the key to the feasibility of sequential imputation. When integrating out θ analytically is not feasible, one can treat θ as an additional missing data, $y_{\text{mis},0}$, and use the same sequential imputation method.

Suppose steps A and B are done repeatedly and independently m times, which results in a *multiple imputation* of the missing data, $\mathbf{y}_{\text{mis}}^{(1)}, \ldots, \mathbf{y}_{\text{mis}}^{(m)}$, and their respective weights $w^{(1)}, \ldots, w^{(m)}$. Then, we can estimate the posterior distribution $f(\theta \mid \mathbf{y}_{\text{obs}})$ by the weighted mixture

$$\frac{1}{W}\sum_{j=1}^{m} w^{(j)} f(\theta \mid \mathbf{y}_{\text{obs}}, \mathbf{y}_{\text{mis}}^{(j)}), \tag{3.5}$$

where $W = \sum w^{(j)}$. Again, the disadvantage of this method is that when the sample size n increases, the importance weights become very skewed. Kong et al. (1994) showed that the normalized weight w_n is a martingale sequence (with respect to n). Thus, the variance of w_n increases stochastically as n increases.

The similarity between the Bayesian missing data problem as the simulation of the SAW is transparent: mathematically, the ith missing component, $y_{\text{mis},i}$, plays the same role as the position of the ith monomer, x_i.

3.3 Nonlinear Filtering

The *state-space model* is a very popular dynamic system and has two major parts: (1) the observation equation, often written as $y_t \sim f_t(\cdot \mid x_t, \phi)$, and (2) the state equation, which can be represented by a Markov process as $x_t \sim q_t(\cdot \mid x_{t-1}, \theta)$. Putting the two together, we have

$$\begin{aligned} &\text{(state equation):} && x_t \sim q_t(\cdot \mid x_{t-1}, \theta), \\ &\text{(observation equation):} && y_t \sim f_t(\cdot \mid x_t, \phi). \end{aligned} \tag{3.6}$$

The y_t are observations and the x_t are referred to as the (unobserved) state variables. This model is sometimes termed as the *hidden Markov model* (HMM) in many applications (most significantly in speech recognition and computational biology). Such a model can be represented graphically as in Figure 3.4. In practice, the x's can be the transmitted digital signals in wireless communication (Liu and Chen 1995), the actual words in speech recognition (Rabiner 1989), the target's position and velocity detected from a radar (Gordon et al. 1993, Gordon, Salmond and Ewing 1995, Avitzour 1995), the underlying volatility in an economical time series (Pitt and Shephard 1999), the structural types in the protein secondary structure prediction problem (Schmidler, Liu and Brutlag 2000), and many others.

A main challenge to researchers is to find efficient methods for on-line estimation and prediction (filtering) of the state variables (i.e., x_t) or other

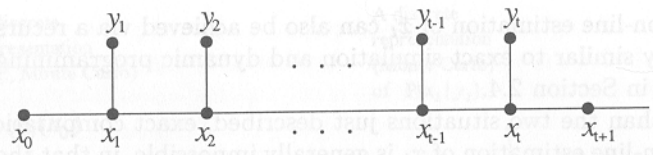

FIGURE 3.4. A graphical representation of the state-space model, also called the hidden Markov model (HMM).

parameters when the signal y_t comes in sequentially. For the simplicity of our presentation, we assume in this section that the system's parameters (i.e., ϕ and θ) are all known and are omitted from the model description. Ideas on estimating model parameters together with the state variables can be found in Liu and Chen (1995), Gordon et al. (1995), and Liu and West (2000).

With known system parameters, the optimal (in terms of the mean squared errors) on-line estimate of x_t is then the *Bayes solution* (West and Harrison 1989)

$$\begin{aligned} \hat{x}_t &= E(x_t \mid y_1, \ldots, y_t) \\ &\equiv \frac{\int \cdots \int x_t \prod_{s=1}^{t} [f(y_s \mid x_s) q_s(x_s \mid x_{s-1})]\, dx_1 \cdots dx_t}{\int \cdots \int \prod_{s=1}^{t} [f(y_s \mid x_s) q_s(x_s \mid x_{s-1})]\, dx_1 \cdots dx_t}. \end{aligned}$$

Hence, of interest at any time t is the "current" posterior distribution of x_t, which can be theoretically derived recursively as

$$\begin{aligned} \pi_t(x_t) &= P(x_t \mid y_1, \ldots, y_t) \\ &\propto \int q_t(x_t \mid x_{t-1}) f_t(y_t \mid x_t) \pi_{t-1}(x_{t-1}) dx_{t-1}, \end{aligned} \tag{3.7}$$

where $\pi_{t-1}(x_{t-1}) = p(x_{t-1} \mid y_1, \ldots, y_{t-1})$ is the "current" posterior distribution of x_{t-1}.

