CHAPTER

PARTICLE FILTERING 17

CHAPTER OUTLINE

17.1 INTRODUCTION

This chapter is a follow-up to Chapter 14, whose focus was on Monte Carlo methods. Our interest now turns to a special type of sampling techniques known as sequential-sampling methods. In contrast to the Monte Carlo methods, considered in Chapter 14, here we will assume that distributions from which we want to sample are time varying, and that sampling will take place in a sequential fashion. The main emphasis of this chapter is on particle filtering techniques for inference in state-space dynamic models. In contrast to the classical form of Kalman filtering, here the model is allowed to be nonlinear and/or the distributions associated with the involved variables non-Gaussians.

17.2 SEQUENTIAL IMPORTANCE SAMPLING

Our interest in this section shifts toward tasks where data are sequentially arriving, and our goal becomes that of sampling from their joint distribution. In other words, we are receiving observations, $\boldsymbol{x}_n \in \mathbb{R}^l$, of random vectors $\mathbf{x}_n$. At some time n, let $\boldsymbol{x}_{1:n} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ denote the set of the available samples

Machine Learning. http://dx.doi.org/10.1016/B978-0-12-801522-3.00015-X

and let the respective joint distribution be denoted as $p_n(\boldsymbol{x}_{1:n})$. No doubt, this new task is to be treated with special care. Not only is the dimensionality of the task (number of random variables, i.e., $\mathbf{x}_{1:n}$) now time varying, but also, after some time has elapsed, the dimensionality will be very large and, in general, we expect the corresponding distribution, $p_n(\boldsymbol{x}_{1:n})$, to be of a rather complex form. Moreover, at time instant n, even if we knew how to sample from $p_n(\boldsymbol{x}_{1:n})$, the required time for sampling would be at least of the order of n. Hence, as n increases, even such a case could not be computationally feasible for large values of n. Sequential sampling is of particular interest in dynamic systems in the context of particle filtering, and we will come to deal with such systems very soon. Our discussion will develop around the importance sampling method, which was introduced in Section 14.5.

17.2.1 IMPORTANCE SAMPLING REVISITED

Recall from Eq. (14.28) that given (a) a function $f(\boldsymbol{x})$, (b) a desired distribution $p(\boldsymbol{x}) = \frac{1}{Z}\phi(\boldsymbol{x})$, and (c) a proposal distribution $q(\boldsymbol{x})$, then

$$\mathbb{E}\left[f(\mathbf{x})\right] := \mu_f \simeq \sum_{i=1}^{N} W(\boldsymbol{x}_i) f(\boldsymbol{x}_i) := \hat{\mu}, \tag{17.1}$$

where $\boldsymbol{x}_i$ are samples drawn from $q(\boldsymbol{x})$. Recall, also, that the estimate

$$\hat{Z} = \frac{1}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_i) \tag{17.2}$$

defines an unbiased estimator of the true normalizing constant Z, where $w(\boldsymbol{x}_i)$ are the nonnormalized weights, $w(\boldsymbol{x}_i) = \frac{\phi(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)}$. Note that the approximation in Eq. (17.1) equivalently implies the following approximation of the desired distribution:

$$\boxed{p(\boldsymbol{x}) \simeq \sum_{i=1}^{N} W(\boldsymbol{x}_i)\delta(\boldsymbol{x} - \boldsymbol{x}_i): \quad \text{Discrete Random Measure Approximation.}} \tag{17.3}$$

In other words, even a continuous pdf is approximated by a set of discrete points and weights assigned to them. We say that the distribution is approximated by a *discrete random measure* defined by the *particles* $\boldsymbol{x}_i$, $i = 1, 2, \ldots, N$, with respective normalized weights $W(\boldsymbol{x}_i) := W^{(i)}$. The approximating random measure is denoted as $\chi = \{\boldsymbol{x}_i, W^{(i)}\}_{i=1}^{N}$.

Also, we have already commented in Section 14.5 that a major drawback of importance sampling is the large variance of the weights, which becomes more severe in high-dimensional spaces, where our interest will be from now on. Let us elaborate on this variance problem a bit more and seek ways to bypass/reduce this undesired behavior.

