to take $R_t^{(k)} = \lfloor r_t \rfloor + B_t^{(k)}$, where the $\{B_t^{(k)}\}$ are independent Bernoulli random variables with parameter $r_t - \lfloor r_t \rfloor$. The variability in the total number of copies made can be reduced by letting the $\{B_t^{(k)}\}_{k=1}^N$ be such that the total number of resampled particles is fixed. It is also possible to combine bootstrap resampling and enrichment, as in the *Pruned-Enriched Rosenbluth* method of [18]. Chapter 9 further explores the merits of such "particle Monte Carlo" methods in which a sequential sampling scheme is combined with a resampling/splitting step.

5.10 NONLINEAR FILTERING FOR HIDDEN MARKOV MODELS

This section describes an application of SIS and SIR to nonlinear filtering. Many problems in engineering, applied sciences, statistics, and econometrics can be formulated as *hidden Markov models* (HMMs). In its simplest form, an HMM is a stochastic process $\{(X_t, Y_t)\}$, where X_t (which may be multidimensional) represents the *true* state of some system and Y_t represents the *observed* state of the system at a discrete time t. It is usually assumed that $\{X_t\}$ is a Markov chain, say with initial distribution $f(x_0)$ and one-step transition probabilities $f(x_t \,|\, x_{t-1})$. It is important to note that the actual state of the Markov chain remains *hidden*, hence the name HMM. All information about the system is conveyed by the process $\{Y_t\}$. We assume that, given $X_0, \ldots, X_t$, the observation Y_t depends only on X_t via some conditional pdf $f(y_t \,|\, x_t)$. Note that we have used here a Bayesian style of notation in which all (conditional) probability densities are represented by the *same symbol* f. We will use this notation throughout the rest of this section. We denote by $\mathbf{X}_{1:t} = (X_1, \ldots, X_t)$ and $\mathbf{Y}_{1:t} = (Y_1, \ldots, Y_t)$ the unobservable and observable sequences up to time t, respectively — and similarly for their lowercase equivalents.

The HMM is represented graphically in Figure 5.4. This is an example of a *Bayesian network*. The idea is that edges indicate the dependence structure between two variables. For example, given the states $X_1, \ldots, X_t$, the random variable Y_t is conditionally independent of $X_1, \ldots, X_{t-1}$, because there is no direct edge from Y_t to any of these variables. We thus have $f(y_t \,|\, \mathbf{x}_{1:t}) = f(y_t \,|\, x_t)$, and more generally

$$f(\mathbf{y}_{1:t} \,|\, \mathbf{x}_{1:t}) = f(y_1 \,|\, x_1) \cdots f(y_t \,|\, x_t) = f(\mathbf{y}_{1:t-1} \,|\, \mathbf{x}_{1:t-1})\, f(y_t \,|\, x_t)\,. \tag{5.86}$$

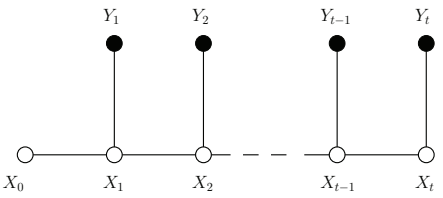

Figure 5.4: A graphical representation of the HMM.

Summarizing, we have

$$\begin{aligned} X_t &\sim f(x_t \,|\, x_{t-1}) \qquad \text{(state equation)}, \\ Y_t &\sim f(y_t \,|\, x_t) \qquad \text{(observation equation)}. \end{aligned} \tag{5.87}$$

■ **EXAMPLE 5.18**

An example of (5.87) is the following popular model:

$$\begin{aligned} X_t &= \varphi_1(X_{t-1}) + \varepsilon_{1t} \,, \\ Y_t &= \varphi_2(X_t) + \varepsilon_{2t} \,, \end{aligned} \tag{5.88}$$

where $\varphi_1(\cdot)$ and $\varphi_2(\cdot)$ are given vector functions and ε_{1t} and ε_{2t} are independent d-dimensional Gaussian random vectors with zero mean and covariance matrices C_1 and C_2, respectively.

Our goal, based on an outcome $\mathbf{y}_{1:t}$ of $\mathbf{Y}_{1:t}$, is to determine, or estimate *on-line*, the following quantities:

1. The joint conditional pdf $f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$ and, as a special case, the marginal conditional pdf $f(x_t \,|\, \mathbf{y}_{1:t})$, which is called the *filtering* pdf.

2. The expected performance

$$\ell = \mathbb{E}_{f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})}[H(\mathbf{X}_{1:t})] = \int H(\mathbf{x}_{1:t})\, f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})\, \mathrm{d}\mathbf{x}_{1:t} \,. \tag{5.89}$$

It is well known [11] that the conditional pdf $f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$ or the filtering pdf $f(x_t \,|\, \mathbf{y}_{1:t})$ can be found explicitly only for the following two particular cases:

(a) When $\varphi_1(x)$ and $\varphi_2(x)$ in (5.88) are linear, the filtering pdf is obtained from the celebrated *Kalman filter*. The Kalman filter is explained in Section A.6 of the Appendix.

(b) When the $\{x_t\}$ can take only a finite number, say K, of possible values, for example, as in binary signals, we can calculate $f(x_t \,|\, \mathbf{y}_{1:t})$ efficiently with complexity $\mathcal{O}(K^2\, t)$. Applications can be found in digital communication and speech recognition; see, for example, Section A.7 of the Appendix.

Because the target pdf $f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$ for the general state space model (5.87) is difficult to obtain exactly, one needs to resort to Monte Carlo methods. To put the nonlinear filtering problem in the sequential Monte Carlo framework of Section 5.8, we first write $f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:n})$ in sequential form, similar to (5.78). A natural candidate for the "auxiliary" pdf at time t is the conditional pdf $f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$. That is, only the observations up to time t are used. By Bayes' rule we have for each $t = 1, \ldots, n$,

$$\begin{aligned} &\frac{f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})}{f(\mathbf{x}_{1:t-1} \,|\, \mathbf{y}_{1:t-1})} \\ &\quad = \frac{f(\mathbf{y}_{1:t} \,|\, \mathbf{x}_{1:t}) f(\mathbf{x}_{1:t})}{f(\mathbf{y}_{1:t})} \frac{f(\mathbf{y}_{1:t-1})}{f(\mathbf{y}_{1:t-1} \,|\, \mathbf{x}_{1:t-1}) f(\mathbf{x}_{1:t-1})} \\ &\quad = \frac{f(\mathbf{y}_{1:t-1} \,|\, \mathbf{x}_{1:t-1})\, f(y_t \,|\, x_t)\, f(\mathbf{x}_{1:t-1})\, f(x_t \,|\, x_{t-1})}{f(\mathbf{y}_{1:t-1})\, f(y_t \,|\, \mathbf{y}_{1:t-1})} \frac{f(\mathbf{y}_{1:t-1})}{f(\mathbf{y}_{1:t-1} \,|\, \mathbf{x}_{1:t-1}) f(\mathbf{x}_{1:t-1})} \\ &\quad = \frac{f(y_t \,|\, x_t)\, f(x_t \,|\, x_{t-1})}{f(y_t \,|\, \mathbf{y}_{1:t-1})} \,, \end{aligned} \tag{5.90}$$

where we have also used (5.86) and the fact that $f(x_t \,|\, \mathbf{x}_{1:t-1}) = f(x_t \,|\, x_{t-1})$, $t = 1, 2, \ldots$, by the Markov property.

This result is of little use for an exact calculation of $f(\mathbf{x}_{1:n} \,|\, \mathbf{y}_{1:n})$, since it requires computation of $f(y_t \,|\, \mathbf{y}_{1:t-1})$, which involves the evaluation of complicated integrals. However, if both functions (pdfs) $f(x_t \,|\, x_{t-1})$ and $f(y_t \,|\, x_t)$ can be evaluated exactly (which is a reasonable assumption), then SIS can be used to approximately simulate from $f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$ as follows: Let $g_t(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$ be the importance sampling pdf. We assume that, similar to (5.73), we can write $g_t(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$ recursively as

$$g_t(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t}) = g_0(x_0 \,|\, y_0) \prod_{s=1}^{t} g_s(x_s \,|\, \mathbf{x}_{s-1}, \mathbf{y}_s) \,. \tag{5.91}$$

Then, by analogy to (5.76), and using (5.90) (dropping the normalization constant $f(y_t \,|\, \mathbf{y}_{1:t-1})$), we can write the importance weight W_t of a path $\mathbf{x}_{1:t}$ generated from $g_t(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$ recursively as

$$W_t = W_{t-1} \frac{f(y_t \,|\, x_t)\, f(x_t \,|\, x_{t-1})}{g_t(x_t \,|\, \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})} = W_{t-1}\, u_t \,. \tag{5.92}$$

A natural choice for the importance sampling pdf is

$$g_t(x_t \,|\, \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t}) = f(x_t \,|\, x_{t-1}) \,, \tag{5.93}$$

in which case the incremental weight simplifies to

$$u_t = f(y_t \,|\, x_t). \tag{5.94}$$

With this choice of sampling distribution, we are simply guessing the values of the hidden process $\{X_t\}$ without paying attention to the observed values.

Once the importance sampling density is chosen, sampling from the target pdf $f(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})$ proceeds as described in Section 5.8. For more details, the interested reader is referred to [11], [27], and [32].

■ EXAMPLE 5.19 Bearings-Only Tracking

Suppose that we want to track an object (e.g., a submarine) via a radar device that only reports the *angle* to the object (see Figure 5.5). In addition, the angle measurements are noisy. We assume that the initial position and velocity are known and that the object moves at a constant speed.

Let $X_t = (p_{1t}, v_{1t}, p_{2t}, v_{2t})^\top$ be the vector of positions and (discrete) velocities of the target object at time $t = 0, 1, 2, \ldots$, and let Y_t be the measured angle. The problem is to track the unknown state of the object X_t based on the measurements $\{Y_t\}$ and the initial conditions.

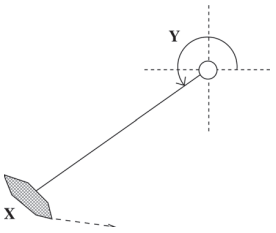

Figure 5.5: Track the object via noisy measurements of the angle.

The process $(X_t, Y_t), t = 0, 1, 2, \ldots$ is described by the following system:

$$\begin{aligned} X_t &= A\, X_{t-1} + \varepsilon_{1t} \\ Y_t &= \arctan(p_{1t}, p_{2t}) + \varepsilon_{2t}\ . \end{aligned}$$

Here $\arctan(u, v)$ denotes the four-quadrant arc-tangent, that is, $\arctan(v/u) + c$, where c is either 0, $\pm\pi$, or $\pm\pi/2$, depending on the quadrant in which (u, v) lies. The random noise vectors $\{\varepsilon_{1t}\}$ are assumed to be $\mathsf{N}(\mathbf{0}, C_1)$ distributed, and the measurement noise ε_{2t} is $\mathsf{N}(0, \sigma_2^2)$ distributed. All noise variables are independent of each other. The matrix A is given by

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}\ .$$

The problem is to find the conditional pdf $f(x_t \,|\, \mathbf{y}_{1:t})$ and, in particular, the expected system state $\mathbb{E}[X_t \,|\, \mathbf{y}_{1:t}]$.

We indicate how this problem can be solved via SIS. Using (5.93) for the sampling distribution means simply that X_t is drawn from a $\mathsf{N}(Ax_{t-1}, C_1)$ distribution. As a consequence of (5.94), the incremental weight, $u_t = f(y_t \,|\, x_t)$, is equal to the value at y_t of the normal pdf with mean $\arctan(p_{1t}, p_{2t})$ and variance σ_2^2. The corresponding SIS procedure is summarized below. Note that the parallel SIS procedure is given, in which the $\{W_t^{(k)}\}$ and $\{X_t^{(k)}\}$ are computed at the same time by running N parallel processes.

Algorithm 5.10.1: SIS Procedure for Bearings-Only Tracking

input : Sample size N, matrices A and C_1, parameter σ_2, and distribution of the starting state.

output: Estimator $\widehat{x}_t$ of the expected position $\mathbb{E}[X_t \,|\, \mathbf{y}_{1:t}]$.

Initialize $X_0^{(k)}$ and set $W_0^{(k)} \leftarrow 0$, $k = 1, \ldots, N$
for $t = 1$ **to** n **do**
  for $k = 1$ **to** N **do**
    $X_t^{(k)} \sim \mathsf{N}(AX_{t-1}^{(k)}, C_1)$
    $u_t \leftarrow \frac{1}{\sigma_2\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{y_t - \arctan(p_{1t}, p_{2t})}{\sigma_2}\right)^2\right\}$
    $W_t^{(k)} \leftarrow W_{t-1}^{(k)} u_t$
$\widehat{x}_t \leftarrow (\sum_{k=1}^N W_t^{(k)} X_t^{(k)}) / \sum_{k=1}^N W_t^{(k)}$
return $\widehat{x}_t$

As a numerical illustration, consider the case where $\sigma_2 = 0.005$ and

$$C_1 = \sigma_1^2 \begin{pmatrix} 1/4 & 1/2 & 0 & 0 \\ 1/2 & 1 & 0 & 0 \\ 0 & 0 & 1/4 & 1/2 \\ 0 & 0 & 1/2 & 1 \end{pmatrix},$$

with $\sigma_1 = 0.001$. Let $X_0 \sim \mathsf{N}(\boldsymbol{\mu}_0, \Sigma_0)$, with $\boldsymbol{\mu}_0 = (-0.05, 0.001, 0.2, -0.055)^\top$, and

$$\Sigma_0 = 0.1^2 \begin{pmatrix} 0.5^2 & 0 & 0 & 0 \\ 0 & 0.005^2 & 0 & 0 \\ 0 & 0 & 0.3^2 & 0 \\ 0 & 0 & 0 & 0.01^2 \end{pmatrix} .$$

The left panel in Figure 5.6 shows how the estimated process $\{\widehat{x}_t\}$, obtained via SIS, tracks the actual process $\{X_t\}$ over $n = 25$ time steps, using a sample size of $N = 10,000$. In the right panel the result of SIR with bootstrap resampling (Algorithm 5.9.2) is shown, for the same sample size as in the SIS case. We see that the actual position is tracked over time more accurately.

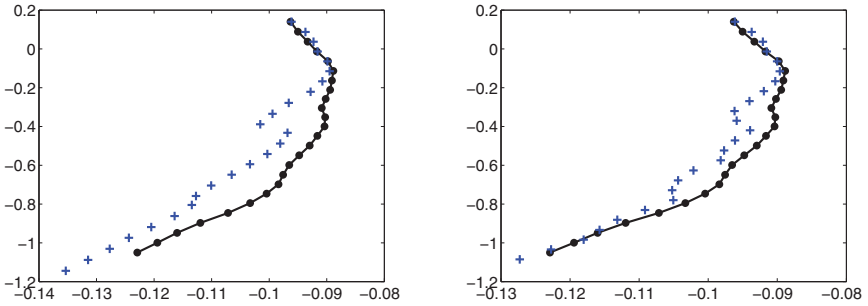

Figure 5.6: Comparison of the performance of the SIS (left) and SIR (right) algorithms for the bearings-only tracking problem, using a sample size of $N = 10^4$ over 25 time steps.

For both SIS and SIR, as time increases, the tracking rapidly becomes more unstable. This is a consequence of the degeneracy of the likelihood ratio. Indeed, after a few iterations, only a handful of samples contain the majority of the importance weight. This yields high variability between many runs and provides less reliable estimates. The resampling step mitigates some of this degeneracy. Several other heuristic resampling techniques have been proposed (e.g., see [11]).

5.11 TRANSFORM LIKELIHOOD RATIO METHOD

The *transform likelihood ratio* (TLR) method is a simple, convenient, and *unifying* way of constructing efficient importance sampling estimators. To motivate the TLR method, we consider the estimation of

$$\ell = \mathbb{E}[H(\mathbf{X})] , \tag{5.95}$$

where $\mathbf{X} \sim f(\mathbf{x})$. Consider first the case where $\mathbf{X}$ is one-dimensional (we write X instead of $\mathbf{X}$). Let F be the cdf of X. According to the IT method, we can write

$$X = F^{-1}(U) , \tag{5.96}$$

where $U \sim \mathsf{U}(0,1)$ and F^{-1} is the inverse of the cdf F. Substituting $X = F^{-1}(U)$ into $\ell = \mathbb{E}[H(X)]$, we obtain

$$\ell = \mathbb{E}[H(F^{-1}(U))] = \mathbb{E}[\widetilde{H}(U)] .$$

Note that in contrast to $\ell = \mathbb{E}[H(X)]$, where the expectation is taken with respect to $f(x)$, in $\ell = [\widetilde{H}(U)]$, the expectation is taken with respect to the uniform $\mathsf{U}(0,1)$ distribution. The extension to the multidimensional case is simple.

Let $h(u;\nu)$ be another density on $(0,1)$, parameterized by some reference parameter ν, with $h(u;\nu) > 0$ for all $0 \leqslant u \leqslant 1$ (note that u is a variable and not a parameter). An example is the $\mathsf{Beta}(\nu,1)$ distribution, with density

$$h(u;\nu) = \nu\, u^{\nu-1}, \quad u \in (0,1)\ ,$$

with $\nu > 0$, or the $\mathsf{Beta}(1,\nu)$ distribution, with density

$$h(u;\nu) = \nu\,(1-u)^{\nu-1}, \quad u \in (0,1)\ .$$

Using $\mathsf{Beta}(1,\nu)$ as the importance sampling pdf, we can write ℓ as

$$\ell = \mathbb{E}_\nu[\widetilde{H}(U)\,\widetilde{W}(U;\nu)]\ , \tag{5.97}$$

where $U \sim h(u;\nu)$, and

$$\widetilde{W}(U;\nu) = \frac{1}{h(U;\nu)} \tag{5.98}$$

is the likelihood ratio. The likelihood ratio estimator of ℓ is given by

$$\widehat{\ell} = N^{-1}\sum_{k=1}^{N} \widetilde{H}(U_k)\,\widetilde{W}(U_k;\nu)\ , \tag{5.99}$$

where $U_1,\ldots,U_N$ is a random sample from $h(u;\nu)$, We call (5.99) the *inverse-transform likelihood ratio* (ITLR) estimator; see Kroese and Rubinstein [23].

Suppose, for example, that $X \sim \mathsf{Weib}(\alpha,\lambda)$, which is to say that X has the density

$$f(x;\alpha,\lambda) = \alpha\lambda(\lambda x)^{\alpha-1}\mathrm{e}^{-(\lambda x)^\alpha}. \tag{5.100}$$

Note that a Weibull random variable can be generated using the transformation

$$X = \lambda^{-1}\, Z^{1/\alpha}, \tag{5.101}$$

where the random variable Z has an $\mathsf{Exp}(1)$ distribution. Applying the IT method, we obtain

$$X = F^{-1}(U) = \lambda^{-1}(-\ln(1-U))^{1/\alpha}, \tag{5.102}$$

and $\widetilde{H}(U_i)\,\widetilde{W}(U_i;\nu)$ in (5.99) reduces to $H(\lambda^{-1}\,(-\ln(1-U_i))^{1/\alpha})/h(U_i;\nu)$.

The TLR method is a natural extension of the ITLR method. It comprises two steps. The first is a simple *change of variable* step, and the second involves an application of the SLR technique to the transformed pdf.

To apply the first step, we simply write $\mathbf{X}$ as a function of another random vector, say as

$$\mathbf{X} = G(\mathbf{Z})\ . \tag{5.103}$$

If we define

$$\widetilde{H}(\mathbf{Z}) = H(G(\mathbf{Z}))\ ,$$

then estimating (5.95) is equivalent to estimating

$$\ell = \mathbb{E}[\widetilde{H}(\mathbf{Z})]\ . \tag{5.104}$$

Note that the expectations in (5.95) and (5.104) are taken with respect to the original density of $\mathbf{X}$ and the transformed density of $\mathbf{Z}$. As an example, consider again a one-dimensional case and let $X \sim \mathsf{Weib}(\alpha, \lambda)$. Recalling (5.101), we have $\widetilde{H}(Z) = H(\lambda^{-1} Z^{1/\alpha})$, and thus $\ell = \mathbb{E}[H(\lambda^{-1} Z^{1/\alpha})]$.

To apply the second step, we assume that $\mathbf{Z}$ has a density $h(\mathbf{z}; \boldsymbol{\theta})$ in some class of densities $\{h(\mathbf{z}; \boldsymbol{\eta})\}$. Then we can seek to estimate ℓ efficiently via importance sampling, for example, using the standard likelihood ratio method. In particular, by analogy to (5.58), we obtain the following estimator:

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^{N} \widetilde{H}(\mathbf{Z}_k)\, \widetilde{W}(\mathbf{Z}_k; \boldsymbol{\theta}, \boldsymbol{\eta}) \;, \tag{5.105}$$

where

$$\widetilde{W}(\mathbf{Z}_k; \boldsymbol{\theta}, \boldsymbol{\eta}) = \frac{h(\mathbf{Z}_k; \boldsymbol{\theta})}{h(\mathbf{Z}_k; \boldsymbol{\eta})}$$

and $\mathbf{Z}_k \sim h(\mathbf{z}; \boldsymbol{\eta})$. We will call the SLR estimator (5.105) based on the transformation (5.103), the *TLR estimator*. As an example, consider again the $\mathsf{Weib}(\alpha, \lambda)$ case. Using (5.101), we could take $h(z; \eta) = \eta\, \mathrm{e}^{-\eta z}$ as the sampling pdf, with $\eta = \theta = 1$ as the nominal parameter. Hence, in this case, $\widehat{\ell}$ in (5.105) reduces to

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^{N} \widetilde{H}\left(\lambda^{-1} Z_k^{1/\alpha}\right) \widetilde{W}(Z_k; \theta, \eta) \;, \tag{5.106}$$

with

$$\widetilde{W}(Z_k; \theta, \eta) = \frac{h(Z_k; \theta)}{h(Z_k; \eta)} = \frac{\theta\, \mathrm{e}^{-\theta Z_k}}{\eta\, \mathrm{e}^{-\eta Z_k}}$$

and $Z_k \sim \mathsf{Exp}(\eta)$.

To find the optimal parameter vector $\boldsymbol{\eta}^*$ of the TLR estimator (5.105), we can solve, by analogy to (5.63), the CE program

$$\max_{\boldsymbol{\eta}} D(\boldsymbol{\eta}) = \max_{\boldsymbol{\eta}} \mathbb{E}_{\boldsymbol{\tau}} \left[\widetilde{H}(\mathbf{Z})\, \widetilde{W}(\mathbf{Z}; \boldsymbol{\theta}, \boldsymbol{\tau}) \ln h(\mathbf{Z}; \boldsymbol{\eta}) \right] \;, \tag{5.107}$$

and similarly for the stochastic counterpart of (5.107).

Since $\mathbf{Z}$ can be distributed quite arbitrarily, its distribution is typically chosen from an exponential family of distributions (see Section A.3 of the Appendix), for which the optimal solution $\boldsymbol{\eta}^*$ of (5.107) can be obtained analytically in a convenient and simple form. Below we present the TLR algorithm for estimating $\ell = \mathbb{E}_f[H(\mathbf{X})]$, assuming that $\mathbf{X}$ is a random vector with independent, continuously distributed components. The key is to find a transformation function G such that $\mathbf{X} = G(\mathbf{Z})$, with $\mathbf{Z} \sim h(\mathbf{z}; \boldsymbol{\theta}$. For example, we can take $\mathbf{Z}$ with all components being iid and distributed according to an exponential family (e.g., $\mathsf{Exp}(1)$).