When both f_t and q_t are linear Gaussian conditional distributions in (3.6), the resulting model is called the *linear state-space model*, or the dynamic linear model, which has long been an active subject of research in automatic control, economics, engineering, and statistics (Harvey 1990, West and Harrison 1989). In this case, the posterior computation can be done analytically via recursion (3.8) because all the "current" posterior distributions are Gaussian. The resulting algorithm is the basis of the celebrated *Kalman filter* (Kalman 1960). When the x_t only takes on a finite, say K, possible values [e.g., binary signals in digital communication or nucleotides (with four different kinds) in DNA sequence analysis], such a state-space model is also referred to as the hidden Markov model (HMM). Its utility in many areas such as digital communication, speech recognition, protein sequence alignment, ion-channel analysis, etc. has been extensively studied.

Optimal on-line estimation of x_t can also be achieved via a recursive algorithm very similar to exact simulation and dynamic programming method described in Section 2.4.

Other than the two situations just described, exact computation of the optimal on-line estimation of x_t is generally impossible, in that the integrations needed to evaluate (3.8) and its normalizing constant quickly becomes infeasible as t increases.

A very general and simple method, named the *bootstrap filter* (and also called *particle filter*), for on-line estimation in a state-space model was proposed by Gordon et al. (1993), which makes use of the idea of sampling-importance-resampling (SIR) (Rubin 1987).

Suppose at time t that we have a random sample $\{x_t^{(1)}, \ldots, x_t^{(m)}\}$ of the state variable which follow approximately the current posterior distribution $\pi_t(x_t) = P(x_t \mid y_1, \ldots, y_t)$ (e.g., those spikes on the leftmost line in Figure 3.5). Gordon et al. (1993) suggested the following updating procedure when y_{t+1} is observed:

(a) Draw $x_{t+1}^{(*j)}$ from the state equation $q_t(x_{t+1} \mid x_t^{(j)})$, $j = 1, \ldots, m$.

(b) Weight each draw by $w^{(j)} \propto f_t(y_{t+1} \mid x_{t+1}^{(*j)})$.

(c) Resample from $\{x_{t+1}^{(*1)}, \ldots, x_{t+1}^{(*m)}\}$ with probability proportional to $w^{(j)}$ to produce a random sample $\{x_{t+1}^{(1)}, \ldots, x_{t+1}^{(m)}\}$ for time $t+1$.

It is easy to show that if the $x_t^{(m)}$ follow the "current" posterior distribution $\pi_t(x_t)$ and if m is large enough, then the new random sample $\{x_{t+1}^{(1)}, \ldots, x_{t+1}^{(m)}\}$ follows the updated posterior distribution $\pi_{t+1}(x_{t+1})$ approximately.

The connection between this problem and the statistical missing data problem analyzed in Section 3.2 is obvious: The state variable x_t can be treated as missing data and imputed sequentially. Under the SIS framework, we see that the "current" target distribution $\pi_t(\cdot)$ satisfies the recursion

$$\pi_t(\mathbf{x}_t) \propto f_t(y_t \mid x_t) q_t(x_t \mid x_{t-1}) \pi_{t-1}(\mathbf{x}_{t-1}). \tag{3.8}$$

A special feature of the state-space model is that the state variable x_t possess a Markovian structure. This feature makes it possible for one to consider only the marginal posterior distribution, $\pi_t(x_t)$, instead of the joint distribution $\pi_t(\mathbf{x}_t)$, and it is essential for the bootstrap filter to be applicable. In fact, if one shifts the step index t for the missing data $y_{\text{mis},t}$ forward by one unit (i.e., treating $y_{\text{mis},t-1}$ as $y_{\text{mis},t}$), then the sequential imputation method with stepwise resampling gives rise to the same algorithm. The advantage of the bootstrap filter is its simplicity, whereas the drawbacks of the method are two: (a) It did not use the current available information y_{t+1} in the sampling step; (b) its excessive use of resampling may decrease its efficiency.