It can be shown (e.g., [33] and Problem 17.1) that the variance of the corresponding estimator, $\hat{\mu}$, in Eq. (17.1) is given by

$$\text{var}[\hat{\mu}] = \frac{1}{N}\left(\int \frac{f^2(\boldsymbol{x})p^2(\boldsymbol{x})}{q(\boldsymbol{x})}d\boldsymbol{x} - \mu_f^2\right). \tag{17.4}$$

Observe that if the numerator $f^2(\boldsymbol{x})p^2(\boldsymbol{x})$ tends to zero slower than $q(\boldsymbol{x})$ does, then for fixed N, the variance $\text{var}[\hat{\mu}] \longrightarrow \infty$. This demonstrates the significance of selecting q very carefully. It is not difficult

to see, by minimizing Eq. (17.4), that the optimal choice for $q(\boldsymbol{x})$, leading to the minimum (zero) variance, is proportional to the product $f(\boldsymbol{x})p(\boldsymbol{x})$. We will make use of this result later on. Note, of course, that the proportionality constant is $1/\mu_f$, which is not known. Thus, this result can only be considered as a benchmark.

Concerning the variance issue, let us turn our attention to the unbiased estimator $\hat{Z}$ of Z in Eq. (17.2). It can be shown (Problem 17.2) that

$$\text{var}[\hat{Z}] = \frac{Z^2}{N}\left(\int \frac{p^2(\boldsymbol{x})}{q(\boldsymbol{x})}d\boldsymbol{x} - 1\right). \tag{17.5}$$

By its definition, the variance of $\hat{Z}$ is directly related to the variance of the weights. It turns out that, in practice, the variance in Eq. (17.5) exhibits an exponential dependence on the dimensionality (e.g., [11, 15] and Problem 17.5). In such cases, the number of samples, N, has to be excessively large in order to keep the variance relatively small. One way to *partially* cope with the variance-related problem is the *resampling* technique.

17.2.2 RESAMPLING

Resampling is a very intuitive approach where one attempts a *randomized* pruning of the available samples (particles), drawn from q, by (most likely) discarding those associated with low weights and replacing them with samples whose weights have larger values. This is achieved by drawing samples from the approximation of $p(\boldsymbol{x})$, denoted as $\hat{p}(\boldsymbol{x})$, which is based on the discrete random measure $\{\boldsymbol{x}_i, W^{(i)}\}_{i=1}^N$, in Eq. (17.3). In importance sampling, the involved particles are drawn from $q(\boldsymbol{x})$ and the weights are appropriately computed in order to "match" the desired distribution. Adding the extra step of resampling, a *new* set of *unweighted* samples is drawn from the discrete approximation $\hat{p}$ of p. Using the resampling step, we still obtain samples that are approximately distributed as p; moreover, particles of low weights have been removed with high probability and thereby, for the next time instant, the probability of exploring regions with larger probability masses is increased. There are different ways of sampling from a discrete distribution.

- *Multinomial resampling.* This method is equivalent to the one presented in Example 14.2. Each particle, $\boldsymbol{x}_i$, is associated with a probability $P_i = W^{(i)}$. Redrawing N (new) particles will generate from each particle, $\boldsymbol{x}_i$, $N^{(i)}$ "offsprings" ($\sum_{i=1}^N N^{(i)} = N$), depending on their respective probability P_i. Hence, $N^{(1)}, \ldots, N^{(N)}$, will follow a multinomial distribution (Section 2.3), that is,

$$P(N^{(1)}, \ldots, N^{(N)}) = \binom{N}{N^{(1)} \ldots N^{(N)}} \prod_{i=1}^N P_i^{N^{(i)}}.$$

  In this way, the higher the probability (weight $W^{(i)}$) of an originally drawn particle, the higher the number of times, $N^{(i)}$, this particle will be redrawn.

  The new discrete estimate of the desired distribution will now be given by

$$\bar{p}(\boldsymbol{x}) = \sum_{i=1}^N \frac{N^{(i)}}{N}\delta(\boldsymbol{x} - \boldsymbol{x}_i).$$

  From the properties of the multinomial distribution, we have that $\mathbb{E}[\mathrm{N}^{(i)}] = NP_i = NW^{(i)}$ and hence $\bar{p}(\boldsymbol{x})$ is an unbiased approximation of $\hat{p}(\boldsymbol{x})$.