Algorithm 5.11.1: Transform likelihood ratio (TLR) method

input : Function G such that $\mathbf{X} = G(\mathbf{Z})$, with $\mathbf{Z} \sim h(\mathbf{z}; \boldsymbol{\theta})$. Sample sizes N and N_2. Initial parameter $\boldsymbol{\tau}$.

output: Estimator $\widehat{\ell}$ of $\ell = \mathbb{E}[H(\mathbf{X})] = \mathbb{E}[\widetilde{H}(\mathbf{Z})]$.

1 Generate a random sample $\mathbf{Z}_1, \ldots, \mathbf{Z}_N$ from $h(\cdot; \boldsymbol{\tau})$.

2 Solve the stochastic counterpart of the program (5.107) (for a one-parameter exponential family parameterized by the mean, apply directly the analytic solution (5.68). Iterate if necessary. Denote the solution by $\widehat{\boldsymbol{\eta}}$.

3 Generate a (larger) random sample $\mathbf{Z}_1, \ldots, \mathbf{Z}_{N_1}$ from $h(\cdot; \widehat{\boldsymbol{\eta}})$ and estimate $\ell = \mathbb{E}[H(G(\mathbf{Z}))]$ via the TLR estimator (5.105), taking $\boldsymbol{\eta} = \widehat{\boldsymbol{\eta}}$.

4 **return** $\widehat{\ell}$

The TLR Algorithm 5.11.1 ensures that as soon as the transformation $\mathbf{X} = G(\mathbf{Z})$ is chosen, one can estimate ℓ using the TLR estimator (5.105) instead of the SLR estimator (5.58). Although the accuracy of both estimators (5.105) and (5.58) is the same (Rubinstein and Kroese [35]), the advantage of the former is its universality and its ability to avoid the computational burden while directly delivering the analytical solution of the stochastic counterpart of the program (5.107).

5.12 PREVENTING THE DEGENERACY OF IMPORTANCE SAMPLING

In this section, we show how to prevent the degeneracy of importance sampling estimators via the *screening method*. The degeneracy of likelihood ratios in high-dimensional Monte Carlo simulation problems is one of the central topics in Monte Carlo simulation; see also Remark 5.7.1.

To motivate the screening method, consider again Example 5.15 and observe that only the first two importance sampling parameters of the five-dimensional vector $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{v}_2, \widehat{v}_3, \widehat{v}_4, \widehat{v}_5)$ are substantially different from those in the nominal parameter vector $\mathbf{u} = (u_1, u_2, u_3, u_4, u_5)$. The reason is that the partial derivatives of ℓ with respect to u_1 and u_2 are significantly larger than those with respect to u_3, u_4, and u_5. We call such elements u_1 and u_2 *bottleneck elements*. Based on this observation, we could use instead of the importance sampling vector $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{v}_2, \widehat{v}_3, \widehat{v}_4, \widehat{v}_5)$ the vector $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{v}_2, u_3, u_4, u_5)$, reducing the number of importance sampling parameters from five to two. This not only has computational advantages — we would then solve a two-dimensional variance or CE minimization program instead of a five-dimensional one — but also leads to further variance reduction, since the likelihood ratio term W with two product terms is less "noisy" than the one with five product terms. Bottleneck elements can be identified as those elements i that have the largest *relative perturbation* $\delta_i = (\widehat{v}_i - u_i)/u_i$.

Algorithm 5.12.1 presents the two-stage screening algorithm for estimating

$$\ell = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})] = \int H(\mathbf{x}) f(\mathbf{x}; \mathbf{u}) \, \mathrm{d}\mathbf{x} \,,$$

based on CE, and denoted as CE-SCR. Its VM counterpart, the VM-SCR algorithm, is similar. For simplicity, we assume that the components of $\mathbf{X}$ are independent and that each component is distributed according to a one-dimensional exponential family that is parameterized by the mean — the dependent case could be treated

similarly. Moreover, $H(\mathbf{x})$ is assumed to be a monotonically increasing function in each component of $\mathbf{x}$. A consequence of the assumptions above is that the parameter vector $\mathbf{v}$ has dimension n.

At the first stage of the algorithm (Lines 1–9), we identify and estimate the bottleneck parameters, leaving the nonbottleneck parameters as they are, and at the second stage (Lines 10–12), we compute and return the importance sampling estimator

$$\widehat{\ell}_B = \frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k)\, W_B(\mathbf{X}_{kB}; \widehat{\mathbf{v}}_B) \,, \tag{5.108}$$

where $\mathbf{X}_{kB}$ is the subvector of bottleneck elements of $\mathbf{X}_k$ and W_B is the corresponding likelihood ratio. Note that, by analogy to Proposition A.4.2 in the Appendix, if each component of the random vector $\mathbf{X}$ is from a one-parameter exponential family parameterized by the mean and if $H(\mathbf{x})$ is a monotonically increasing function in each component of $\mathbf{x}$, then each element of the CE optimal parameter $\mathbf{v}^*$ is at least as large as the corresponding one of $\mathbf{u}$. We leave the proof as an exercise for the reader.

Algorithm 5.12.1: CE-SCR Two-Stage Screening Algorithm

input : Sample size N, performance function H, tolerance δ, number of repetitions d.
output: Estimator $\widehat{\ell}_B$ of the expected performance $\mathbb{E}[H(\mathbf{X})]$.

1 Initialize $B \leftarrow \{1, \ldots, n\}$.
2 **for** $t = 1$ **to** d **do**
3 Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{u})$.
4 **for** $i = 1$ **to** n **do**
5 $\widehat{v}_i \leftarrow \frac{\sum_{k=1}^{N} H(\mathbf{X}_k)\, X_{ki}}{\sum_{k=1}^{N} H(\mathbf{X}_k)}$
6 $\delta_i \leftarrow \frac{\widehat{v}_i - u_i}{u_i}$ `// calculate the relative perturbation`
7 **if** $\delta_i < \delta$ **then**
8 $\widehat{v}_i = u_i$
9 $B \leftarrow B \setminus \{i\}$ `// remove` i `from the bottleneck set`
10 Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x}; \widehat{\mathbf{v}})$
11 $\widehat{\ell}_B \leftarrow \frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k)\, W_B(\mathbf{X}_{kB}; \widehat{\mathbf{v}}_B)$.
12 **return** $\widehat{\ell}_B$

It is important to note the following:

1. As mentioned, under the present assumptions (independent components, each from a one-parameter exponential family parameterized by the mean, and $H(\mathbf{x})$ monotonically increasing in each component), the components of the $\mathbf{v}^*$ are at least as large as the corresponding elements of $\mathbf{u}$. Algorithm 5.12.1 takes this into account and always identifies all elements i corresponding to $\delta_i < 0$ as nonbottleneck ones.

2. Recall that Lines 3–9 are purposely performed d times. This allows us to better determine the nonbottleneck parameters, since it is likely that they will fluctuate around their nominal value u_i and therefore δ_i will become negative or very small in one of the replications.

In general, large-dimensional, complex simulation models contain both bottleneck and nonbottleneck parameters. The number of bottleneck parameters is typically smaller than the number of nonbottleneck parameters. Imagine a situation where the size (dimension) of the vector $\mathbf{u}$ is large, say 100, and the number of bottleneck elements is only about 10–15. Then, clearly, an importance sampling estimator based on bottleneck elements alone will not only be much more accurate than its standard importance sampling counterpart involving all 100 likelihood ratios (containing both bottleneck and nonbottleneck ones) but, in contrast to the latter, will not be degenerated.

The bottleneck phenomenon often occurs when one needs to estimate the probability of a nontypical event in the system, like a rare-event probability. This will be treated in Chapter 8. For example, if one observes a failure in a reliability system with highly reliable elements, then it is very likely that several elements (typically the less reliable ones) forming a minimal cut in the model all fail simultaneously. Another example is the estimation of a buffer overflow probability in a queueing network, that is, the probability that the total number of customers in all queues exceeds some large number. Again, if a buffer overflow occurs, it is quite likely that this has been caused by a buildup in the bottleneck queue, which is the most congested one in the network.

5.12.0.1 Numerical Results We next present numerical studies with Algorithm 5.12.1 for a generalization of the bridge system in Example 5.1, depicted in Figure 5.7. We will implement screening for both CE and VM methods. Recall that for the CE method the parameter vector $\widehat{\mathbf{v}}$ (and $\widehat{\mathbf{v}}_B$) can often be updated analytically, in particular when the sampling distribution comes from an exponential family. In contrast, for VM the updating typically involves a numerical procedure.

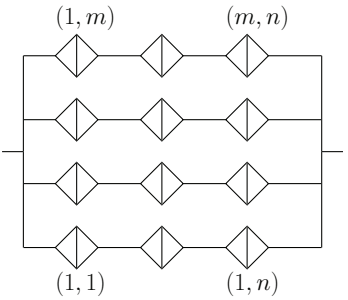

Figure 5.7: An $m \times n$ bridge system.

The system consists of $m \times n$ bridges arranged in a grid, and all bridges are of the form in Figure 5.1. Denote the lengths of the edges within the (i, j)-th bridge by $X_{ij1}, \ldots, X_{ij5}$. Then the length of the shortest path through bridge (i, j) is

$$\begin{aligned} Y_{ij} = \min\{&X_{ij1} + X_{ij4},\ X_{ij2} + X_{ij5}, \\ &X_{ij1} + X_{ij3} + X_{ij5},\ X_{ij2} + X_{ij3} + X_{ij4}\}\,. \end{aligned} \tag{5.109}$$

Suppose that we want to estimate the expected maximal length ℓ of the shortest paths in all rows, that is, $\ell = \mathbb{E}[H(\mathbf{X})]$, with

$$H(\mathbf{X}) = \max\{Y_{11} + \cdots + Y_{1n}, \ldots, Y_{m1} + \cdots + Y_{mn}\} . \tag{5.110}$$

In our numerical results, we assume that the components X_{ijk} of the random vector $\mathbf{X}$ are independent and that each component has a $\mathsf{Weib}(\alpha, u)$ distribution, which is to say X_{ijk} has the density

$$f(x; \alpha, u) = \alpha u (ux)^{\alpha-1} \mathrm{e}^{-(ux)^\alpha},$$

with $u = u_{ijk}$. Recall that such a Weibull random variable can be generated using the transformation $X = u^{-1} Z^{1/\alpha}$, where Z is a random variable distributed $\mathsf{Exp}(1)$. We also assume that only u is controllable, while α is fixed and equals 0.2. We purposely selected some elements of $\mathbf{u}$ to be bottleneck ones and set $\delta = 0.1$. It is important to note that the $\{X_{ijk}\}$ are here *not* parameterized by the mean. However, by taking $1/u_{ijk}$ as the parameter, we are in the framework described above. In particular, the relative perturbations are carried out with respect to $\alpha/\widehat{v}_{ijk}$ and α/u_{ijk}.

Table 5.2 presents the performance of Algorithm 5.12.1 for the 1×1 (single-bridge) model (5.110). Here $u_{111} = 1$ and $u_{112} = 1$ are chosen to be the bottleneck parameters, whereas the remaining (nonbottleneck) ones are set equal to 2. The notations in Table 5.2 are as follows:

1. *Mean, max*, and *min* $\widehat{\ell}$ denote the sample mean, maximum, and minimum values of 10 independently generated estimates of $\widehat{\ell}$.
2. *RE* denotes the sample relative error for $\widehat{\ell}$, averaged over the 10 runs.
3. *CPU* denotes the average CPU time in seconds based on 10 runs.

Table 5.2: Performance of Algorithm 5.12.1 for the single-bridge model with samples $N = N_1 = 500$.

	CMC	CE	VM	CE-SCR	VM-SCR
Mean $\widehat{\ell}$	4.052	3.970	3.734	3.894	3.829
Max $\widehat{\ell}$	8.102	4.327	4.201	4.345	4.132
Min $\widehat{\ell}$	1.505	3.380	3.395	3.520	3.278
RE	0.519	0.070	0.078	0.076	0.068
CPU	0.00	0.04	0.21	0.05	0.13

From the results of Table 5.2, it follows that for this relatively small model both CE and VM perform similarly to their screening counterparts. We will further see (in Chapter 8) that as the complexity of the model increases, VM-SCR outperforms its three alternatives, in particular CE-SCR. Note that for this model, both CE and VM detected correctly, at the first stage, the two bottleneck parameters. In particular, Table 5.3 presents a typical dynamics of detecting the two bottleneck parameters at the first stage of Algorithm 5.12.1 for a single-bridge model having

a total of 5 parameters. In Table 5.3, t denotes the replication number at the first stage, while the 0s and 1s indicate whether the corresponding parameters are identified as nonbottleneck or bottleneck parameters, respectively. As can be seen, after two replications four bottleneck parameters are left, after six replications three are identified as bottleneck parameters, and after seven replications the process stabilizes, detecting correctly the two true bottleneck parameters.

Table 5.3: Typical dynamics for detecting the bottleneck parameters at the first stage of Algorithm 5.12.1 for the bridge model.

t	u_1	u_2	u_3	u_4	u_5
0	1	1	1	1	1
1	1	1	0	1	1
2	1	1	0	1	1
3	1	1	0	1	0
4	1	1	0	1	0

t	u_1	u_2	u_3	u_4	u_5
5	1	1	0	1	0
6	1	1	0	1	0
7	1	1	0	0	0
8	1	1	0	0	0
9	1	1	0	0	0

Table 5.4 presents a typical evolution of $\{\widehat{\mathbf{v}}_t\}$ in the single-bridge model for the VM and VM-SCR methods at the second stage of Algorithm 5.12.1.

Table 5.4: Typical evolution of $\{\widehat{\mathbf{v}}_t\}$ for the VM and VM-SCR methods.

	VM					VM-SCR					
t	$\widehat{v}_1$	$\widehat{v}_2$	$\widehat{v}_3$	$\widehat{v}_4$	$\widehat{v}_5$	t	$\widehat{v}_1$	$\widehat{v}_2$	$\widehat{v}_3$	$\widehat{v}_4$	$\widehat{v}_5$
	1.000	1.000	2.000	2.000	2.000		1.000	1.000	2	2	2
1	0.537	0.545	2.174	2.107	1.615	1	0.555	0.599	2	2	2
2	0.346	0.349	2.071	1.961	1.914	2	0.375	0.402	2	2	2
3	0.306	0.314	1.990	1.999	1.882	3	0.315	0.322	2	2	2

As is clearly seen, the bottleneck parameters decrease about three times after the third iteration, while the nonbottleneck ones fluctuate about their nominal value $u = 2$.

Table 5.5 presents the performance of Algorithm 5.12.1 for the 3×10 bridge model with six bottlenecks corresponding to the elements u_{111}, u_{112}, u_{211}, u_{212}, u_{311}, u_{312}. We set $u_{111} = u_{112} = u_{211} = u_{212} = u_{311} = u_{312} = 1$, while the remaining (nonbottlenecks) values are set equal to 2. Note again that in this case both CE and VM found the true six bottlenecks.

Table 5.5: Performance of Algorithm 5.12.1 for the 3×10 model with six bottleneck elements and sample size $N = N_1 = 1000$.

	CMC	CE	VM	CE-SCR	VM-SCR
Mean $\widehat{\ell}$	16.16	16.11	14.84	16.12	15.67
Max $\widehat{\ell}$	22.65	26.85	16.59	18.72	17.20
Min $\widehat{\ell}$	11.13	7.007	12.59	14.63	14.80
RE	0.20	0.34	0.075	0.074	0.049
CPU	0.00	0.49	68.36	0.73	27.54

From the results in Table 5.5, it follows that without screening even the naive Monte Carlo outperforms the standard CE. However, using screening results in substantial improvement of CE. Finally, VM-SCR outperforms all four remaining alternatives.

PROBLEMS

5.1 Consider the integral $\ell = \int_a^b H(x)\,\mathrm{d}x = (b-a)\,\mathbb{E}[H(X)]$, with $X \sim \mathsf{U}(a,b)$. Let $X_1,\ldots,X_N$ be a random sample from $\mathsf{U}(a,b)$. Consider the estimators $\widehat{\ell} = \frac{1}{N}\sum_{i=1}^N H(X_i)$ and $\widehat{\ell}_1 = \frac{1}{2N}\sum_{i=1}^N \{H(X_i) + H(b+a-X_i)\}$. Prove that if $H(x)$ is monotonic in x, then

$$\mathrm{Var}(\widehat{\ell}_1) \leqslant \frac{1}{2}\mathrm{Var}(\widehat{\ell})\ .$$

In other words, using antithetic random variables is more accurate than using CMC.

5.2 Estimate the expected length of the shortest path for the bridge network in Example 5.1. Use both the CMC estimator (5.8) and the antithetic estimator (5.9). For both cases, take a sample size of N =100,000. Suppose that the lengths of the links $X_1,\ldots,X_5$ are exponentially distributed, with means $1, 1, 0.5, 2, 1.5$. Compare the results.

5.3 Common random variables (CRVs) are often used when estimating derivatives or gradients of functions. As a toy example, consider the estimation of the derivative $\ell'(u)$ of the function $\ell(u) = \mathbb{E}[Y]$, where $Y \sim \mathsf{Exp}(2u)$. Hence, $\ell'(u) = -1/(2u^2)$. A simple way to estimate $\ell'(u)$ is to first approximate it with the *forward difference*

$$\frac{\ell(u+h) - \ell(u)}{h}$$

for small h, and then to estimate both $\ell(u+h)$ and $\ell(u)$ via Monte Carlo simulation. For example, generate $X_1,\ldots,X_N$ from $\mathsf{Exp}(u+h)$ and $Y_1,\ldots,Y_n$ from $\mathsf{Exp}(u)$ independently, and take $(\bar{X} - \bar{Y})/h$ as an estimator of the forward difference (and hence as a biased estimator of $\ell'(u)$). However, it is better to draw each pair (X_i, Y_i) with CRNs, for example, by letting $X_i = -\ln(U_i)/(2(u+h))$ and $Y_i = -\ln(U_i)/(2u)$, where $U_i \sim \mathsf{U}(0,1), i = 1,\ldots,N$.

a) Implement the two forward difference estimators for the case $u = 1$ and $h = 10^{-2}$, taking $N = 10^6$. Estimate and compare the relative errors of both estimators.

b) Compute the relative errors exactly (not using simulation).

5.4 Use the batch means method to estimate the expected stationary waiting time in a $GI/G/1$ queue via Lindley's equation for the case where the interarrival times are $\mathsf{Exp}(1/2)$ distributed and the service times are $\mathsf{U}[0.5, 2]$ distributed. Take a simulation run of M = 10,000 customers, discarding the first $K = 100$ observations. Examine to what extent variance reduction can be achieved by using antithetic random variables.

5.5 Run the stochastic shortest path problem in Example 5.4 and estimate the performance $\ell = \mathbb{E}[H(\mathbf{X})]$ from 1000 independent replications, using the given (C_1, C_2, C_3, C_4) as the vector of control variables, assuming that $X_i \sim \mathsf{Exp}(1)$, $i = 1,\ldots,5$. Compare the results with those obtained with the CMC method.

5.6 Estimate the expected waiting time of the fourth customer in a $GI/G/1$ queue for the case where the interarrival times are $\mathsf{Exp}(1/2)$ distributed and the service times are $\mathsf{U}[0.5, 2]$ distributed. Use Lindley's equation and control variables, as described in Example 5.5. Generate $N = 1000$ replications of W_4 and provide a 95% confidence interval for $\mathbb{E}[W_4]$.

5.7 Prove that for any pair of random variables (U, V),

$$\mathrm{Var}(U) = \mathbb{E}[\,\mathrm{Var}(U \,|\, V)] + \mathrm{Var}(\,\mathbb{E}[U \,|\, V]\,) \;.$$

[Hint: Use the facts that $\mathbb{E}[U^2] = \mathbb{E}[\,\mathbb{E}[U^2 \,|\, V]\,]$ and $\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$.]

5.8 Let $R \sim \mathsf{G}(p)$ and define $S_R = \sum_{i=1}^{R} X_i$, where $X_1, X_2, \ldots$ is a sequence of iid $\mathsf{Exp}(\lambda)$ random variables that are independent of R.

a) Show, that $S_R \sim \mathsf{Exp}(\lambda\, p)$. [Hint: The easiest way is to use transform methods and conditioning.]

b) For $\lambda = 1$ and $p = 1/10$, estimate $\mathbb{P}(S_R > 10)$ using CMC with a sample size of $N = 1000$.

c) Repeat b), now using the conditional Monte Carlo estimator (5.22). Compare the results with those of a) and b).

5.9 Consider the random sum S_R in Problem 5.8, with parameters $p = 0.25$ and $\lambda = 1$. Estimate $\mathbb{P}(S_R > 10)$ via stratification using strata corresponding to the partition of events $\{R = 1\}$, $\{R = 2\}$, $\ldots$,$\{R = 7\}$, and $\{R > 7\}$. Allocate a total of $N = 10{,}000$ samples via both $N_i = p_i N$ and the optimal N_i^* in (5.35), and compare the results. For the second method, use a simulation run of size 1000 to estimate the standard deviations $\{\sigma_i\}$.

5.10 Show that the solution to the minimization program

$$\min_{N_1,\ldots,N_m} \sum_{i=1}^{m} \frac{p_i^2\, \sigma_i^2}{N_i} \qquad \text{such that} \qquad N_1 + \cdots + N_m = N \;,$$

is given by (5.35). This justifies the stratified sampling Theorem 5.5.1.

5.11 Use Algorithm 5.4.2 and (5.26) to estimate the reliability of the bridge reliability network in Example 4.2 on page 110 via permutation Monte Carlo. Consider two cases, where the link reliabilities are given by $\mathbf{p} = (0.3, 0.1, 0.8, 0.1, 0.2)$ and $\mathbf{p} = (0.95, 0.95, 0.95, 0.95, 0.95)$, respectively. Take a sample size of $N = 2000$.

5.12 Repeat Problem 5.11, using Algorithm 5.4.3. Compare the results.

5.13 This exercise discusses the counterpart of Algorithm 5.4.3 involving minimal paths rather than minimal cuts. A state vector $\mathbf{x}$ in the reliability model of Section 5.4.1 is called a *path vector* if $H(\mathbf{x}) = 1$. If in addition $H(\mathbf{y}) = 0$ for all $\mathbf{y} < \mathbf{x}$, then $\mathbf{x}$ is called the *minimal path vector*. The corresponding set $A = \{i : x_i = 1\}$ is called the *minimal path set*; that is, a minimal path set is a minimal set of components whose *functioning* ensures the functioning of the system. If $A_1, \ldots, A_m$ denote all the minimal paths sets, then the system is functioning if and only if all the components of at least one minimal path set are functioning.

a) Show that

$$H(\mathbf{x}) = \max_k \prod_{i \in A_k} x_i = 1 - \prod_{k=1}^{m} \left(1 - \prod_{i \in A_k} x_i\right). \tag{5.111}$$

b) Define

$$Y_k = \prod_{i \in A_k} X_i, \; k = 1, \ldots, m,$$

that is, Y_k is the indicator of the event that all components in A_i are functioning. Apply Proposition 5.4.1 to the sum $S = \sum_{k=1}^{m} Y_k$ and devise an algorithm similar to Algorithm 5.4.3 to estimate the reliability $r = \mathbb{P}(S > 0)$ of the system.

c) Test this algorithm on the bridge reliability network in Example 4.2.

5.14 Prove (see (5.44)) that the solution of

$$\min_g \mathrm{Var}_g \left(H(\mathbf{X}) \frac{f(\mathbf{X})}{g(\mathbf{X})} \right)$$

is

$$g^*(\mathbf{x}) = \frac{|H(\mathbf{x})| \, f(\mathbf{x})}{\int |H(\mathbf{x})| \, f(\mathbf{x}) \, \mathrm{d}\mathbf{x}} \, .$$

5.15 Let $Z \sim \mathsf{N}(0,1)$. Estimate $\mathbb{P}(Z > 4)$ via importance sampling, using the following shifted exponential sampling pdf:

$$g(x) = \mathrm{e}^{-(x-4)}, \quad x \geqslant 4 \, .$$

Choose N large enough to obtain accuracy to at least three significant digits and compare with the exact value.