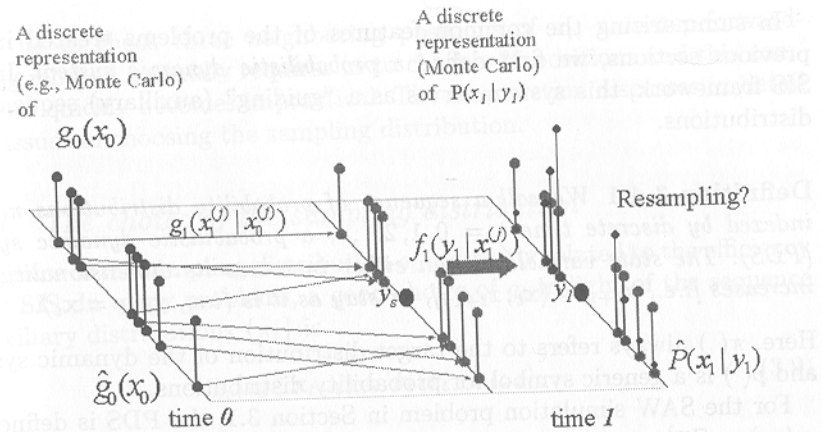

FIGURE 3.5. A graphical illustration of the bootstrap/particle filter. At time 0, one has a discrete representation of the current distribution. One propagates to time 1 by first sampling those $x_1^{(m)}$'s from the state equation (equivalent to sampling from its prior distribution) and then correcting this sampling by an importance reweighting and resampling.

Those models discussed in Sections 2.7, 3.1, and 3.2 and dealt with by the general SIS methodology do not share the special structure enjoyed by the state-space model (i.e., the x_t's are not Markovian). However, the generic SIS formulation forces one to think deeper into the general issues of efficiency improvement for importance sampling. For example, one may want to construct trial sampling distributions that are more efficient than the one used in the bootstrap filter and may think of applying techniques such as marginalization and rejection controls (Sections 3.4.6 and 2.6.2). Liu and Chen (1995, 1998) also discuss the role of resampling in a generic SIS framework.

3.4 A General Framework

The connection of the methods described in the previous three sections can be briefly summarized as follows. The growth method is a special application of the general SIS methodology described in Section 2.6.3. The sequential imputation method of Kong et al. (1994) is an equivalent form of the SIS under the statistical missing data setting. The bootstrap filter uses an SIS construction but has an additional resampling step inserted before each sequential sampling step. In this section, we show that the modifications to the growth method made by Wall and Erpenbeck (1959) and Grassberger (1997) is practically equivalent to an SIS with resampling (Liu and Chen 1995).

In summarizing the common features of the problems treated in the previous sections, we first define a *probabilistic dynamic system*. In the SIS framework, this system serves as a "guiding" (auxiliary) sequence of distributions.

Definition 3.4.1 *We call a sequence of probability distributions $\pi_t(\mathbf{x}_t)$, indexed by discrete time $t = 0, 1, 2, \ldots$, a probabilistic dynamic system (PDS). The state variable $\mathbf{x}_t$ can either increases its dimensionality as t increases [i.e., $\mathbf{x}_{t+1} = (\mathbf{x}_t, x_{t+1})$], or stay as it is (i.e., $\mathbf{x}_{t+1} = \mathbf{x}_t$).*

Here, $\pi(\ )$ always refers to the target distribution of the dynamic system and $p(\ )$ is a generic symbol for probability distributions.

For the SAW simulation problem in Section 3.1, the PDS is defined as $\pi_t(\mathbf{x}_t) = Z_t^{-1}$ on the set of all self-avoiding walks with t notes, where Z_t is the total number of such configurations. In words, π_t is the uniform distribution on the set of all SAWs of $t-1$ steps. In the literature of molecular simulation, one is often interested in more complex structures [e.g., protein structures, see Leach (1996)] in which the π_t is taken as the *Boltzmann distribution* of the form $\exp(-U_t/k_BT)/Z$. Here, U_t is the *interacting energy* generated by the monomers of the chain polymer (with t monomers), T is the absolute temperature, and k_B is the Boltzmann constant.

The state-space model example (Section 3.3) also fits very well into our PDS framework in that we can define $\pi_t(\mathbf{x}_t)$ as the posterior distribution $p(\mathbf{x}_t \mid \mathbf{y}_t, x_0)$. With this PDS, the estimation of x_t can be thought of as evaluating statistical (Monte Carlo) average of a random variable under distribution π_t. The methodology developed in this section will be generally called *sequential Monte Carlo*.

It is natural to think of the PDS as a sequence of posterior distributions conditional on the information "up to time t" for a system with random variables $x_1, x_2, \ldots$. In other words, the final probability distribution is "built up" sequentially as more and more structural details (i.e., information) are incorporated. In the traditional Bayesian analysis (Gelman, Carlin, Stern and Rubin 1995), the dimensionality of the random vector (i.e., $\mathbf{x}_t$) involved in the system does not vary when new information comes in; whereas in our PDS, the "configuration" space changes along with t. Once a proper PDS is employed, this PDS can further help us choose good sampling distributions $g_t(x_t \mid \mathbf{x}_{t-1})$.

