishing House.

Artzner, P, F. Delbaen, J.-M. Eber, and D. Heath. 1999. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228.

Asmussen, S. 1985. Conjugate processes and the simulation of ruin problems. *Stochastic Processes and their Applications*, 20:213–229.

Asmussen, S. 1987. *Applied Probability and Queues*. Wiley.

Asmussen, S. 1988. Ruin probabilities expressed in terms of storage process. *Advances in Applied Probability*, 20:913–916.

Asmussen, S. 2000. *Ruin Probabilities*. Singapore: World Scientific Publishing.

Asmussen, S. 2002. Large deviations in rare events simulation: Examples, counterexamples, and alternatives. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, ed. K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 1–9, Berlin. Springer-Verlag.

Asmussen, S. and P. W. Glynn. 2007. *Stochastic Simulation*. New York: Springer-Verlag.

Asmussen, S. and R. Rubinstein. 1995. Complexity properties of steady-state rare events simulation in queueing models. In *Advances in Queueing: Theory, Methods, and Open Problems*, ed. J. Dshalalow, 429–462. CRC Press.

Asmussen, S and R. Y. Rubinstein. 2000. Sensitivity analysis of insurance risk models via simulation. *Management Science*. To appear.

Athreya, K. B. and P. Ney. 1978. A new approach to the limit theory of recurrent markov chains. *Trans. Amer. Math. Soc.*, 245:493–501.

Atlason, J., M. A. Epelman, and S. G. Henderson. 2004. Call center staffing with simulation and cutting plane methods. *Annals of Operations Research*, 127:333–358.

Avramidis, A. N., W. Chan, M. Gendreau, P. L'Ecuyer, and O. Pisacane. 2010. Optimizing daily agent scheduling in a multiskill call centers. *European Journal of Operational Research*, 200(3):822–832.

Avramidis, A. N., N. Channouf, and P. L'Ecuyer. 2009. Efficient correlation matching for fitting discrete multivariate distributions with arbitrary marginals and normal copula dependence. *INFORMS Journal of Computing*, 21:88–106.

Avramidis, A. N., A. Deslauriers, and P. L'Ecuyer. 2004. Modeling daily arrivals to a telephone call center. *Management Science*, 50(7):896–908.

Avramidis, A. N. and P. L'Ecuyer. 2005. Modeling and simulation of call centers. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 144–152. IEEE Press.

Avramidis, A. N. and P. L'Ecuyer. 2006. Efficient Monte Carlo and quasi-Monte Carlo option pricing under the variance-gamma model. *Management Science*, 52(12):1930–1944.

Avramidis, A. N., P. L'Ecuyer, and P.-A. Tremblay. 2003. Efficient simulation of gamma and variance-gamma processes. In *Proceedings of the 2003 Winter Simulation Conference*, 319–326, Piscataway, New Jersey. IEEE Press.

Avramidis, A. N. and J. R. Wilson. 1993. A splitting scheme for control variates. *Operations Research Letters*, 14:187–198.

Avramidis, A. N. and J. R. Wilson. 1994. A flexible method for estimating inverse distribution functions in simulation experiments. *ORSA Journal of Computing*, 6:342–355.

Avramidis, A. N. and J. R. Wilson. 1996. Integrated variance reduction strategies for simulation. *Operations Research*, 44:327–346.

Avramidis, A. N. and J. R. Wilson. 1998. Correlation-induction techniques for estimating quantiles in simulation experiments. *Operations Research*, 46(4):574–591.

Balci, O. 1998. Verification, validation, and testing. In *Handbook of Simulation*, ed. J. Banks, 335–393. Wiley. chapter 10.

Ball, M. O, C. J. Colbourn, and J. S. Provan. 1995. Network reliability. In *Handbook of Operations Research: Network Models*, 673–762. Amsterdam: Elsevier, North-Holland.

Banks, J. 1998. *Handbook of Simulation*. New York, NY: Wiley.

Barndorff-Nielsen, O. E. 1997. Normal inverse Gaussian distributions and stochastic volatility modelling. *Scandinavian Journal of Statistic*, 24(1):1–13.

Barndorff-Nielsen, O. E. 1998. Processes of normal inverse gaussian type. *Finance and Stochastics*, 2:41–68.

Barndorff-Nielsen, O. E, T. Mikosch, and S. I. Resnick. 2013. *Lévy Processes: Theory and Applications*. Birkhäuser.

Barraquand, J. 1995. Numerical valuation of high-dimensional multivariate European securities. *Management Science*, 41:1882–1891.

Barton, R. R and M. Meckesheimer. 2006. Metamodel-based simulation optimization. In *Simulation*, ed. S. G. Henderson and B. L. Nelson, Handbooks in Operations Research and Management Science, 535–574. Amsterdam, The Netherlands: Elsevier. Chapter 18.

Bastin, F., C. Cirillo, and Ph. L. Toint. 2006. Convergence theory for nonconvex stochastic programming with an application to mixed logit. *Mathematical Programming, Series B*, 108(2–3):207–234.

Bauer Jr., K. W. 1987. Control-variate selection for multiresponse simulation. PhD thesis, School of Industrial Engineering, Purdue University, West Lafayette, Indiana.

Bauer Jr., K. W and J. R. Wilson. 1992. Control-variate selection criteria. *Naval Research Logistics*, 39:307–321.

Bayes, A. J. 1972. A minimum variance technique for simulation models. *Journal of the ACM*, 19:734–741.

Bechhofer, R, T. Santner, and D. Goldsman. 1995. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. New York, NY: Wiley.

Ben Abdellah, A., P. L'Ecuyer, A. Owen, and F. Puchhammer. 2021. Density estimation by randomized quasi-Monte Carlo. *SIAM Journal on Uncertainty Quantification*, 9(1):280–301.

Ben Ameur, H., M. Breton, and P. L'Ecuyer. 1999. Partial hedging for options based on extreme values and passage times. Technical Report G–99–15, GERAD, Ecole des Hautes Études Commerciales, Montréal.

Ben Ameur, H., M. Breton, and P. L'Ecuyer. 2002. A dynamic programming procedure for pricing american-style asian options. *Management Science*, 48:625–643.

Ben-Ameur, H., P. L'Ecuyer, and C. Lemieux. 2004. Combination of general antithetic transformations and control variables. *Mathematics of Operations Research*, 29(4):946–960.

Benth, F., M. Groth, and P. Kettler. 2006. A quasi-Monte Carlo algorithm for the normal inverse Gaussian distribution and valuation of financial derivatives. *International Journal of Theoretical and Applied Finance*, 9(6):843–867.

Berlinet, A. and L. Devroye. 1994. A comparison of kernel density estimates. *Publications de l'Institut de Statistique de l'Université de Paris*, 38(3):3–59.

Bertoin, J. 1996. *Lévy Processes*. Cambridge: Cambridge University Press.

Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control, Volumes I and II*. Belmont, Mass.: Athena Scientific.

Beyer, W. A, R. B. Roof, and D. Williamson. 1971. The lattice structure of multiplicative congruential pseudo-random vectors. *Mathematics of Computation*, 25(114):345–363.

Bézier, P. 1970. *Emploi des Machines à Commande Numérique*. Paris: Masson et cie.

Biller, B and B. L. Nelson. 2002. Parameter estimation for the ARTA processes. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 255–262. IEEE Press.

Billingsley, P. 1968. *Convergence of Probability Measures*. New York, NY: John Wiley.

Birge, J. R and F. Louveaux. 2011. *Introduction to Stochastic Programming*. second ed. New York, NY, USA: Springer-Verlag.

Birtwistle, G. M., O. Dahl, B. Myhrhaug, and K. Nygaard. 1979. *Simula begin*. Lund:Studentlitteratur.

Black, F. and M. Scholes. 1973. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–654.

Blair, J. M., C. A. Edwards, and J. H. Johnson. 1976. Rational Chebyshev approximations for the inverse of the error function. *Mathematics of Computation*, 30:827–830.

Blanchet, J. and D. Rudoy. 2009. Rare event simulation and counting problems. In *Rare Event Simulation Using Monte Carlo Methods*, ed. G. Rubino and B. Tuffin, 171–192. Wiley. Chapter 8.

Blomqvist, N. 1967. The covariance function of the $M/G/1$ queueing system. *Skandinavisk Aktuarietidskrift*, 50:157–174.

Blum, L, M. Blum, and M. Schub. 1986. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383.

Blum, M. 1986. Independent unbiased coin flips from a correlated biased source—a finite state Markov chain. *Combinatorica*, 6(2):97–108.

Blum, P, A. Dias, and P. Embrechts. 2002. The ART of dependence modelling: the latest advances in correlation analysis. In *Alternative Risk Strategies*, ed. M. Lane, 339–356. London: Risk Books.

Borak, S, W. Härdle, and R. Weron. 2005. Stable distributions. In *Statistical Tools for Finance and Insurance*, ed. P. Čižek, W. Härdle, and R. Weron, 21–44. Springer-Verlag.

Botev, Z. I., P. L'Ecuyer, G. Rubino, R. Simard, and B. Tuffin. 2013. Static network reliability estimation via generalized splitting. *INFORMS Journal on Computing*, 25(1):56–71.

Botev, Z. I., P. L'Ecuyer, and B. Tuffin. 2016. Static network reliability estimation under the Marshall-Olkin copula. *ACM Transactions on Modeling and Computer Simulation*, 26(2):Article 14, 28 pages.

Box, G. E. P., G. M. Jenkins, and G. C. Reinsel. 1994. *Time Series Analysis, Forecasting, and Control*. third ed. Englewood Cliffs, New Jersey: Prentice-Hall.

Box, G. E. P. and M. E. Muller. 1958. A note on the generation of random normal deviates. *Annals of Mathematical Statistics*, 29:610–611.

Boyle, P. 1977. Options: a Monte Carlo approach. *Journal of Financial Economics*, 4:323–338.

Boyle, P, M. Broadie, and P. Glasserman. 1997a. Monte Carlo methods for security pricing. *Journal of Economic Dynamics and Control*, 21(8-9):1267–1321.

Boyle, P., M. Broadie, and P. Glasserman. 1997b. Monte Carlo methods for security pricing. *Journal of Economic Dynamics and Control*, 21:1267–1321.

Boyle, P., Y. Lai, and K. S. Tan. 2005. Pricing options using lattice rules. *North American Actuarial Journal*, 9(3):50–76.

Brassard, G. and P. Bratley. 1988. *Algorithmics, Theory and Practice*. Prentice-Hall.

Bratley, P., B. L. Fox, and L. E. Schrage. 1983. *A Guide to Simulation*. First ed. New York, NY: Springer-Verlag.

Bratley, P., B. L. Fox, and L. E. Schrage. 1987. *A Guide to Simulation*. Second ed. New York, NY: Springer-Verlag.

Brémaud, P., R. Kannurpatti, and R. Mazumdar. 1992. Event and time averages: A review. *Advances in Applied Probability*, 24:377–411.

Brent, R. P. 1971. An algorithm with guaranteed convergence for finding a zero of a function. *Computer Journal*, 14:422–425.

Brent, R. P. 1973. *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ: Prentice-Hall.

Brent, R. P. 2004. Note on Marsaglia's xorshift random number generators. *Journal of Statistical Software*, 11(5):1–4.

Brown, M and H. Solomon. 1979. On combining pseudorandom number generators. *Annals of Statistics*, 1:691–695.

Bucklew, J. 1990. *Large Deviation Techniques in Decision, Simulation and Estimation*. New York: John Wiley and Sons.

Bucklew, J. A. 2004. *Introduction to Rare Event Simulation*. New York: Springer-Verlag.

Buist, E and P. L'Ecuyer. 2005. A Java library for simulating contact centers. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 556–565. IEEE Press.

Buzacott, J. A. and J. G. Shantikumar. 1993. *Stochastic Models of Manufacturing Systems*. Upper Saddle River, NJ: Prentice-Hall.

Caflisch, R. E., W. Morokoff, and A. Owen. 1997. Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. *J. of Computational Finance*, 1(1):27–46.

Calafiore, G. and M. Campi. 2005. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102:25–46.

Cambanis, S., S. Huang, and G. Simons. 1981. On the theory of elliptically contoured distributions. *Journal of Multivariate Analysis*, 11:368–385.