5.16 Pearson's χ^2 discrepancy measure between densities g and h is defined as

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

Verify that the VM program (5.43) is equivalent to minimizing the Pearson χ^2 discrepancy measure between the zero-variance pdf g^* in (5.45) and the importance sampling density g. In this sense, the CE and VM methods are similar, since the CE method minimizes the Kullback–Leibler distance between g^* and g.

5.17 Repeat Problem 5.2 using importance sampling, where the lengths of the links are exponentially distributed with means $v_1, \ldots, v_5$. Write down the deterministic CE updating formulas, and estimate these via a simulation run of size 1000 using $\mathbf{w} = \mathbf{u}$.

5.18 Consider the natural exponential family ((A.9) in the Appendix). Show that (5.61), with $\mathbf{u} = \boldsymbol{\theta}_0$ and $\mathbf{v} = \boldsymbol{\theta}$, reduces to solving

$$\mathbb{E}_{\boldsymbol{\theta}_0} \left[H(\mathbf{X}) \left(\frac{\nabla c(\boldsymbol{\theta})}{c(\boldsymbol{\theta})} + \mathbf{t}(\mathbf{X}) \right) \right] = \mathbf{0} \, . \tag{5.112}$$

5.19 As an application of (5.112), suppose that we want to estimate the expectation of $H(X)$, with $X \sim \mathsf{Exp}(\lambda_0)$. Show that the corresponding CE optimal parameter is

$$\lambda^* = \frac{\mathbb{E}_{\lambda_0}[H(X)]}{\mathbb{E}_{\lambda_0}[H(X)X]} \, .$$

Compare with (A.15) in the Appendix. Explain how to estimate λ^* via simulation.

5.20 Let $X \sim \mathsf{Weib}(\alpha, \lambda_0)$. We wish to estimate $\ell = \mathbb{E}_{\lambda_0}[H(X)]$ via the SLR method, generating samples from $\mathsf{Weib}(\alpha, \lambda)$ — thus changing the scale parameter λ but keeping the scale parameter α fixed. Use (5.112) and Table A.1 in the Appendix to show that the CE optimal choice for λ is

$$\lambda^* = \left(\frac{\mathbb{E}_{\lambda_0}[H(X)]}{\mathbb{E}_{\lambda_0}[H(X)\, X^\alpha]} \right)^{1/\alpha} .$$

Explain how we can estimate λ^* via simulation.

5.21 Let $X_1, \ldots, X_n$ be independent $\mathsf{Exp}(1)$ distributed random variables. Let $\mathbf{X} = (X_1, \ldots, X_n)$ and $S(\mathbf{X}) = X_1 + \cdots + X_n$. We wish to estimate $\mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$ via importance sampling, using $X_i \sim \mathsf{Exp}(\theta)$, for all i. Show that the CE optimal parameter θ^* is given by

$$\theta^* = \frac{\mathbb{E}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}]}{\mathbb{E}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}\, \overline{X}]} ,$$

with $\overline{X} = (X_1 + \cdots + X_n)/n$ and $\mathbb{E}$ indicating the expectation under the original distribution (where each $X_i \sim \mathsf{Exp}(1)$).

5.22 Consider Problem 5.20. Define $G(z) = z^{1/\alpha}/\lambda_0$ and $\widetilde{H}(z) = H(G(z))$.

a) Show that if $Z \sim \mathsf{Exp}(1)$, then $G(Z) \sim \mathsf{Weib}(\alpha, \lambda_0)$.

b) Explain how to estimate ℓ via the TLR method.

c) Show that the CE optimal parameter for Z is given by

$$\theta^* = \frac{\mathbb{E}_\eta[\widetilde{H}(Z)\, W(Z; 1, \eta)]}{\mathbb{E}_\eta[\widetilde{H}(Z)\, Z\, W(Z; 1, \eta)]},$$

where $W(Z; 1, \eta)$ is the ratio of the $\mathsf{Exp}(1)$ and $\mathsf{Exp}(\eta)$ pdfs.

5.23 Assume that the expected performance can be written as $\ell = \sum_{i=1}^m a_i \ell_i$, where $\ell_i = \int H_i(\mathbf{x})\, \mathrm{d}\mathbf{x}$, and the a_i, $i = 1, \ldots, m$ are known coefficients. Let $Q(\mathbf{x}) = \sum_{i=1}^m a_i\, H_i(\mathbf{x})$. For any pdf g dominating $Q(\mathbf{x})$, the random variable

$$L = \sum_{i=1}^m a_i \frac{H_i(\mathbf{X})}{g(\mathbf{X})} = \frac{Q(\mathbf{X})}{g(\mathbf{X})} ,$$

where $\mathbf{X} \sim g$, is an unbiased estimator of ℓ — note that there is only one sample. Prove that L attains the smallest variance when $g = g^*$, with

$$g^*(\mathbf{x}) = |Q(\mathbf{x})| / \int |Q(\mathbf{x})|\, \mathrm{d}\mathbf{x} ,$$

and that

$$\mathrm{Var}_{g^*}(L) = \left(\int |Q(\mathbf{x})|\, \mathrm{d}\mathbf{x} \right)^2 - \ell^2 .$$

5.24 The Hit-or-Miss Method. Suppose that the sample performance function, H, is bounded on the interval $[0, b]$, say, $0 \leqslant H(x) \leqslant c$ for $x \in [0, b]$. Let

$\ell = \int H(x)\,\mathrm{d}x = b\,\mathbb{E}[H(X)]$, with $X \sim \mathsf{U}[0,b]$. Define an estimator of ℓ by

$$\widehat{\ell}^h = \frac{b\,c}{N} \sum_{i=1}^{N} I_{\{Y_i < H(X_i)\}} \;,$$

where $\{(X_i, Y_i) : j = 1, \ldots, N\}$ is a sequence of points uniformly distributed over the rectangle $[0,b] \times [0,c]$ (see Figure 5.8). The estimator $\widehat{\ell}^h$ is called the *hit-or-miss estimator*, since a point (X,Y) is accepted or rejected depending on whether that point falls inside or outside the shaded area in Figure 5.8, respectively. Show that the hit-or-miss estimator has a larger variance than the CMC estimator,

$$\widehat{\ell} = \frac{b}{N} \sum_{i=1}^{N} H(X_i) \;,$$

with $X_1, \ldots, X_N$ a random sample from $\mathsf{U}[0,b]$.

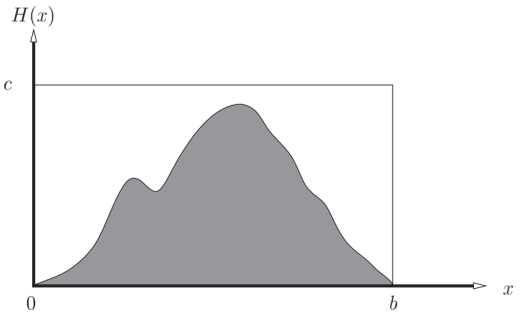

Figure 5.8: The hit-or-miss method.

Further Reading

The fundamental paper on variance reduction techniques is Kahn and Marshal [20]. There are a plenty of good Monte Carlo textbooks with chapters on variance reduction techniques. Among them are [13], [17], [21], [22], [25], [29], [32], [33], and [40]. For a comprehensive study of variance reduction techniques, see Fishman [13] and Rubinstein [34]. Asmussen and Glynn [2] provide a modern treatment of variance reduction and rare-event simulation. See also Chapter 9 of [24], which explains in detail the wide variety of variance reduction techniques that can be used to solve a single estimation problem. Influential books on sequential importance resampling are [27] and [11]. Multilevel Monte Carlo is a burgeoning area of research. The recent paper by Rhee and Glynn [31] explains how randomization of the levels of an infinite-level multilevel Monte Carlo algorithm can achieve theoretically a zero-bias estimator. Botev [6] describes the state-of-the art variance reduction technique for sampling from high-dimensional truncated multivariate normal distributions.

An introduction to reliability models may be found in [15]. For more information on variance reduction in the presence of heavy-tailed distributions, see also [1], [3], [4], and [10].

There is a large literature on estimating the number of SAWs. Although there is currently no known formula for the exact number of SAWs of length n, many approximating methods exist. The most advanced algorithms are the pivot ones. They can handle SAWs of size 10^7; see [8], [28]. For a recent survey, see [41].

A precursor of Algorithm 5.12.1 was given in [26], where the bottleneck elements were identified by estimating gradients of the performance function.

REFERENCES

1. S. Asmussen. Stationary distributions via first passage times. In J. H. Dshalalow, editor, *Advances in Queueing: Theory, Methods and Open Problems*, pages 79–102, New York, 1995. CRC Press.
2. S. Asmussen and P. W. Glynn. *Stochastic Simulation*. Springer-Verlag, New York, 2007.
3. S. Asmussen and D. P. Kroese. Improved algorithms for rare event simulation with heavy tails. *Advances in Applied Probability*, 38(2):545–558, 2006.
4. S. Asmussen, D. P. Kroese, and R. Y. Rubinstein. Heavy tails, importance sampling and cross-entropy. *Stochastic Models*, 21(1):57–76, 2005.
5. S. Asmussen and R. Y. Rubinstein. Complexity properties of steady-state rare-events simulation in queueing models. In J. H. Dshalalow, editor, *Advances in Queueing: Theory, Methods and Open Problems*, pages 429–462, New York, 1995. CRC Press.
6. Z. I. Botev. The normal law under linear restrictions: simulation and estimation via minimax tilting. *Journal of the Royal Statistics Society (B)*, 79:1–24, 2017.
7. H.-P. Chan and T.-L. Lai. A general theory of particle filters in hidden Markov models and some applications. *Annals of Statistics*, 41(6):2877–2904, 2013.
8. N. Clisby. Efficient implementation of the pivot algorithm for self-avoiding walks. *Journal of Statistical Physics*, 140:349–392, 2010.
9. W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, New York, 3rd edition, 1977.
10. P. T. de Boer, D. P. Kroese, and R. Y. Rubinstein. A fast cross-entropy method for estimating buffer overflows in queueing networks. *Management Science*, 50(7):883–895, 2004.
11. A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.
12. T. Elperin, I. B. Gertsbakh, and M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40(5):572–581, 1991.
13. G. S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, New York, 1996.
14. S. Gal, R. Y. Rubinstein, and A. Ziv. On the optimality and efficiency of common random numbers. *Math. Comput. Simul.*, 26(6):502–512, 1984.
15. I. B. Gertsbakh. *Statistical Reliability Theory*. Marcel Dekker, New York, 1989.
16. M. B. Giles. Multilevel Monte Carlo path simulation. *Operations Research*, 56(8):607–617, 2008.
17. P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, New York, 2004.

18. P. Grassberger. Pruned-enriched Rosenbluth method: Simulations of θ polymers of chain length up to 1000000. *Phys. Rev. E*, 56:3682–3693, Sep 1997.

19. D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 2nd edition, 1985.

20. M. Kahn and A. W. Marshall. Methods of reducing sample size in Monte Carlo computations. *Operations Research*, 1:263–278, 1953.

21. J. P. C. Kleijnen. *Statistical Techniques in Simulation, Part 1*. Marcel Dekker, New York, 1974.

22. J. P. C. Kleijnen. Analysis of simulation with common random numbers: A note on Heikes et al. *Simuletter*, 11:7–13, 1976.

23. D. P. Kroese and R. Y. Rubinstein. The transform likelihood ratio method for rare event simulation with heavy tails. *Queueing Systems*, 46:317–351, 2004.

24. D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*. John Wiley & Sons, 2011.

25. A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, 3rd edition, 2000.

26. D. Lieber, R. Y. Rubinstein, and D. Elmakis. Quick estimation of rare events in stochastic networks. *IEEE Transaction on Reliability*, 46:254–265, 1997.

27. J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, New York, 2001.

28. N. N. Madras. *Lectures on Monte Carlo Methods*. American Mathematical Society, 2002.

29. D. L. McLeish. *Monte Carlo Simulation and Finance*. John Wiley & Sons, New York, 2005.

30. M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Dover Publications, New York, 1981.

31. C.-H. Rhee and P. W. Glynn. Unbiased estimation with square root convergence for SDE models. *Operations Research*, 63(5):1026–1043, 2015.

32. C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, New York, 2nd edition, 2004.

33. S. M. Ross. *Simulation*. Academic Press, New York, 3rd edition, 2002.

34. R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, New York, 1981.

35. R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*. Springer-Verlag, New York, 2004.

36. R. Y. Rubinstein and R. Marcus. Efficiency of multivariate control variables in Monte Carlo simulation. *Operations Research*, 33:661–667, 1985.

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

38. R. Y. Rubinstein and A. Shapiro. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization via the Score Function Method*. John Wiley & Sons, New York, 1993.

39. R.Y. Rubinstein, M. Samorodnitsky, and M. Shaked. Antithetic variables, multivariate dependence and simulation of complex stochastic systems. *Management Science*, 31:66–77, 1985.

40. I. M. Sobol. *A Primer for the Monte Carlo Method*. CRC Press, Boca Raton, FL, 1994.

41. E. J. J. van Rensburg. Monte Carlo methods for the self-avoiding walk. *Journal of Physics A: Mathematical and Theoretical*, 42(32):323001, 2009.

42. F. T. Wall and J. J. Erpenbeck. New method for the statistical computation of polymer dimensions. *Journal of Chemical Physics*, 30(3):634–637, 1959.

43. P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons, 1992.

44. W. Whitt. Bivariate distributions with given marginals. *Annals of Statistics*, 4(6):1280–1289, 1976.

CHAPTER 6

MARKOV CHAIN MONTE CARLO

6.1 INTRODUCTION

In this chapter we present a powerful generic method, called *Markov chain Monte Carlo* (MCMC), for *approximately* generating samples from an arbitrary distribution. This, as we learned in Section 2.5, is typically not an easy task, in particular when $\mathbf{X}$ is a random vector with dependent components. An added advantage of MCMC is that it only requires specification of the target pdf up to a (normalization) constant.

The MCMC method is due to Metropolis et al. [18]. They were motivated by computational problems in statistical physics, and their approach uses the idea of generating a Markov chain whose limiting distribution is equal to the desired target distribution. There are many modifications and enhancement of the original Metropolis [18] algorithm, notably the algorithm introduced by Hastings [11]. Nowadays, any approach that produces an ergodic Markov chain whose stationary distribution is the target distribution is referred to as MCMC or *Markov chain sampling* [20]. The prominent MCMC algorithms are the Metropolis–Hastings and the Gibbs samplers, the latter being particularly useful in Bayesian analysis. Finally, MCMC sampling is the main ingredient in the popular *simulated annealing* technique [1] for discrete and continuous optimization.

The rest of this chapter is organized as follows: In Section 6.2 we present the classic Metropolis–Hastings algorithm, which simulates a Markov chain such that its stationary distribution coincides with the target distribution. An important special case is the *hit-and-run* sampler, discussed in Section 6.3. Section 6.4 deals with the *Gibbs sampler*, where the underlying Markov chain is constructed based

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

on a sequence of conditional distributions. Section 6.5 explains how to sample from distributions arising in the Ising and Potts models, which are extensively used in statistical mechanics, and Section 6.6 deals with applications of MCMC in Bayesian statistics. In Section 6.7 we show that both the Metropolis–Hastings and Gibbs samplers can be viewed as special cases of a general MCMC algorithm and present the slice and reversible jump samplers. Section 6.8 deals with the classic simulated annealing method for finding the global minimum of a multiextremal function, which is based on the MCMC method. Finally, Section 6.9 presents the perfect sampling method, for sampling exactly from a target distribution rather than approximately.

6.2 METROPOLIS–HASTINGS ALGORITHM

The main idea behind the Metropolis–Hastings algorithm is to simulate a Markov chain such that the stationary distribution of this chain coincides with the target distribution.

To motivate the MCMC method, assume that we want to generate a random variable X taking values in $\mathscr{X} = \{1, \ldots, m\}$, according to a target distribution $\{\pi_i\}$, with

$$\pi_i = \frac{b_i}{C}, \quad i \in \mathscr{X}, \tag{6.1}$$

where it is assumed that all $\{b_i\}$ are strictly positive, m is large, and the normalization constant $C = \sum_{i=1}^{m} b_i$ is difficult to calculate. Following Metropolis et al. [18], we construct a Markov chain $\{X_t,\ t = 0, 1, \ldots\}$ on $\mathscr{X}$ whose evolution relies on an arbitrary transition matrix $\mathbf{Q} = (q_{ij})$ in the following way:

- When $X_t = i$, generate a random variable Y satisfying $\mathbb{P}(Y = j) = q_{ij},\ j \in \mathscr{X}$. Thus, Y is generated from the m-point distribution given by the i-th row of $\mathbf{Q}$.
- If $Y = j$, let

$$X_{t+1} = \begin{cases} j & \text{with probability } \alpha_{ij} = \min\left\{\frac{\pi_j\, q_{ji}}{\pi_i\, q_{ij}}, 1\right\} = \min\left\{\frac{b_j\, q_{ji}}{b_i\, q_{ij}}, 1\right\}, \\ i & \text{with probability } 1 - \alpha_{ij}. \end{cases}$$

It follows that $\{X_t, t = 0, 1, \ldots\}$ has a one-step transition matrix $\mathbf{P} = (p_{ij})$ given by

$$p_{ij} = \begin{cases} q_{ij}\, \alpha_{ij} & \text{if } i \neq j\ , \\ 1 - \sum_{k \neq i} q_{ik}\, \alpha_{ik} & \text{if } i = j\ . \end{cases} \tag{6.2}$$

Now it is easy to check (see Problem 6.1) that, with α_{ij} as above,

$$\pi_i\, p_{ij} = \pi_j\, p_{ji}, \quad i, j \in \mathscr{X}\ . \tag{6.3}$$

In other words, the *detailed balance equations* (1.38) hold, and hence the Markov chain is time reversible and has stationary probabilities $\{\pi_i\}$. Moreover, this stationary distribution is also the *limiting* distribution if the Markov chain is irreducible and aperiodic. Note that there is no need for the normalization constant C in (6.1) to define the Markov chain.

The extension of the MCMC approach above for generating samples from an arbitrary multidimensional pdf $f(\mathbf{x})$ (instead of π_i) is straightforward. In this case, the nonnegative probability transition function $q(\mathbf{x}, \mathbf{y})$ (taking the place of q_{ij}

above) is often called the *proposal* or *instrumental* function. In viewing this function as a conditional pdf, we can also write $q(\mathbf{y} \mid \mathbf{x})$ instead of $q(\mathbf{x}, \mathbf{y})$. The probability $\alpha(\mathbf{x}, \mathbf{y})$ is called the *acceptance probability*. The original Metropolis algorithm [18] was suggested for symmetric proposal functions, that is, for $q(\mathbf{x}, \mathbf{y}) = q(\mathbf{y}, \mathbf{x})$. Hastings modified the original MCMC algorithm to allow nonsymmetric proposal functions. Such an algorithm is called a *Metropolis–Hastings algorithm*. We call the corresponding Markov chain the *Metropolis–Hastings Markov chain*.

In summary, the Metropolis–Hastings algorithm, which, like the acceptance–rejection method, is based on a trial-and-error strategy, is given as follows:

Algorithm 6.2.1: Metropolis–Hastings Algorithm

input : Initial state $\mathbf{X}_0$ and sample size N. Target pdf $f(\mathbf{x})$ and proposal function $q(\mathbf{x}, \mathbf{y})$.
output: Markov chain $\mathbf{X}_1, \ldots, \mathbf{X}_N$ approximately distributed according to $f(\mathbf{x})$.

for $t = 0$ **to** $N - 1$ **do**
  Generate $\mathbf{Y} \sim q(\mathbf{X}_t, \mathbf{y})$ `// draw a proposal`
  $\alpha \leftarrow \min\left\{ \frac{f(\mathbf{Y})\, q(\mathbf{Y}, \mathbf{X}_t)}{f(\mathbf{X}_t)\, q(\mathbf{X}_t, \mathbf{Y})},\ 1 \right\}$ `// acceptance probability`
  Generate $U \sim \mathsf{U}(0, 1)$
  if $U \leqslant \alpha$ **then**
    $\mathbf{X}_{t+1} \leftarrow \mathbf{Y}$
  else
    $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$
return $\mathbf{X}_1, \ldots, \mathbf{X}_N$

The algorithm produces a sequence $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ of *dependent* random variables, with $\mathbf{X}_t$ approximately distributed according to $f(\mathbf{x})$, for large t.

Since Algorithm 6.2.1 is of the acceptance–rejection type, its efficiency depends on the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$. Ideally, we would like $q(\mathbf{x}, \mathbf{y})$ to reproduce the desired pdf $f(\mathbf{y})$ as faithfully as possible. A common approach [20] is to first parameterize $q(\mathbf{x}, \mathbf{y})$ as $q(\mathbf{x}, \mathbf{y}; \theta)$ and then use stochastic optimization methods to maximize this with respect to θ. Below we consider several particular choices of $q(\mathbf{x}, \mathbf{y})$.

EXAMPLE 6.1 Independence Sampler

The simplest Metropolis-type MCMC algorithm is obtained by choosing the proposal function $q(\mathbf{x}, \mathbf{y})$ to be independent of $\mathbf{x}$, that is, $q(\mathbf{x}, \mathbf{y}) = g(\mathbf{y})$ for some pdf $g(\mathbf{y})$. Thus, starting from a previous state $\mathbf{X}$ a candidate state $\mathbf{Y}$ is generated from $g(\mathbf{y})$ and accepted with probability

$$\alpha(\mathbf{X}, \mathbf{Y}) = \min\left\{ \frac{f(\mathbf{Y})\, g(\mathbf{X})}{f(\mathbf{X})\, g(\mathbf{Y})},\ 1 \right\}.$$

This procedure is very similar to the original acceptance–rejection methods of Chapter 2, and, as in that method, it is important that the proposal distribution g be close to the target f. Note that, in contrast to the acceptance–rejection method, the independence sampler produces *dependent* samples.

EXAMPLE 6.2 Uniform Sampling

Being able to sample *uniformly* from some discrete set $\mathscr{Y}$ is important in many applications; see, for example, the algorithms for counting in Chapter 9. A simple general procedure is as follows: Define a *neighborhood* structure on $\mathscr{Y}$. Any neighborhood structure is allowed, as long as the resulting Metropolis–Hastings Markov chain is irreducible and aperiodic. Let $n_{\mathbf{x}}$ be the number of neighbors of a state $\mathbf{x}$. For the proposal distribution, we simply choose each possible neighbor of the current state $\mathbf{x}$ with equal probability. That is, $q(\mathbf{x}, \mathbf{y}) = 1/n_{\mathbf{x}}$. Since the target pdf $f(\mathbf{x})$ here is constant, the acceptance probability is

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\{n_{\mathbf{x}}/n_{\mathbf{y}}, 1\} .$$

By construction, the limiting distribution of the Metropolis–Hastings Markov chain is the uniform distribution on $\mathscr{Y}$.

EXAMPLE 6.3 Random Walk Sampler

In the random walk sampler the proposal state $\mathbf{Y}$, for a given current state $\mathbf{x}$, is given by $\mathbf{Y} = \mathbf{x} + \mathbf{Z}$, where $\mathbf{Z}$ is typically generated from some spherically symmetrical distribution (in the continuous case), such as $\mathsf{N}(\mathbf{0}, \Sigma)$. Note that the proposal function is symmetrical in this case; thus,

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{\frac{f(\mathbf{y})}{f(\mathbf{x})}, 1\right\}. \tag{6.4}$$

EXAMPLE 6.4

Let the random vector $\mathbf{X} = (X_1, X_2)$ have the following two-dimensional pdf:

$$f(\mathbf{x}) = c \exp(-(x_1^2 x_2^2 + x_1^2 + x_2^2 - 8x_1 - 8x_2)/2) , \tag{6.5}$$

where $c \approx 1/20216.335877$ is a normalization constant. The graph of this density is depicted in Figure 6.1.