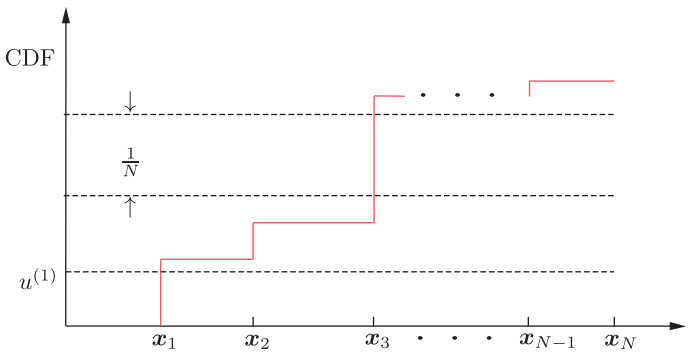

FIGURE 17.1

The sample $u^{(1)}$ drawn from $\mathcal{U}\left(0, \frac{1}{N}\right)$ determines the first point that defines the set of N equidistant lines, which are drawn and cut across the cumulative distribution function (CDF). The respective intersections determine the number of times, $N^{(i)}$, the corresponding particle, $\boldsymbol{x}_i$, will be represented in the set. For the case of the figure, $\boldsymbol{x}_1$ will appear once, $\boldsymbol{x}_2$ is missed, $\boldsymbol{x}_3$ two times.

- *Systematic resampling*. Systematic resampling is a variant of the multinomial approach. Recall from Example 14.2 that every time a particle is to be (re)drawn, a new sample is generated from the uniform distribution $\mathcal{U}(0, 1)$. In contrast, in systematic resampling, the process is not entirely random. To generate N particles, we only select randomly one sample, $u^{(1)} \sim \mathcal{U}(0, \frac{1}{N})$. Then, define

$$u^{(j)} = u^{(1)} + \frac{j-1}{N}, \quad j = 2, 3, \ldots, N,$$

and set

$$N^{(i)} = \text{card}\left\{\text{All } j : \sum_{k=1}^{i-1} W^{(k)} \leq u^{(j)} < \sum_{k=1}^{i} W^{(k)}\right\},$$

where card$\{\cdot\}$ denotes the cardinality of the respective set. Figure 17.1 illustrates the method. The resampling algorithm is summarized next.

Algorithm 17.1 (Resampling).

- Initialization
  - Input the samples $\boldsymbol{x}_i$ and respective weights $W^{(i)}$, $i = 1, 2, \ldots, N$.
  - $c_0 = 0$, $N^{(i)}$=0, $i = 1, 2, \ldots, N$.
- **For** $i = 1, 2, \ldots, N$, **Do**
  - $c_i = c_{i-1} + W^{(i)}$; Construct CDF.
- **End For**
- Draw $u^{(1)} \sim \mathcal{U}(0, \frac{1}{N})$
- $i = 1$
- **For** $j = 1, 2, \ldots, N$, **Do**
  - $u^{(j)} = u^{(1)} + \frac{j-1}{N}$

- **While** $u^{(j)} > c_i$
  - $i = i + 1$
- **End While**
- $\bar{\boldsymbol{x}}_j = \boldsymbol{x}_i$; Assign sample.
- $N^{(i)} = N^{(i)} + 1$

- **End For**

The output comprises the new samples $\bar{\boldsymbol{x}}_j$, $j = 1, 2, \ldots, N$, and all the weights are set equal to $\frac{1}{N}$. The sample $\boldsymbol{x}_i$ will now appear, after resampling, $N^{(i)}$ times.

The previously stated two resampling methods are not the only possibilities (see, e.g., [12]). However, the systematic resampling is the one that is usually adopted due mainly to its easy implementation. Systematic resampling was introduced in [25].

Resampling schema result in estimates that converge to their true values, as long as the number of particles tends to infinity (Problem 17.3).

17.2.3 SEQUENTIAL SAMPLING

Let us now apply the experience we have gained in importance sampling to the case of sequentially arriving particles. The first examples of such techniques date back to the fifties (e.g., [19, 36]). At time n, our interest is to draw samples from the joint distribution

$$p_n(\boldsymbol{x}_{1:n}) = \frac{\phi_n(\boldsymbol{x}_{1:n})}{Z_n}, \tag{17.6}$$

based on a proposal distribution $q_n(\boldsymbol{x}_{1:n})$, where Z_n is the normalizing constant at time n. However, we are going to set the same goal that we have adopted for any time-recursive setting throughout this book, that is, to keep computational complexity *fixed, independent* of the time instant, n. Such a rationale dictates a *time-recursive computation* of the involved quantities. To this end, we select a proposal distribution of the form

$$q_n(\boldsymbol{x}_{1:n}) = q_{n-1}(\boldsymbol{x}_{1:n-1}) q_n(\boldsymbol{x}_n | \boldsymbol{x}_{1:n-1}). \tag{17.7}$$