It is helpful to recall how the growth method is used to simulate a SAW. At step t, a monomer is added to the existing partial chain $(\mathbf{x}_{t-1})$ according to a distribution that makes use of the information generated from a forward-looking step. This is equivalent to using the sequence of distributions, $\{\pi_t(\mathbf{x}_t), t = 1, 2, \ldots\}$, as the PDS, where π_t is the uniform distribution on all SAWs with t nodes, and letting $g_t(x_t \mid \mathbf{x}_{t-1}) = \pi_t(x_t \mid \mathbf{x}_{t-1})$. There is no reason why we cannot use information generated by more forward-looking steps. But as the one-step-forward-looking strategy requires us to check three neighboring positions of x_t, a k-step-forward-looking method generally requires us to check $\sim 3^k$ positions and this computation quickly becomes impractical. The next section discusses in detail some issues in choosing the sampling distribution.

3.4.1 The choice of the sampling distribution

The choice of the sampling distribution g_t is directly related to the efficiency of the SIS. In many problems, a good choice of g_t in light of the sequence of auxiliary distributions $\{\pi_t\}$ is

$$g_t(x_t \mid \mathbf{x}_{t-1}) = \pi_t(x_t \mid \mathbf{x}_{t-1}), \tag{3.9}$$

with the incremental weight

$$u_t = \frac{\pi_t(\mathbf{x}_{t-1})}{\pi_{t-1}(\mathbf{x}_{t-1})}. \tag{3.10}$$

Note that u_t in (3.10) does not depend on the value of x_t and this feature is important to several issues discussed later. The reason that drawing x_t from $\pi_t(x_t \mid \mathbf{x}_{t-1})$ is more desirable than from a more or less arbitrary function is clear from rewriting the incremental weight (3.10) as

$$u_t = \frac{\pi_t(\mathbf{x}_{t-1})}{\pi_{t-1}(\mathbf{x}_{t-1})} \frac{\pi_t(x_t \mid \mathbf{x}_{t-1})}{g_t(x_t \mid \mathbf{x}_{t-1})}.$$

Intuitively, the second ratio is needed to correct the discrepancy between $g_t(x_t \mid \mathbf{x}_{t-1})$ and $\pi_t(x_t \mid \mathbf{x}_{t-1})$ when they are different.

Other choices of g_t can also be desirable. For example, one may also consider a two-step-forward sampling distribution

$$g_t(x_t \mid \mathbf{x}_{t-1}) \propto \int \pi_{t+1}(x_{t+1}, \mathbf{x}_t) dx_{t+1}.$$

In the polymer simulation, this corresponds to a two-step look-ahead strategy (Section 3.1.2). In the content of polymer simulation, this idea was also proposed by Meirovitch (1982).

Another strategy is to "coarsen" the dynamic systems; that is, one can group, say, $(x_{bt+1}, \ldots, x_{b(t+1)})$ together as a new "mega"-state x'_t in the system. Then, the SIS method can be applied to draw a block of b x's at a time. This method is also proven useful in simulating long polymer chains (Wall, Rubin and Isaacson 1957).

3.4.2 Normalizing constant

In many scientific problems, computing the normalizing constant (i.e., the partition function) of an unnormalized probability distribution function is

of critical importance. Examples in physics and chemistry are abundant. Sometimes, even mathematicians and statisticians are interested in such problems, one of which is to count the total number Z of distinctive tables that contain only 0's and 1's and have the fixed row sums $r_1, \ldots, r_m$ and column sums $c_1, \ldots, c_n$. For example, there are twenty-seven 0-1 tables with the row sums 2, 2, 2, 3 and the column sums 3, 2, 3, 1. One such table is given in Table 3.1. This problem can be formulated as one of estimating

0	0	1	1
1	1	0	0
1	0	1	0
1	1	1	0

TABLE 3.1. A typical 0-1 table with given respective column sums: 3, 2, 3, 1 and row sums: 2, 2, 2, 3.

the normalizing constant Z in that the uniform distribution on space of all such tables is of the form $1/Z$. Problems such as evaluating likelihood (Section 3.2.1) and computing the Bayes factor (Kong et al. 1994, Meng and Wong 1996) also belong to this class.