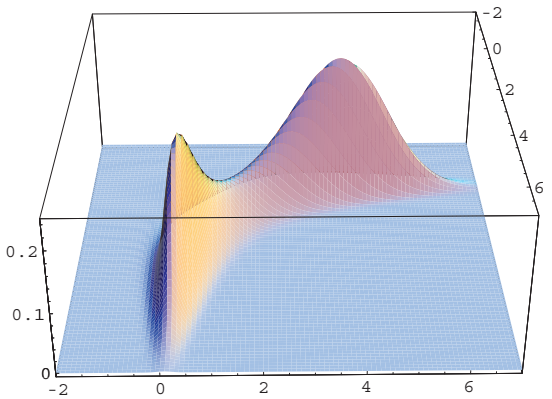

Figure 6.1: The density $f(x_1, x_2)$.

Suppose that we wish to estimate $\ell = \mathbb{E}[X_1]$ via the CMC estimator

$$\widehat{\ell} = \frac{1}{N} \sum_{t=1}^{N} X_{t1} \ ,$$

using the random walk sampler to generate a dependent sample $\{\mathbf{X}_t\}$ from $f(\mathbf{x})$. A simple choice for the increment $\mathbf{Z}$ is to draw the components of $\mathbf{Z}$ independently, from a $\mathsf{N}(0, a^2)$ distribution for some $a > 0$. Note that, if a is chosen too small, say less than 0.5, the components of the samples will be strongly positively correlated, which will lead to a large variance for $\widehat{\ell}$. On the other hand, for a too large, say greater than 10, most of the samples will be rejected, leading again to low efficiency. Below we choose a moderate value of a, say $a = 2$. The random walk sampler is now summarized as follows:

Algorithm 6.2.2: Random Walk Sampler

Initialize $\mathbf{X}_0 \leftarrow (X_{01}, X_{02})$.
for $t = 0$ **to** $N - 1$ **do**
  Draw $Z_1, Z_2 \sim \mathsf{N}(0, 1)$ independently.
  Set $\mathbf{Z} \leftarrow (Z_1, Z_2)$ and $\mathbf{Y} \leftarrow \mathbf{X}_t + 2\,\mathbf{Z}$.
  Set $\alpha \leftarrow \min\{\frac{f(\mathbf{Y})}{f(\mathbf{X}_t)}, 1\}$.
  Generate $U \sim \mathsf{U}(0, 1)$.
  if $U \leqslant \alpha$ **then**
    $\mathbf{X}_{t+1} \leftarrow \mathbf{Y}$
  else
    $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$

We ran this algorithm to produce $N = 10^5$ samples. The last few hundred of these are displayed in the left plot of Figure 6.2. We see that the samples closely follow the contour plot of the pdf, indicating that the correct region has been sampled. This is corroborated by the right plot of Figure 6.2, where we see that the histogram of the x_1 values is close to the true pdf (solid line).

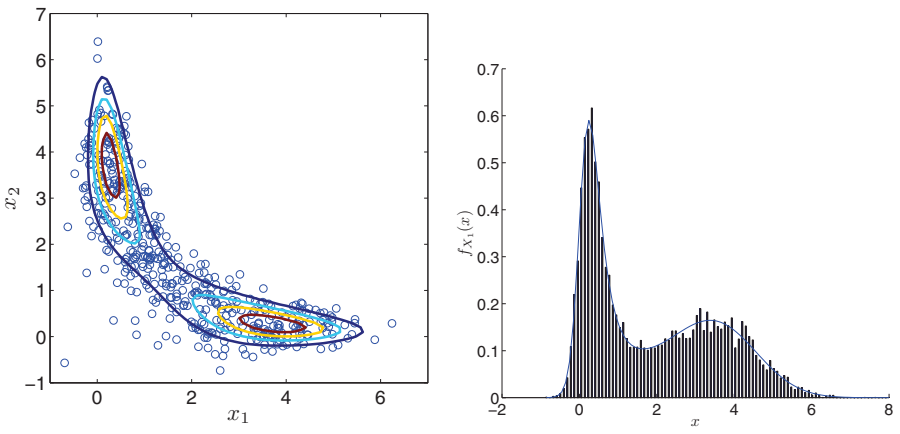

Figure 6.2: The left plot shows some samples of the random walk sampler along with several contour lines of f. The right plot shows the histogram of the x_1 values along with the true density of X_1.

We obtained an estimate $\widehat{\ell} = 1.89$ (the true value is $\mathbb{E}[X_1] \approx 1.85997$). To obtain a CI, we can use (4.21), where $\tilde{S}$ estimates the asymptotic variance, or employ the batch means method of Section 4.4.2.1. Figure 6.3 displays the estimated (auto)covariance function $\widehat{R}(k)$ for $k = 0, 1, \ldots, 400$. We see that up to about 100 the covariances are nonnegligible. Thus, to estimate the variance of $\widehat{\ell}$, we need to include all nonzero terms in (4.20), not only the variance $R(0)$ of X_1. Summing over the first 400 lags, we obtained an estimate of 10.41 for the asymptotic variance. This gives an estimated relative error for $\widehat{\ell}$ of 0.0185 and an 95% CI of $(1.82, 1.96)$. A similar CI was found when using the batch means method with 500 batches of size 200.

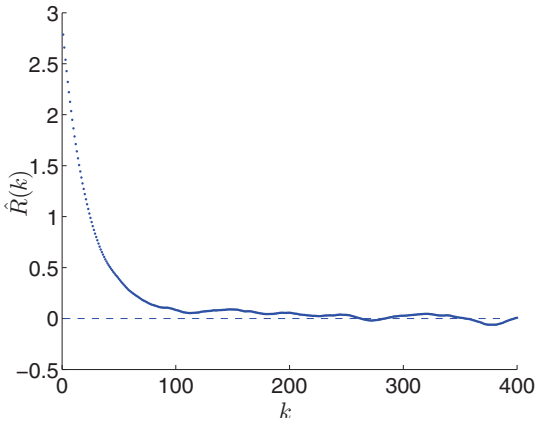

Figure 6.3: The estimated covariance function for the $\{X_{t1}\}$ for lags k up to 400.

While MCMC is a generic method and can be used to generate random samples virtually from any target distribution, regardless of its dimensionality and complexity, potential problems with the MCMC method are:

1. The resulting samples are often highly correlated.
2. Typically, it takes a considerable amount of time until the underlying Markov chain settles down to its steady state.
3. The estimates obtained via MCMC samples often tend to have much greater variances than those obtained from independent sampling of the target distribution. Various attempts have been made to overcome this difficulty. For details see, for example, [14] and [20].

Remark 6.2.1 At this point we must stress that although it is common practice to use MCMC to sample from $f(\mathbf{x})$ in order to estimate any expectation $\ell = \mathbb{E}_f[H(\mathbf{X})]$, the *actual* target for estimating ℓ is $g^*(\mathbf{x}) \propto |H(\mathbf{x})| f(\mathbf{x})$. Namely, sampling from $g^*(\mathbf{x})$ gives a minimum variance estimator (zero variance in the case $H(\mathbf{x}) \geqslant 0$). Thus, it is important to distinguish clearly between using MCMC for generating from some difficult pdf $f(\mathbf{x})$ and using MCMC to estimate a quantity such as ℓ. For the latter problem, much more efficient techniques can be used, such as importance sampling; moreover, a good importance sampling pdf can be obtained adaptively, as with the CE and TLR methods.

6.3 HIT-AND-RUN SAMPLER

The *hit-and-run* sampler, pioneered by Robert Smith [25], is among the first MCMC samplers in the category of *line samplers* [2]. As in the previous section, the objective is to sample from a target distribution $f(\mathbf{x})$ on $\mathscr{X} \subset \mathbb{R}^n$. Line samplers afford the opportunity to reach across the entire feasible region $\mathscr{X}$ in one step.

We first describe the original hit-and-run sampler for generating from a *uniform* distribution on a bounded open region $\mathscr{X}$ of $\mathbb{R}^n$. At each iteration, starting from a current point $\mathbf{x}$, a *direction vector* $\mathbf{d}$ is generated uniformly on the surface of an n-dimensional hypersphere. The intersection of the corresponding bidirectional line (through $\mathbf{x}$) and the enclosing box of $\mathscr{X}$ defines a line segment $\mathscr{L}$. The next point $\mathbf{y}$ is then selected uniformly from the intersection of $\mathscr{L}$ and $\mathscr{X}$.

Figure 6.4 illustrates the hit-and-run algorithm for generating uniformly from the set $\mathscr{X}$ (the gray region), which is bounded by a square. Given the point $\mathbf{x}$ in $\mathscr{X}$, a random direction $\mathbf{d}$ is generated, which defines the line segment $\mathscr{L} = uv$. Then a point $\mathbf{y}$ is chosen uniformly on $\mathscr{M} = \mathscr{L} \cap \mathscr{X}$, for example, by the acceptance–rejection method; that is, one generates a point uniformly on $\mathscr{L}$ and then accepts this point only if it lies in $\mathscr{X}$.

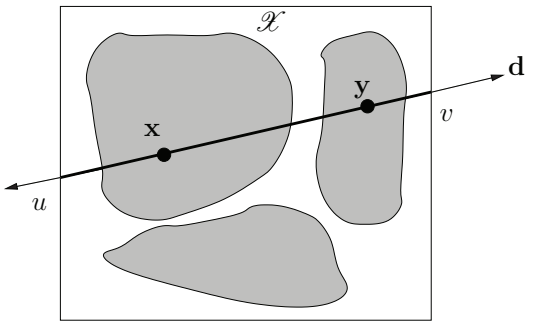

Figure 6.4: Illustration of the hit-and-run algorithm on a square in two dimensions.

Smith [25] showed that hit-and-run asymptotically generates uniformly distributed points over *arbitrary* open regions of $\mathbb{R}^n$. One desirable property of hit-and-run is that it can globally reach any point in the set in one step; that is, there is a strictly positive probability of sampling any neighborhood in the set. This property, coupled with a symmetry property, is important in deriving the limiting distribution. Lovász [15] proved that hit-and-run on a convex body in n dimensions produces an approximately uniformly distributed sample point in polynomial time, $\mathcal{O}(n^3)$, the best-known bound for such a sampling algorithm. He noted that the hit-and-run algorithm appears in practice to offer the most rapid convergence to a uniform distribution [15, 16]. Hit-and-run is unique in that it only takes polynomial time to get out of a corner; in contrast, *ball walk* takes exponential time to get out of a corner [17].

Note that the hit-and-run algorithm described above is a special case of the Metropolis–Hastings Algorithm 6.2.1, where the proposal function $q(\mathbf{x}, \mathbf{y})$ is symmetric and the target $f(\mathbf{x})$ is constant. It follows that each candidate point is

accepted with probability 1. To generate from a *general* strictly positive continuous pdf $f(\mathbf{x})$, one can simply modify the uniform hit-and-run algorithm above by accepting the candidate $\mathbf{y}$ with probability

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\{f(\mathbf{y})/f(\mathbf{x}), 1\}\,, \tag{6.6}$$

as in Algorithm 6.2.1 (note that $q(\mathbf{y}, \mathbf{x})/q(\mathbf{x}, \mathbf{y})$ equals 1). Thus the general hit-and-run algorithm with the Metropolis acceptance criterion above is summarized as follows [21]:

Algorithm 6.3.1: Hit-and-Run

input : Bounded region $\mathscr{X}$, target pdf f on $\mathscr{X}$, sample size N.
output: $\mathbf{X}_1, \ldots, \mathbf{X}_N$ approximately distributed according to f.

Initialize $\mathbf{X}_0 \in \mathscr{X}$.
for $t = 0$ **to** $N - 1$ **do**
  repeat
    Generate a random direction $\mathbf{d}_t$ according to a uniform distribution on the unit n-dimensional hypersphere.
    Let $\mathscr{M}_t \leftarrow \{\mathbf{x} \in \mathscr{X} : \mathbf{x} = \mathbf{X}_t + \lambda \mathbf{d}_t,\ \lambda \in \mathbb{R}\}$.
  until $\mathscr{M}_t \neq \emptyset$
  Generate a candidate point $\mathbf{Y}$ uniformly distributed over the line set $\mathscr{M}_t$.
  Generate $U \sim \mathsf{U}(0, 1)$.
  if $U \leqslant \min\{f(\mathbf{Y})/f(\mathbf{X}_t),\, 1\}$ **then**
    $\mathbf{X}_{t+1} \leftarrow \mathbf{Y}$
  else
    $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$
return $\mathbf{X}_1, \ldots, \mathbf{X}_N$

Chen and Schmeiser [5] describe how the hit-and-run sampler can be generalized to sample from any pdf on any bounded or unbounded region $\mathscr{X}$.

The hit-and-run algorithm can be embedded within an optimization framework to yield two global optimization algorithms: *hide-and-seek* [21] and *improving hit-and-run* [27]. The latter has been applied successfully to practical problems including composite material design and shape optimization, and it has been shown to have polynomial complexity, on average, for a class of quadratic programs. In Section 6.8 we show how to turn an MCMC sampler into an optimization algorithm by using simulated annealing.

6.4 GIBBS SAMPLER

The *Gibbs sampler* (Geman and Geman [7]) uses a somewhat different methodology from the Metropolis–Hastings algorithm and is particularly useful for generating n-dimensional random vectors. The distinguishing feature of the Gibbs sampler is that the underlying Markov chain is constructed, in a deterministic or random fashion, from a sequence of conditional distributions.

Gibbs sampling is advantageous if it is easier to sample from the conditional distributions than from the joint distribution. The essential idea of the Gibbs sampler — updating one part of the previous element while keeping the other parts

fixed — is useful in many instances where the state variable is a random variable taking values in a general space, not just in $\mathbb{R}^n$ (see [12]).

Suppose that we wish to sample a random vector $\mathbf{X} = (X_1, \ldots, X_n)$ according to a target pdf $f(\mathbf{x})$. Let $f(x_i \,|\, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ represent the conditional pdf of the i-th component, X_i, given the other components $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. The Gibbs sampler is given next.

Algorithm 6.4.1: Gibbs Sampler

input : Initial point $\mathbf{X}_0$, sample size N, and target pdf f.
output: $\mathbf{X}_1, \ldots, \mathbf{X}_N$ approximately distributed according to f.

for $t = 0$ **to** $N - 1$ **do**
  Draw Y_1 from the conditional pdf $f(y_1 \,|\, X_{t,2}, \ldots, X_{t,n})$.
  for $i = 2$ **to** n **do**
    Draw Y_i from the conditional pdf $f(y_i \,|\, Y_1, \ldots, Y_{i-1}, X_{t,i+1}, \ldots, X_{t,n})$.
  $\mathbf{X}_{t+1} \leftarrow \mathbf{Y}$
return $\mathbf{X}_1, \ldots, \mathbf{X}_N$

Note that in the Gibbs sampler *all* samples are accepted, in contrast to the Metropolis–Hastings algorithm. We will see in Section 6.7 that under mild conditions the limiting distribution of the process $\{\mathbf{X}_t, t = 1, 2, \ldots\}$, generated via the Gibbs sampler, is precisely $f(\mathbf{x})$. Moreover, under some other simple conditions, it can be shown (see [14], [20]) that the convergence to the desired pdf is geometrically fast.

■ **EXAMPLE 6.5 Example 6.4 (Continued)**

We will show how to sample easily from the pdf f in (6.5) via the Gibbs sampler. We start by writing

$$f(x, y) = c_1(y) \exp\left(-\frac{1 + y^2}{2} \left(x - \frac{4}{1 + y^2} \right)^2 \right),$$

where $c_1(y)$ depends only on y; we see that, conditional on y, X has a normal distribution with expectation $4/(1 + y^2)$ and variance $1/(1 + y^2)$. The conditional distribution of Y given x follows in the same way. The corresponding Gibbs sampler is thus as follows:

Algorithm 6.4.2: Gibbs Sampler for Example 6.4

Initialize Y_0.
for $t = 0$ **to** $N - 1$ **do**
  Draw $Z_1, Z_2 \sim \mathsf{N}(0, 1)$.
  $X_{t+1} \leftarrow Z_1 / \sqrt{1 + Y_t^2} + 4/(1 + Y_t^2)$
  $Y_{t+1} \leftarrow Z_2 / \sqrt{1 + X_{t+1}^2} + 4/(1 + X_{t+1}^2)$

Remark 6.4.1 (Systematic and Random Gibbs Samplers) Note that Algorithm 6.4.1 presents a *systematic* coordinatewise Gibbs sampler. That is, the vector

$\mathbf{X}$ is updated in a deterministic order: $1, 2, \ldots, n, 1, 2, \ldots$. In the *random* coordinatewise Gibbs sampler, the coordinates are chosen randomly, such as by generating them independently from a discrete uniform n-point pdf. In that case the Gibbs sampler can be viewed as an instance of the Metropolis–Hastings sampler, namely with the transition function

$$q(\mathbf{x}, \mathbf{y}) = \frac{1}{n} f(y_i \,|\, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \frac{1}{n} \frac{f(\mathbf{y})}{\sum_{y_i} f(\mathbf{y})} ,$$

where $\mathbf{y} = (x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$. Since $\sum_{y_i} f(\mathbf{y})$ can also be written as $\sum_{x_i} f(\mathbf{x})$, we have

$$\varrho(\mathbf{x}, \mathbf{y}) = \frac{f(\mathbf{y})\, q(\mathbf{y}, \mathbf{x})}{f(\mathbf{x})\, q(\mathbf{x}, \mathbf{y})} = \frac{f(\mathbf{y})\, f(\mathbf{x})}{f(\mathbf{x})\, f(\mathbf{y})} = 1 ,$$

so that the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$ is 1 in this case.

Here is another example of an application of the Gibbs sampler.

EXAMPLE 6.6 Closed Network of Queues in a Product Form

Consider m customers moving among n queues in a closed queueing network. Denote by $X_i(t)$ the number of customers in queue i, $i = 1, \ldots, n$, and let $\mathbf{X}(t) = (X_1(t), \ldots, X_n(t))$ and $\mathbf{x} = (x_1, \ldots, x_n)$. It is well known [22] that if the limit

$$\lim_{t \to \infty} \mathbb{P}(\mathbf{X}(t) = \mathbf{x}) = \pi(\mathbf{x})$$

exists, then, for exponentially distributed service times, the joint discrete pdf $\pi(\mathbf{x})$ can be written in *product form* as

$$\pi(\mathbf{x}) = C \prod_{i=1}^{n} f_i(x_i), \quad \text{for } \sum_{i=1}^{n} x_i = m , \tag{6.7}$$

where the $\{f_i(x_i),\ x_i \geq 0\}$ are *known* discrete pdfs, and C is a normalization constant. For a concrete example, see Problem 6.11.

The constant C is in general difficult to compute. To proceed, writing $S(\mathbf{x}) = \sum_{i=1}^{n} x_i$ and $\mathscr{X}^* = \{\mathbf{x} : S(\mathbf{x}) = m\}$, we have

$$C^{-1} = \sum_{\mathbf{x} \in \mathscr{X}^*} \prod_{i=1}^{n} f_i(x_i) , \tag{6.8}$$

which requires the evaluation of the product of n pdfs for each $\mathbf{x}$ in the set $\mathscr{X}^*$. This set has a total of $|\mathscr{X}^*| = \binom{m+n-1}{n-1}$ elements (see Problem 6.10), which rapidly grows very large.

We now show how to compute C based on Gibbs sampling. To apply the Gibbs sampler, we need to be able to generate samples from the conditional distribution of X_i given the other components. Note that we only have to generate $X_1, \ldots, X_{n-1}$, since $X_n = m - \sum_{k=1}^{n-1} X_k$. For $i = 1, \ldots, n-1$, we have

$$f(x_i \,|\, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n-1}) \propto f_i(x_i)\, f_n\left(m - \sum_{k=1}^{n-1} x_i\right) \tag{6.9}$$

for $x_i \in \{0, 1, \ldots, m - x_1 - \cdots - x_{i-1} - x_{i+1} - \cdots - x_{n-1}\}$. Sampling from these conditional pdfs can often be done efficiently, in particular when the $\{f_i\}$ are members of an exponential family; see also Problem 6.11.

Now that we can sample (approximately) from $\pi(\mathbf{x})$, it is straightforward to estimate the normalization constant C by observing that

$$\mathbb{E}_\pi \left[\frac{1}{\prod_{i=1}^n f_i(X_i)} \right] = \sum_{\mathbf{x} \in \mathscr{X}^*} \frac{1}{\prod_{i=1}^n f_i(x_i)} C \prod_{i=1}^n f_i(x_i) = |\mathscr{X}^*| \, C \, .$$

This suggests the following estimator for C, obtained from a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from π:

$$\widehat{C} = \binom{m+n-1}{n-1}^{-1} \frac{1}{N} \sum_{k=1}^N \prod_{i=1}^n \frac{1}{f_i(X_{ki})} \, ,$$

where X_{ki} is the i-th component of $\mathbf{X}_k$.

6.5 ISING AND POTTS MODELS

6.5.1 Ising Model

The Ising model is one of the most popular and most extensively studied models in statistical mechanics. It describes the interaction of idealized magnets, called *spins*, that are located on a two- or three-dimensional lattice. In the basic two-dimensional case the spins are located on the lattice $\{1, \ldots, n\} \times \{1, \ldots, n\}$, and each of the n^2 sites has four nearest neighbors, possibly including boundary sites, which "wrap around" to the other side of the lattice, creating a so-called *torus*. See Figure 6.5, where the four light gray sites are the neighbors of the dark gray site.

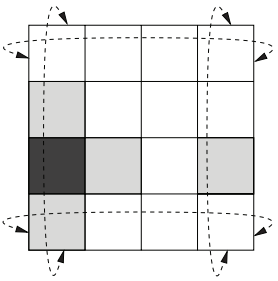

Figure 6.5: The boundary sites wrap around. The neighbors of the dark gray site are the light gray sites.

Let $\{1, \ldots, n^2\}$ be an enumeration of the sites. Each spin can be in one of two states: -1 or 1. Each of the 2^{n^2} *configurations* of spins $\mathbf{s} = (s_1, \ldots, s_{n^2})$ carries an amount of *total energy*

$$E(\mathbf{s}) = -A \sum_{i \leftrightarrow j} s_i \, s_j - B \sum_i s_i \, ,$$

where A and B are constants; in most studies $A = 1$ and $B = 0$, which we will now assume. The quantities $\sum_{i \leftrightarrow j} s_i\, s_j$ and $\sum_i s_i$ are called the *interaction energy* and *magnetization*, respectively. The notation $\sum_{i \leftrightarrow j}$ indicates that the summation is taken over neighboring pairs (i, j).

In thermal equilibrium the distribution of the spins, say π, follows the Boltzmann law: $\pi(\mathbf{s}) \propto \exp(-E(\mathbf{s})/T)$, where T is a fixed temperature. In other words, we have

$$\pi(\mathbf{s}) = \frac{\mathrm{e}^{\frac{1}{T}\sum_{i \leftrightarrow j} s_i\, s_j}}{\mathcal{Z}} \,,$$

where $\mathcal{Z}$ is the normalization constant, called the *partition function*. Apart from $\mathcal{Z}$, particular quantities of interest are the *mean energy per spin* $\mathbb{E}_\pi[\sum_{i \leftrightarrow j} S_i\, S_j/n^2]$ and the *mean magnetization per spin* $\mathbb{E}_\pi[\sum_i S_i/n^2]$. These quantities can be obtained via Monte Carlo simulation, provided that one can sample efficiently from the target distribution π (see below).