Cario, M. C. and B. L. Nelson. 1996. Autoregressive to anything: Time series input processes for simulation. *Operations Research Letters*, 19:51–58.

Cario, M. C. and B. L. Nelson. 1997. Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. working paper.

Cario, M. C. and B. L. Nelson. 1998. Numerical methods for fitting and simulating autoregressive-to-anything processes. *INFORMS Journal on Computing*, 10:72–81.

Carr, P., H. Geman, D. Madan, and M. Yor. 2002. The fine structure of asset returns: An empirical investigation. *The Journal of Business*, 75:305–332.

Carson, J. S. and A. M. Law. 1980. Conservation equations and variance reduction in queueing simulations. *Operations Research*, 28:535–546.

Cash, C. R., B. L. Nelson, J. M. Long, D. G. Dippold, and W. P. Pollard. 1992. Evaluation of tests for initial-condition bias. In *Proceedings of the 1992 Winter Simulation Conference*, 577–585. IEEE Press.

Ceẓik, M. T. and P. L'Ecuyer. 2006. Staffing multiskill call centers via linear programming and simulation. *Management Science*. To appear.

Ceẓik, M. T. and P. L'Ecuyer. 2008. Staffing multiskill call centers via linear programming and simulation. *Management Science*, 54(2):310–323.

Chan, T. F., G. Golub, and R. J. LeVeque. 1983. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37:242–247.

Chan, W., G. Koole, and P. L'Ecuyer. 2014. Dynamic call center routing policies using call waiting and agent idle times. *Manufacturing & Service Operations Management*, 16(4):544–560.

Chang, C. S., P. Heidelberger, S. Juneja, and P. Shahabuddin. 1994. Effective bandwidth and fast simulation of ATM intree networks. *Performance Evaluation*, 20:45–65.

Chang, H. S., M. C. Fu, J. Hu, and S. I. Marcus. 2007. *Simulation-Based Algorithms for Markov Decision Processes*. London: Springer-Verlag.

Channouf, N. and P. L'Ecuyer. 2012. A normal copula model for the arrival process in a call center. *International Transactions in Operational Research*, 19(6):771–787.

Chapuis, K., P. Taillandier, and A. Drogoul. 2022. Generation of synthetic populations in social simulations: A review of methods and practices. *Journal of Artificial Societies and Social Simulation*, 25(2):Article 6.

Charnes, J. M. 1991. Multivariate simulation output analysis. In *Proceedings of the 1991 Winter Simulation Conference*, ed. B. L. Nelson, W. D. Kelton, and G. M. Clark, 187–193. IEEE Press.

Charnes, J. M. 1995. Analyzing multivariate output. In *Proceedings of the 1995 Winter Simulation Conference*, 201–208. IEEE Press.

Chen, H and C. Jeng. 1998. RA algorithms for generation of multivariate random vectors. working paper.

Chen, H. C. and Y. Asau. 1974. On generating random variates from an empirical distribution. *AIEE Transactions*, 6:163–166.

Cheng, R. C. H. 1982. The use of antithetic variates in computer simulations. *Journal of the Operational Research Society*, 33:229–237.

Cheng, R. C. H. 1984. Antithetic variate methods for simulation of processes with peaks and troughs. *European Journal of Operational Research*, 15:227–236.

Cheng, R. C. H. 1998. Random variate generation. In *Handbook of Simulation*, ed. J. Banks, 139–172. Wiley. chapter 5.

Chernick, M. R. 1999. *Bottstrap Methods: A Practitioner's Guide*. New York, NY: John Wiley.

Chhikara, R. S and J. L. Folks. 1989. *The Inverse Gaussian Distribution: Theory, Methodology and Applications*. New York, NY: Marcel Dekker.

Chib, S. 2004. Markov chain Monte Carlo technology. In *Handbook of Computational Statistics*, ed. J. E. Gentle, W. Haerdle, and Y. Mori, 71–102. Berlin: Springer-Verlag. Chapter II.3.

Chien, C.-H. 1989. Small sample theory for stead-state confidence intervals. Technical Report 37, Department of Operations Research, Standord University, Stanford, CA.

Chien, C.-H, D. Goldsman, and B. Melamed. 1997. Large-sample results for batch means. *Management Science*, 43:1288–1295.

Choquet, D., P. L'Ecuyer, and C. Léger. 1999. Bootstrap confidence intervals for ratios of expectations. *ACM Transactions on Modeling and Computer Simulation*, 9(4):326–348.

Chor, B. and O. Goldreich. 1988. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computation*, 17(2):230–261.

Chow, Y. S. and H. Robbins. 1965. On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *Ann. Math. Stat.*, 36:457–462.

Chung, K. L. 1954. On a stochastic approximation method. *Annals of Mathematical Statistics*, 25:463–483.

Çinlar, E. 1975. *Introduction to Stochastic Processes*. Englewood Cliffs, N. J.: Prentice-Hall.

Clewlow, L and A. Carverhill. 1994. On the simulation of contingent claims. *Journal of Derivatives*, 999:66–74.

Cochran, W. G. 1977. *Sampling Techniques*. Second ed. New York, NY: John Wiley and Sons.

Collings, B. J. 1987. Compound random number generators. *Journal of the American Statistical Association*, 82(398):525–527.

Compagner, A. 1991. The hierarchy of correlations in random binary sequences. *Journal of Statistical Physics*, 63:883–896.

Conover, W. J. 1980. *Practical Nonparametric Statistics*. second ed. New York, NY: John Wiley.

Cont, R and P. Tankov. 2004. *Financial Modeling with Jump Processes*. Chapman and Hall/CRC.

Conway, J. H. and N. J. A. Sloane. 1999. *Sphere Packings, Lattices and Groups*. 3rd ed. Grundlehren der Mathematischen Wissenschaften 290, New York: Springer-Verlag.

Couture, R. and P. L'Ecuyer. 1994. On the lattice structure of certain linear congruential sequences related to AWC/SWB generators. *Mathematics of Computation*, 62(206):798–808.

Couture, R. and P. L'Ecuyer. 1996. Orbits and lattices for linear random number generators with composite moduli. *Mathematics of Computation*, 65(213):189–201.

Couture, R. and P. L'Ecuyer. 1997. Distribution properties of multiply-with-carry random number generators. *Mathematics of Computation*, 66(218):591–607.

Couture, R. and P. L'Ecuyer. 2000. Lattice computations for random numbers. *Mathematics of Computation*, 69(230):757–765.

Couture, R., P. L'Ecuyer, and S. Tezuka. 1993. On the distribution of k-dimensional vectors for simple and combined Tausworthe sequences. *Mathematics of Computation*, 60(202):749–761, S11–S16.

Coveyou, R. R. and R. D. MacPherson. 1967. Fourier analysis of uniform random number generators. *Journal of the ACM*, 14:100–119.

Cox, J. C., J. E. Ingersoll, and S. A. Ross. 1985. A theory of the term structure of interest rates. *Econometrica*, 53:385–407.

Cranley, R. and T. N. L. Patterson. 1976. Randomization of number theoretic methods for multiple integration. *SIAM Journal on Numerical Analysis*, 13(6):904–914.

Cristea, L. L., J. Dick, G. Leobacher, and F. Pillichshammer. 2007. The tent transformation can improve the convergence rate of quasi-Monte Carlo algorithms using digital nets. *Numerische Mathematik*, 105:413–455.

Daley, D. J. 1968. The serial coorrelation coefficients of waiting times in a stationary single-server queue. *Journal of the Australian Mathematical Society*, 8:683–699.

Damerdji, H. 1994. Strong consistency of the variance estimator in steady-state simulation output analysis. *Mathematics of Operations Research*, 19:494–512.

Darling, D. A. 1960. On the theorems of Kolmogorov-Smirnov. *Theory of Probability and Its Applications*, V(4):356–360.

David, H. A. 1981. *Order Statistics*. Second ed. Wiley.

Davis, P and P. Rabinowitz. 1984. *Methods of Numerical Integration*. second ed. New York: Academic Press.

De Matteis, A. and S. Pagnutti. 1988. Parallelization of random number generators and long-range correlations. *Numerische Mathematik*, 53:595–608.

de Mello, T. H. 2003. Variable-sample methods for stochastic optimization. *ACM Transactions on Modeling and Computer Simulation*, 13(2):108–133.

DeBrota, D. J, R. S. Dittus, S. D. Roberts, and J. R. Wilson. 1989a. Visual interactive fitting of of bounded Johnson distributions. *Simulation*, 52:199–205.

DeBrota, D. J., R. S. Dittus, J. J. Swain, S. D. Roberts, J. R. Wilson, and S. Venkatraman. 1989b. Modeling input processes with Johnson distributions. In *Proceedings of the 1989 Winter Simulation Conference*, 308–318. IEEE Press.

Deler, B. and B. L. Nelson. 2001. Modeling and generating multivariate time series with arbitrary marginals and autocorrelation structures. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 275–282. IEEE Press.

Demers, V., P. L'Ecuyer, and B. Tuffin. 2005. A combination of randomized quasi-Monte Carlo with splitting for rare-event simulation. In *Proceedings of the 2005 European Simulation and Modeling Conference*, 25–32, Ghent, Belgium. EUROSIS.

Deng, L.-Y. and E. O. George. 1990. Generation of uniform variates from several nearly uniformly distributed variables. *Communications in Statistics*, B19(1):145–154.

Deng, L.-Y. and D. K. J. Lin. 2000. Random number generation for the new century. *The American Statistician*, 54(2):145–150.

Deng, L.-Y. and H. Xu. 2003. A system of high-dimensional, efficient, long-cycle and portable uniform random number generators. *ACM Transactions on Modeling and Computer Simulation*, 13(4):299–309.

Deo, N. 1974. *Graph Theory with Applications to Engineering and Computer Science*. Englewood Cliffs, N.J.: Prentice-Hall.

Derflinger, G, W. Hörmann, and J. Leydold. 2010. Random variate generation by numerical inversion when only the density is known. *ACM Transactions on Modeling and Computer Simulation*, 20(4):Article 18.

Devroye, L. 1986. *Non-Uniform Random Variate Generation*. New York, NY: Springer-Verlag.

Devroye, L. 1996. Random variate generation in one line of code. In *Proceedings of the 1996 Winter Simulation Conference*, 265–271. IEEE Press.

Devroye, L, L. Györfi, and G. Lugosi. 1996. *A Probabilistic Theory of Pattern Recognition*. New York, NY: Springet-Verlag.

Dieter, U. 1975. How to calculate shortest vectors in a lattice. *Mathematics of Computation*, 29(131):827–833.

Dieudonné, J. 1969. *Foundations of Modern Analysis*. second ed. New York: Academic Press.

Dion, M and P. L'Ecuyer. 2010. Americal option pricing with randomized quasi-Monte Carlo simulation. In *Proceedings of the 2010 Winter Simulation Conference*, 2705–2720, Piscataway, NJ. IEEE Press.

Drouet Mori, D. and S. Kotz. 2001. *Correlation and Dependence*. Imperial College Press.

Dudewicz, E. J. 1995. The heteroscedastic method: Fifty+ years of progress 1945–2000, and professor minoru siotani's award-winning contributions. *American Journal of Mathematical and Management Sciences*, 15:170–197.

Duffie, D. 1996. *Dynamic Asset Pricing Theory*. second ed. Princeton University Press.

Durbin, J. 1973. *Distribution Theory for Tests Based on the Sample Distribution Function*. SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, PA: SIAM.

Dutré, P, K. Bala, and P. Bekaert. 2006. *Advanced Global Illumination*. second ed. Boca Raton, FL: CRC Press.

Dwork, C., F. McSherry, K. Nissim, and A. Smith. 2017. Calibrating noise to sensitivity in private data analysis. *Journal of Privacy and Confidentiality*, 7(3):17–51.

Dwork, C. and A. Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and trends in theoretical computer science*, 9(3–4):211–407.

Efron, B. 1982. *The Jackknife, The Bootstrap and Other Resampling Plans*. volume 38 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Philadelphia: SIAM.

Efron, B and R. J. Tibshirani. 1994. *An Introduction to the Bootstrap*. New York, NY: Chapman & Hall.

Eichenauer-Herrmann, J. 1995. Pseudorandom number generation by nonlinear methods. *International Statistical Reviews*, 63:247–255.