The SIS approach can be an effective means for estimating the normalizing constant. Suppose the target probability distribution is $\pi(\mathbf{x})$. An auxiliary PDS is found to be $\Pi = \{\pi_t(\mathbf{x}_t),\ t = 1, \ldots, N\}$ (with $\pi_N \equiv \pi$), and each of its member's density function, π_t, is known up to a normalizing constant. In other words, we can write down the unnormalized distribution function, $q_t(\mathbf{x}_t) = Z_t \pi_t(\mathbf{x}_t)$, for everyone in the PDS. In statistics, Z_t often corresponds to the *Bayes factor* or the likelihood function evaluated at a given parameter value.

Suppose we implement an SIS scheme with the sequential sampling system $\{g_t(x_t \mid \mathbf{x}_{t-1}),\ t = 1, \ldots, N\}$. Then, the incremental importance weight u_t is computed as

$$u_t = \frac{q_t(\mathbf{x}_t)}{q_{t-1}(\mathbf{x}_{t-1}) g_t(x_t \mid \mathbf{x}_{t-1})} = \frac{Z_t \pi_t(\mathbf{x}_t)}{Z_{t-1} \pi_{t-1}(\mathbf{x}_{t-1}) g_t(x_t \mid \mathbf{x}_{t-1})}.$$

The final weight takes the form

$$w_N = \prod_{t=1}^{N} u_t = \frac{Z_N}{Z_1} \frac{\pi_N(\mathbf{x}_N)}{g_1(x_1) \cdots g_N(x_N \mid \mathbf{x}_{N-1})}. \tag{3.11}$$

Thus, the sample average of w_N gives us an unbiased estimate of $E(w_N) = Z_N/Z_1$, the ratio of two normalizing constants. In many cases, π_1 is a very simple system in which Z_1 can be easily obtained either analytically or numerically. Then, we can obtain a rather accurate estimate of Z_N. Based on this procedure, Chen (2001) designed an SIS algorithm to approximately

count the total number of 0-1 tables with given marginal sums (more in Section 4.3).

In recent years, many techniques have been developed to improve the SIS-based methods for nonlinear filtering problems. But the estimation problem for the normalizing constants has been mostly neglected. We note, however, that expression (3.11) is broadly useful for estimating the normalizing constant with the following improvement methods.

3.4.3 Pruning, enrichment, and resampling

When the system grows, the variance of the importance weights w_t increases. Thus, after a certain number of steps, many of the weights become very small and a few very large. One of the earliest methods for improving the SIS procedure in polymer simulations, called the *enrichment* method, was proposed in Wall and Erpenbeck (1959). Their work, according to Kremer and Binder (1988), led to "one of the very first intriguing successes of Monte Carlo methods." Briefly, Wall and Erpenbeck suggested splitting those successfully simulated partial SAW chains into, say, r chains, each assigned $1/r$ of the original weight, and then continuing them in r independent ways. More precisely, once we have successfully simulated by the SIS method a chain up to step t, say, $\mathbf{x}_t$ with weight w_t, we can make r_t copies of it and assign each a weight w_t/r_t. Then, all the copies will be continued as if they were independently built up from scratch. An undesirable, and also unavoidable, feature of this enhancement is that the resulting full-length chains are no longer statistically independent.

Because choosing r_t can be a difficult task in general, Grassberger (1997) introduces some learning strategy for chain splitting and *pruning*. Instead of slitting each chain $\mathbf{x}_t$ indiscriminately, he suggested using a step-dependent upper cutoff value C_t and a lower cutoff value c_t. Then, if the weight w_t is greater than C_t, the chain $\mathbf{x}_t$ is split into r copies, each with weight w_t/r, whereas if $w_t \le c_t$, one flips a fair coin to decide whether to keep it. If it is kept, its weight w_t is doubled to $2w_t$. He named the algorithm "PERM" for *Prune-Enriched Rosenbluth Method*. However, choosing the cutoff values c_t and C_t can also be tricky.

Independently, a resampling strategy (to be described in the next section) for improving SIS has been developed in the statistics and engineering community (Gordon et al. 1993, Liu and Chen 1995). Liu and Chen (1995) introduced a monitored resampling procedure for the general SIS and applied the method to a blind deconvolution problem. Their method produces the same enrichment and pruning effects on the SIS samples as that of Grassberger (1997) and is arguably more flexible than the PERM. In treating the state-space models, Gordon et al. (1993) also used a resampling strategy. In fact, a resampling step is enforced at every time step t of their *bootstrap filter*. The success of this method, however, relies heavily on the Markovian structure among the state variable $x_1, x_2, \ldots$; that is, given the

realization of x_t, the next variable, x_{t+1}, is statistically independent of all the previous states, $\mathbf{x}_{t-1}$. Thus, resampling from set $\{\mathbf{x}_{t-1}^{(j)}, j = 1, \ldots, m\}$ is equivalent to resampling from $\{x_{t-1}^{(j)}, j = 1, \ldots, m\}$, the set of the "current state." Otherwise, frequent resampling will rapidly impoverish diversity of the partial samples produced earlier.