In Figure 6.6 a sample from π is given (black = 1, white = -1) for $n = 30$ at the so-called *critical temperature* $T = 2/\ln(1 + \sqrt{2}) \approx 2.269$.

Figure 6.6: Ising configuration at the critical temperature.

We next define the *Potts model* — which can be viewed as a generalization of the Ising model — and explain how to generate samples from this extended model and thus, in particular, how to generate Figure 6.6.

6.5.2 Potts Model

Let $\{1, \ldots, J\}$ be an enumeration of spatial positions (sites), and let ψ_{ij} be some symmetrical and positive function relating the sites to each other, for example,

$$\psi_{ij} = \begin{cases} \beta\,(>0) & \text{if } i \text{ and } j \text{ are neighbors,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.10}$$

Assign to each site i a "color" x_i. Suppose that there are K such colors, labeled $\{1,\dots,K\}$. Define $\mathbf{x} = (x_1,\dots,x_J)$, and let $\mathscr{X}$ be the space of such configurations. On $\mathscr{X}$ we define the target pdf $f(\mathbf{x}) \propto \mathrm{e}^{H(\mathbf{x})}$ with

$$H(\mathbf{x}) = \sum_{i<j} \psi_{ij}\, I_{\{x_i = x_j\}} \,.$$

To see that the Ising model is a special case of the Potts model, define $x_i = I_{\{s_i=1\}}$ and ψ_{ij} as in (6.10), with $\beta = 4/T$. Then

$$\frac{1}{T}\sum_{i\leftrightarrow j} s_i\, s_j = \frac{1}{T}\sum_{i\leftrightarrow j} 2\left(I_{\{x_i=x_j\}} - \frac{1}{2}\right) = \sum_{i<j} \psi_{ij}\, I_{\{x_i=x_j\}} + \text{const}\,,$$

so that $\pi(\mathbf{s}) = f(\mathbf{x})$.

Next, we show how to generate a sample from the target pdf $f(\mathbf{x})$. To do so, we define auxiliary random variables $Y_{ij}, 1 \leqslant i < j \leqslant J$, such that conditional on $\mathbf{X} = \mathbf{x}$ the $\{Y_{ij}\}$ are independent, and each Y_{ij} is uniformly distributed on the interval $[0, a_{ij}]$, with $a_{ij} = \exp(\psi_{ij}\, I_{\{x_i=x_j\}}) \geqslant 1$. In other words, the conditional pdf of $\mathbf{Y} = \{Y_{ij}\}$ given $\mathbf{X} = \mathbf{x}$ is

$$f(\mathbf{y}\,|\,\mathbf{x}) = \prod_{i<j} \frac{I_{\{y_{ij}\leqslant a_{ij}\}}}{a_{ij}} = \prod_{i<j} I_{\{y_{ij}\leqslant a_{ij}\}}\, \mathrm{e}^{-H(\mathbf{x})}\,.$$

The significance of this is that the joint pdf of $\mathbf{X}$ and $\mathbf{Y}$ is now simply

$$f(\mathbf{x},\mathbf{y}) = f(\mathbf{x})\, f(\mathbf{y}|\mathbf{x}) \propto \begin{cases} 1 & \text{if } y_{ij} \leqslant a_{ij}, \text{ for all } i<j\,, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $(\mathbf{X},\mathbf{Y})$ is *uniformly* distributed. More important, because $f(\mathbf{x}\,|\,\mathbf{y}) \propto f(\mathbf{x},\mathbf{y})$, we find that $\mathbf{X}\,|\,\mathbf{y}$ is uniformly distributed over the set $\mathscr{A} = \{\mathbf{x} : y_{ij} \leqslant \exp(\psi_{ij}\, I_{\{x_i=x_j\}})$ for all $i<j\}$. Now, either $y_{ij} \in [0,1]$ or $y_{ij} \in (1, \mathrm{e}^{\psi_{ij}}]$. In the former case, for any $\mathbf{x} \in \mathscr{A}$, the coordinates x_i and x_j range over all the colors, and by the uniformity, each color is equally likely. But in the latter case, x_i must be equal to x_j. Thus, for a given $\mathbf{y}$, the sites i,j (with $i<j$) for which $y_{ij} > 1$ can be gathered into clusters, and within each such cluster, the sites have identical colors. Moreover, given $\mathbf{y}$, the colors within the clusters are independent and uniformly distributed on $\{1,\dots,K\}$. The same holds for the colors of the remaining positions, which can be viewed as one-cluster sites.

Hence, we can easily generate both $\mathbf{X}\,|\,\mathbf{y}$ and $\mathbf{Y}\,|\,\mathbf{x}$. As a consequence, we can use the Gibbs sampler to (approximately) sample from $f(\mathbf{x},\mathbf{y})$; that is, we iteratively sample from $f(\mathbf{x}\,|\,\mathbf{y})$ and $f(\mathbf{y}\,|\,\mathbf{x})$. Finally, to obtain a sample $\mathbf{X}$ from $f(\mathbf{x})$, we generate $(\mathbf{X},\mathbf{Y})$ via the Gibbs sampler and simply ignore $\mathbf{Y}$.

To simplify matters further, note that instead of the exact value Y_{ij} it suffices to know only the variable $B_{ij} = I_{\{Y_{ij}\geqslant 1\}}$. Given $\mathbf{X} = \mathbf{x}$, B_{ij} has a $\mathsf{Ber}(1-\mathrm{e}^{-\psi_{ij}})$ distribution if $x_i = x_j$, and $B_{ij} = 0$ otherwise. This leads to the following so-called Swendsen–Wang algorithm:

Algorithm 6.5.1: Swendsen–Wang

1 Given $\{X_i\}$, generate $B_{ij} \sim \mathsf{Ber}(I_{\{X_i=X_j\}}(1-\mathrm{e}^{-\psi_{ij}}))$ for $1 \leqslant i < j \leqslant J$.
2 Given $\{B_{ij}\}$, generate $X_i, i = 1,\dots,J$ by clustering all the sites and choosing each cluster color independently and uniformly from $\{1,\dots,K\}$.

Remark 6.5.1 (Data Augmentation) The idea above of introducing an *auxiliary* variable $\mathbf{y}$ to make sampling from $f(\mathbf{x})$ easier is also known as *data augmentation*. The composition method described in Section 2.3.3 can be viewed as another example of data augmentation. To illustrate, suppose that we want to sample from the mixture pdf

$$f(x) = \sum_{i=1}^{K} p_i \, f_i(x) \; .$$

Let Y be the discrete random variable taking values in $\{1, \ldots, K\}$ corresponding to the probabilities $\{p_i\}$. The composition method makes it easy to sample from the joint pdf of X and Y: first, draw Y according to $\{p_i\}$ and then sample X conditional on $Y = i$; that is, sample from $f_i(x)$. By simply ignoring Y, we obtain a sample from $f(x)$.

6.6 BAYESIAN STATISTICS

One of the main application areas of the MCMC method is Bayesian statistics. The mainstay of the Bayesian approach is Bayes' rule (1.6), which, in terms of pdfs, can be written as

$$f(\mathbf{y} \,|\, \mathbf{x}) = \frac{f(\mathbf{x} \,|\, \mathbf{y}) \, f(\mathbf{y})}{\int f(\mathbf{x} \,|\, \mathbf{y}) \, f(\mathbf{y}) \, \mathrm{d}\mathbf{y}} \propto f(\mathbf{x} \,|\, \mathbf{y}) \, f(\mathbf{y}) \; . \tag{6.11}$$

In other words, for any two random variables $\mathbf{X}$ and $\mathbf{Y}$, the conditional distribution of $\mathbf{Y}$ given $\mathbf{X} = \mathbf{x}$ is proportional to the product of the conditional pdf of $\mathbf{X}$ given $\mathbf{Y} = \mathbf{y}$ and the pdf of $\mathbf{Y}$. Note that instead of writing f_X, f_Y, $f_{X \,|\, Y}$, and $f_{Y \,|\, X}$ in the formula above, we have used the *same letter* f for the pdf of $\mathbf{X}$, $\mathbf{Y}$, and the conditional pdfs. This particular style of notation is typical in Bayesian analysis and can be of great descriptive value, despite its apparent ambiguity. We will use this notation whenever we work in a Bayesian setting.

The significance of (6.11) becomes clear when it is employed in the context of Bayesian parameter estimation, sometimes referred to as *Bayesian learning*. The following example explains the ideas.

■ EXAMPLE 6.7 Coin Flipping and Bayesian Learning

Recall the basic random experiment in Example 1.1 on Page 3, where we toss a biased coin n times. Suppose that the outcomes are $x_1, \ldots, x_n$, with $x_i = 1$ if the i-th toss is heads and $x_i = 0$ otherwise, $i = 1, \ldots, n$. Let p denote the probability of heads. We want to obtain information about p from the data $\mathbf{x} = (x_1, \ldots, x_n)$, for example, construct a CI.

The crucial idea is to summarize the information about p via a probability density $f(p)$. For example, if we know nothing about p, we take $f(p)$ uniformly distributed on the $(0, 1)$ interval, that is, $f(p) = 1, 0 \leqslant p \leqslant 1$. In effect, we treat p as a random variable. Now, obviously, the data $\mathbf{x}$ will affect our knowledge of p, and the way to update this information is to use Bayes' formula:

$$f(p \,|\, \mathbf{x}) \propto f(\mathbf{x} \,|\, p) \, f(p) \; .$$

The density $f(p)$ is called the *prior* density, $f(p \,|\, \mathbf{x})$ is called the *posterior density*, and $f(\mathbf{x} \,|\, p)$ is referred to as the *likelihood*. In our case, given p, the

$\{X_i\}$ are independent and $\mathsf{Ber}(p)$ distributed, so

$$f(\mathbf{x}\,|\,p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^s\,(1-p)^{n-s}\ ,$$

with $s = x_1 + \cdots + x_n$ representing the total number of successes. Then using a uniform prior ($f(p) = 1$) we get a posterior pdf

$$f(p\,|\,\mathbf{x}) = c\,p^s\,(1-p)^{n-s}\ ,$$

which is the pdf of the $\mathsf{Beta}(s+1, n-s+1)$ distribution. The normalization constant is $c = (n+1)\binom{n}{s}$.

A Bayesian CI for p is now formed by taking the appropriate quantiles of the posterior pdf. As an example, suppose that $n = 100$ and $s = 1$. Then, a left one-sided 95% CI for p is $[0, 0.0461]$, where 0.0461 is the 0.95 quantile of the $\mathsf{Beta}(2, 100)$ distribution. To estimate p, we can take the value for which the pdf is maximal, the so-called *mode* of the pdf. In this problem, the mode is 0.01, coinciding with the sample mean. Figure 6.7 gives a plot of the posterior pdf for this problem.

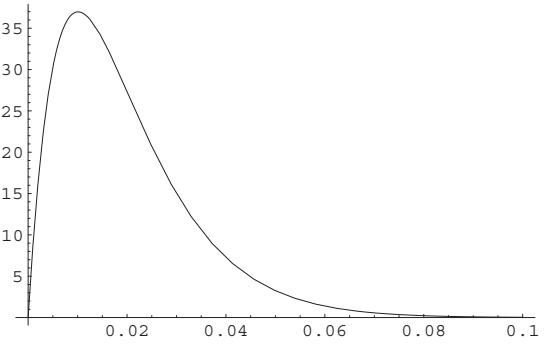

Figure 6.7: Posterior pdf for p, with $n = 100$ and $s = 1$.

Generalizing the previous example, a typical situation where MCMC (in particular, Gibbs sampling) can be used in Bayesian statistics is the following: Suppose that we want to sample from a posterior density $f(\boldsymbol{\theta}\,|\,\mathbf{x})$, where the data $\mathbf{x}$ are given (fixed) and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ is the parameter of interest. Suppose that it is easy to sample from $f(\theta_i\,|\,\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_k, \mathbf{x})$ for all i. Then, we can use the Gibbs sampler to obtain a sample $\boldsymbol{\Theta}$ from $f(\boldsymbol{\theta}\,|\,\mathbf{x})$. The next example, adapted from Gelman et al. [6], illustrates the general idea.

■ EXAMPLE 6.8 Poisson Disruption Problem

Suppose that the random variables $X_1, \ldots, X_n$ describe the number of disasters in n subsequent years. In some random year K the rate of disasters changes from λ_1 to λ_2. Such a K is often called a *change point*. Our prior knowledge of λ_i is summarized by a $\mathsf{Gamma}(a_i, \eta_i)$, where shape parameter a_i is known. In turn, η_i is given by a $\mathsf{Gamma}(b_i, c_i)$ distribution, where both

b_i and c_i are known. Let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ and $\boldsymbol{\eta} = (\eta_1, \eta_2)$. We are given the data $\mathbf{x} = (x_1, \ldots, x_n)$, and the objective is to simulate from the posterior distribution of $\boldsymbol{\theta} = (\lambda_1, \lambda_2, \eta_1, \eta_2, K)$ given $\mathbf{x}$.

For the model we have the following hierarchical structure:

1. K has some discrete pdf $f(K)$ on $1, \ldots, n$.
2. Given K, the $\{\eta_i\}$ are independent and have a $\mathsf{Gamma}(b_i, c_i)$ distribution for $i = 1, 2$.
3. Given K and $\boldsymbol{\eta}$, the $\{\lambda_i\}$ are independent and have a $\mathsf{Gamma}(a_i, \eta_i)$ distribution for $i = 1, 2$.
4. Given $K, \boldsymbol{\eta}$, and $\boldsymbol{\lambda}$, the $\{X_i\}$ are independent and have a $\mathsf{Poi}(\lambda_1)$ distribution for $i = 1, \ldots, K$, and a $\mathsf{Poi}(\lambda_2)$ distribution for $i = K+1, \ldots, n$.

It follows from point 4. that

$$
\begin{aligned}
f(\mathbf{x} \,|\, \boldsymbol{\lambda}, \boldsymbol{\eta}, K) &= \prod_{i=1}^{K} \mathrm{e}^{-\lambda_1} \frac{\lambda_1^{x_i}}{x_i!} \prod_{i=K+1}^{n} \mathrm{e}^{-\lambda_2} \frac{\lambda_2^{x_i}}{x_i!} \\
&= \mathrm{e}^{-\lambda_1 K} \lambda_1^{\sum_{i=1}^{K} x_i} \mathrm{e}^{-\lambda_2 (n-K)} \lambda_2^{\sum_{i=K+1}^{n} x_i} \prod_{i=1}^{n} \frac{1}{x_i!} \,.
\end{aligned}
$$

Moreover, by the product rule (1.4), the joint pdf is given by

$$
\begin{aligned}
f(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\eta}, K) \propto f(K) \, &\mathrm{e}^{-\lambda_1 K} \lambda_1^{\sum_{i=1}^{K} x_i} \mathrm{e}^{-\lambda_2 (n-K)} \lambda_2^{\sum_{i=K+1}^{n} x_i} \prod_{i=1}^{n} \frac{1}{x_i!} \\
&\times \mathrm{e}^{-\eta_1 \lambda_1} \lambda_1^{a_1 - 1} \eta_1^{a_1} \times \mathrm{e}^{-\eta_2 \lambda_2} \lambda_2^{a_2 - 1} \eta_2^{a_2} \\
&\times \mathrm{e}^{-c_1 \eta_1} \eta_1^{b_1 - 1} c_1^{b_1} \times \mathrm{e}^{-c_2 \eta_2} \eta_2^{b_2 - 1} c_2^{b_2} \,.
\end{aligned}
$$

As a consequence,

$$
f(\lambda_1 \,|\, \lambda_2, \boldsymbol{\eta}, K, \mathbf{x}) \propto \mathrm{e}^{-\lambda_1 (K + \eta_1)} \lambda_1^{a_1 - 1 + \sum_{i=1}^{K} x_i} \,.
$$

In other words, $(\lambda_1 \,|\, \lambda_2, \boldsymbol{\eta}, K, \mathbf{x}) \sim \mathsf{Gamma}(a_1 + \sum_{i=1}^{K} x_i,\, K + \eta_1)$. In a similar way, we have

$$
\begin{aligned}
(\lambda_2 \,|\, \lambda_1, \boldsymbol{\eta}, K, \mathbf{x}) &\sim \mathsf{Gamma}\left(a_2 + \textstyle\sum_{i=K+1}^{n} x_i,\; n - K + \eta_2\right) \,, \\
(\eta_1 \,|\, \boldsymbol{\lambda}, \eta_2, K, \mathbf{x}) &\sim \mathsf{Gamma}(a_1 + b_1,\; \lambda_1 + c_1) \,, \\
(\eta_2 \,|\, \boldsymbol{\lambda}, \eta_1, K, \mathbf{x}) &\sim \mathsf{Gamma}(a_2 + b_2,\; \lambda_2 + c_2) \,, \\
f(K \,|\, \boldsymbol{\lambda}, \boldsymbol{\eta}, \mathbf{x}) &\propto f(K) \, \mathrm{e}^{-K(\lambda_1 - \lambda_2)} \, (\lambda_1 / \lambda_2)^{\sum_{i=1}^{K} x_i} \,.
\end{aligned}
$$

Thus, Gibbs sampling can be used to sample from the posterior pdf $f(\boldsymbol{\lambda}, \boldsymbol{\eta}, K \,|\, \mathbf{x})$.

6.7 OTHER MARKOV SAMPLERS

There exist many variants of the Metropolis–Hastings and Gibbs samplers. However, all of the known MCMC algorithms can be described via the following framework: Consider a Markov chain $\{(\mathbf{X}_n, \mathbf{Y}_n), n = 0, 1, 2, \ldots\}$ on the set $\mathscr{X} \times \mathscr{Y}$, where

$\mathscr{X}$ is the target set and $\mathscr{Y}$ is an auxiliary set. Let $f(\mathbf{x})$ be the target pdf. Each transition of the Markov chain consists of two parts. The first is $(\mathbf{x}, \tilde{\mathbf{y}}) \to (\mathbf{x}, \mathbf{y})$, according to a transition matrix $\mathbf{Q}$; the second is $(\mathbf{x}, \mathbf{y}) \to (\mathbf{x}', \mathbf{y}')$, according to a transition matrix $\mathbf{R}$. In effect, the transition matrix $\mathbf{P}$ of the Markov chain is given by the product $\mathbf{Q}\,\mathbf{R}$. Both steps are illustrated in Figure 6.8 and explained below.

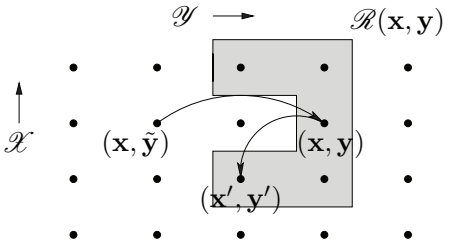

Figure 6.8: Each transition of the Markov chain consists of two steps: the Q-step, followed by the R-step.

The first step, the *Q-step*, changes the $\mathbf{y}$-coordinate but leaves the $\mathbf{x}$-coordinate intact. In particular, $\mathbf{Q}$ is of the form $\mathbf{Q}[(\mathbf{x}, \tilde{\mathbf{y}}), (\mathbf{x}, \mathbf{y})] = \mathbf{Q}_{\mathbf{x}}(\tilde{\mathbf{y}}, \mathbf{y})$, where $\mathbf{Q}_{\mathbf{x}}$ is a transition matrix on $\mathscr{Y}$. Let $q_{\mathbf{x}}$ be a stationary distribution for $\mathbf{Q}_{\mathbf{x}}$, assuming that it exists.

The second step, the *R-step*, is determined by the stationary distribution $q_{\mathbf{x}}$ and the neighborhood structure on the set $\mathscr{X} \times \mathscr{Y}$. Specifically, we define for each point $(\mathbf{x}, \mathbf{y})$ a set of neighbors $\mathscr{R}(\mathbf{x}, \mathbf{y})$ such that *if* $(\mathbf{x}', \mathbf{y}')$ *is a neighbor of* $(\mathbf{x}, \mathbf{y})$ *then the converse is also true*; see Figure 6.8, where the shaded area indicates the neighborhood set of $(\mathbf{x}, \mathbf{y})$. The crucial step is now to define the transition matrix $\mathbf{R}$ as

$$\mathbf{R}[(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')] = c(\mathbf{x}, \mathbf{y})\, f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}') \quad \text{for all} \quad (\mathbf{x}', \mathbf{y}') \in \mathscr{R}(\mathbf{x}, \mathbf{y}) \;,$$

where $c(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x}', \mathbf{y}') \in \mathscr{R}(\mathbf{x}, \mathbf{y})} f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}')$. Note that $c(\mathbf{x}, \mathbf{y}) = c(\mathbf{x}', \mathbf{y}')$ when $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$ belong to the same neighborhood set. With this choice of $\mathbf{Q}$ and $\mathbf{R}$ it can be shown (see Problem 6.15) that the Markov chain has a stationary distribution

$$\mu(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})\, q_{\mathbf{x}}(\mathbf{y}) \;, \tag{6.12}$$

which is also the limiting distribution, provided that the chain is irreducible and aperiodic. In particular, by ignoring the $\mathbf{y}$-coordinate, we see that the limiting pdf of $\mathbf{X}_n$ is the required target $f(\mathbf{x})$. This leads to the following *generalized Markov sampler* [12]:

Algorithm 6.7.1: Generalized Markov Sampler

1 Initialize $(\mathbf{X}_0, \mathbf{Y}_0)$.
2 **for** $t = 0$ **to** $N - 1$ **do**
3 Given $(\mathbf{X}_t, \mathbf{Y}_t)$, generate $\mathbf{Y}$ from $\mathbf{Q}_{\mathbf{x}}(\mathbf{Y}_t, \mathbf{y})$ `// Q-step`
4 Given $\mathbf{Y}$, generate $(\mathbf{X}_{t+1}, \mathbf{Y}_{t+1})$ from $\mathbf{R}[(\mathbf{X}_t, \mathbf{Y}), (\mathbf{x}, \mathbf{y})]$ `// R-step`

Remark 6.7.1 Denoting $\mathscr{R}^-(\mathbf{x},\mathbf{y}) = \mathscr{R}(\mathbf{x},\mathbf{y}) \setminus \{(\mathbf{x},\mathbf{y})\}$, the sampler can be generalized further (see [12]) by redefining $\mathbf{R}$ as

$$\mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')] = \begin{cases} s(\mathbf{x},\mathbf{y})\, c(\mathbf{x},\mathbf{y})\, f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}') & \text{if } (\mathbf{x}',\mathbf{y}') \in \mathscr{R}^-(\mathbf{x},\mathbf{y}) \,, \\ 1 - \displaystyle\sum_{(\mathbf{z},\mathbf{k}) \in \mathscr{R}^-(\mathbf{x},\mathbf{y})} \mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{z},\mathbf{k})] & \text{if } (\mathbf{x}',\mathbf{y}') = (\mathbf{x},\mathbf{y}) \,, \end{cases} \tag{6.13}$$

where s is an arbitrary function such that, first, $s(\mathbf{x},\mathbf{y}) = s(\mathbf{x}',\mathbf{y}')$ for all $(\mathbf{x}',\mathbf{y}') \in \mathscr{R}(\mathbf{x},\mathbf{y})$ and, second, the quantities above are all probabilities.

The generalized Markov sampler framework makes it possible to obtain many different samplers in a simple and unified manner. We give two examples: the slice sampler and the reversible jump sampler.