Eichenauer-Herrmann, J, H. Grothe, and J. Lehn. 1989. On the period length of pseudorandom vector sequences generated by matrix generators. *Mathematics of Computation*, 52(185):145–148.

Eichenauer-Herrmann, J., E. Herrmann, and S. Wegenkittl. 1998. A survey of quadratic and inversive congruential pseudorandom numbers. In *Monte Carlo and Quasi-Monte Carlo Methods 1996*, ed. P. Hellekalek, G. Larcher, H. Niederreiter, and P. Zinterhof, volume 127 of *Lecture Notes in Statistics*, 66–97. New York, NY: Springer.

Elmaghraby, S. 1977. *Activity Networks*. New York: Wiley.

Embrechts, P, F. Lindskog, and A. McNeil. 2003. Modelling dependence with copulas and applications to risk management. In *Handbook of Heavy Tailed Distributions in Finance*, ed. S. Rachev, 329–384. Elsevier. chapter 8.

Embrechts, P, A. McNeil, and D. Straumann. 2002. Correlation and dependence in risk management: properties and pitfalls. In *Risk Management: Value at Risk and Beyond*, ed. M. A. H. Dempster, 176–223. Cambridge: Cambridge University Press.

Esary, J. D, F. Proschan, and D. W. Walkup. 1967. Association of random variables with applications. *Annals of Mathematical Statistics*, 38:1466–1474.

Evans, M. and T. Swartz. 2000. *Approximating Integrals via Monte Carlo and Deterministic Methods*. Oxford, UK: Oxford University Press.

Fabian, V. 1968. On asymptotic normality in stochastic approximation. *Annals of Mathematical Statistics*, 39:1327–1332.

Fang, K.-T, S. Kotz, and K. W. Ng. 1987. *Statistical Multivariate and Related Distributions*. New York: Chapman and Hall.

Fang, K.-T., S. Kotz, and K. W. Ng. 1990. *Symmetric Multivariate and Related Distributions*. London: Chapman and Hall.

Faure, H. 1982. Discrépance des suites associées à un système de numération en dimension s. *Acta Arithmetica*, 61:337–351.

Faure, H and C. Lemieux. 2009. Generalized Halton sequences in 2008: A comparative study. *ACM Transactions on Modeling and Computer Simulation*, 19(4):Article 15.

Faure, H. and S. Tezuka. 2002. Another random scrambling of digital (t, s)-sequences. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, ed. K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 242–256, Berlin. Springer-Verlag.

Fedele, M., R. Piersanti, F. Regazzoni, M. Salvador, P. C. Africa, M. Bucelli, A. Zingaro, L. Dede', and A. Quarteroni. 2023. A comprehensive and biophysically detailed computational model of the whole human heart electromechanics. *Computer Methods in Applied Mechanics and Engineering*, 410:115983.

Ferrenberg, A. M., D. P. Landau, and Y. J. Wong. 1992. Monte Carlo simulations: Hidden errors from "good" random number generators. *Physical Review Letters*, 69(23):3382–3384.

Fincke, U. and M. Pohst. 1985. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44:463–471.

Fishman, G. S. 1976. Sampling from the gamma distribution on a computer. *Communications of the ACM*, 19:407–409.

Fishman, G. S. 1990. Multiplicative congruential random number generators with modulus 2^β: An exhaustive analysis for $\beta = 32$ and a partial analysis for $\beta = 48$. *Mathematics of Computation*, 54(189):331–344.

Fishman, G. S. 1996. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer Series in Operations Research, New York, NY: Springer-Verlag.

Fishman, G. S. 1998. LABATCH.2: Software for statistical analysis of simulation sample path data. In *Proceedings of the 1998 Winter Simulation Conference*, 131–140, Piscataway, NJ. IEEE Press.

Fishman, G. S. 2001. *Discrete Event Simulation: Modeling, Programming, and Analysis*. Springer Series in Operations Research, New York, NY: Springer-Verlag.

Fishman, G. S and V. G. Kulkarni. 1992. Improving Monte Carlo efficiency by increasing variance. *Management Science*, 38:1432–1444.

Fishman, G. S. and L. S. Moore III. 1986. An exhaustive analysis of multiplicative congruential random number generators with modulus $2^{31} - 1$. *SIAM Journal on Scientific and Statistical Computing*, 7(1):24–45.

Fishman, G. S. and B. D. Wang. 1983. Antithetic variates revisited. *Communications of the ACM*, 26(11):964–971.

Fleming, P. J., D. Schaeffer, and B. Simon. 1995. Efficient Monte Carlo simulation of a product-form model for a cellular system with dynamic resource sharing. *ACM Transactions on Modeling and Computer Simulation*, 5(1):3–21.

Foley, R. D. and D. Golsman. 1999. Confidence intervals using orthonormally weighted standardized time series. *ACM Transactions on Modeling and Computer Simulation*, 9(4):297–325.

Fox, B. L. 1993. Shortening future-event lists. *ORSA Journal on Computing*, 5:147–150.

Fox, B. L. 1999. *Strategies for Quasi-Monte Carlo*. Boston, MA: Kluwer Academic.

Fox, B. L and P. W. Glynn. 1986. Discrete-time conversion for simulating semi-Markov processes. *Operations Research Letters*, 5:191–196.

Fox, B. L. and P. W. Glynn. 1990. Discrete-time conversion for simulating finite-horizon Markov processes. *SIAM Journal on Applied Mathematics*, 50:1457–1473.

Fréchet, M. 1951. Sur les tableaux de corrélation dont les marges sont données. *Annales de L'Université de Lyon, Sciences Mathématiques et Astronomie*, 14:53–77.

Fréchet, M. 1957. Les tableaux de corrélation dont les marges et les bornes sont données. *Annales de L'Université de Lyon, Sciences Mathématiques et Astronomie*, 20:13–31.

Fu, M and J.-Q. Hu. 1997. *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*. Boston: Kluwer Academic.

Fu, M. C. 2002. Optimization for simulation: Theory vs practice. *INFORMS Journal on Computing*, 14(3):192–215. (see also the commentaries on pages 217–227).

Fu, M. C. 2006. Gradient estimation. In *Simulation*, ed. S. G. Henderson and B. L. Nelson, Handbooks in Operations Research and Management Science, 575–616. Amsterdam, The Netherlands: Elsevier. Chapter 19.

Fu, M. C. and S. G. Henderson. 2017. History of seeking better solutions, aka simulation optimization. In *Proceedings of the 2017 Winter Simulation Conference*, 131–157. IEEE Press.

Fu, M. C., D. B. Madan, and T. Wang. 1998. Pricing continuous Asian options: A comparison of Monte Carlo and Laplace transform inversion methods. *Journal of Computational Finance*, 2:49–74.

Galambos, J. 1978. *The Asymptotic Theory of Extreme Order Statistics*. New York, NY: Wiley.

Gans, N, G. Koole, and A. Mandelbaum. 2003. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management*, 5:79–141.

Gao, R. and A. J. Kleywegt. 2022. Distributionally robust stochastic optimization with Wasserstein distance. *Mathematics of Operations Research*. To appear.

Garvels, M. J. J. 2000. The splitting method in rare event simulation. PhD thesis, Faculty of mathematical Science, University of Twente, The Netherlands.

Garvels, M. J. J and D. P. Kroese. 1998. A comparison of RESTART implementations. In *Proceedings of the 1998 Winter Simulation Conference*, 601–609. IEEE Press.

Garvels, M. J. J., D. P. Kroese, and J.-K. C. W. Van Ommeren. 2002. On the importance function in splitting simulation. *European Transactions on Telecommunications*, 13(4):363–371.

Gentle, J. E. 2003. *Random Number Generation and Monte Carlo Methods*. second ed. New York, NY: Springer.

Gertsbakh, I. B. 1989. *Statistical Reliability Theory*. New York and Basel: Marcel Dekker.

Gertsbakh, I. B and Y. Shpungin. 2010. *Models of Network Reliability*. Boca Raton, FL: CRC Press.

Ghosh, M., N. Mukhopadhyay, and P. K. Sen. 2001. *Sequential Estimation*. New York, NY: Wiley.

Ghosh, S. and S. G. Henderson. 2001. Chessboard distributions. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 385–393. IEEE Press.

Ghosh, S. and S. G. Henderson. 2002. Properties of the NORTA method in higher dimensions. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 263–269. IEEE Press.

Glasserman, P. 1988. Performance continuity and differentiability in Monte Carlo optimization. In *Proceedings of the 1988 Winter Simulation Conference*, 518–524, Piscataway, NJ. IEEE Press.

Glasserman, P. 1991. *Gradient Estimation Via Perturbation Analysis*. Norwell, MA: Kluwer Academic.

Glasserman, P. 1993. Filtered Monte Carlo. *Mathematics of Operations Research*, 18(3):610–634.

Glasserman, P. 2004. *Monte Carlo Methods in Financial Engineering*. New York: Springer-Verlag.

Glasserman, P, P. Heidelberger, P. Shahabuddin, and T. Zajic. 1998. A large deviations perspective on the efficiency of multilevel splitting. *IEEE Transactions on Automatic Control*, AC-43(12):1666–1679.

Glasserman, P., P. Heidelberger, P. Shahabuddin, and T. Zajic. 1999. Multilevel splitting for estimating rare event probabilities. *Operations Research*, 47(4):585–600.

Glasserman, P. and S.-G. Kou. 1995. Analysis of an importance sampling estimator for tandem queues. *ACM Transactions on Modeling and Computer Simulation*, 5(1):22–42.

Glasserman, P. and Y. Wang. 1997. Counterexamples in importance sampling for large deviations probabilities. *The Annals of Applied Probability*, 7(3):731–746.

Glasserman, P. and D. D. Yao. 1992. Some guidelines and guarantees for common random numbers. *Management Science*, 38(6):884–908.

Glover, F., J. P. Kelly, and M. Laguna. 1999. New advances for wedding optimization and simulation. In *Proceedings of the 1999 Winter Simulation Conference*, 255–260, Piscataway, NJ. IEEE Press.

Glynn, P. W. 1985. Regenerative structure of Markov chains simulated via common random numbers. *Operations Research Letters*, 4:49–53.

Glynn, P. W. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

Glynn, P. W. 1994. Efficiency improvement techniques. *Annals of Operations Research*, 53:175–197.

Glynn, P. W and P. Heidelberger. 1990. Bias properties of budget constrained simulations. *Operations Research*, 38(5):801–814.

Glynn, P. W. and D. L. Iglehart. 1988. Simulation methods for queues: An overview. *Queueing Systems*, 3:221–256.

Glynn, P. W. and D. L. Iglehart. 1989. Importance sampling for stochastic simulations. *Management Science*, 35:1367–1392.

Glynn, P. W. and D. L. Iglehart. 1990. Simulation output analysis using standardized time series. *Mathematics of Operations Research*, 15(1):1–16.

Glynn, P. W., P. L'Ecuyer, and M. Adès. 1991. Gradient estimation for ratios. In *Proceedings of the 1991 Winter Simulation Conference*, ed. B. L. Nelson, W. D. Kelton, and G. M. Clark, 986–993, Piscataway, NJ. IEEE Press.

Glynn, P. W. and R. Szechtman. 2002. Some new perspectives on the method of control variates. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, ed. K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 27–49, Berlin. Springer-Verlag.

Glynn, P. W. and W. Whitt. 1989. Indirect estimation via $L = \lambda w$. *Operations Research*, 37:82–103.

Glynn, P. W. and W. Whitt. 1992. The asymptotic efficiency of simulation estimators. *Operations Research*, 40:505–520.

Goldsman, D. and B. L. Nelson. 1998. Comparing systems via simulation. In *Handbook of Simulation*, ed. J. Banks, 273–306. Wiley. chapter 8.

Goldsman, D, B. L. Nelson, and B. Schmeiser. 1991. Methods for selecting the best system. In *Proceedings of the 1991 Winter Simulation Conference*, ed. B. L. Nelson, W. D. Kelton, and G. M. Clark, 177–186, Piscataway, NJ. IEEE Press.

Goresky, M. and A. Klapper. 2003. Efficient multiply-with-carry random number generators with maximal period. *ACM Transactions on Modeling and Computer Simulation*, 13(4):310–321.