3.4.4 *More about resampling*

Suppose at step t we have a collection of m partial samples of length t, $\mathcal{S}_t = \{\mathbf{x}_t^{(j)}, j = 1, \ldots, m\}$, which are properly weighted by the collection of weights $\mathcal{W}_t = \{w_t^{(j)}, j = 1, \ldots m\}$ with respect to the PDS π_t. Each of these partial samples, $\mathbf{x}_t^{(j)}$, will also be called a *stream*. Instead of carrying the weight $w_t^{(j)}$ as the system evolves, it is also legitimate, and sometimes advantageous, to insert the following *resampling* step between SIS recursions (Liu and Chen 1998), and such a procedure is referred to as the *SIS with resampling*. The following two schemes are just two of many possible choices to achieve the resampling goal.

Simple Random Resampling

1. Sample a new set of streams (i.e., partial samples), denoted as $\mathcal{S}_t'$, from $\mathcal{S}_t$ according to the weights $w_t^{(j)}$.

2. Assign equal weights, W_t/m, to the streams in $\mathcal{S}_t'$, where $W_t = w_t^{(1)} + \cdots + w_t^{(m)}$.

Residual Resampling:

1′ Retain $k_j = [mw_t^{(*j)}]$ copies of $\mathbf{x}_t^{(j)}$, where $w_t^{(*j)} = w_t^{(j)}/W_t$, and $j = 1, \ldots, m$. Let $m_r = m - k_1 - \cdots - k_m$.

2′ Obtain m_r i.i.d. draws from $\mathcal{S}_t$ with probabilities proportional to $mw_t^{(*j)} - k_j, j = 1, \ldots, m$.

3′ Reset all the weights to W_t/m.

Note that one can still use the average of the final weights to estimate the ratio Z_n/Z_1 of two normalizing constants (see Section 3.4.2) even after several resampling steps have been incurred. This is the primary reason why we set the weights of all the streams to their *current average* after each resampling step. It is easily shown that the residual sampling dominates the simple random sampling in having smaller Monte Carlo variance and favorable computation time, and it does not seem to have disadvantages in other aspects. Some new resampling method has recently been proposed by Doucet, Godsill and Andrieu (2000).

As one can clearly see here, the residual resampling scheme is very much similar to the enrichment and pruning (PERM) method of Grassberger (Wall and Erpenbeck 1959, Grassberger 1997). Both the PERM algorithm and the resampling scheme construct a proper set of importance samples which can be used to estimate any quantity of interest with respect to π_T (the target distribution). In particular, PERM will be proper as long as the cutoff values c_t and C_t are prescribed in advance; and resampling will cause no conceptual problem as long as the decision of resampling does not depend on the configurations. PERM has more flexibility in dealing with the weights, whereas the resampling method is more disciplined and requires less tuning. A prominent advantage of the resampling method over PERM is that the choice of cutoff values for splitting the streams is automatically and implicitly achieved by the cross-stream comparison of the weights. More precisely, in residual resampling, we split a stream into k more copies precisely because its weight is k times better than the average weight. This feature makes the resampling method very generally applicable and powerful. As we will show in the next section, resampling can be used, to a great advantage, in applications ranging from target tracking to population genetics and to biological sequence analysis.

Instead of using the w_t in resampling, we can also implement the following more general resampling strategy.

- For $j' = 1, \ldots, \tilde{m}$:
  - Draw $\tilde{\mathbf{x}}_t^{(j')}$ independently from the current sample $\{\mathbf{x}_t^{(j)}, j = 1, \ldots, m\}$ according to the probability vector $(a^{(1)}, \ldots, a^{(m)})$; suppose we obtain that $\tilde{\mathbf{x}}_t^{(j')} = \mathbf{x}_t^{(j)}$.
  - A new weight $\tilde{w}_t^{(j')} = w_t^{(j)}/a^{(j)}$ is assigned to this sample.
- Return the new representation $\tilde{\mathcal{S}}_t = \{\tilde{\mathbf{x}}_t^{(j')}, j' = 1, \ldots, \tilde{m}\}$ and $\tilde{\mathcal{W}}_t = \{\tilde{w}_t^{(j')}, j' = 1, \ldots, \tilde{m}\}$.