6.7.1 Slice Sampler

Suppose that we wish to generate samples from the pdf

$$f(\mathbf{x}) = b \prod_{k=1}^{m} f_k(\mathbf{x}) \,, \tag{6.14}$$

where b is a known or unknown constant and the $\{f_k\}$ are known positive functions — not necessarily densities. We employ Algorithm 6.7.1, where at the Q-step we generate, for a given $\mathbf{X} = \mathbf{x}$, a vector $\mathbf{Y} = (Y_1,\ldots,Y_m)$ by independently drawing each component Y_k from the uniform distribution on $[0, f_k(\mathbf{x})]$. Thus $q_{\mathbf{x}}(\mathbf{y}) = 1/\prod_{k=1}^m f_k(\mathbf{x}) = b/f(\mathbf{x})$. Next, we let $\mathscr{R}(\mathbf{x},\mathbf{y}) = \{(\mathbf{x}',\mathbf{y}) : f_k(\mathbf{x}') \geqslant y_k,\ k = 1,\ldots,m\}$. Then, (since $f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}) = b$),

$$\mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y})] = \frac{1}{|\mathscr{R}(\mathbf{x},\mathbf{y})|} \,.$$

In other words, in the R-step, given $\mathbf{x}$ and $\mathbf{y}$, we draw $\mathbf{X}'$ uniformly from the set $\{\mathbf{x}' : f_k(\mathbf{x}') \geqslant y_k, k = 1,\ldots,m\}$. This gives the following *slice sampler*:

Algorithm 6.7.2: Slice Sampler

input : Pdf f of the form (6.14), initial point $\mathbf{X}_0$, and sample size N.
output: $\mathbf{X}_1,\ldots,\mathbf{X}_N$ approximately distributed according to f.

```
for t = 0 to N − 1 do
    for k = 1 to m do
        Draw U_k ∼ U(0, 1).
        Y_k ← U_k f_k(X_t)
    Draw X_{t+1} uniformly from the set {x : f_k(x) ⩾ Y_k, k = 1, ..., m}.
return X_1, ..., X_N
```

■ **EXAMPLE 6.9 Slice Sampler**

Suppose that we now need to generate a sample from the target pdf

$$f(x) = c\,\frac{x\,\mathrm{e}^{-x}}{1+x}, \quad x \geqslant 0\,,$$

using the slice sampler with $f_1(x) = x/(1+x)$ and $f_2(x) = \mathrm{e}^{-x}$.

Suppose that at iteration t, $X_t = z$, and u_1 and u_2 are generated in Lines 2–4. In Line 5, X_{t+1} is drawn uniformly from the set $\{x : f_1(x)/f_1(z) \geqslant u_1,\ f_2(x)/f_2(z) \geqslant u_2\}$, which implies the bounds $x \geqslant \frac{u_1 z}{1+z-u_1 z}$ and $x \leqslant z - \ln u_2$. Since for $z > 0$ and $0 \leqslant u_1, u_2 \leqslant 1$, the latter bound is larger than the former, the interval to be drawn from in Line 5 is $(\frac{u_1 z}{1+z-u_1 z},\ z - \ln u_2)$. Figure 6.9 depicts a histogram of $N = 10^5$ samples generated via the slice sampler, along with the true pdf $f(x)$. We see that the two are in close agreement.

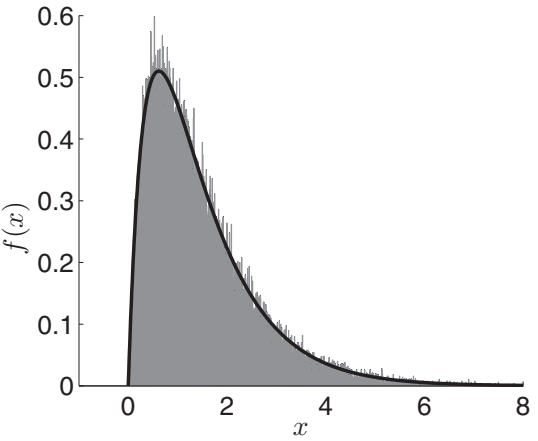

Figure 6.9: True density and histogram of samples produced by the slice sampler.

6.7.2 Reversible Jump Sampler

Reversible jump samplers [9] are useful for sampling from target spaces that contain vectors of different dimensions. This often occurs in Bayesian inference when different models for the data are considered.

■ **EXAMPLE 6.10 Regression Data**

Let data $y_1, \ldots, y_n$ be the outcomes of independent random variables $\{Y_i\}$ of the form

$$Y_i = \sum_{j=0}^{M} \beta_j\, u_i^j + \varepsilon_i, \qquad \varepsilon_i \sim \mathsf{N}(0,1), \quad i = 1, \ldots, n\,, \tag{6.15}$$

where $u_1, \ldots, u_n$ are known variables, and $M \in \{0, \ldots, M_{\max}\}$ and the parameters $\{\beta_m\}$ are unknown. Let $\mathbf{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_M)$. Taking uniform (i.e., constant) priors for $\{\beta_m\}$ and M, we have the joint pdf

$$f(\mathbf{y}, m, \boldsymbol{\beta}) \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{m}\beta_j\, u_i^j\right)^2\right] . \tag{6.16}$$

Let $\mathbf{x} = (m, \boldsymbol{\beta})$. Our objective is to draw from the posterior pdf $f(\mathbf{x}\,|\,\mathbf{y}) = f(m, \boldsymbol{\beta}\,|\,\mathbf{y})$. This yields information not only about the parameters, but also about which model (expressed by M) is more appropriate. Note here that the dimensionality of $\mathbf{x}$ depends crucially on m, so the standard Gibbs or Metropolis–Hastings sampling is not appropriate.

The reversible jump sampler jumps between spaces of different dimensionalities according to a set of allowed jumps (also called *moves*). In the example above, we could, for instance, allow only jumps between vectors that differ in dimension by at most 1; that is, $\beta_0 \to \beta_0'$, $\beta_0 \to (\beta_0', \beta_1')$, $(\beta_0, \beta_1) \to \beta_0'$, and so on.

To formulate the reversible jump sampler in the generalized Markov sampler framework, we define $\mathscr{Y} = \mathscr{X} \times \mathscr{M}$, where $\mathscr{M}$ is the set of moves, and write a generic element as $(\mathbf{z}, m)$. In the Q-step we take $\mathbf{Q}_{\mathbf{x}}(\cdot, (\mathbf{z}, m)) = p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})$. That is, a move of type m is selected according to some discrete pdf $p_{\mathbf{x}}(m)$. For example, the dimension of $\mathbf{x}$ is decreased, increased, or left unchanged. Then a new $\mathbf{z}$ is selected according to some transition function $q_m(\mathbf{x}, \mathbf{z})$. Note that the stationary pdf for the Q-step at $(\mathbf{z}, m)$ then becomes $p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})$. The R-step is determined by defining $\mathscr{R}(\mathbf{x}, (\mathbf{z}, m)) = \{(\mathbf{x}, (\mathbf{z}, m)), (\mathbf{z}, (\mathbf{x}, m'))\}$, where m' is the reverse move of m, that is, from $\mathbf{z}$ to $\mathbf{x}$. Then (6.13) reduces to

$$\mathbf{R}[(\mathbf{x}, (\mathbf{z}, m)), (\mathbf{z}, (\mathbf{x}, m'))] = \frac{s(\mathbf{x}, (\mathbf{z}, m))}{1 + 1/\varrho} , \tag{6.17}$$

with $\varrho = \frac{f(\mathbf{z})\, p_{\mathbf{z}}(m')\, q_{m'}(\mathbf{z}, \mathbf{x})}{f(\mathbf{x})\, p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})}$. Taking $s(\mathbf{x}, (\mathbf{z}, m)) = \min\{1 + \varrho,\, 1 + 1/\varrho\}$ reduces the right-hand side of (6.17) further to $\min\{\varrho, 1\}$. The transition $(\mathbf{x}, (\mathbf{z}, m)) \to (\mathbf{z}, (\mathbf{x}, m'))$ can thus be interpreted as acceptance of the proposed element $\mathbf{z}$. In effect, $\mathbf{Q}$ is used to propose a new element in accordance with the move m and transition function q, and $\mathbf{R}$ is used to accept or reject it in accordance with the acceptance ratio above. The reversible jump sampler can thus be viewed as a generalization of the Metropolis–Hastings sampler. This gives Algorithm 6.7.3.

Remark 6.7.2 (Dimension Matching) When dealing with continuous random variables, it is important to ensure that the transition densities are properly defined. Suppose that $\dim(\mathbf{x}) = d$ and $\dim(\mathbf{z}) = d' > d$. A possible way to generate a transition $\mathbf{x} \to \mathbf{z}$ is to first draw a $(d' - d)$-dimensional random vector $\mathbf{U}$ according to some density $g(\mathbf{u})$ and then let $\mathbf{z} = \phi(\mathbf{x}, \mathbf{U})$ for some bijection ϕ. This is known as *dimension matching* — the dimension of $(\mathbf{x}, \mathbf{u})$ must match that of $\mathbf{z}$. Note that by (1.20) the transition density is given by $q(\mathbf{x}, \mathbf{z}) = g(\mathbf{u})/|J_{(\mathbf{x},\mathbf{u})}(\phi)|$, where $|J_{(\mathbf{x},\mathbf{u})}(\phi)|$ is the absolute value of the determinant of the matrix of Jacobi of ϕ at $(\mathbf{x}, \mathbf{u})$.

Algorithm 6.7.3: Reversible Jump Sampler

input : Pdfs $\{p_{\mathbf{x}}\}$, transition functions $\{q_m\}$, target pdf f, initial point $\mathbf{X}_0$, and sample size N.
output: $\mathbf{X}_1, \ldots, \mathbf{X}_N$ approximately distributed according to f.

for $t = 0$ **to** $N - 1$ **do**
  Generate $m \sim p_{\mathbf{X}_t}(m)$.
  Generate $\mathbf{Z} \sim q_m(\mathbf{X}_t, \mathbf{z})$. Let m' be the reverse move from $\mathbf{Z}$ to $\mathbf{X}_t$.
  $\alpha \leftarrow \min\left\{ \dfrac{f(\mathbf{Z})\, p_{\mathbf{Z}}(m')\, q_{m'}(\mathbf{Z}, \mathbf{X}_t)}{f(\mathbf{X}_t)\, p_{\mathbf{X}_t}(m)\, q_m(\mathbf{X}_t, \mathbf{Z})},\ 1 \right\}$ `// acceptance probability`
  Generate $U \sim \mathsf{U}(0,1)$.
  if $U \leqslant \alpha$ **then**
    $\mathbf{X}_{t+1} \leftarrow \mathbf{Z}$
  else
    $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$
return $\mathbf{X}_1, \ldots, \mathbf{X}_N$

■ **EXAMPLE 6.11 Example 6.10 (Continued)**

We illustrate the reversible jump sampler using regression data $\mathbf{y} = (y_1, \ldots, y_n)$ of the form (6.15), with $u_i = (i-1)/20$, $i = 1, \ldots, 101$, $\beta_0 = 1$, $\beta_1 = 0.3$, and $\beta_2 = -0.2$. The data are depicted in Figure 6.10. Although it is obvious that a constant model ($m = 0$) does not fit the data, it is not clear if a linear model ($m = 1$) or a quadratic model ($m = 2$) is more appropriate. To assess the different models, we can run a reversible jump sampler to produce samples from the posterior pdf $f(\mathbf{x} \,|\, \mathbf{y})$, which (up to a normalization constant) is given by the right-hand side of (6.16). A very basic implementation is the following:

Algorithm 6.7.4: Reversible Jump Sampler for Example 6.10

Initialize $\mathbf{X}_0 = (m', \boldsymbol{\beta}')$
for $t = 0$ **to** $N - 1$ **do**
  Generate $m \in \{0, 1, 2\}$ with equal probability.
  Generate $\boldsymbol{\beta}$ from an $(m+1)$-dimensional normal pdf g_m with independent components, with means 0 and variances σ^2.
  $\mathbf{Z} \leftarrow (m, \boldsymbol{\beta})$
  $\alpha \leftarrow \min\left\{ \dfrac{f(\mathbf{Z} \,|\, \mathbf{y})\ g_{m'}(\boldsymbol{\beta}')}{f(\mathbf{X}_t \,|\, \mathbf{y})\ g_m(\boldsymbol{\beta})},\ 1 \right\}$ `// acceptance probability`
  Generate $U \sim \mathsf{U}(0,1)$.
  if $U \leqslant \alpha$ **then**
    $\mathbf{X}_{t+1} \leftarrow \mathbf{Z}$
  else
    $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$
  $(m', \boldsymbol{\beta}') \leftarrow \mathbf{X}_{t+1}$

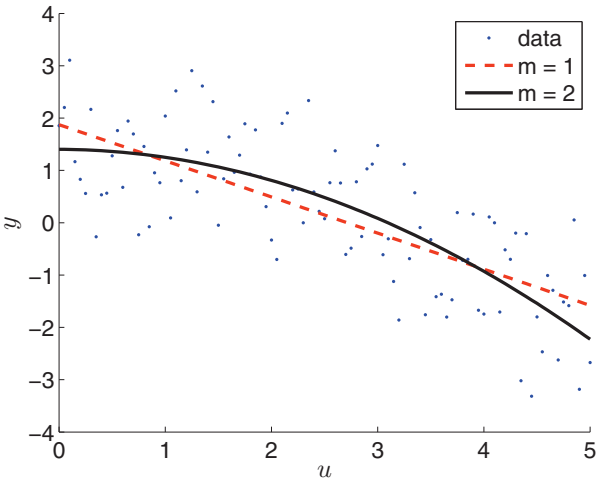

Figure 6.10: Regression data and fitted curves.

The procedure above, with $N = 10^5$ and $\sigma = 2$, produced 22,136 two-dimensional vectors $\boldsymbol{\beta}$ and 77,834 three-dimensional vectors, giving posterior probabilities 0.221 and 0.778 for models 1 and 2, respectively. The posterior probability for the constant model was negligible (0.0003). This indicates that the quadratic model has the best fit. The regression parameters $\boldsymbol{\beta}$ are estimated via the sample means of the $\{\boldsymbol{\beta}_t\}$ for $m_t = 1$ or 2 and are found to be $(1.874, -0.691)$ and $(1.404, -0.011, -0.143)$. The corresponding regression curves are depicted in Figure 6.10.

6.8 SIMULATED ANNEALING

Simulated annealing is a popular optimization technique based on MCMC. This technique uses MCMC sampling to find a mode of a density $f(\mathbf{x})$ (a point $\mathbf{x}^*$ where $f(\mathbf{x})$ is maximal). It involves defining a family of densities of the form $f_T(\mathbf{x}) \propto [f(\mathbf{x})]^{1/T}$, where the parameter T is called the *temperature* of the distribution. MCMC sampling is used to draw a single element $\mathbf{X}^{(k)}$ from f_{T_k} for successively lower temperatures $T_1, T_2, \ldots$. Each element $\mathbf{X}^{(k)}$ is used as the initial element of the next chain. As the temperature is reduced, the distributions become sharply peaked at the global maxima of f. Thus, the $\{\mathbf{X}^{(k)}\}$ converge to a point. They can converge to a local maximum, but this possibility is reduced by careful selection of successive temperatures. The sequence of temperatures, or *annealing schedule*, is therefore critical to the success of the method. A common choice for the annealing schedule is a geometric progression, starting with a specified initial temperature and multiplying by a *cooling factor* in the interval $(0, 1)$ after each iteration.

Simulated annealing can also be applied to nonprobabilistic optimization problems. Given an objective function $S(\mathbf{x})$, a Boltzmann distribution is defined via the density $f(\mathbf{x}) \propto \mathrm{e}^{-S(\mathbf{x})}$ or $f(\mathbf{x}) \propto \mathrm{e}^{S(\mathbf{x})}$, depending on whether the objective is to minimize or maximize S. Global optima of S are then obtained by searching for the mode of the Boltzmann distribution. We illustrate the method via two

worked examples, one based on the Metropolis–Hastings sampler and the other on the Gibbs sampler.

EXAMPLE 6.12 Traveling Salesman Problem

The traveling salesman problem (TSP) can be formulated as follows: Consider a weighted graph G with n nodes, labeled $1, 2, \ldots, n$. The nodes represent cities, and the edges represent the roads between the cities. Each edge from i to j has weight or cost c_{ij}, representing the length of the road. The problem is to find the shortest *tour* that visits all the cities exactly once except the starting city, which is also the terminating city. An example is given in Figure 6.11, where the bold lines form a possible tour.

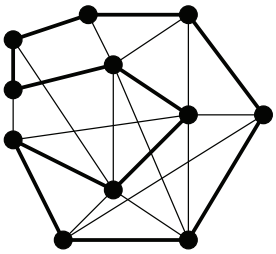

Figure 6.11: Find the shortest tour $\mathbf{x}$ visiting all nodes.

Without loss of generality, we can assume that the graph is *complete* (fully connected) because, if it is not complete, we can always add some costs (distances) equal to $+\infty$. Let $\mathscr{X}$ be the set of all possible tours, and let $S(\mathbf{x})$ the total length of tour $\mathbf{x} \in \mathscr{X}$. We can represent each tour via a *permutation* of $(1, \ldots, n)$. For example, for $n = 4$, the permutation $(1, 3, 2, 4)$ represents the tour $1 \to 3 \to 2 \to 4 \to 1$. Therefore, we will identify a tour with its corresponding permutation. The objective is thus to minimize

$$\min_{\mathbf{x} \in \mathscr{X}} S(\mathbf{x}) \quad = \quad \min_{\mathbf{x} \in \mathscr{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i, x_{i+1}} + c_{x_n, 1} \right\} . \tag{6.18}$$

Note that the number of elements in $\mathscr{X}$ is typically very large, since $|\mathscr{X}| = n!$.

The TSP can be solved via simulated annealing. First, we define the target pdf to be the Boltzmann pdf $f(\mathbf{x}) = c\, \mathrm{e}^{-S(\mathbf{x})/T}$. Second, we define a neighborhood structure on the space of permutations $\mathscr{X}$, called *2-opt*. Here the neighbors of an arbitrary permutation $\mathbf{x}$ are found by (1) selecting two different indexes from $\{1, \ldots, n\}$ and (2) reversing the path of $\mathbf{x}$ between those two indexes. For example, if $\mathbf{x} = (1, 2, \ldots, 10)$ and indexes 4 and 7 are selected, then $\mathbf{y} = (1, 2, 3, 7, 6, 5, 4, 8, 9, 10)$; see Figure 6.12. Another example is: if $\mathbf{x} = (6, 7, 2, 8, 3, 9, 10, 5, 4, 1)$ and indexes 6 and 10 are selected, then $\mathbf{y} = (6, 7, 2, 8, 3, 1, 4, 5, 10, 9)$.

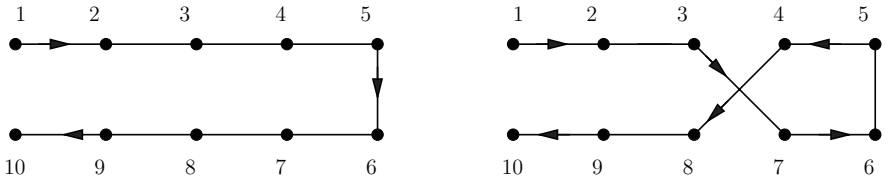

Figure 6.12: Illustration of the 2-opt neighborhood structure.

Third, we apply the Metropolis–Hastings algorithm to sample from the target. We need to supply a transition function $q(\mathbf{x}, \mathbf{y})$ from $\mathbf{x}$ to one of its neighbors. Typically, the two indexes for the 2-opt neighborhood are selected uniformly. This can be done, for example, by drawing a uniform permutation of $(1, \ldots, n)$ (see Section 2.10) and then selecting the first two elements of this permutation. The transition function is here constant: $q(\mathbf{x}, \mathbf{y}) = q(\mathbf{y}, \mathbf{x}) = 1/\binom{n}{2}$. It follows that in this case the acceptance probability is

$$\alpha = \min\left\{\frac{f(\mathbf{y})}{f(\mathbf{x})},\, 1\right\} = \begin{cases} 1 & \text{if } S(\mathbf{y}) \leqslant S(\mathbf{x}) \\ \mathrm{e}^{-(S(\mathbf{y})-S(\mathbf{x}))/T} & \text{if } S(\mathbf{y}) > S(\mathbf{x})\,. \end{cases} \tag{6.19}$$

As we gradually decrease the temperature T, the Boltzmann distribution becomes more and more concentrated around the global minimizer. Common practice is to decrease the temperature as $T_{t+1} = \beta\, T_t$ for some $\beta < 1$ close to 1, such as $\beta = 0.99$. This leads to the following generic simulated annealing algorithm with Metropolis–Hastings sampling:

Algorithm 6.8.1: Simulated Annealing: Metropolis–Hastings Sampling

input : Objective function S, starting state $\mathbf{X}_0$, initial temperature T_0, number of iterations N, symmetric proposal function $q(\mathbf{x}, \mathbf{y})$, constant β.

output: Approximate minimum value of S and corresponding minimizer.

for $t = 0$ **to** $N - 1$ **do**
  Generate a new state $\mathbf{Y}$ from the symmetric proposal $q(\mathbf{X}_t, \mathbf{y})$.
  if $S(\mathbf{Y}) < S(\mathbf{X}_t)$ **then**
    $\mathbf{X}_{t+1} \leftarrow \mathbf{Y}$
  else
    Draw $U \sim \mathsf{U}(0, 1)$.
    if $U \leqslant \mathrm{e}^{-(S(\mathbf{Y})-S(\mathbf{X}_t))/T_t}$ **then**
      $\mathbf{X}_{t+1} \leftarrow \mathbf{Y}$
    else
      $\mathbf{X}_{t+1} \leftarrow \mathbf{X}_t$
  $T_{t+1} \leftarrow \beta\, T_t$
return $S(\mathbf{X}_N)$ and $\mathbf{X}_N$

Instead of stopping after a fixed number N of iterations, it is useful to stop when consecutive function values are closer than some distance ε to each other, or when the best found function value has not changed over a fixed number d of iterations.

EXAMPLE 6.13 *n*-Queens Problem

In the n-queens problem, the objective is to arrange n queens on a $n \times n$ chessboard in such a way that no queen can capture another queen. An illustration is given in Figure 6.13 for the case $n = 8$. Note that the configuration in Figure 6.13 does not solve the problem. We take $n = 8$ from now on. Note that each row of the chessboard must contain exactly one queen. Denote the position of the queen in the i-th row by x_i; this way each configuration can be represented by a vector $\mathbf{x} = (x_1, \ldots, x_8)$. For example, $\mathbf{x} = (2, 3, 7, 4, 8, 5, 1, 6)$ corresponds to the large configuration in Figure 6.13. Two other examples are given in the same figure. We can now formulate the problem in terms of minimizing a function $S(\mathbf{x})$ that represents the amount of "threat" of the queens. For this we simply add the number of queens *minus* 1, for each column and diagonal that have at least two queens present. For the large configuration in Figure 6.13 there are only two diagonals with two queens, so the score is $(2-1)+(2-1) = 2$. Note that the minimal S value is 0. One of the optimal solutions is $\mathbf{x}^* = (5, 1, 8, 6, 3, 7, 2, 4)$.

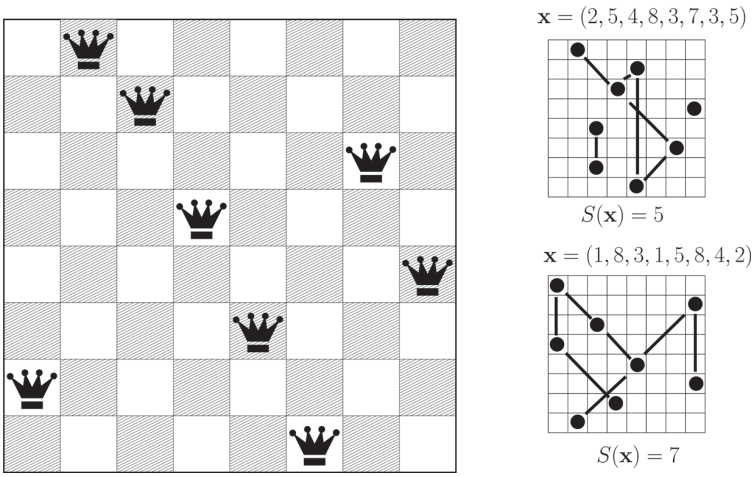

Figure 6.13: Position the eight queens such that no queen can capture another.