Goyal, A., P. Shahabuddin, P. Heidelberger, V. F. Nicola, and P. W. Glynn. 1992. A unified framework for simulating Markovian models of highly reliable systems. *IEEE Transactions on Computers*, C-41:36–51.

Gross, D. and C. M. Harris. 1998. *Fundamentals of Queueing Theory*. Third ed. New York, NY: Wiley.

Gumbel, E. J. 1960. Distribution des valeurs extrêmes en plusieurs dimensions. *Publications de l'Institut de Statistique de l'Université de Paris*, 9:171–173.

Gut, A. 1988. *Stopped Random Walks*. New York, NY: Springer-Verlag.

Haber, S. 1967. A modified Monte Carlo quadrature II. *Mathematics of Computation*, 21:388–397.

Häggström, O. 2002. *Finite Markov Chains and Algorithmic Applications*. Cambridge, U.K.: Cambridge University Press.

Hall, P. 1986. On powerful distributional tests based on sample spacings. *Journal of Multivariate Analysis*, 19:201–224.

Hall, P. 1988. Theoretical comparison of bootstrap confidence intervals. *Annals of Statistics*, 16:927–985.

Hall, P. 1992. *The Bootstrap and Edgeworth Expansion*. New York, NY: Springer-Verlag.

Halton, J. H. 1960. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90.

Hammersley, J. M. 1960. Monte Carlo methods for solving multivariate problems. *Annals of the New York Academy of Science*, 86:844–874.

Hammersley, J. M and D. C. Handscomb. 1964. *Monte Carlo Methods*. London: Methuen.

Haramoto, H., M. Matsumoto, T. Nishimura, F. Panneton, and P. L'Ecuyer. 2008. Efficient jump ahead for $\mathbf{F}_2$-linear random number generators. *INFORMS Journal on Computing*, 20(3):290–298.

Heidelberger, P. 1995. Fast simulation of rare events in queueing and reliability models. *ACM Transactions on Modeling and Computer Simulation*, 5(1):43–85.

Heidelberger, P and D. L. Iglehart. 1979. Comparing stochastic systems using regenerative simulations with common random numbers. *Advances in Applied Probabability*, 11:804–819.

Heinrich, S., F. J. Hickernell, and R. X. Yue. 2004. Optimal quadrature for Haar wavelet spaces. *Mathematics of Computation*, 73:259–277.

Hellekalek, P. 1998. On the assessment of random and quasi-random point sets. In *Random and Quasi-Random Point Sets*, ed. P. Hellekalek and G. Larcher, volume 138 of *Lecture Notes in Statistics*, 49–108. New York, NY: Springer.

Hellekalek, P. and S. Wegenkittl. 2003. Empirical evidence concerning AES. *ACM Transactions on Modeling and Computer Simulation*, 13(4):322–333.

Henderson, S. G. 2001. Mathematics of simulation. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 83–94. IEEE Press.

Henderson, S. G., B. A. Chiera, and R. M. Cooke. 2000. Generating "dependent" quasi-random numbers. In *Proceedings of the 2000 Winter Simulation Conference*, 527–536. IEEE Press.

Hickernell, F. J. 1998a. A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, 67(221):299–322.

Hickernell, F. J. 1998b. Lattice rules: How well do they measure up? In *Random and Quasi-Random Point Sets*, ed. P. Hellekalek and G. Larcher, volume 138 of *Lecture Notes in Statistics*, 109–166. New York: Springer-Verlag.

Hickernell, F. J. 2002. Obtaining $O(N^{-2+\epsilon})$ convergence for lattice quadrature rules. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, ed. K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 274–289, Berlin. Springer-Verlag.

Hickernell, F. J., H. S. Hong, P. L'Ecuyer, and C. Lemieux. 2001. Extensible lattice sequences for quasi-Monte Carlo quadrature. *SIAM Journal on Scientific Computing*, 22(3):1117–1138.

Hickernell, F. J., C. Lemieux, and A. B. Owen. 2005. Control variates for quasi-Monte Carlo. *Statistical Science*, 20(1):1–31.

Hillier, F. S. and G. J. Lieberman. 2021. *Introduction to Operations Research*. eleventh ed. McGraw-Hill.

Ho, Y.-C., C. G. Cassandras, C. H. Chen, and L. Y. Dai. 2000. Ordinal optimization and simulation. *Journal of the Operational Research Society*, 51:490–500.

Hoeffding, W. 1940. Maßstabinvariante korrelationstheorie. *Schriften Math. Inst. Univ. Berlin*, 5:181–233.

Hoeffding, W. 1948. A class of statistics with asymptotically normal distributions. *Annals of Mathematical Statistics*, 19:293–325.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–29.

Hogg, R. V and A. F. Craig. 1995. *Introduction to Mathematical Statistics*. 5th ed. Prentice-Hall.

Hong, H. S. and F. J. Hickernell. 2003. Algorithm 823: Implementing scrambled digital sequences. *ACM Transactions on Mathematical Software*, 29:95–109.

Hong, L. J., Z. Hu, and G. Liu. 2014. Monte Carlo methods for value-at-risk and conditional value-at-risk: A review. *ACM Transactions on Modeling and Computer Simulation*, 24(4):Article 22.

Hora, S. C. 1983. Estimation of the inverse function for random number generation. *Communications of the ACM*, 26:590–594.

Hörmann, W and G. Derflinger. 1994. The transformed rejection method for generating random variables, an alternative to the ratio-of-uniforms method. *Communications in Statistics: Simulation and Computation*, 23(3):847–860.

Hörmann, W. and G. Derflinger. 1996. Rejection-inversion to generate variates from monotone discrete distributions. *ACM Transactions on Modeling and Computer Simulation*, 6(3):169–184.

Hörmann, W. and J. Leydold. 2000. Automatic random variate generation for simulation input. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 675–682, Piscataway, NJ. IEEE Press.

Hörmann, W. and J. Leydold. 2003. Continuous random variate generation by fast numerical inversion. *ACM Transactions on Modeling and Computer Simulation*, 13(4):347–362.

Hörmann, W., J. Leydold, and G. Derflinger. 2004. *Automatic Nonuniform Random Variate Generation*. Berlin: Springer-Verlag.

Hsu, J. C. and B. L. Nelson. 1990. Control variates for quantile estimation. *Management Science*, 36(7):835–851.

Hua, L. and Y. Wang. 1981. *Applications of Number Theory to Numerical Analysis*. Berlin: Springer-Verlag.

Hull, J. C. 1997. *Options, Futures, and Other Derivative Securities*. third ed. Englewood-Cliff, N.J.: Prentice-Hall.

Hull, J. C. 2000. *Options, Futures, and Other Derivative Securities*. fourth ed. Englewood-Cliff, N.J.: Prentice-Hall.

Hull, J. C. 2006. *Options, Futures, and Other Derivatives*. sixth ed. Upper Saddle River, N.J.: Prentice-Hall.

Hull, J. C and A. White. 1987. The pricing of options on assets with stochastic volatilities. *Journal of Finance*, 42:281–300.

Hyndman, R. J. and Y. Fan. 1996. Sample quantiles in statistical packages. *American Statistician*, 50(4):361–365.

Ibrahim, R., P. L'Ecuyer, H. Shen, and M. Thiongane. 2016. Inter-dependent, heterogeneous, and time-varying service-time distributions in call centers. *European Journal of Operational Research*, 250:480–492.

Imai, J. and K. S. Tan. 2002. Enhanced quasi-Monte Carlo methods with dimension reduction. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C. H. Chen, J. L. Snowdon, and J. M. Charnes, 1502–1510, Piscataway, New Jersey. IEEE Press.

Imai, J. and K. S. Tan. 2004. Minimizing effective dimension using linear transformation. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, ed. H. Niederreiter, 275–292, Berlin. Springer-Verlag.

Imai, J and K. S. Tan. 2006. A general dimension reduction technique for derivative pricing. *Journal of Computational Finance*, 10(2):129–155.

Iman, R. and W. Conover. 1982. A distribution-free approach to inducing rank correlation among input variables. *Communications in Statistics—Simulation and Computation*, 11:311–334.

Jäckel, P. 2002. *Monte Carlo Methods in Finance*. Chichester, U.K.: John Wiley.

Jagerman, D. L and B. Melamed. 1992. The transition and autocorrelation structure of TES process. *Communications in Statistics: Stochastic Models*, 8(2):193–219.

Jaoua, A., P. L'Ecuyer, and L. Delorme. 2013. Call type dependence in multiskill call centers. *Simulation*, 89(6):722–734.

Joe, H. 1997. *Multivariate Models and Dependence Concepts*. London: Chapman and Hall.

Joe, S and F. Y. Kuo. 2008. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654.

Joe, S. and I. H. Sloan. 1992. Imbedded lattice rules for multidimensional integration. *SIAM Journal on Numerical Analysis*, 29:1119–1135.

Johnson, M. E. 1987. *Multivariate Statistical Simulation*. New York, NY: John Wiley.

Johnson, N. L. 1949. Systems of frequency curves generated by methods of translation. *Biometrika*, 36:149–176.

Johnson, N. L and S. Kotz. 1969–1972a. *Distributions in Statistics*. New York, NY: Wiley. In four volumes.

Johnson, N. L. and S. Kotz. 1972b. *Distributions in Statistics: Continuous Multivariate Distributions*. New York, NY: John Wiley.

Johnson, N. L., S. Kotz, and N. Balakrishnan. 1994–1995. *Continuous Univariate Distributions*. 2nd ed., volume 1–2. Wiley.

Jones, M. C., J. S. Marron, and S. J. Sheather. 1996. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91(433):401–407.

Jørgensen, B. 1982. *Statistical Properties of the Generalized Inverse Gaussian Distribution*. volume 9 of *Lecture Notes in Statistics*. New York–Berlin: Springer-Verlag.

Juneja, S and P. Shahabuddin. 1999. Simulating heavy-tailed processes using delayed hazard rate twisting. Manuscript.

Juneja, S. and P. Shahabuddin. 2006. Rare event simulation techniques: An introduction and recent advances. In *Simulation*, ed. S. G. Henderson and B. L. Nelson, Handbooks in Operations Research and Management Science, 291–350. Amsterdam, The Netherlands: Elsevier. Chapter 11.

Kahn, H. and T. E. Harris. 1951. Estimation of particle transmission by random sampling. *National Bureau of Standards Applied Mathematical Series*, 12:27–30.

Karatzas, I. and S. E. Shreve. 1998. *Methods of Mathematical Finance*. New York: Springer.

Kemna, A. G. Z. and A. C. F. Vorst. 1990. A pricing method for options based on average asset values. *Journal of Banking and Finance*, 14:113–129.

Kibria, B. M. G. and A. H. Joarder. 2006. A short review of multivariate t-distribution. *Journal of Statistical Research*, 40(1):59–72.

Kim, S., R. Pasupathy, and S. G. Henderson. 2015. A guide to sample average approximation. In *Handbook of Simulation Optimization*, ed. M. C. Fu, 207–243. New York, USA: Springer.

Kim, S.-H and B. L. Nelson. 2006. Selecting the best system. In *Simulation*, ed. S. G. Henderson and B. L. Nelson, Handbooks in Operations Research and Management Science, 501–534. Amsterdam, The Netherlands: Elsevier. Chapter 17.

Kinderman, A. J. and J. F. Monahan. 1977. Computer generation of random variables using the ratio of uniform deviates. *ACM Transactions on Mathematical Software*, 3:257–260.

King, A. J. 1989. Generalized delta theorems for multivalued mappings and measurable selections. *Mathematics of Operations Research*, 14:720–736.

Kleijnen, J. P. C. 1974. *Statistical Techniques in Simulation, Part. 1*. New York, NY: Dekker.

Kleijnen, J. P. C. 1995. Verification and validation of simulation models. *European Journal of Operations Research*, 82:145–162.

Kleijnen, J. P. C. 1998. Experimental design for sensitivity analysis, optimization, and validation of simulation models. In *Handbook of Simulation*, ed. J. Banks, 173–223. Wiley. chapter 6.

Kleinrock, L. 1975. *Queueing Systems, Vol. 1.* New York, NY: Wiley.

Kleywegt, A. J, A. Shapiro, and T. Homem-de Mello. 2002. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.