The new set $\tilde{\mathcal{S}}_t$ thus formed is also properly weighted by $\tilde{\mathcal{W}}_t$ (approximately) with respect to π (Rubin 1987). Because the role of resampling is to prune away "bad" samples and to split good ones, we should choose $a^{(j)}$ as a monotone function of $w_t^{(j)}$. Having an additional flexibility in choosing the sampling weights $a^{(j)}$ is rather intriguing and can be potentially very useful. For example, the $a^{(j)}$ can be chosen to reflect certain "future trend" (Pitt and Shephard 1999) or be chosen to balance between the need of diversity (i.e., having multiple distinct samples) and the need of focus (i.e., giving more presence to those samples with large weights). A generic choice is

$$a^{(j)} = [w_t^{(j)}]^\alpha, \tag{3.12}$$

where $0 < \alpha \leq 1$ can vary according to the coefficient of variation of the w_t. Professor W.H. Wong suggested choosing $\alpha = 1/2$, which seems to work well in several examples.

The *bootstrap filter* and its variations (Gordon et al. 1993, Kitagawa 1996, Pitt and Shephard 1999, Berzuini, Best, Gilks and Larizza 1997) can be seen as SIS with special choices of g_t and with resampling at every step. Since resampling at every step is neither necessary nor efficient, it is desirable to prescribe a schedule for the resampling step to take place. In the state-space models, frequent resampling does not produce much detrimental effect because of the Markov structure of the unobserved state variables. However, resampling too often gives very bad results in a general PDS, where no simple Markovian structure is present. These systems include the SAW model for polymer studies (Kremer and Binder 1988), the demographic tree model in population genetics (Stephens and Donnelly 2000, Chen and Liu 2000a), the model for wireless signal detection in flat-fading channels (Chen and Liu 2000a, Chen, Wang and Liu 2000), and many Bayesian missing data problems.

The resampling schedule (i.e., when to resample) can be either deterministic or dynamic. When the schedule is dynamic, some small bias may be introduced. However, our experience showed that this bias did not produce any adverse effect. With a *deterministic schedule*, we conduct resampling at time $t_0, 2t_0, \ldots$, where t_0 is given in advance and it may be tuned to suit for the particular problem of interest. In a *dynamic schedule*, a sequence of thresholds, $c_1, c_2, \ldots$, are given in advance. We monitor the coefficient of variation of the weights cv_t^2 and invoke the resampling step when event $cv_t^2 > c_t$ occurs. A typical sequence of c_t can be $c_t = a + bt^\alpha$. We summarize the resampling scheme in the following pseudo-code:

1. Check the weight distribution by performing one of the following two methods at time t:
   - Dynamic: go to step 2 if the coefficient of variation of the weight is modest [i.e., $cv_t^2(w) < c_t$]; otherwise go to step 3.
   - Deterministic: go to step 2 if $t \neq kt_0$ for integer k; otherwise go to step 3.
2. Invoke an SIS step. Set $t = t + 1$ and go to step 1.
3. Invoke a resampling step. Go to step 2.

It is not immediately clear why one needs resampling at a certain stage t. As much detailed theoretical discussion is given by Liu and Chen (1995), we only mention a few heuristics on the issue. First, if the weights $w_t^{(j)}$ are constant (or near constant) for all t (such a case occurs when one can draw from π_t directly), resampling only reduces the number of distinctive streams and introduces extra Monte Carlo variation. This suggests that one should

not perform resampling when the coefficient of variation [as defined in (2.5)], cv_t^2, for the $w_t^{(j)}$ is small. As argued in Kong et al. (1994), the *effective sample size* is inversely proportional to $1+\text{cv}_t^2$. Second, it can be shown that as the system evolves, cv_t^2 *increases* stochastically (Kong et al. 1994). When the weights get very skewed at time t, carrying many streams with very small weights is apparently a waste. Resampling can provide chances for the good (i.e., "important") streams to amplify themselves and hence "rejuvenate" the sampler. The resampling step tends to result in a better group of "ancestors" so as to produce better "descendants" (i.e., Monte Carlo samples of the future states), although it does not improve inferences on the *current* state, $\mathbf{x}_t$.

3.4.5 *Partial rejection control*

As t increases, the distribution of w_t typically becomes more and more skewed, implying that the χ^2 distance between the sampling distribution of $\mathbf{x}_t$ and its current guiding distribution π_t increases. (This distance is equivalent to the coefficient of variation of w_t.) As a consequence, many streams carried by the SIS will have minimal impact on the final estimation. It is thus desirable to prune them away at an earlier stage. In the previous two sections, we discussed the ideas of PERM and resampling. In Section 2.6.4, we saw how the rejection control method can be used to achieve pruning without creating bias or correlations. However, the implementation of a full rejection control requires that we make up the lost streams by restarting from stage 1 [i.e., sampling from $g_1(\ )$ and proceeding forward] and passing through all the intermediate rejection steps (i.e., check-points). This procedure can be extremely time-consuming and is not very practical for a large probabilistic system.