We show next how this optimization problem can be solved via simulated annealing using the Gibbs sampler. As in the previous TSP example, each iteration of the algorithm consists of sampling from the Boltzmann pdf $f(\mathbf{x}) = \mathrm{e}^{-S(\mathbf{x})/T}$ via the Gibbs sampler, followed by decreasing the temperature. This leads to the following generic simulated annealing algorithm using Gibbs sampling:

Algorithm 6.8.2: Simulated Annealing: Gibbs Sampling

```
1  Initialize X_0.
2  t ← 0
3  while S(X_t) > 0 do
4      Draw Y_1 from the conditional pdf f(y_1 | X_{t,2}, ..., X_{t,n}).
5      for i = 2 to n do
6          Draw Y_i from the conditional pdf f(y_i | Y_1, ..., Y_{i-1}, X_{t,i+1}, ..., X_{t,n}).
7      X_{t+1} ← Y
8      T_{t+1} ← β T_t
9      t ← t + 1
10 return X_t
```

Note that in Line 6 each Y_i is drawn from a discrete distribution on $\{1,\ldots,n\}$ with probabilities proportional to $\mathrm{e}^{-S(\mathbf{Z}_1)/T_t},\ldots,\mathrm{e}^{-S(\mathbf{Z}_n)/T_t}$, where each $\mathbf{Z}_k$ is equal to the vector $(Y_1,\ldots,Y_{i-1},k,X_{t,i+1},\ldots,X_{t,n})$.

Other MCMC samplers can be used in simulated annealing. For example, in the *hide-and-seek* algorithm [21] the general hit-and-run sampler (Section 6.3) is used. Research motivated by the use of hit-and-run and discrete hit-and-run in simulated annealing, has resulted in the development of a theoretically derived cooling schedule that uses the recorded values obtained during the course of the algorithm to adaptively update the temperature [23, 24].

6.9 PERFECT SAMPLING

Returning to the beginning of this chapter, suppose that we wish to generate a random variable X taking values in $\{1,\ldots,m\}$ according to a target distribution $\pi=\{\pi_i\}$. As mentioned, one of the main drawbacks of the MCMC method is that each sample X_t is only *asymptotically* distributed according to π, that is, $\lim_{t\to\infty}\mathbb{P}(X_t=i)=\pi_i$. In contrast, *perfect sampling* is an MCMC technique that produces exact samples from π.

Let $\{X_t\}$ be a Markov chain with state space $\{1,\ldots,m\}$, transition matrix P, and stationary distribution π. We wish to generate $\{X_t,\, t=0,-1,-2,\ldots\}$ in such a way that X_0 has the desired distribution. We can draw X_0 from the m-point distribution corresponding to the X_{-1}-th row of P; see Algorithm 2.7.1. This can be done via the IT method, which requires the generation of a random variable $U_0\sim\mathsf{U}(0,1)$. Similarly, X_{-1} can be generated from X_{-2} and $U_{-1}\sim\mathsf{U}(0,1)$. In general, we see that for any negative time $-t$, the random variable X_0 depends on X_{-t} and the independent random variables $U_{-t+1},\ldots,U_0\sim\mathsf{U}(0,1)$.

Next, let us consider m dependent copies of the Markov chain, starting from each of the states $1,\ldots,m$ and using the *same* random numbers $\{U_i\}$ — as in the CRV method. Then, if two paths coincide, or *coalesce*, at some time, from that time on, both paths will be identical. The paths are said to be *coupled*. The main point of the perfect sampling method is that if the chain is ergodic (in particular, if it is aperiodic and irreducible), then *with probability 1 there exists a negative time $-T$ such that all m paths will have coalesced before or at time 0.* The situation is illustrated in Figure 6.14.

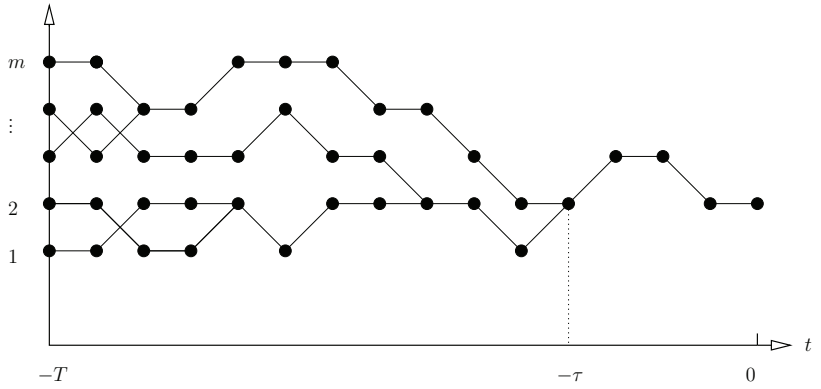

Figure 6.14: All Markov chains have coalesced at time $-\tau$.

Let $\mathbf{U}$ represent the vector of all $U_t, t \leqslant 0$. For each $\mathbf{U}$ we know there exists, with probability 1, a $-T(\mathbf{U}) < 0$ such that by time 0 all m coupled chains defined by $\mathbf{U}$ have coalesced. Moreover, if we start at time $-T$, a *stationary* version of the Markov chain, using again the same $\mathbf{U}$, this stationary chain must, at time $t = 0$, have coalesced with the other ones. Thus, any of the m chains has at time 0 the same distribution as the stationary chain, which is π.

Note that in order to construct T, we do not need to know the whole (infinite vector) $\mathbf{U}$. Instead, we can work backward from $t = 0$ by generating U_{-1} first, and checking if $-T = -1$. If this is not the case, generate U_{-2} and check if $-T = -2$, and so on. This leads to the following algorithm, due to Propp and Wilson [19], called *coupling from the past*:

Algorithm 6.9.1: Coupling from the Past

```
Generate U_0 ~ U(0,1).
U_0 ← U_0
t ← −1
NotCoalesced ← true
while NotCoalesced do
    Generate m Markov chains {X_i, i = −t, ..., 0}, starting at t from each of
     the states 1, ..., m, and using the same random vector U_{t+1}.
    if all chains have coalesced before or at time 0 then
        NotCoalesced ← false
    else
        Generate U_t ~ U(0,1).
        U_t ← (U_t, U_{t+1})
        t ← t − 1
return X_0
```

Although perfect sampling seems indeed "perfect" in that it returns an exact sample from the target π rather than an approximate one, practical applications of the technique are, presently, quite limited. Not only is the technique difficult or impossible to use for most continuous simulation systems, it is also much more computationally intensive than simple MCMC.

PROBLEMS

6.1 Verify that the local balance equation (6.3) holds for the Metropolis–Hastings algorithm.

6.2 When running an MCMC algorithm, it is important to know when the transient (or *burn-in*) period has finished; otherwise, steady-state statistical analyses such as those in Section 4.4.2 may not be applicable. In practice, this is often done via a visual inspection of the sample path. Run the random walk sampler with normal target distribution $\mathsf{N}(10, 1)$ and proposal $Y \sim \mathsf{N}(x, 0.01)$. Take a sample size of $N = 5000$. Determine roughly when the process reaches stationarity.

6.3 A useful tool for examining the behavior of a stationary process $\{X_t\}$ obtained, for example, from an MCMC simulation, is the covariance function $R(t) = \mathrm{Cov}(X_t, X_0)$; see Example 6.4. Estimate the covariance function for the process in Problem 6.2 and plot the results. In Matlab's *signal processing* toolbox, this is implemented under the M-function `xcov.m`. Try different proposal distributions of the form $\mathsf{N}(x, \sigma^2)$ and observe how the covariance function changes.

6.4 Implement the independence sampler with an $\mathsf{Exp}(1)$ target and an $\mathsf{Exp}(\lambda)$ proposal distribution for several values of λ. Similar to the importance sampling situation, things go awry when the sampling distribution gets too far from the target distribution, in this case when $\lambda > 2$. For each run, use a sample size of 10^5 and start with $x = 1$.

- **a)** For each value $\lambda = 0.2, 1, 2$, and 5, plot a histogram of the data and compare it with the true pdf.
- **b)** For each value of λ given above, calculate the sample mean and repeat this for 20 independent runs. Make a dotplot of the data (plot them on a line) and notice the differences. Observe that for $\lambda = 5$ most of the sample means are below 1, and thus underestimate the true expectation 1, but a few are significantly greater. Observe also the behavior of the corresponding auto-covariance functions, both between the different λs and, for $\lambda = 5$, within the 20 runs.

6.5 Implement the random walk sampler with an $\mathsf{Exp}(1)$ target distribution, where Z (in the proposal $Y = x + Z$) has a double exponential distribution with parameter λ. Carry out a study similar to that in Problem 6.4 for different values of λ, say $\lambda = 0.1, 1, 5, 20$. Observe that (in this case) the random walk sampler has a more stable behavior than the independence sampler.

6.6 Let $\mathbf{X} = (X, Y)^\top$ be a random column vector with a bivariate normal distribution with expectation vector $\mathbf{0} = (0, 0)^\top$ and covariance matrix

$$\Sigma = \begin{pmatrix} 1 & \varrho \\ \varrho & 1 \end{pmatrix} .$$

- **a)** Show that $(Y \mid X = x) \sim \mathsf{N}(\varrho\, x, 1 - \varrho^2)$ and $(X \mid Y = y) \sim \mathsf{N}(\varrho\, y, 1 - \varrho^2)$.
- **b)** Write a systematic Gibbs sampler to draw 10^4 samples from the bivariate distribution $\mathsf{N}(\mathbf{0}, \Sigma)$ and plot the data for $\varrho = 0, 0.7$ and 0.9.

6.7 A remarkable feature of the Gibbs sampler is that the conditional distributions in Algorithm 6.4.1 contain sufficient information to generate a sample from the joint distribution. The following result (by Hammersley and Clifford [10]) shows

that it is possible to directly express the joint pdf in terms of the conditional pdfs. Namely,

$$f(x,y) = \frac{f_{Y|X}(y|x)}{\int \frac{f_{Y|X}(y|x)}{f_{X|Y}(x|y)}\,\mathrm{d}y}\,.$$

Prove this. Generalize this to the n-dimensional case.

6.8 In the Ising model the *expected magnetization per spin* is given by

$$M(T) = \frac{1}{n^2}\,\mathbb{E}_{\pi_T}\left[\sum_i S_i\right],$$

where π_T is the Boltzmann distribution at temperature T. Estimate $M(T)$, for example via the Swendsen–Wang algorithm, for various values of $T \in [0,5]$, and observe that the graph of $M(T)$ changes sharply around the critical temperature $T \approx 2.61$. Take $n = 20$ and use periodic boundaries.

6.9 Run Peter Young's Java applet in

`http://physics.ucsc.edu/~peter/java/ising/ising.html`

to gain a better understanding of how the Ising model works.

6.10 As in Example 6.6, let $\mathscr{X}^* = \{\mathbf{x} : \sum_{i=1}^n x_i = m,\ x_i \in \{0,\ldots,m\},\ i = 1,\ldots,n\}$. Show that this set has $\binom{m+n-1}{n-1}$ elements.

6.11 In a simple model for a closed queueing network with n queues and m customers, it is assumed that the service times are independent and exponentially distributed, with rate μ_i for queue i, $i = 1,\ldots,n$. After completing service at queue i, the customer moves to queue j with probability p_{ij}. The $\{p_{ij}\}$ are the so-called *routing probabilities*.

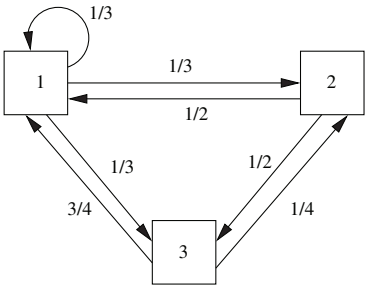

Figure 6.15: A closed queueing network.

It can be shown (e.g., see [13]) that the stationary distribution of the number of customers in the queues is of product form (6.7), with f_i being the pdf of the $\mathsf{G}(1 - y_i/\mu_i)$ distribution; thus $f_i(x_i) \propto (y_i/\mu_i)^{x_i}$. Here the $\{y_i\}$ are constants that are obtained from the following set of *flow balance* equations:

$$y_i = \sum_j y_j\, p_{ji}\,, \quad i = 1,\ldots,n\,, \tag{6.20}$$

which has a one-dimensional solution space. Without loss of generality, y_1 can be set to 1 to obtain a unique solution.

Consider now the specific case of the network depicted in Figure 6.15, with $n = 3$ queues. Suppose that the service rates are $\mu_1 = 2$, $\mu_2 = 1$, and $\mu_3 = 1$. The routing probabilities are given in the figure.

a) Show that a solution to (6.20) is $(y_1, y_2, y_3) = (1, 10/21, 4/7)$.

b) For $m = 50$ determine the exact normalization constant C.

c) Implement the procedure of Example 6.6 to estimate C via MCMC, and compare the estimate for $m = 50$ with the exact value.

6.12 Let $X_1, \ldots, X_n$ be a random sample from the $\mathsf{N}(\mu, \sigma^2)$ distribution. Consider the following Bayesian model:

- $f(\mu, \sigma^2) = 1/\sigma^2$;
- $(\mathbf{x}_i \,|\, \mu, \sigma) \sim \mathsf{N}(\mu, \sigma^2)$, $i = 1, \ldots, n$ independently.

Note that the prior for (μ, σ^2) is *improper*. That is, it is not a pdf in itself, but if we obstinately apply Bayes' formula, it does yield a proper posterior pdf. In some sense it conveys the least amount of information about μ and σ^2. Let $\mathbf{x} = (x_1, \ldots, x_n)$ represent the data. The posterior pdf is given by

$$f(\mu, \sigma^2 \,|\, \mathbf{x}) = \left(2\,\pi\,\sigma^2\right)^{-n/2} \exp\left\{-\frac{1}{2}\frac{\sum_i (x_i - \mu)^2}{\sigma^2}\right\} \frac{1}{\sigma^2}\,.$$

We wish to sample from this distribution via the Gibbs sampler.

a) Show that $(\mu \,|\, \sigma^2, \mathbf{x}) \sim \mathsf{N}(\bar{x},\, \sigma^2/n)$, where $\bar{x}$ is the sample mean.

b) Prove that

$$f(\sigma^2 \,|\, \mu, \mathbf{x}) \propto \frac{1}{(\sigma^2)^{n/2+1}} \exp\left(-\frac{n}{2}\frac{V_\mu}{\sigma^2}\right), \tag{6.21}$$

where $V_\mu = \sum_i (x_i - \mu)^2/n$ is the classical sample variance for known μ. In other words, $(1/\sigma^2 \,|\, \mu, \mathbf{x}) \sim \mathsf{Gamma}(n/2, nV_\mu/2)$.

c) Implement a Gibbs sampler to sample from the posterior distribution, taking $n = 100$. Run the sampler for 10^5 iterations. Plot the histograms of $f(\mu \,|\, \mathbf{x})$ and $f(\sigma^2 \,|\, \mathbf{x})$, and find the sample means of these posteriors. Compare them with the classical estimates.

d) Show that the true posterior pdf of μ, given the data, is

$$f(\mu \,|\, \mathbf{x}) \propto \left((\mu - \bar{x})^2 + V\right)^{-n/2},$$

where $V = \sum_i (x_i - \bar{x})^2/n$. [Hint: To evaluate the integral

$$f(\mu \,|\, \mathbf{x}) = \int_0^\infty f(\mu, \sigma^2 \,|\, \mathbf{x})\, \mathrm{d}\sigma^2$$

write it first as $(2\pi)^{-n/2} \int_0^\infty t^{n/2-1} \exp(-\frac{1}{2}\, t\, c)\, \mathrm{d}t$, where $c = n\, V_\mu$, by applying the change of variable $t = 1/\sigma^2$. Show that the latter integral is proportional to $c^{-n/2}$. Finally, apply the decomposition $V_\mu = (\bar{x} - \mu)^2 + V$.]

6.13 Suppose that $f(\boldsymbol{\theta} \,|\, \mathbf{x})$ is the posterior pdf for some Bayesian estimation problem. For example, $\boldsymbol{\theta}$ could represent the parameters of a regression model based on the data $\mathbf{x}$. An important use for the posterior pdf is to make predictions

about the distribution of other random variables. For example, suppose that the pdf of some random variable $\mathbf{Y}$ depends on $\boldsymbol{\theta}$ via the conditional pdf $f(\mathbf{y} \,|\, \boldsymbol{\theta})$. The *predictive pdf* of Y given $\mathbf{x}$ is defined as

$$f(\mathbf{y} \,|\, \mathbf{x}) = \int f(\mathbf{y} \,|\, \boldsymbol{\theta}) f(\boldsymbol{\theta} \,|\, \mathbf{x}) \, \mathrm{d}\boldsymbol{\theta} \,,$$

which can be viewed as the expectation of $f(\mathbf{y} \,|\, \boldsymbol{\theta})$ under the posterior pdf. Therefore, we can use Monte Carlo simulation to approximate $f(\mathbf{y} \,|\, \mathbf{x})$ as

$$f(\mathbf{y} \,|\, \mathbf{x}) \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{y} \,|\, \boldsymbol{\theta}_i) \,,$$

where the sample $\{\boldsymbol{\theta}_i, i = 1, \ldots, N\}$ is obtained from $f(\boldsymbol{\theta} \,|\, \mathbf{x})$ (e.g., via MCMC).

As a concrete application, suppose that the independent measurement data $-0.4326, -1.6656, 0.1253, 0.2877, -1.1465$ come from some $\mathsf{N}(\mu, \sigma^2)$ distribution. Define $\boldsymbol{\theta} = (\mu, \sigma^2)$. Let $Y \sim \mathsf{N}(\mu, \sigma^2)$ be a new measurement. Estimate and draw the predictive pdf $f(y \,|\, \mathbf{x})$ from a sample $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ obtained via the Gibbs sampler of Problem 6.12. Take $N = 10{,}000$. Compare this with the "common-sense" Gaussian pdf with expectation $\bar{x}$ (sample mean) and variance s^2 (sample variance).

6.14 In the *zero-inflated Poisson* (ZIP) model, random data $X_1, \ldots, X_n$ are assumed to be of the form $X_i = R_i Y_i$, where the $\{Y_i\}$ have a $\mathsf{Poi}(\lambda)$ distribution and the $\{R_i\}$ have a $\mathsf{Ber}(p)$ distribution, all independent of each other. Given an outcome $\mathbf{x} = (x_1, \ldots, x_n)$, the objective is to estimate both λ and p. Consider the following hierarchical Bayes model:

- $p \sim \mathsf{U}(0, 1)$ (prior for p),
- $(\lambda \,|\, p) \sim \mathsf{Gamma}(a, b)$ (prior for λ),
- $(r_i \,|\, p, \lambda) \sim \mathsf{Ber}(p)$ independently (from the model above),
- $(x_i \,|\, \mathbf{r}, \lambda, p) \sim \mathsf{Poi}(\lambda\, r_i)$ independently (from the model above),

where $\mathbf{r} = (r_1, \ldots, r_n)$ and a and b are known parameters. It follows that

$$f(\mathbf{x}, \mathbf{r}, \lambda, p) = \frac{b^a \lambda^{a-1} \mathrm{e}^{-b\lambda}}{\Gamma(a)} \prod_{i=1}^{n} \frac{\mathrm{e}^{-\lambda\, r_i} (\lambda\, r_i)^{x_i}}{x_i!} \, p^{r_i} (1-p)^{1-r_i} \,.$$

We wish to sample from the posterior pdf $f(\lambda, p, \mathbf{r} \,|\, \mathbf{x})$ using the Gibbs sampler.

a) Show that

1. $(\lambda \,|\, p, \mathbf{r}, \mathbf{x}) \sim \mathsf{Gamma}(a + \sum_i x_i,\ b + \sum_i r_i)$.
2. $(p \,|\, \lambda, \mathbf{r}, \mathbf{x}) \sim \mathsf{Beta}(1 + \sum_i r_i,\ n + 1 - \sum_i r_i)$.
3. $(r_i \,|\, \lambda, p, \mathbf{x}) \sim \mathsf{Ber}\left(\frac{p\, \mathrm{e}^{-\lambda}}{p\, \mathrm{e}^{-\lambda} + (1-p) I_{\{x_i = 0\}}}\right)$.

b) Generate a random sample of size $n = 100$ for the ZIP model using parameters $p = 0.3$ and $\lambda = 2$.

c) Implement the Gibbs sampler, generate a large (dependent) sample from the posterior distribution, and use this to construct 95% Bayesian CIs for p and λ using the data in b). Compare your results with the true values.

6.15 Show that μ in (6.12) satisfies the local balance equations

$$\mu(\mathbf{x},\mathbf{y})\,\mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')] = \mu(\mathbf{x}',\mathbf{y}')\,\mathbf{R}[(\mathbf{x}',\mathbf{y}'),(\mathbf{x},\mathbf{y})]\ .$$

Thus μ is stationary with respect to $\mathbf{R}$, that is, $\mu\mathbf{R} = \mu$. Show that μ is also stationary with respect to $\mathbf{Q}$. Show also that μ is stationary with respect to $\mathbf{P} = \mathbf{QR}$.

6.16 To show that the systematic Gibbs sampler is a special case of the generalized Markov sampler, take $\mathscr{Y}$ to be the set of indexes $\{1,\ldots,n\}$, and define for the Q-step

$$\mathbf{Q}_{\mathbf{x}}(y,y') = \begin{cases} 1 & \text{if } y' = y+1 \text{ or } y' = 1, y = n, \\ 0 & \text{otherwise.} \end{cases}$$

Let the set of possible transitions $\mathscr{R}(\mathbf{x},y)$ be the set of vectors $\{(\mathbf{x}',y)\}$ such that all coordinates of $\mathbf{x}'$ are the same as those of $\mathbf{x}$ except for possibly the y-th coordinate.

a) Show that the stationary distribution of $\mathbf{Q}_{\mathbf{x}}$ is $q_{\mathbf{x}}(y) = 1/n$, for $y = 1,\ldots,n$.

b) Show that

$$\mathbf{R}[(\mathbf{x},y),(\mathbf{x}',y)] = \frac{f(\mathbf{x}')}{\displaystyle\sum_{(\mathbf{z},y)\in\mathscr{R}(\mathbf{x},y)} f(\mathbf{z})}, \qquad \text{for } (\mathbf{x}',y)\in\mathscr{R}(\mathbf{x},y)\ .$$

c) Compare with Algorithm 6.4.1.

6.17 Prove that the Metropolis–Hastings algorithm is a special case of the generalized Markov sampler. [Hint: Let the auxiliary set $\mathscr{Y}$ be a copy of the target set $\mathscr{X}$, let $\mathbf{Q}_{\mathbf{x}}$ correspond to the transition function of the Metropolis–Hastings algorithm (i.e., $\mathbf{Q}_{\mathbf{x}}(\cdot,\mathbf{y}) = q(\mathbf{x},\mathbf{y})$), and define $\mathscr{R}(\mathbf{x},\mathbf{y}) = \{(\mathbf{x},\mathbf{y}),(\mathbf{y},\mathbf{x})\}$. Use arguments similar to those for the Markov jump sampler (see (6.17)) to complete the proof.]

6.18 Barker's and Hastings' MCMC algorithms differ from the symmetric Metropolis sampler only in that they define the acceptance ratio $\alpha(\mathbf{x},\mathbf{y})$ to be, respectively, $f(\mathbf{y})/(f(\mathbf{x}) + f(\mathbf{y}))$ and $s(\mathbf{x},\mathbf{y})/(1 + 1/\varrho(\mathbf{x},\mathbf{y}))$ instead of $\min\{f(\mathbf{y})/f(\mathbf{x}),1\}$. Here, $\varrho(\mathbf{x},\mathbf{y}) = f(\mathbf{y})q(\mathbf{y},\mathbf{x})/(f(\mathbf{x})q(\mathbf{x},\mathbf{y}))$ and s is any symmetric function such that $0 \leqslant \alpha(\mathbf{x},\mathbf{y}) \leqslant 1$. Show that both are special cases of the generalized Markov sampler. [Hint: Take $\mathscr{Y} = \mathscr{X}$.]