Kloeden, P. E. and E. Platen. 1992. *Numerical Solutions of Stochastic Differential Equations*. Berlin: Springer-Verlag.

Knuth, D. E. 1969. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. First ed. Reading, MA: Addison-Wesley.

Knuth, D. E. 1981. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Second ed. Reading, MA: Addison-Wesley.

Knuth, D. E. 1998. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Third ed. Reading, MA: Addison-Wesley.

Kollman, C, K. Baggerly, D. Cox, and R. Picard. 1999. Adaptive importance sampling on discrete Markov chains. *Annals of Applied Probability*, 9(2):391–412.

Koole, G. 2013. *Call Center Optimization*. MG books, Amsterdam.

Korobov, N. M. 1959. The approximate computation of multiple integrals. *Dokl. Akad. Nauk SSSR*, 124:1207–1210. in Russian.

Kroese, D. P, T. Taimre, and Z. I. Botev. 2011. *Handbook of Monte Carlo Methods*. New York: John Wiley and Sons.

Kronmal, R. A. and A. V. Peterson. 1979. On the alias method for generating random variables from discrete a distribution. *The American Statistician*, 33(4):214–218.

Kronmal, R. A. and A. V. Peterson. 1984. An acceptance-complement analogue of the mixture-plus-acceptance-rejection method for generating random variables. *ACM Transactions on Mathematical Software*, 10:271–281.

Kuhl, M. E., J. R. Wilson, and M. A. Johnson. 1997. Estimating and simulating Poisson processes having trends or multiple periodicities. *IIE Transactions*, 29:201–211.

Kuhl, M. E., J. R. Wilson, and M. A. Johnson. 1998. Least-squares estimation of nonhomogeneous Poisson processes. In *Proceedings of the 1998 Winter Simulation Conference*, 637–645. IEEE Press.

Kuipers, L. and H. Niederreiter. 1974. *Uniform Distribution of Sequences*. New York, NY: John Wiley.

Kushner, H. J. and G. Yin. 2003. *Stochastic Approximation Algorithms and Applications*. second ed. New York: Springer Verlag.

Lagarias, J. C. 1993. Pseudorandom numbers. *Statistical Science*, 8(1):31–39.

Larcher, G, A. Lauss, H. Niederreiter, and W. C. Schmid. 1996. Optimal polynomials for (t, m, s)-nets and numerical integration of multivariate Walsh series. *SIAM Journal on Numerical Analysis*, 33(6):2239–2253.

Larcher, G., H. Niederreiter, and W. C. Schmid. 1996. Digital nets and sequences constructed over finite rings and their application to quasi-Monte Carlo integration. *Monatshefte für Mathematik*, 121(3):231–253.

Larcher, G. and G. Pirsic. 1999. Base change problems for generalized Walsh series and multivariate numerical integration. *Pacific Journal of Mathematics*, 189:75–105.

Lavenberg, S. S. and P. D. Welch. 1981. A perspective on the use of control variables to increase the efficiency of Monte Carlo simulations. *Management Science*, 27:322–335.

Law, A. M. 1974. Efficient estimators for simulated queueing systems. Technical Report ORC 74-7, Operations Research Center, University of California at Berkeley.

Law, A. M. 2014. *Simulation Modeling and Analysis*. Fifth ed. New York: McGraw-Hill.

Law, A. M and W. D. Kelton. 1991. *Simulation Modeling and Analysis*. Second ed. New York, NY: McGraw-Hill.

Law, A. M. and W. D. Kelton. 2000. *Simulation Modeling and Analysis*. Third ed. New York, NY: McGraw-Hill.

Law, A. M., W. D. Kelton, and L. W. Koenig. 1981. Relative width sequential confidence intervals for the mean. *Communications in Statistics, Series B*, 10:29–39.

Le, D.-T., G. Cernicchiaro, C. Zegras, and J. Ferreira. 2016. Constructing a synthetic population of establishments for the simmobility microsimulation platform. *Transportation Research Procedia*, 19:81–93.

L'Ecuyer, P. 1983. Processus de décision markoviens à étapes discrètes: Application à des problèmes de remplacement d'équipement. PhD thesis, Dépt. d'IRO, Université de Montréal.

L'Ecuyer, P. 1988. Efficient and portable combined random number generators. *Communications of the ACM*, 31(6):742–749 and 774. See also the correspondence in the same journal, 32, 8 (1989) 1019–1024.

L'Ecuyer, P. 1988. SIMOD: Définition fonctionnelle et guide d'utilisation (version 2.0). Technical Report DIUL-RT-8804, Département d'informatique, Université Laval.

L'Ecuyer, P. 1990a. Random numbers for simulation. *Communications of the ACM*, 33(10):85–97.

L'Ecuyer, P. 1990b. A unified view of the IPA, SF, and LR gradient estimation techniques. *Management Science*, 36(11):1364–1383.

L'Ecuyer, P. 1991. An overview of derivative estimation. In *Proceedings of the 1991 Winter Simulation Conference*, ed. B. L. Nelson, W. D. Kelton, and G. M. Clark, 207–217, Piscataway, NJ. IEEE Press.

L'Ecuyer, P. 1992. Convergence rates for steady-state derivative estimators. *Annals of Operations Research*, 39:121–136.

L'Ecuyer, P. 1993. Two approaches for estimating the gradient in a functional form. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. E. Biles, 338–346, Piscataway, NJ. IEEE Press.

L'Ecuyer, P. 1994a. Efficiency improvement via variance reduction. In *Proceedings of the 1994 Winter Simulation Conference*, ed. J. D. Tew, M. Manivannan, D. A. Sadowski, and A. F. Seila, 122–132. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

L'Ecuyer, P. 1994b. Uniform random number generation. *Annals of Operations Research*, 53:77–120.

L'Ecuyer, P. 1996a. Combined multiple recursive random number generators. *Operations Research*, 44(5):816–822.

L'Ecuyer, P. 1996b. Maximally equidistributed combined Tausworthe generators. *Mathematics of Computation*, 65(213):203–213.

L'Ecuyer, P. 1997. Bad lattice structures for vectors of non-successive values produced by some linear recurrences. *INFORMS Journal on Computing*, 9(1):57–60.

L'Ecuyer, P. 1999a. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1):159–164.

L'Ecuyer, P. 1999b. Tables of maximally equidistributed combined LFSR generators. *Mathematics of Computation*, 68(225):261–269.

L'Ecuyer, P. 2001. Software for uniform random number generation: Distinguishing the good and the bad. In *Proceedings of the 2001 Winter Simulation Conference*, 95–105, Piscataway, NJ. IEEE Press.

L'Ecuyer, P. 2004. Polynomial integration lattices. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, ed. H. Niederreiter, 73–98, Berlin. Springer-Verlag.

L'Ecuyer, P. 2005. Comment on "control variates for quasi-Monte Carlo". *Statistical Science*, 20(1):19–21.

L'Ecuyer, P. 2008. *SSJ: A Java Library for Stochastic Simulation.* Software user's guide, available at `http://www.iro.umontreal.ca/~lecuyer`.

L'Ecuyer, P. 2009. Quasi-Monte Carlo methods with applications in finance. *Finance and Stochastics*, 13(3):307–349.

L'Ecuyer, P. 2010. Pseudorandom number generators. In *Simulation Methods in Financial Engineering*, ed. E. Platen and P. Jaeckel, Encyclopedia of Quantitative Finance, 1431–1437. Chicester, UK: Wiley.

L'Ecuyer, P. 2015. Random number generation with multiple streams for sequential and parallel computers. In *Proceedings of the 2015 Winter Simulation Conference*, 31–44. IEEE Press.

L'Ecuyer, P. 2017. History of uniform random number generation. In *Proceedings of the 2017 Winter Simulation Conference*, 202–230. IEEE Press.

L'Ecuyer, P. 2018. Randomized quasi-Monte Carlo: An introduction for practitioners. In *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC 2016*, ed. P. W. Glynn and A. B. Owen, 29–52, Berlin. Springer.

L'Ecuyer, P. 2023. SSJ: Stochastic simulation in Java. `https://github.com/umontreal-simul/ssj`.

L'Ecuyer, P and T. H. Andres. 1997. A random number generator based on the combination of four LCGs. *Mathematics and Computers in Simulation*, 44:99–107.

L'Ecuyer, P., F. Blouin, and R. Couture. 1993. A search for good multiple recursive random number generators. *ACM Transactions on Modeling and Computer Simulation*, 3(2):87–98.

L'Ecuyer, P. and E. Buist. 2005. Simulation in Java with SSJ. In *Proceedings of the 2005 Winter Simulation Conference*, 611–620. IEEE Press.

L'Ecuyer, P. and E. Buist. 2008. On the interaction between stratification and control variates, with illustrations in a call center simulation. *Journal of Simulation*, 2(1):29–40.

L'Ecuyer, P. and Y. Champoux. 2001. Estimating small cell-loss ratios in atm switches via importance sampling. *ACM Transactions on Modeling and Computer Simulation*, 11(1):76–105.

L'Ecuyer, P., J.-F. Cordeau, and R. Simard. 2000. Close-point spatial tests and their application to random number generators. *Operations Research*, 48(2):308–317.

L'Ecuyer, P. and S. Côté. 1991. Implementing a random number package with splitting facilities. *ACM Transactions on Mathematical Software*, 17(1):98–111.

L'Ecuyer, P. and R. Couture. 1997. An implementation of the lattice and spectral tests for multiple recursive linear random number generators. *INFORMS Journal on Computing*, 9(2):206–217.

L'Ecuyer, P. and N. Giroux. 1987. A process-oriented simulation package based on Modula-2. In *1987 Winter Simulation Proceedings*, 165–174.

L'Ecuyer, P., N. Giroux, and P. W. Glynn. 1994. Stochastic optimization by simulation: Numerical experiments with the $M/M/1$ queue in steady-state. *Management Science*, 40(10):1245–1261.

L'Ecuyer, P. and P. W. Glynn. 1994. Stochastic optimization by simulation: Convergence proofs for the $GI/G/1$ queue in steady-state. *Management Science*, 40(11):1562–1578.

L'Ecuyer, P. and J. Granger-Piché. 2003. Combined generators with components from different families. *Mathematics and Computers in Simulation*, 62:395–404.

L'Ecuyer, P., K. Gustavsson, and L. Olsson. 2018. Modeling bursts in the arrival process to an emergency call center. In *Proceedings of the 2018 Winter Simulation Conference*, 525–536. IEEE Press.

L'Ecuyer, P. and P. Hellekalek. 1998. Random number generators: Selection criteria and testing. In *Random and Quasi-Random Point Sets*, ed. P. Hellekalek and G. Larcher, volume 138 of *Lecture Notes in Statistics*, 223–265. New York, NY: Springer-Verlag.

L'Ecuyer, P., C. Lécot, and B. Tuffin. 2008. A randomized quasi-Monte Carlo simulation method for Markov chains. *Operations Research*, 56(4):958–975.

L'Ecuyer, P. and C. Lemieux. 2000. Variance reduction via lattice rules. *Management Science*, 46(9):1214–1235.

L'Ecuyer, P. and C. Lemieux. 2002. Recent advances in randomized quasi-Monte Carlo methods. In *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, ed. M. Dror, P. L'Ecuyer, and F. Szidarovszky, 419–474. Boston: Kluwer Academic.

L'Ecuyer, P., L. Meliani, and J. Vaucher. 2002. SSJ: A framework for stochastic simulation in Java. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 234–242. IEEE Press.

L'Ecuyer, P., D. Munger, B. Oreshkin, and R. Simard. 2017. Random numbers for parallel computers: Requirements and methods, with emphasis on GPUs. *Mathematics and Computers in Simulation*, 135:3–17.

L'Ecuyer, P., D. Munger, and B. Tuffin. 2010. On the distribution of integration error by randomly-shifted lattice rules. *Electronic Journal of Statistics*, 4:950–993.

L'Ecuyer, P., O. Nadeau-Chamard, Y.-F. Chen, and J. Lebar. 2021. Multiple streams with recurrence-based, counter-based, and splittable random number generators. In *Proceedings of the 2021 Winter Simulation Conference*, 1–16. IEEE Press.