Instead of employing the full rejection control, we can opt for the following more practical method, the *partial rejection control*:

1. At each check point t_k, start $\text{RC}(t_k)$ as described in Section 2.6.4 with the threshold value $c = c_k$. If stream $\mathbf{x}_{t_k}^{(j)}$ with weight $w_{t_k}^{(j)}$ passes this check point, one proceeds the same way as in a standard SIS with the old weight replaced by $w_{t_k}^{(*j)} = \max\{w_{t_k}^{(j)}, c_k\}$.
2. When rejected, go back to check point t_{k-1} to draw a stream $x_{t_{k-1}}^{(j)}$ from the pool $\mathcal{S}_{t_{k-1}}$, with probability proportional to $w_{t_{k-1}}^{(j)}$. Reset its weight as $\bar{w}_{t_{k-1}}$ and proceed with the SIS procedure. If the new stream formed in this way pass the check point t_k, then its weight is set as $w_{t_k}^{(*j)} = \max\{w_{t_k}^{(j)}, c_k\}$.
3. Reset all the weights as $w_{t_k}^{(j)} = \hat{p_c} w_{t_k}^{(*j)}$.

The partial rejection control method combines the three main ideas used in designing sequential Monte Carlo algorithms: weighting, rejection, and resampling. It is also related to the method of (Hurzeler and Kunsch 1998).

3.4.6 Marginalization, look-ahead, and delayed estimate

Two other general methods for improving the performance of an SIS algorithm are *marginalization* and *iterative updating*. The former is an analytical method and, when achievable, it can improve almost all Monte Carlo methods. Its general principle was described by Hammersley and Handscomb (1964) and Rubinstein (1981) as a dimension reduction technique. Using marginalization to improve a static importance sampling method has been discussed in Section 2.5.5. Its use with Markov chain Monte Carlo methods will be discussed in the next chapter. The latter approach is to make use of Metropolis or Gibbs sampling (heat-bath) algorithm to improve the underlying importance sampling distribution. MacEachern et al. (1999) provide some theory on why these operations help to improve the performance of the algorithm.

Another useful strategy in constructing the trial sampling distribution is to "look-ahead" for a few steps. This method has been successfully implemented for polymer simulations (Meirovitch 1982, Meirovitch 1985, Batoulis and Kremer 1988) and for nonlinear state-space models (Liu and Chen 1995). It is also called the "scanning future method" in statistical physics literature. More precisely, we can construct our sampling distribution as close to the future target as possible. One such construction is to let

$$g_t(x_t \mid \mathbf{x}_{t-1}) = \pi_{t+s}(x_t \mid \mathbf{x}_{t+1}).$$

Note that

$$\pi_{t+s}(x_t \mid \mathbf{x}_{t-1}) = \int \pi_{t+s}(x_{t+s}, \ldots, x_t \mid \mathbf{x}_{t-1}) dx_{t+1} \cdots dx_{t+s}.$$

If we regard each $\pi_t(\mathbf{x})$ as a "posterior distribution" conditional on the information up to time t, then the above approach is just trying to make use of as much future information as possible. If we had chosen s so that $t + s = n$, for example, then the resulting sampling distribution g_t from "forward-looking" is in fact the ideal conditional distribution for x_t under the target distribution (so that the product of g_t is equal to π_n). However, the forward-looking distribution can be difficult to deal with as s becomes large, which is the reason why we adopt the SIS approach in the first place. In Meirovitch (1985), he proposed the *mean-field scanning method*, in which one constructs g_t by making use of the future states information generated by a mean-field approximation.

3.5 Problems

1. Restate the growth method as a special form of SIS. Describe a proper PDS and the implied sampling distributions used for the growth method for simulating SAWs.

2. Describe a proper PDS for the nonlinear filtering problem with model (3.6) and the corresponding SIS strategy.

3. Design a SIS method for counting the number of 0-1 tables with given margins (Chen 2001).

4. Show that the weighting strategy in sequential imputation (Section 3.2) is proper and show that the average of the weight can be used to estimate the likelihood value.

5. Show that the (normalized) importance weight sequence in SIS is a martingale sequence and the variances increase stochastically (Kong et al. 1994).

6. Show that one can still use a similar method as in Section 3.4.2 to estimate the ratio of two normalizing constants, Z_n/Z_1, when a few resampling steps are incurred between time 1 and n.

7. Discuss why resampling can be useful in sequential Monte Carlo.