From Eq. (17.7), it is readily seen that

$$q_n(\boldsymbol{x}_{1:n}) = q_1(\boldsymbol{x}_1) \prod_{k=2}^{n} q_k(\boldsymbol{x}_k | \boldsymbol{x}_{1:k-1}). \tag{17.8}$$

This means that one has only to choose $q_k(\boldsymbol{x}_k | \boldsymbol{x}_{1:k-1})$, $k = 2, 3, \ldots, n$, together with the initial (prior) $q_1(\boldsymbol{x}_1)$. Note that the dimensionality of the involved random vector in $q_k(\cdot|\cdot)$, given the past, remains *fixed* for all time instants. Equation (17.8), viewed from another angle, reveals that in order to draw a single (multivariate) sample that spans the time interval up to time n, that is, $\boldsymbol{x}_{1:n}^{(i)} = \{\boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, \ldots, \boldsymbol{x}_n^{(i)}\}$, we build it up *recursively*; we first draw $\boldsymbol{x}_1^{(i)} \sim q_1(\boldsymbol{x})$ and then draw $\boldsymbol{x}_k^{(i)} \sim q_k(\boldsymbol{x} | \boldsymbol{x}_{1:k-1}^{(i)})$, $k = 2, 3, \ldots, n$. The corresponding nonnormalized weights are also computed recursively [15]. Indeed,

$$\begin{aligned} w_n(\boldsymbol{x}_{1:n}) &:= \frac{\phi_n(\boldsymbol{x}_{1:n})}{q_n(\boldsymbol{x}_{1:n})} = \frac{\phi_{n-1}(\boldsymbol{x}_{1:n-1})}{q_n(\boldsymbol{x}_{1:n})} \frac{\phi_n(\boldsymbol{x}_{1:n})}{\phi_{n-1}(\boldsymbol{x}_{1:n-1})} \\ &= \frac{\phi_{n-1}(\boldsymbol{x}_{1:n-1})}{q_{n-1}(\boldsymbol{x}_{1:n-1})} \frac{\phi_n(\boldsymbol{x}_{1:n})}{\phi_{n-1}(\boldsymbol{x}_{1:n-1}) q_n(\boldsymbol{x}_n | \boldsymbol{x}_{1:n-1})} \end{aligned}$$

$$= w_{n-1}(\boldsymbol{x}_{1:n-1})a_n(\boldsymbol{x}_{1:n})$$
$$= w_1(\boldsymbol{x}_1)\prod_{k=2}^{n} a_k(\boldsymbol{x}_{1:k}), \tag{17.9}$$

where

$$a_k(\boldsymbol{x}_{1:k}) := \frac{\phi_k(\boldsymbol{x}_{1:k})}{\phi_{k-1}(\boldsymbol{x}_{1:k-1})q_k(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1})}, \quad k = 2, 3, \ldots, n. \tag{17.10}$$

The question that is now raised is how to choose $q_n(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1})$, $n = 2, 3, \ldots$. A sensible strategy is to select it in order to minimize the variance of the weight $w_n(\boldsymbol{x}_{1:n})$, given the samples $\boldsymbol{x}_{1:n-1}$. It turns out that the optimal value, which actually makes the variance zero (Problem 17.4), is given by

$$\boxed{q_n^{opt}(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}) = p_n(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}) : \text{ Optimal Proposal Distribution.}} \tag{17.11}$$

However, most often in practice, $p_n(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1})$ is not easy to sample and one has to be content with adopting some approximation of it. We are now ready to state our first algorithm for sequential sampling.

Algorithm 17.2 (Sequential importance sampling (SIS)).