L'Ecuyer, P. and F. Panneton. 2002. Construction of equidistributed generators based on linear recurrences modulo 2. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, ed. K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 318–330. Berlin: Springer-Verlag.

L'Ecuyer, P. and F. Panneton. 2009. $\mathbf{F}_2$-linear random number generators. In *Advancing the Frontiers of Simulation: A Festschrift in Honor of George Samuel Fishman*, ed. C. Alexopoulos, D. Goldsman, and J. R. Wilson, 169–193. New York: Springer-Verlag.

L'Ecuyer, P. and G. Perron. 1994. On the convergence rates of IPA and FDC derivative estimators. *Operations Research*, 42(4):643–656.

L'Ecuyer, P. and R. Proulx. 1989. About polynomial-time "unpredictable" generators. In *Proceedings of the 1989 Winter Simulation Conference*, ed. K. J. Musselman, P. Heidelberger, and E. A. MacNair, 467–476. IEEE Press.

L'Ecuyer, P., G. Rubino, S. Saggadi, and B. Tuffin. 2011. Approximate zero-variance importance sampling for static network reliability estimation. *IEEE Transactions on Reliability*, 8(4):590–604.

L'Ecuyer, P. and R. Simard. 1999. Beware of linear congruential generators with multipliers of the form $a = \pm 2^q \pm 2^r$. *ACM Transactions on Mathematical Software*, 25(3):367–374.

L'Ecuyer, P. and R. Simard. 2001. On the performance of birthday spacings tests for certain families of random number generators. *Mathematics and Computers in Simulation*, 55(1–3):131–137.

L'Ecuyer, P. and R. Simard. 2002. *TestU01: A Software Library in ANSI C for Empirical Testing of Random Number Generators*. Software user's guide. Available at `http://www.iro.umontreal.ca/~lecuyer`.

L'Ecuyer, P. and R. Simard. 2007. TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 33(4):Article 22.

L'Ecuyer, P., R. Simard, E. J. Chen, and W. D. Kelton. 2002. An object-oriented random-number package with many long streams and substreams. *Operations Research*, 50(6):1073–1075.

L'Ecuyer, P., R. Simard, and S. Wegenkittl. 2002. Sparse serial tests of uniformity for random number generators. *SIAM Journal on Scientific Computing*, 24(2):652–668.

L'Ecuyer, P. and S. Tezuka. 1991. Structural properties for two classes of combined random number generators. *Mathematics of Computation*, 57(196):735–746.

L'Ecuyer, P. and R. Touzin. 2000. Fast combined multiple recursive generators with multipliers of the form $a = \pm 2^q \pm 2^r$. In *Proceedings of the 2000 Winter Simulation Conference*, 683–689. IEEE Press.

L'Ecuyer, P. and R. Touzin. 2004. On the Deng-Lin random number generators and related methods. *Statistics and Computing*, 14:5–9.

L'Ecuyer, P. and B. Tuffin. 2007. Effective approximation of zero-variance simulation in a reliability setting. In *Proceedings of the 2007 European Simulation and Modeling Conference*, 48–54, Ghent, Belgium. EUROSIS.

L'Ecuyer, P. and B. Tuffin. 2008. Approximate zero-variance simulation. In *Proceedings of the 2008 Winter Simulation Conference*, 170–181. IEEE Press.

L'Ecuyer, P. and F. Vázquez-Abad. 1997. Functional estimation with respect to a threshold parameter via dynamic split-and-merge. *Discrete Event Dynamic Systems: Theory and Applications*, 7(1):69–92.

L'Ecuyer, P. and G. Yin. 1998. Budget-dependent convergence rate for stochastic approximation. *SIAM Journal on Optimization*, 8(1):217–247.

Lee, S., J. R. Wilson, and M. M. Crawford. 1991. Modeling and simulation of a nonhomogeneous Poisson process having cyclic behavior. *Communications in Statistics—Simulation and Computation*, 20:777–809.

Leeb, H. 1995. Random numbers for computer simulation. Master's thesis, University of Salzburg.

Leemis, L. 2001. Input modeling techniques for discrete-event simulations. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 62–73. IEEE Press.

Leemis, L. M. 1991. Nonparametric estimation of the cumulative intensity function for a nonhomogeneous poisson process. *Management Science*, 37(7):886–900.

Léger, C, D. Politis, and J. Romano. 1992. Bootstrap technology and applications. *Technometrics*, 34:378–398.

Lehmann, E. L. 1966. Some concepts of dependence. *Annals of Mathematical Statistics*, 37:1137–1153.

Lehmann, E. L and G. Casella. 1998. *Theory of Point Estimation*. third ed. New York, NY: Springer-Verlag.

Lehmer, D. H. 1951. Mathematical methods in large scale computing units. *The Annals of the Computation Laboratory of Harvard University*, 26:141–146.

Lehtonen, T and H. Nyrhinen. 1992a. On asymptotically efficient simulation of ruin probabilities in a Markovian environment. *Scandinavian Actuarial Journal*, 24:858–874. rare events, ruin, risk.

Lehtonen, T. and H. Nyrhinen. 1992b. Simulating level-crossing probabilities by importance sampling. *Advances in Applied Probability*, 24:858–874. is, rare events.

Lemieux, C., M. Cieslak, and K. Luttmer. 2004. *RandQMC User's Guide: A Package for Randomized Quasi-Monte Carlo Methods in C*. Software user's guide, available at `http://www.math.uwaterloo.ca/~clemieux/randqmc.html`.

Lemieux, C. and P. L'Ecuyer. 1998. Efficiency improvement by lattice rules for pricing Asian options. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, 579–586, Piscataway, NJ. IEEE Press.

Lemieux, C. and P. L'Ecuyer. 2001. On selection criteria for lattice rules and other low-discrepancy point sets. *Mathematics and Computers in Simulation*, 55(1–3):139–148.

Lemieux, C. and P. L'Ecuyer. 2003. Randomized polynomial lattice rules for multivariate integration and simulation. *SIAM Journal on Scientific Computing*, 24(5):1768–1789.

Lévy, P. 1925. *Calcul des Probabilités*. Gauthier Villars.

Lewis, P. A. W and E. McKenzie. 1991. Minification processes and their transformations. *Journal of Applied Probability*, 28:45–57.

Lewis, T. G. and W. H. Payne. 1973. Generalized feedback shift register pseudorandom number algorithm. *Journal of the ACM*, 20(3):456–468.

Leydold, J. and W. Hörmann. 2002. *UNU.RAN—A Library for Universal Non-Uniform Random Number Generators*. Available at `http://statistik.wu-wien.ac.at/unuran`.

Leydold, J., E. Janka, and W. Hörmann. 2002. Variants of transformed density rejection and correlation induction. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, ed. K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 345–356, Berlin. Springer-Verlag.

Li, S. T. and J. L. Hammond. 1975. Generation of pseudorandom numbers with specified univariate distribution and correlation coefficients. *IEEE Transactions on Systems, Man, and Cybernetics*, 5:557–561.

Lidl, R. and H. Niederreiter. 1986. *Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge University Press.

Lindholm, J. H. 1968. An analysis of the pseudo-randomness properties of subsequences of long m-sequences. *IEEE Transactions on Information Theory*, IT-14(4):569–576.

Lindley, D. V. 1952. The theory of queues with a single server. *Proceedings of the Cambridge Philosophical Society*, 48(2):277–289.

Liu, J. S. 2001. *Monte Carlo Strategies in Scientific Computing*. New York, NY: Springer-Verlag.

Liu, R and A. B. Owen. 2006. Estimating mean dimensionality of analysis of variance decompositions. *Journal of the American Statistical Association*, 101(474):712–721.

Livny, M., B. Melamed, and A. K. Tsiolis. 1993. The impact of autocorrelation on queuing systems. *Management Science*, 39(3):322–339.

Loh, W.-L. 1996. A combinatorial central limit theorem for randomized orthogonal sampling designs. *Annals of Statistics*, 24:1209–1224.

Longstaff, F. A and E. S. Schwartz. 2001. Valuing american options by simulation: A simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147.

Luby, M. 1996. *Pseudorandomness and Cryptographic Applications*. Princeton: Princeton University Press.

Lurie, P. M and M. S. Goldberg. 1998. An approximate method for sampling correlated random variables from partially-specified distributions. *Management Science*, 44:203–218.

Lüscher, M. 1994. A portable high-quality random number generator for lattice field theory simulations. *Computer Physics Communications*, 79:100–110.

Macal, C. M, P. L'Ecuyer, S. Chick, B. L. Nelson, S. Brailsford, and S. Taylor. 2013. Grand challenges in modeling and simulation: An OR/MS perspective. In *Proceedings of the 2013 Winter Simulation Conference*, 1269–1282. IEEE Press.

Madan, D. B., P. P. Carr, and E. C. Chang. 1998. The variance gamma process and option pricing. *European Finance Review*, 2:79–105.

Madan, D. B. and F. Milne. 1991. Option pricing with V.G. martingale components. *Mathematical Finance*, 1:39–55.

Mardia, K. V. 1970. A translation family of bivariate distributions and Fréchet's bounds. *Sankhya*, A32:119–122.

Marsaglia, G. 1962. Improving the polar method for generating a pair of random variables. Technical report, Boeing Scientific Research Laboratory, Seattle, Washington.

Marsaglia, G. 1968. Random numbers fall mainly in the planes. *Proceedings of the National Academy of Sciences of the United States of America*, 60:25–28.

Marsaglia, G. 1985. A current view of random number generators. In *Computer Science and Statistics, Sixteenth Symposium on the Interface*, ed. L. Billard, 3–10, North-Holland, Amsterdam. Elsevier Science Publishers.

Marsaglia, G. 1996. The Marsaglia random number CDROM including the DIEHARD battery of tests of randomness. See `http://www.stat.fsu.edu/pub/diehard`.

Marsaglia, G. 2003. Xorshift RNGs. *Journal of Statistical Software*, 8(14):1–6.

Marsaglia, G and J. Marsaglia. 2004. Evaluating the Anderson-Darling distribution. *Journal of Statistical Software*, 9(2):1–5. See `http://www.jstatsoft.org/v09/i02/`.

Marsaglia, G., W. W. Tsang, and J. Wang. 2003. Evaluating Kolmogorov's distribution. *Journal of Statistical Software*, 8(18):1–4. URL is `http://www.jstatsoft.org/v08/i18/`.

Marsaglia, G. and A. Zaman. 1991. A new class of random number generators. *The Annals of Applied Probability*, 1:462–480.

Marsaglia, G., A. Zaman, and J. C. W. Marsaglia. 1994. Rapid evaluation of the inverse normal distribution function. *Statistics and Probability Letters*, 19:259–266.

Marshall, A. and I. Olkin. 1967. A multivariate exponential distribution. *Journal of the American Statistical Association*, 62:30–44.

Massart, P. 1990. The tight constant in the dvoretzky-kiefer-wolfowitz inequality. *The Annals of Probability*, 18(3):1269–1283.

Massey, J. L. 1969. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theor*, IT-15:122–127.

Matoušek, J. 1999. *Geometric Discrepancy: An Illustrated Guide*. Berlin: Springer-Verlag.

Matsumoto, M and Y. Kurita. 1992. Twisted GFSR generators. *ACM Transactions on Modeling and Computer Simulation*, 2(3):179–194.

Matsumoto, M. and Y. Kurita. 1994. Twisted GFSR generators II. *ACM Transactions on Modeling and Computer Simulation*, 4(3):254–266.

Matsumoto, M. and Y. Kurita. 1996. Strong deviations from randomness in m-sequences based on trinomials. *ACM Transactions on Modeling and Computer Simulation*, 6(2):99–106.

Matsumoto, M. and T. Nishimura. 1998. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30.

McKay, M. D., R. J. Beckman, and W. J. Conover. 1979. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245.

Meketon, M. S. and P. Heidelberger. 1982. A renewal theoretic approach to bias reduction in regenerative simulations. *Management Science*, 26:173–181.

Melamed, B. 1991. TES: A class of methods for generating autocorrelated uniform variates. *ORSA Journal on Computing*, 3(4):317–329.

Meyn, S. P and R. L. Tweedie. 1993. *Markov Chains and Stochastic Stability*. Springer-Verlag.