6.19 Implement the simulated annealing algorithm for the n-queens problem suggested in Example 6.13. How many solutions can you find?

6.20 Implement the Metropolis–Hastings based simulated annealing algorithm for the TSP in Example 6.12. Run the algorithm on some test problems in

`http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

6.21 Write a simulated annealing algorithm based on the random walk sampler to maximize the function

$$S(x) = \left|\frac{\sin^8(10x) + \cos^5(5x+1)}{x^2 - x + 1}\right|, \quad x\in\mathbb{R}\ .$$

Use a $\mathsf{N}(x, \sigma^2)$ proposal function, given the current state x. Start with $x = 0$. Plot the current best function value against the number of evaluations of S for various values of σ and various annealing schedules. Repeat the experiments several times to assess what works best.

Further Reading

MCMC is one of the principal tools of statistical computing and Bayesian analysis. A comprehensive discussion of MCMC techniques can be found in [20], and practical applications are discussed in [8]. See also [4]. For more details on the use of MCMC in Bayesian analysis, we refer to [6]. A classical reference on simulated annealing is [1]. More general global search algorithms may be found in [26]. An influential paper on stationarity detection in Markov chains, which is closely related to perfect sampling, is [3].

REFERENCES

1. E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, Chichester, 1989.
2. D. J. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs*. In preparation. `http://www.stat.berkeley.edu /users/aldous/book.html`, 2007.
3. S. Asmussen, P. W. Glynn, and H. Thorisson. Stationary detection in the initial transient problem. *ACM Transactions on Modeling and Computer Simulation*, 2(2):130–157, 1992.
4. S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.
5. M.-H. Chen and B. W. Schmeiser. General hit-and-run Monte Carlo sampling for evaluating multidimensional integrals. *Operations Research Letters*, 19(4):161–169, 1996.
6. A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall, New York, 2nd edition, 2003.
7. S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on PAMI*, 6:721–741, 1984.
8. W.R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, New York, 1996.
9. P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.
10. J. Hammersley and M. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1970.
11. W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:92–109, 1970.
12. J. M. Keith, D. P. Kroese, and D. Bryant. A generalized Markov chain sampler. *Methodology and Computing in Applied Probability*, 6(1):29–53, 2004.
13. F. P. Kelly. *Reversibility and Stochastic Networks*. John Wiley & Sons, Chichester, 1979.

14. J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, New York, 2001.

15. L. Lovász. Hit-and-run mixes fast. *Mathematical Programming*, 86:443–461, 1999.

16. L. Lovász and S. S. Vempala. Hit-and-run is fast and fun. Technical report, Microsoft Research, SMS-TR, 2003.

17. L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.

18. M. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

19. J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 1 & 2:223–252, 1996.

20. C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 2nd edition, 2004.

21. H. E. Romeijn and R. L. Smith. Simulated annealing for constrained global optimization. *Journal of Global Optimization*, 5:101–126, 1994.

22. S. M. Ross. *Simulation*. Academic Press, New York, 3rd edition, 2002.

23. Y. Shen. *Annealing Adaptive Search with Hit-and-Run Sampling Methods for Stochastic Global Optimization Algorithms*. PhD thesis, University of Washington, 2005.

24. Y. Shen, S. Kiatsupaibul, Z. B. Zabinsky, and R. L. Smith. An analytically derived cooling schedule for simulated annealing. *Journal of Global Optimization*, 38(3):333–365, 2007.

25. R. L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32:1296–1308, 1984.

26. Z. B. Zabinsky. *Stochastic Adaptive Search for Global Optimization*. Kluwer Academic Publishers, Dordrecht, 2003.

27. Z. B. Zabinsky, R. L. Smith, J. F. McDonald, H. E. Romeijn, and D. E. Kaufman. Improving hit-and-run for global optimization. *Journal of Global Optimization*, 3:171–192, 1993.

CHAPTER 7

SENSITIVITY ANALYSIS AND MONTE CARLO OPTIMIZATION

7.1 INTRODUCTION

As discussed in Chapter 3, many real-world complex systems in science and engineering can be modeled as *discrete-event systems*. The behavior of such systems is identified via a sequence of discrete events, which causes the system to change from one state to another. Examples include traffic systems, flexible manufacturing systems, computer-communications systems, inventory systems, production lines, coherent lifetime systems, Program Evaluation and Review Technique (PERT) networks, and flow networks. A discrete-event system can be classified as either *static* or *dynamic*. The former are called *discrete-event static systems* (DESS), while the latter are called *discrete-event dynamic systems* (DEDS). The main difference is that DESS do not evolve over time, while DEDS do. The PERT network is a typical example of a DESS, with the sample performance being, e.g., the shortest path in the network. A queueing network, such as the Jackson network in Section 3.4.1, is an example of a DEDS, with the sample performance being, for example, the delay (waiting time of a customer) in the network. In this chapter we will deal mainly with DESS. For a comprehensive study of both DESS and DEDS, the reader is referred to [12], [17], and [21].

Because of their complexity, the performance evaluation of discrete-event systems is usually done by simulation, and it is often associated with the estimation of a response function $\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})]$, where the distribution of the sample performance $H(\mathbf{X})$ depends on the control or reference parameter $\mathbf{u} \in \mathscr{V}$. *Sensitivity*

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

analysis is concerned with evaluating sensitivities (gradients, Hessians, etc.) of the response function $\ell(\mathbf{u})$ with respect to parameter vector $\mathbf{u}$, and is based on the score function and Fisher information. It provides guidance for design and operational decisions and plays an important role in selecting system parameters that optimize certain performance measures.

To illustrate, consider the following examples:

1. **Stochastic networks.** Sensitivity analysis can be employed to minimize the mean shortest path in the network with respect, say, to network link parameters, subject to certain constraints. PERT networks and flow networks are common examples. In the former, input and output variables may represent activity durations and minimum project duration, respectively. In the latter, they may represent flow capacities and maximal flow capacities.

2. **Traffic light systems.** The performance measure might be a vehicle's average delay as it proceeds from a given origin to a given destination or the average number of vehicles waiting for a green light at a given intersection. The sensitivity and decision parameters might be the average rate at which vehicles arrive at intersections and the rate of light changes from green to red. Some performance issues of interest are:

   (a) What will the vehicle's average delay be if the interarrival rate at a given intersection increases (decreases), say, by 10–50%? What would be the corresponding impact of adding one or more traffic lights to the system?

   (b) Which parameters are most significant in causing bottlenecks (high congestion in the system), and how can these bottlenecks be prevented or removed most effectively?

   (c) How can the average delay in the system be minimized, subject to certain constraints?

We will distinguish between the so-called *distributional* sensitivity parameters and the *structural* ones. In the former case, we are interested in sensitivities of the expected performance

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})] = \int H(\mathbf{x}) f(\mathbf{x}; \mathbf{u}) \, \mathrm{d}\mathbf{x} \tag{7.1}$$

with respect to the parameter vector $\mathbf{u}$ of the pdf $f(\mathbf{x}; \mathbf{u})$, while in the latter case, we are interested in sensitivities of the expected performance

$$\ell(\mathbf{u}) = \mathbb{E}[H(\mathbf{X}; \mathbf{u})] = \int H(\mathbf{x}; \mathbf{u}) f(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{7.2}$$

with respect to the parameter vector $\mathbf{u}$ in the sample performance $H(\mathbf{x}; \mathbf{u})$. As an example, consider a $GI/G/1$ queue. In the first case, $\mathbf{u}$ might be the vector of the inter-arrival and service rates, and in the second case, $\mathbf{u}$ might be the buffer size. Note that often the parameter vector $\mathbf{u}$ includes both the distributional and structural parameters. In such a case we will use the following notation:

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_2}[H(\mathbf{X}; \mathbf{u}_1)] = \int H(\mathbf{x}; \mathbf{u}_1) f(\mathbf{x}; \mathbf{u}_2) \, \mathrm{d}\mathbf{x} \,, \tag{7.3}$$

where $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$. Note that $\ell(\mathbf{u})$ in (7.1) and (7.2) can be considered particular cases of $\ell(\mathbf{u})$ in (7.3), where the corresponding sizes of the vectors $\mathbf{u}_1$ or $\mathbf{u}_2$ equal 0.

EXAMPLE 7.1

Let $H(\mathbf{X}; u_3, u_4) = \max\{X_1 + u_3, X_2 + u_4\}$, where $\mathbf{X} = (X_1, X_2)$ is a two-dimensional vector with independent components and $X_i \sim f_i(X; u_i)$, $i = 1, 2$. In this example u_1 and u_2 are distributional parameters, and u_3 and u_4 are structural ones.

Consider the following minimization problem using representation (7.3):

$$
\begin{array}{lll}
\text{minimize} & \ell_0(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_0(\mathbf{X}; \mathbf{u}_2)], & \mathbf{u} \in \mathscr{V}, \\
\text{subject to:} & \ell_j(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_j(\mathbf{X}; \mathbf{u}_2)] \leqslant 0, & j = 1, \ldots, k, \\
& \ell_j(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_j(\mathbf{X}; \mathbf{u}_2)] = 0, & j = k+1, \ldots, M,
\end{array}
\qquad (\mathrm{P}_0) \qquad (7.4)
$$

where $H_j(\mathbf{X})$ is the j-th sample performance, driven by an input vector $\mathbf{X} \in \mathbb{R}^n$ with pdf $f(\mathbf{x}; \mathbf{u}_1)$, and $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ is a decision parameter vector belonging to some parameter set $\mathscr{V} \subset \mathbb{R}^m$.

When the objective function $\ell_0(\mathbf{u})$ and the constraint functions $\ell_j(\mathbf{u})$ are available analytically, (P_0) becomes a standard nonlinear programming problem, which can be solved either analytically or numerically by standard nonlinear programming techniques. For example, the Markovian queueing system optimization falls within this domain. Here, however, it will be assumed that the objective function and some of the constraint functions in (P_0) are not available analytically (typically due to the complexity of the underlying system), so that one must resort to stochastic optimization methods, particularly Monte Carlo optimization.

The rest of this chapter is organized as follows: Section 7.2 deals with sensitivity analysis of DESS with respect to the distributional parameters. Here we introduce the celebrated *score function* (SF) method. Section 7.3 deals with simulation-based optimization for programs of type (P_0) when the expected values $\mathbb{E}_{\mathbf{u}_1}[H_j(\mathbf{X}, \mathbf{u}_2)]$ are replaced by their corresponding sample means. The simulation-based version of (P_0) is called the *stochastic counterpart* of the original program (P_0). The main emphasis will be placed on the stochastic counterpart of the unconstrained program (P_0). Here we show how the stochastic counterpart method can approximate quite efficiently the true unknown optimal solution of the program (P_0) using a single simulation. Our results are based on [16, 18, 19], where theoretical foundations of the stochastic counterpart method are established. It is interesting to note that Geyer and Thompson [3] independently discovered the stochastic counterpart method in 1995. They used it to make statistical inference for a particular unconstrained setting of the general program (P_0). Section 7.4 presents an introduction to sensitivity analysis and simulation-based optimization of DEDS. Particular emphasis is placed on sensitivity analysis with respect to the distributional parameters of Markov chains using the dynamic version of the SF method. For a comprehensive study of sensitivity analysis and optimization of DEDS, including different types of queueing and inventory models, the reader is referred to [17].

7.2 SCORE FUNCTION METHOD FOR SENSITIVITY ANALYSIS OF DESS

In this section we introduce the celebrated *score function (SF) method* for sensitivity analysis of DESS. The goal of the SF method is to estimate the gradient and higher derivatives of $\ell(\mathbf{u})$ with respect to the distributional parameter vector $\mathbf{u}$, where the expected performance is given (see (7.1)) by

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})] \,,$$

with $\mathbf{X} \sim f(\mathbf{x};\mathbf{u})$. As we will see below, the SF approach permits the estimation of *all* sensitivities (gradients, Hessians, etc.) from a *single simulation run* (experiment) for a DESS with tens and quite often with hundreds of parameters. We closely follow [17].

Consider first the case where $\mathbf{u}$ is scalar (denoted therefore u instead of $\mathbf{u}$) and assume that the parameter set $\mathscr{V}$ is an open interval on the real line. Suppose that for all $\mathbf{x}$ the pdf $f(\mathbf{x};u)$ is continuously differentiable in u and that there exists an integrable function $h(\mathbf{x})$ such that

$$\left| H(\mathbf{x}) \frac{\mathrm{d}f(\mathbf{x};u)}{\mathrm{d}u} \right| \leqslant h(\mathbf{x}) \tag{7.5}$$

for all $u \in \mathscr{V}$. Then under mild conditions [19] the differentiation and expectation (integration) operators are interchangeable, so that differentiation of $\ell(u)$ yields

$$\begin{aligned} \frac{\mathrm{d}\ell(u)}{\mathrm{d}u} &= \frac{\mathrm{d}}{\mathrm{d}u} \int H(\mathbf{x}) f(\mathbf{x};u) \mathrm{d}\mathbf{x} = \int H(\mathbf{x}) \frac{\mathrm{d}f(\mathbf{x};u)}{\mathrm{d}u} \, \mathrm{d}\mathbf{x} \\ &= \int H(\mathbf{x}) \frac{\frac{\mathrm{d}f(\mathbf{x};u)}{\mathrm{d}u}}{f(\mathbf{x};u)} f(\mathbf{x};u) \, \mathrm{d}\mathbf{x} = \mathbb{E}_u \left[H(\mathbf{X}) \frac{\mathrm{d} \ln f(\mathbf{X};u)}{\mathrm{d}u} \right] \\ &= \mathbb{E}_u \left[H(\mathbf{X}) \, \mathcal{S}(u;\mathbf{X}) \right] , \end{aligned}$$

where

$$\mathcal{S}(u;\mathbf{x}) = \frac{\mathrm{d} \ln f(\mathbf{x};u)}{\mathrm{d}u}$$

is the *score function* (SF); see also (1.57). This is viewed as a function of u for a given $\mathbf{x}$.

Consider next the multidimensional case. Similar arguments allow us to represent the gradient and the higher order derivatives of $\ell(\mathbf{u})$ in the form

$$\nabla^k \ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}} \left[H(\mathbf{X}) \, \mathcal{S}^{(k)}(\mathbf{u};\mathbf{X}) \right] , \tag{7.6}$$

where

$$\mathcal{S}^{(k)}(\mathbf{u};\mathbf{x}) = \frac{\nabla^k f(\mathbf{x};\mathbf{u})}{f(\mathbf{x};\mathbf{u})} \tag{7.7}$$

is the *k-th order score function*, $k = 0, 1, 2, \ldots$. In particular, $\mathcal{S}^{(0)}(\mathbf{u};\mathbf{x}) = 1$ (by definition), $\mathcal{S}^{(1)}(\mathbf{u};\mathbf{x}) = \mathcal{S}(\mathbf{u};\mathbf{x}) = \nabla \ln f(\mathbf{x};\mathbf{u})$, and $\mathcal{S}^{(2)}(\mathbf{u};\mathbf{x})$ can be represented as

$$\begin{aligned} \mathcal{S}^{(2)}(\mathbf{u};\mathbf{x}) &= \nabla \mathcal{S}(\mathbf{u};\mathbf{x}) + \mathcal{S}(\mathbf{u};\mathbf{x}) \, \mathcal{S}(\mathbf{u};\mathbf{x})^\top \\ &= \nabla^2 \ln f(\mathbf{x};\mathbf{u}) + \nabla \ln f(\mathbf{x};\mathbf{u}) \, \nabla \ln f(\mathbf{x};\mathbf{u})^\top , \end{aligned} \tag{7.8}$$

where $\nabla \ln f(\mathbf{x};\mathbf{u})^\top$ represents that transpose of the column vector $\nabla \ln f(\mathbf{x};\mathbf{u})$ of partial derivatives of $\ln f(\mathbf{x};\mathbf{u})$. Note that all partial derivatives are taken with respect to the components of the parameter vector $\mathbf{u}$.

Table 7.1 displays the score functions $\mathcal{S}(\mathbf{u};x)$ calculated from (7.6) for the commonly used distributions given in Table A.1 in the Appendix. We take $\mathbf{u}$ to be the usual parameters for each distribution. For example, for the $\mathsf{Gamma}(\alpha,\lambda)$ and $\mathsf{N}(\mu,\sigma^2)$ distributions, we take $\mathbf{u}=(\alpha,\lambda)$ and $\mathbf{u}=(\mu,\sigma)$, respectively.

Table 7.1: Score functions for commonly used distributions.

Distribution	$f(x;\mathbf{u})$	$\mathcal{S}(\mathbf{u};x)$
$\mathsf{Exp}(\lambda)$	$\lambda\,\mathrm{e}^{-\lambda x}$	$\lambda^{-1}-x$
$\mathsf{Gamma}(\alpha,\lambda)$	$\dfrac{\lambda^\alpha x^{\alpha-1}\mathrm{e}^{-\lambda x}}{\Gamma(\alpha)}$	$\Big(\ln(\lambda x)-\frac{\Gamma'(\alpha)}{\Gamma(\alpha)},\ \alpha\lambda^{-1}-x\Big)$
$\mathsf{N}(\mu,\sigma^2)$	$\dfrac{1}{\sigma\sqrt{2\pi}}\,\mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$	$\big(\sigma^{-2}(x-\mu),\ -\sigma^{-1}+\sigma^{-3}(x-\mu)^2\big)$
$\mathsf{Weib}(\alpha,\lambda)$	$\alpha\lambda\,(\lambda x)^{\alpha-1}\,\mathrm{e}^{-(\lambda x)^\alpha}$	$\big(\alpha^{-1}+\ln(\lambda x)[1-(\lambda x)^\alpha],\ \frac{\alpha}{\lambda}[1-(\lambda x)^\alpha]\big)$
$\mathsf{Bin}(n,p)$	$\binom{n}{x}p^x(1-p)^{n-x}$	$\dfrac{x-np}{p(1-p)}$
$\mathsf{Poi}(\lambda)$	$\dfrac{\lambda^x\mathrm{e}^{-\lambda}}{x!}$	$\dfrac{x}{\lambda}-1$
$\mathsf{G}(p)$	$p(1-p)^{x-1}$	$\dfrac{1-p\,x}{p(1-p)}$

In general, the quantities $\nabla^k\ell(\mathbf{u})$, $k=0,1,\ldots$, are not available analytically, since the response $\ell(\mathbf{u})$ is not available. They can be estimated, however, via simulation as

$$\widehat{\nabla^k\ell}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i)\,\mathcal{S}^{(k)}(\mathbf{u};\mathbf{X}_i). \tag{7.9}$$

It is readily seen that the function $\ell(\mathbf{u})$ and *all* the sensitivities $\nabla^k\ell(\mathbf{u})$ can be estimated from a single simulation, since in (7.6) all of them are expressed as expectations with respect to the same pdf, $f(\mathbf{x};\mathbf{u})$.

The following two toy examples provide more details on the estimation of $\nabla\ell(\mathbf{u})$. Both examples are only for illustration, since $\nabla^k\ell(\mathbf{u})$ is available analytically.

EXAMPLE 7.2

Let $H(\mathbf{X})=X$, with $X\sim\mathsf{Ber}(p=u)$, where $u\in[0,1]$. Using Table 7.1 for the $\mathsf{Bin}(1,p)$ distribution, we easily find that the estimator of $\nabla\ell(u)$ is

$$\widehat{\nabla\ell}(u) = \frac{1}{N}\sum_{i=1}^{N} X_i\,\frac{X_i-u}{u(1-u)} \;=\frac{1}{u\,N}\sum_{i=1}^{N} X_i \approx 1\,, \tag{7.10}$$

where $X_1, \ldots, X_N$ is a random sample from $\mathsf{Ber}(u)$. In the second equation we use the fact that $X_i^2 = X_i$. The approximation sign in (7.10) follows from the law of large numbers.

Suppose that $u = \frac{1}{2}$. Suppose also that we took a sample of size $N = 20$ from $\mathsf{Ber}(\frac{1}{2})$ and obtained the following values:

$$\{x_1, \ldots, x_{20}\} = \{0,1,0,0,1,0,0,1,1,1,0,1,0,1,1,0,1,0,1,1\}.$$

From (7.10) we see that the sample derivative is $\widehat{\nabla \ell}(\frac{1}{2}) = 1.1$, while the true one is clearly $\nabla \ell(\frac{1}{2}) = 1$.

■ **EXAMPLE 7.3**

Let $H(\mathbf{X}) = X$, with $X \sim \mathsf{Exp}(\lambda = u)$. This is also a toy example, since $\nabla \ell(u) = -1/u^2$. We see from Table 7.1 that $\mathcal{S}(u; x) = u^{-1} - x$, and therefore

$$\widehat{\nabla \ell}(u) = \frac{1}{N} \sum_{i=1}^{N} X_i \, (u^{-1} - X_i) \approx -\frac{1}{u^2} \tag{7.11}$$

is an estimator of $\nabla \ell(u)$, where $X_1, \ldots, X_N$ is a random sample from $\mathsf{Exp}(u)$.

■ **EXAMPLE 7.4 Example 7.1 (Continued)**

As before, let $H(\mathbf{X}; u_3, u_4) = \max\{X_1 + u_3, X_2 + u_4\}$, where $\mathbf{X} = (X_1, X_2)$ is a two-dimensional vector with independent components and $X_i \sim f_i(X, u_i)$, $i = 1, 2$. Suppose we are interested in estimating $\nabla \ell(\mathbf{u}_1)$ with respect to the distributional parameter vector $\mathbf{u}_1 = (u_1, u_2)$. We have

$$\widehat{\nabla \ell}(\mathbf{u}_1) = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i; u_3, u_4) \, \mathcal{S}(\mathbf{u}_1; \mathbf{X}_i) \, ,$$

where $\mathcal{S}(\mathbf{u}_1; \mathbf{X}_i)$ is the column vector $(\mathcal{S}(u_1; X_{1i}), \ \mathcal{S}(u_2; X_{2i}))^\top$.

Next, we will apply the importance sampling technique to estimate the sensitivities $\nabla^k \ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X}) \, \mathcal{S}^{(k)}(\mathbf{u}; \mathbf{X})]$ simultaneously for several values of $\mathbf{u}$. To this end, let $g(\mathbf{x})$ be the importance sampling density, and assume, as usual, that the support of $g(\mathbf{x})$ contains the support of $H(\mathbf{x}) f(\mathbf{x}; \mathbf{u})$ for all $\mathbf{u} \in \mathscr{V}$. Then $\nabla^k \ell(\mathbf{u})$ can be written as

$$\nabla^k \ell(\mathbf{u}) = \mathbb{E}_g[H(\mathbf{X}) \, \mathcal{S}^{(k)}(\mathbf{u}; \mathbf{X}) \, W(\mathbf{X}; \mathbf{u})] \, , \tag{7.12}$$

where

$$W(\mathbf{x}; \mathbf{u}) = \frac{f(\mathbf{x}; \mathbf{u})}{g(\mathbf{x})} \tag{7.13}$$

is the likelihood ratio of $f(\mathbf{x}; \mathbf{u})$ and $g(\mathbf{x})$. The likelihood ratio estimator of $\nabla^k \ell(\mathbf{u})$ can be written as

$$\widehat{\nabla^k \ell}(\mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i) \, \mathcal{S}^{(k)}(\mathbf{u}; \mathbf{X}_i) \, W(\mathbf{X}_i; \mathbf{u}) \, , \tag{7.14}$$