- Select $q_1(\cdot)$, $q_n(\cdot|\cdot)$, $n = 2, 3, \ldots$
- Select number of particles, N.
- **For** $i = 1, 2, \ldots, N$, **Do**; Initialize N different realizations/streams.
  - Draw $\boldsymbol{x}_1^{(i)} \sim q_1(\boldsymbol{x})$
  - Compute the weights $w_1(\boldsymbol{x}_1^{(i)}) = \frac{\phi_1(\boldsymbol{x}_1^{(i)})}{q_1(\boldsymbol{x}_1^{(i)})}$
- **End For**
- **For** $i = 1, 2, \ldots, N$, **Do**
  - Compute the normalized weights $W_1^{(i)}$.
- **End For**
- **For** $n = 2, 3, \ldots,$ **Do**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - Draw $\boldsymbol{x}_n^{(i)} \sim q_n(\boldsymbol{x}|\boldsymbol{x}_{1:n-1}^{(i)})$
    - Compute the weights

$$w_n(\boldsymbol{x}_{1:n}^{(i)}) = w_{n-1}(\boldsymbol{x}_{1:n-1}^{(i)})a_n(\boldsymbol{x}_{1:n}^{(i)}); \text{ from Eq. (17.10).}$$

  - **End For**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - $W_n^{(i)} \propto w_n(\boldsymbol{x}_{1:n}^{(i)})$
  - **End For**
- **End For**

Once the algorithm has been completed, we can write

$$\hat{p}_n(\boldsymbol{x}_{1:n}) = \sum_{i=1}^{N} W_n^{(i)}\delta(\boldsymbol{x}_{1:n} - \boldsymbol{x}_{1:n}^{(i)}).$$

However, as we have already said, the variance of the weights has the tendency to increase with n (see Problem 17.5). Thus, the resampling version of the sequential importance sampling is usually employed.

Algorithm 17.3 (SIS with resampling).

- Select $q_1(\cdot)$, $q_n(\cdot|\cdot)$, $n = 1, 2, \ldots$
- Select number of particles N.
- **For** $i = 1, 2, \ldots, N$, **Do**
  - Draw $\boldsymbol{x}_1^{(i)} \sim q_1(\boldsymbol{x})$
  - Compute the weights $w_1(\boldsymbol{x}_1^{(i)}) = \dfrac{\phi_1(\boldsymbol{x}_1^{(i)})}{q_1(\boldsymbol{x}_1^{(i)})}$.
- **End For**
- **For** $i = 1, \ldots, N$, **Do**
  - Compute the normalized weights $W_1^{(i)}$
- **End For**
- Resample $\{\boldsymbol{x}_1^{(i)}, W_1^{(i)}\}_{i=1}^N$ to obtain $\{\bar{\boldsymbol{x}}_1^{(i)}, \frac{1}{N}\}_{i=1}^N$, using Algorithm 17.1
- **For** $n = 2, 3, \ldots,$ **Do**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - Draw $\boldsymbol{x}_n^{(i)} \sim q_n(\boldsymbol{x}|\bar{\boldsymbol{x}}_{1:n-1}^{(i)})$
    - Set $\boldsymbol{x}_{1:n}^{(i)} = \{\boldsymbol{x}_n^{(i)}, \bar{\boldsymbol{x}}_{1:n-1}^{(i)}\}$
    - Compute $w_n(\boldsymbol{x}_{1:n}^{(i)}) = \frac{1}{N} a_n(\boldsymbol{x}_{1:n}^{(i)})$; (Eq. (17.10)).
  - **End For**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - Compute $W_n^{(i)}$
  - **End Do**
  - Resample $\{\boldsymbol{x}_{1:n}^{(i)}, W_n^{(i)}\}_{i=1}^N$ to obtain $\{\bar{\boldsymbol{x}}_{1:n}^{(i)}, \frac{1}{N}\}_{i=1}^N$
- **End For**

Remarks 17.1.

- Convergence results concerning sequential importance sampling can be found in, for example, [5–7]. It turns out that, in practice, the use of resampling leads to substantially smaller variances.
- From a practical point of view, sequential importance methods with resampling are expected to work reasonably well, if the desired successive distributions at different time instants do not differ much and the choice of $q_n(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1})$ is *close to the optimal* one (see, e.g., [15]).

17.3 KALMAN AND PARTICLE FILTERING

Particle filtering is an instance of the sequential Monte Carlo methods. Particle filtering is a technique born in the 1990s and it was first introduced in [18] as an attempt to solve estimation tasks in the context of state-space modeling for the more general nonlinear and non-Gaussian scenarios. The term "particle filtering" was coined in [3], although the term "particle" had been used in [25].