Michael, J. R., W. R. Schuchany, and R. W. Haas. 1976. Generating random variates using transformations with multiple roots. *The American Statistician*, 30:88–90.

Morokoff, W. J. 1998. Generating quasi-random paths for stochastic processes. *SIAM Review*, 40(4):765–788.

Morrison, D. F. 1990. *Multivariate Statistical Methods*. third ed. McGraw-Hill.

Moskowitz, B and R. E. Caflisch. 1996. Smoothness and dimension reduction in quasi-Monte Carlo methods. *Journal of Mathematical and Computer Modeling*, 23:37–54.

Müller, K. and K. W. Axhausen. 2011. Population synthesis for microsimulation: State of the art. In TRB 90th Annual Meeting Compendium of Papers DVD, number 11-1789, Washington DC, United States.

Myers, R. H. and D. C. Montgomery. 2002. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. second ed. New York, NY: Wiley.

Nelsen, R. B. 1999. *An Introduction to Copulas*. volume 139 of *Lecture Notes in Statistics*. New York, NY: Springer-Verlag.

Nelson, B. L. 1987. On control variate estimators. *Computers and Operations Research*, 14(3):219–225.

Nelson, B. L. 1989. Batch size effects on the efficiency of control variates in simulation. *European Journal of Operational Research*, 43:184–196.

Nelson, B. L. 1990. Control-variate remedies. *Operations Research*, 38:974–992.

Nelson, B. L. 1992. Statistical analysis of simulation results. In *Handbook of Industrial Engineering*, ed. G. Salvendy, chapter 102, 2567–2593. New York, NY: Wiley.

Nelson, B. L and M. Yamnitsky. 1998. Input modeling tools for complex problems. In *Proceedings of the 1998 Winter Simulation Conference*, 105–112, Piscataway, NJ. IEEE Press.

Nicola, V. F., M. K. Nakayama, P. Heidelberger, and A. Goyal. 1991. Fast simulation of highly dependable systems with general failure and repair processes. *IEEE Transactions on Computers*, 42(8):1440–1452.

Niederreiter, H. 1987. Point sets and sequences with small discrepancy. *Monatshefte für Mathematik*, 104:273–337.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. volume 63 of *SIAM CBMS-NSF Reg. Conf. Series in Applied Mathematics*. SIAM.

Niederreiter, H and I. E. Shparlinski. 2002. Recent advances in the theory of nonlinear pseudorandom number generators. In *Monte Carlo and Quasi-Monte Carlo Methods 2000*, ed. K.-T. Fang, F. J. Hickernell, and H. Niederreiter, 86–102, Berlin. Springer-Verlag.

Niederreiter, H. and C. Xing. 1998. The algebraic-geometry approach to low-discrepancy sequences. In *Monte Carlo and Quasi-Monte Carlo Methods 1996*, ed. P. Hellekalek, G. Larcher, H. Niederreiter, and P. Zinterhof, volume 127 of *Lecture Notes in Statistics*, 139–160, New York, NY. Springer-Verlag.

Nishimura, T. 2000. Tables of 64-bit Mersenne twisters. *ACM Transactions on Modeling and Computer Simulation*, 10(4):348–357.

Nygaard, K and O.-J. Dahl. 1978. The development of the SIMULA languages. *ACM SIGPLAN Notices*, 13(8):245–272.

Ólafsson, S. 2006. Metaheuristics. In *Simulation*, ed. S. G. Henderson and B. L. Nelson, Handbooks in Operations Research and Management Science, 633–654. Amsterdam, The Netherlands: Elsevier. Chapter 21.

Oreshkin, B., N. Régnard, and P. L'Ecuyer. 2016. Rate-based daily arrival process models with application to call centers. *Operations Research*, 64(2):510–527.

Owen, A. B. 1992. A central limit theorem for Latin hypercube sampling. *Journal of the Royal Statistical Society B*, 54(2):541–551.

Owen, A. B. 1995. Randomly permuted (t, m, s)-nets and (t, s)-sequences. In *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, ed. H. Niederreiter and P. J.-S. Shiue, volume 106 of *Lecture Notes in Statistics*, 299–317. Springer-Verlag.

Owen, A. B. 1997. Scrambled net variance for integrals of smooth functions. *Annals of Statistics*, 25(4):1541–1562.

Owen, A. B. 1998. Latin supercube sampling for very high-dimensional simulations. *ACM Transactions on Modeling and Computer Simulation*, 8(1):71–102.

Owen, A. B. 2003. Variance with alternative scramblings of digital nets. *ACM Transactions on Modeling and Computer Simulation*, 13(4):363–378.

Panneton, F. 2004. Construction d'ensembles de points basée sur des récurrences linéaires dans un corps fini de caractéristique 2 pour la simulation Monte Carlo et l'intégration quasi-Monte Carlo. PhD thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada.

Panneton, F and P. L'Ecuyer. 2004. Random number generators based on linear recurrences in F_{2^w}. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, ed. H. Niederreiter, 367–378, Berlin. Springer-Verlag.

Panneton, F and P. L'Ecuyer. 2005. On the xorshift random number generators. *ACM Transactions on Modeling and Computer Simulation*, 15(4):346–361.

Panneton, F. and P. L'Ecuyer. 2010. Resolution-stationary random number generators. *Mathematics and Computers in Simulation*, 80(6):1096–1103.

Panneton, F., P. L'Ecuyer, and M. Matsumoto. 2006. Improved long-period generators based on linear recurrences modulo 2. *ACM Transactions on Mathematical Software*, 32(1):1–16.

Papageorgiou, A. 2002. The Brownian bridge does not offer a consistent advantage in quasi-Monte Carlo integration. *Journal of Complexity*, 18:171–186.

Parekh, S and J. Walrand. 1989. A quick simulation method for excessive backlogs in networks of queues. *IEEE Transactions on Automatic Control*, AC-34:54–66.

Pharr, M., W. Jakob, and G. Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation*. third ed. Cambridge, MA: Morgan Kaufmann.

Polyak, B. T. and A. B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Porta Nova, A. and J. R. Wilson. 1989. Estimation of multiresponse simulation metamodels using control variates. *Management Science*, 35:1316–1333.

Porta Nova, A. and J. R. Wilson. 1993. Selecting control variates to estimate multiresponse simulation metamodels. *European Journal of Operational Research*, 71:80–94.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. second ed. New York, NY: Cambridge University Press.

Pritsker, A. A. B. 1998. Principles of simulation modeling. In *Handbook of Simulation*, ed. J. Banks, 31–51. Wiley. chapter 2.

Qiao, H and C. P. Tsokos. 1994. Parameter estimation of the Weibull probability distribution. *Mathematics and Computers in Simulation*, 37:47–55.

Raykar, V. C. and R. Duraiswami. 2006. Fast optimal bandwidth selection for kernel density estimation. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, 524–528.

Read, T. R. C. and N. A. C. Cressie. 1988. *Goodness-of-Fit Statistics for Discrete Multivariate Data*. Springer Series in Statistics, New York, NY: Springer-Verlag.

Ribeiro, C. and N. Webber. 2003. Correcting for simulation bias in Monte Carlo methods to value exotic options in models driven by Lévy processes. Working Paper, Cass Business School, London, UK.

Ripley, B. D. 1987. *Stochastic Simulation*. New York, NY: Wiley.

Robbins, H and S. Monro. 1951. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Robert, C. P. and G. Casella. 2004. *Monte Carlo Statistical Methods*. second ed. New York, NY: Springer-Verlag.

Robinson, S. M. 1996. Analysis of sample path optimization. *Mathematics of Operations Research*, 21:513–528.

Rosiński, J. 2007. Tempering stable processes. *Stochastic Processes and their Applications*, 117(6):677–707.

Ross, S. M. 1970. *Applied Probability Models with Optimization Applications*. San Francisco: Holden-Day.

Rubino, G. 1998. Network reliability evaluation. In *The state-of-the art in performance modeling and simulation*, ed. K. Bagchi and J. Walrand. Gordon and Breach Books. Chapter 11.

ed. G. Rubino and B. Tuffin. 2009. *Rare Event Simulation using Monte Carlo Methods*. Wiley.

Rubinstein, R. Y. 1981. *Simulation and the Monte Carlo Method*. New York, NY: John Wiley.

Rubinstein, R. Y and R. Marcus. 1985. Efficiency of multivariate control variates in Monte Carlo simulation. *Operations Research*, 33:661–667.

Rubinstein, R. Y. and A. Shapiro. 1993. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*. New York: Wiley.

Rukhin, A., J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. 2001. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST special publication 800-22, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA. See `http://csrc.nist.gov/rng/`.

Ruppert, R. 1988. Efficient estimators from a slowly converging Robbins-Monro process. Technical Report No. 781, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York.

ed. A. Ruszczyński and A. Shapiro. 2003. *Stochastic Programming*. Handbooks in Operations Research and Management Science, Amsterdam, The Netherlands: Elsevier.

Rydberg, T. H. 1997. The normal inverse Gaussian Lévy process: Simulation and approximation. *Communications in Statistics: Stochastic Models*, 13(4):887–910.

Sadowsky, J. S. 1991. Large deviations and efficient simulation of excessive backlogs in a $GI/G/m$ queue. *IEEE Transactions on Automatic Control*, AC-36:1383–1394.

Salmon, J. K, M. A. Moraes, R. O. Dror, and D. E. Shaw. 2011. Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 16:1–16:12, New York. Association for Computing Machinery.

Samorodnitsky, G. and M. L. Taqqu. 1994. *Non-Gaussian Stable Processes*. Chapman and Hall.

Sargent, R. G. 2001. Some approaches and paradigms for verifying and validating simulation models. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 106–114. IEEE Press.

Sato, K.-I. 1999. *Lévy Processes and Infinitely Divisible Distributions*. Cambridge, UK: Cambridge University Press.

Scarsini, M. 1984. On measures of concordance. *Stochastica*, 8:201–218.

Schervish, M. J. 1995. *Theory of Statistics*. New York, NY: Springer-Verlag.

Schmeiser, B. 1999. Advanced input modeling for simulation optimization. In *Proceedings of the 1999 Winter Simulation Conference*, 110–115. IEEE Press.

Schmeiser, B. W, M. R. Taafe, and J. Wang. 2001. Biased control-variate estimation. *IIE Transactions*, 33(3):219–228.

Schmid, W. C. and R. Schürer. 2005. MinT, the database for optimal (t, m, s)-net parameters. `http://mint.sbg.ac.at`.

Schoutens, W. 2003. *Lévy Processes in Finance: Pricing Financial Derivatives*. Wiley.

Schriber, T. J. 1974. *Simulation Using GPSS*. New York, NY: John Wiley.

Scott, D. W. 1979. On optimal and data-based histograms. *Biometrika*, 66:605–610.

Scott, D. W. 1985a. Averaged shifted histograms: Effective nonparametric density estimators in several dimensions. *The Annals of Statistics*, 13(3):1024–1040.

Scott, D. W. 1985b. Frequency polygons. *Journal of the American Statistical Association*, 80:348–354.

Scott, D. W. 2004. Multivariate density estimation and visualization. In *Handbook of Computational Statistics*, ed. J. E. Gentle, W. Haerdle, and Y. Mori, 517–538. Berlin: Springer-Verlag. Chapter III.4.

Scott, D. W. 2015. *Multivariate Density Estimation*. second ed. Wiley.

Serfling, R. J. 1980. *Approximation Theorems for Mathematical Statistics*. New York, NY: Wiley.

Seshadri, V. 1993. *The Inverse Gaussian Distribution*. Clarendon Press.

Shannon, R. E. 1998. Introduction to the art and science of simulation. In *Proceedings of the 1998 Winter Simulation Conference*, 7–14. IEEE Press.

Shao, J and D. Tu. 1995. *The Jackknife and Bootstrap*. New York, NY: Springer.

Shapiro, A. 2003. Monte Carlo sampling methods. In *Stochastic Programming*, ed. A. Ruszczyński and A. Shapiro, Handbooks in Operations Research and Management Science, 353–425. Amsterdam, The Netherlands: Elsevier. Chapter 6.

Shapiro, A., D. Dentcheva, and A. Ruszczyński. 2014. *Lecture Notes on Stochastic Programming: Modeling and Theory*. Second ed. Handbooks in Operations Research and Management Science, Philadelphia: Society for Industrial and Applied Mathematics.

Shapiro, S. S., M. B. Wilk, and H. J. Chen. 1968. A comparative study of various tests of normality. *Journal of the American Statistical Association*, 63:1343–1372.

Sigman, K. and R. W. Wolff. 1993. A review of regenerative processes. *SIAM Review*, 35(2):269–288.

Silverman, B. 1986. *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall.

Simard, R and P. L'Ecuyer. 2011. Computing the two-sided Kolmogorov-Smirnov distribution. *Journal of Statistical Software*, 39(11):1–18. URL is `http://www.jstatsoft.org/v39/i11`.

SimPy, T. 2020. *SimPy Documentation*. `https://simpy.readthedocs.io/en/latest/`.

Sivakumar, A, C. R. Bhat, and G. Ökten. 2005. Simulation estimation of mixed discrete choice models with the use of randomized quasi-Monte Carlo sequences: A comparative study. *Transportation Research Record*, 1921:112–122.

Sklar, A. 1959. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231.

Sloan, I. H and S. Joe. 1994. *Lattice Methods for Multiple Integration*. Oxford: Clarendon Press.

Sobol', I. M. 1967. The distribution of points in a cube and the approximate evaluation of integrals. *U.S.S.R. Comput. Math. and Math. Phys.*, 7(4):86–112.

Sobol', I. M. 1994. *A Primer for the Monte Carlo Method*. Boca Raton, Florida: CRC Press. First published in Russian in 1968.

Song, W. T and B. W. Schmeiser. 1995. Optimal mean-squared error batch sizes. *Management Science*, 41(1):110–123.

Srikant, R. and W. Whitt. 1999. Variance reduction in simulations of loss models. *Operations Research*, 47(4):509–523.

Stanfield, P. M., J. R. W. G. A. Mirka, N. F. Glasscock, J. P. Psihogios, and J. R. Davis. 1996. Multivariate input modeling with Johnson distributions. In *Proceedings of the 1996 Winter Simulation Conference*, 1457–1464, Piscataway, New Jersey. IEEE Press.

Staum, J. 2009. Monte Carlo computation in finance. In *Monte Carlo and Quasi-Monte Carlo Methods 2008*, ed. P. L'Ecuyer and A. B. Owen, 19–44, Berlin. Springer-Verlag.

Stefanov, A., N. Gisin, O. Guinnard, L. Guinnard, and H. Zbinden. 2000. Optical quantum random number generator. *Journal of Modern Optics*, 47(4):595–598.

Steiger, N. M., C. Alexopoulos, D. Goldsman, E. K. Lada, J. R. Wilson, and F. Zouaoui. 2002. ASAP2: An improved batch means procedure for simulation output analysis. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 336–344. IEEE Press.

Steiger, N. M. and J. R. Wilson. 2000. Experimental performance evaluation of batch means procedures for simulation output analysis. In *Proceedings of the 2000 Winter Simulation Conference*, 627–636, Piscataway, New Jersey. IEEE Press.

Steiger, N. M. and J. R. Wilson. 2001a. Convergence properties of the batch means method for simulation output analysis. manuscript.

Steiger, N. M. and J. R. Wilson. 2001b. An improved batch means procedure for simulation output analysis. manuscript.

Stein, M. 1987. Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29(2):143–151.

Stephan, F. F. 1945. The expected value and variance of the reciprocal and other negative powers of a positive bernoullian variate. *Annals of Mathematical Statistics*, 16:50–61.

Stephens, M. A. 1970. Use of the Kolmogorov-Smirnov, Cramér-Von Mises and related statistics without extensive tables. *Journal of the Royal Statistical Society, Series B*, 33(1):115–122.

Stephens, M. S. 1986. Tests based on EDF statistics. In *Goodness-of-Fit Techniques*, ed. R. B. D'Agostino and M. S. Stephens. New York and Basel: Marcel Dekker.

Suematsu, C., N. Namekata, I. Shimada, and S. Inoue. 2007. Generation of physical random numbers by means of photon counting. *Electronics and Communications in Japan, Part 3*, 90(2):1–8.

Sun, L. and A. Erath. 2015. A Bayesian network approach for population synthesis. *Transportation Research Part C*, 61:49–62.

Sun, L., A. Erath, and M. Cai. 2018. A hierarchical mixture modeling framework for population synthesis. *Transportation Research Part B*, 114:199–212.

Suri, R. 1987. Infinitesimal perturbation analysis for general discrete event systems. *Journal of the ACM*, 34(3):686–717.

Sutton, R. S and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. second ed. Cambridge, MA, USA: MIT Press.

Swain, J. J., S. Venkatraman, and J. R. Wilson. 1988. Least-squares estimation of distribution functions in Johnson's translation system. *Journal of Statistical Computation and Simulation*, 29:271–297.

Tausworthe, R. C. 1965. Random numbers generated by linear recurrence modulo two. *Mathematics of Computation*, 19:201–209.

Taylor, H. M and S. Karlin. 1998. *An Introduction to Stochastic Modeling*. third ed. San Diego: Academic Press.

Tezuka, S. 1995. *Uniform Random Numbers: Theory and Practice*. Kluwer Academic.

Tezuka, S and H. Faure. 2002. i-binomial scrambling of digital nets and sequences. Technical report, IBM Research, Tokyo Research Laboratory.

Tezuka, S. and P. L'Ecuyer. 1991. Efficient and portable combined Tausworthe random number generators. *ACM Transactions on Modeling and Computer Simulation*, 1(2):99–112.

Tezuka, S., P. L'Ecuyer, and R. Couture. 1993. On the add-with-carry and subtract-with-borrow random number generators. *ACM Transactions of Modeling and Computer Simulation*, 3(4):315–331.

Tokol, G., D. Goldsman, D. H. Ockerman, and J. J. Swain. 1998. Standardized time series L_p-norm variance estimators for simulation. *Management Science*, 44(2):234–245.

Tong, Y. L. 1990. *The Multivariate Normal Distribution*. New York, NY: Springer-Verlag.

Tootill, J. P. R, W. D. Robinson, and D. J. Eagle. 1973. An asymptotically random Tausworthe sequence. *Journal of the ACM*, 20:469–481.

Train, K. 2003. *Discrete Choice Methods with Simulation*. New York, USA: Cambridge University Press.

Vattulainen, I, T. Ala-Nissila, and K. Kankaala. 1995. Physical models as tests of randomness. *Physical Review E*, 52(3):3205–3213.

Venkatraman, S. and J. R. Wilson. 1986. The efficiency of control variates in multiresponse simulation. *Operations Research Letters*, 5:37–42.

Villén-Altamirano, M., A. Martinez-Marrón, J. Gamo, and F. Fernández-Cuesta. 1994. Enhancement of the accelerated simulation method restart by considering multiple thresholds. In *Proceedings of the 14th International Teletraffic Congress*, 797–810. Elsevier Science.

Villén-Altamirano, M. and J. Villén-Altamirano. 1991. Restart: A method for accelerating rare events simulations. In *Proceedings of the 13th International Teletraffic Congress*, 71–76. North-Holland.

Villén-Altamirano, M. and J. Villén-Altamirano. 1994. RESTART: A straightforward method for fast simulation of rare events. In *Proceedings of the 1994 Winter Simulation Conference*, 282–289. IEEE Press.

von Neumann, J. 1951. Various techniques used in connection with random digits. In *The Monte Carlo Method*, ed. A. S. Householder et al., volume 12, 36–38. National Bureau of Standards, Applied Mathematics Series.

Wagner, M. A. F and J. R. Wilson. 1995. Graphical interactive simulation input modeling with bivariate Bézier distributions. *ACM Transactions of Modeling and Computer Simulation*, 5:163–189.

Wagner, M. A. F. and J. R. Wilson. 1996. Using univariate Bézier distributions to model simulation input processes. *IIE Transactions*, 28:699–711.

Wakefield, J. C., A. E. Gelfand, and A. F. M. Smith. 1991. Efficient generation of random variates via the ratio-of-uniforms method. *Statistical Computing*, 1(2):129–133.

Walker, A. J. 1974. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronic Letters*, 10:127–128.

Walker, A. J. 1977. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3:253–256.

Wallace, C. S. 1976. Transformed rejection generators for the gamma and normal pseudorandom variables. *Australian Computer Journal*, 8:103–105.

Wang, D and A. Compagner. 1993. On the use of reducible polynomials as random number generators. *Mathematics of Computation*, 60:363–374.

Wang, X. and F. J. Hickernell. 2000. Randomized Halton sequences. *Mathematical and Computer Modelling*, 32:887–899.

Wang, X. and I. H. Sloan. 2008. Low discrepancy sequences in high dimensions: How well are their projections distributed. *Journal of Computational and Applied Mathematics*, 213(2):366–386.

Wang, X. and D. Xu. 2010. An inverse Gaussian process model for degradation data. *Technometrics*, 52(2):188–197.

Wegenkittl, S. and M. Matsumoto. 1999. Getting rid of correlations among pseudorandom numbers: Discarding versus tempering. *ACM Transactions on Modeling and Computer Simulation*, 9(3):282–294.

Welch, P. D. 1983. The statistical analysis of simulation results. In *The Computer Performance Modeling Handbook*, ed. S. Lavenberg, 268–328. San Diego, Cal.: Academic Press.

Weron, R. 2004. Computationally intensive value at risk calculations. In *Handbook of Computational Statistics*, ed. J. E. Gentle, W. Haerdle, and Y. Mori, 911–950. Berlin: Springer-Verlag. Chapter IV.1.

Whitt, W. 1976. Bivariate distributions with given marginals. *Annals of Statistics*, 4:1280–1289.

Whitt, W. 1989. Planning queueing simulations. *Management Science*, 35:1341–1366.

Whitt, W. 1991. The efficiency of one long run versus independent replications in steady-state simulation. *Management Science*, 37(6):645–666.

Whitt, W. 1999. Dynamic staffing in a telephone call center aiming to immediately answer all calls. *Operations Research Letters*, 24(5):205–212.

Willard, G. A. 1997. Calculating prices and sensitivities for path-dependent derivatives securities in multifactor models. *Journal of derivatives*, 5:45–61.

Wilson, J. R. 1983. Antithetic sampling with multivariate inputs. *American Journal of Mathematical and Management Sciences*, 3:121–144.

Wilson, J. R. 1984. Variance reduction techniques for digital simulation. *American Journal of Mathematical and Management Sciences*, 4:277–312.

Wilson, J. R. 1997. Modeling dependencies in stochastic simulation inputs. In *Proceedings of the 1997 Winter Simulation Conference*, 47–52. IEEE Press.

Wilson, J. R and A. A. B. Pritsker. 1984a. Experimental evaluation of variance reduction techniques for queueing simulation using generalized concomitant variables. *Management Science*, 30:1459–1472.

Wilson, J. R. and A. A. B. Pritsker. 1984b. Variance reduction in queueing simulation using generalized concomitant variables. *Journal of Statistical Computation and Simulation*, 19:129–153.

Wolff, R. W. 1989. *Stochastic Modeling and the Theory of Queues*. New York, NY: Prentice-Hall.

Wu, P.-C. 1997. Multiplicative, congruential random number generators with multiplier $\pm 2^{k_1} \pm 2^{k_2}$ and modulus $2^p - 1$. *ACM Transactions on Mathematical Software*, 23(2):255–265.

Yakowitz, S, J. E. Krimmel, and F. Szidarovszky. 1978. Weighted Monte Carlo integration. *SIAM Journal on Numerical Analysis*, 15:1289–1300.

Yakowitz, S., P. L'Ecuyer, and F. Vázquez-Abad. 2000. Global stochastic optimization with low-discrepancy point sets. *Operations Research*, 48(6):939–950.

Yang, W. and B. L. Nelson. 1992. Multivariate batch means and control variates. *Management Science*, 38(10):1415–1431.

Ye, Z.-S. and N. Chen. 2014. The inverse Gaussian process as a degradation model. *Technometrics*, 56(3):302–311.

Yin, M., M. Sheehan, S. Feygin, J.-F. Paiement, and A. Pozdnoukhov. 2018. A generative model of urban activities from cellular data. *IEEE Transactions on Intelligent Transportation Systems*, 19:1682–1696.

Index

Index