Hidden Markov models, which are treated in Section 16.4, and Kalman filters, treated in Chapter 4, are special types of the state-space (state-observation) modeling. The former address the case of discrete state (latent) variables and the latter the continuous case, albeit at the very special case of linear and Gaussian scenario. In particle filtering, the interest shifts to models of the following form:

$$\mathbf{x}_n = \boldsymbol{f}_n(\mathbf{x}_{n-1}, \boldsymbol{\eta}_n): \quad \text{State Equation} \tag{17.12}$$

$$\mathbf{y}_n = \boldsymbol{h}_n(\mathbf{x}_n, \mathbf{v}_n): \quad \text{Observations Equation,} \tag{17.13}$$

where $\boldsymbol{f}_n(\cdot,\cdot)$ and $\boldsymbol{h}_n(\cdot,\cdot)$ are nonlinear, in general, (vector) functions; $\boldsymbol{\eta}_n$ and $\mathbf{v}_n$ are noise sequences; and the dimensions of $\mathbf{x}_n$ and $\mathbf{y}_n$ can be different. The random vector $\mathbf{x}_n$ is the (latent) state vector and $\mathbf{y}_n$ corresponds to the observations. There are two inference tasks that are of interest in practice.

Filtering: Given the set of measurements, $\boldsymbol{y}_{1:n}$, in the time interval $[1, n]$, compute

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}).$$

Smoothing: Given the set of measurements $\boldsymbol{y}_{1:N}$ in a time interval $[1, N]$, compute

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:N}), \quad 1 \le n \le N.$$

Before we proceed to our main goal, let us review the simpler case, that of Kalman filters, this time from a Bayesian viewpoint.

17.3.1 KALMAN FILTERING: A BAYESIAN POINT OF VIEW

Kalman filtering was first discussed in Section 4.10 in the context of linear estimation methods and the mean-square error criterion. In the current section, the Kalman filtering algorithm will be rederived following concepts from the theory of graphical models and Bayesian networks, which are treated in Chapters 15 and 16. This probabilistic view will then be used for the subsequent nonlinear generalizations in the framework of particle filtering. For the linear case model, Eqs. (17.12) and (17.13) become

$$\mathbf{x}_n = F_n\mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \tag{17.14}$$

$$\mathbf{y}_n = H_n\mathbf{x}_n + \mathbf{v}_n, \tag{17.15}$$

where F_n and H_n are matrices of appropriate dimensions. We further assume that the two noise sequences are statistically independent and of a Gaussian nature, that is,

$$p(\boldsymbol{\eta}_n) = \mathcal{N}(\boldsymbol{\eta}_n|\mathbf{0}, Q_n), \tag{17.16}$$

$$p(\boldsymbol{\upsilon}_n) = \mathcal{N}(\boldsymbol{\upsilon}_n|\mathbf{0}, R_n). \tag{17.17}$$

The kick-off point for deriving the associated recursions is the Bayes rule,

$$\begin{aligned} p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}) &= \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{y}_{1:n-1})p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})}{Z_n} \\ &= \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})}{Z_n}, \end{aligned} \tag{17.18}$$

where

$$\begin{aligned} Z_n &= \int p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})\,\mathrm{d}\boldsymbol{x}_n \\ &= p(\boldsymbol{y}_n|\boldsymbol{y}_{1:n-1}), \end{aligned} \tag{17.19}$$

and we have used the fact that $p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{y}_{1:n-1}) = p(\boldsymbol{y}_n|\boldsymbol{x}_n)$, which is a consequence of (17.15). For those who have already read Chapter 15, recall that Kalman filtering is a special case of a Bayesian network

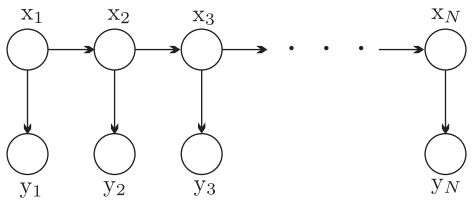

FIGURE 17.2

Graphical model corresponding to the state-space modeling for Kalman and particle filters.

and corresponds to the graphical model given in Figure 17.2. Hence, due to the Markov property, $\mathbf{y}_n$ is independent of the past given the values in $\boldsymbol{x}_n$. Moreover, note that

$$\begin{aligned} p(\boldsymbol{x}_n|\mathbf{y}_{1:n-1}) &= \int p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1},\mathbf{y}_{1:n-1})p(\boldsymbol{x}_{n-1}|\mathbf{y}_{1:n-1})\,\mathrm{d}\boldsymbol{x}_{n-1} \\ &= \int p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{x}_{n-1}|\mathbf{y}_{1:n-1})\,\mathrm{d}\boldsymbol{x}_{n-1}, \end{aligned} \tag{17.20}$$

where, once more, the Markov property (i.e., Eq. (17.14)) has been used.

Equations (17.18)–(17.20) comprise the set of recursions, which lead to the update

$$p(\boldsymbol{x}_{n-1}|\mathbf{y}_{1:n-1}) \longrightarrow p(\boldsymbol{x}_n|\mathbf{y}_{1:n}),$$

starting from the initial (prior) $p(\boldsymbol{x}_0|\mathbf{y}_0) := p(\boldsymbol{x}_0)$. If $p(\boldsymbol{x}_0)$ is chosen to be Gaussian, then all the involved pdfs turn out to be Gaussian due to Eqs. (17.16) and (17.17), and the linearity of Eqs. (17.14) and (17.15); this makes the computational of the integrals a trivial task following the recipe rules in the Appendix in Section 12.9.

Before we proceed further, note that the recursions in Eqs. (17.18) and (17.20) are an instance of the sum-product algorithm for graphical models. Indeed, to put our current discussion in this context, let us compactly write the previous recursions as

$$\boxed{p(\boldsymbol{x}_n|\mathbf{y}_{1:n}) = \underbrace{\frac{p(\mathbf{y}_n|\boldsymbol{x}_n)}{Z_n}}_{\text{corrector}} \underbrace{\int p(\boldsymbol{x}_{n-1}|\mathbf{y}_{1:n-1})p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})\,\mathrm{d}\boldsymbol{x}_{n-1}}_{\text{predictor}} : \text{ Filtering.}} \tag{17.21}$$

Note that this is of exactly the same form, within the normalizing factor, as Eq. (16.43) of Chapter 16; just replace summation with integration. One can rederive Eq. (17.21) using the sum-product rule, following similar steps as for Eq. (16.43). The only difference is that the normalizing constant has to be involved in all respective definitions and we replace summations with integrations. Because all the involved pdfs are Gaussians, the computation of the involved normalizing constants is trivially done; moreover, it suffices to derive recursions only for the respective mean values and covariances.

In Eq. (17.20), we have that

$$p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}) = \mathcal{N}(\boldsymbol{x}_n|F_n\boldsymbol{x}_{n-1}, Q_n).$$

Let, also, $p(\boldsymbol{x}_{n-1}|\mathbf{y}_{1:n-1})$ be Gaussian with mean and covariance matrix,

$$\boldsymbol{\mu}_{n-1|n-1}, \quad P_{n-1|n-1},$$

respectively, where the notation is chosen so that in order for the derived recursions to comply with the resulting algorithm in Section 4.10. Then, according to the Appendix in Section 12.9, $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})$ is a Gaussian marginal pdf with mean and covariance given by (see Eqs. (12.147) and (12.148))

$$\boldsymbol{\mu}_{n|n-1} = F_n\boldsymbol{\mu}_{n-1|n-1}, \tag{17.22}$$

$$P_{n|n-1} = Q_n + F_nP_{n-1|n-1}F_n^T. \tag{17.23}$$

Also, in Eq. (17.18) we have that

$$p(\boldsymbol{y}_n|\boldsymbol{x}_n) = \mathcal{N}(\boldsymbol{y}_n|H_n\boldsymbol{x}_n, R_n).$$

From Section 12.9, and taking into account (Eqs. (17.22) and (17.23)), we get that $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n})$ is the posterior (Gaussian) with mean and covariance given by (see Eqs. (12.145) and (12.146))

$$\boldsymbol{\mu}_{n|n} = \boldsymbol{\mu}_{n|n-1} + K_n(\boldsymbol{y}_n - H_n\boldsymbol{\mu}_{n|n-1}), \tag{17.24}$$

$$P_{n|n} = P_{n|n-1} - K_nH_nP_{n|n-1}, \tag{17.25}$$

where

$$K_n = P_{n|n-1}H_n^TS_n^{-1}, \tag{17.26}$$

and

$$S_n = R_n + H_nP_{n|n-1}H_n^T. \tag{17.27}$$

Note that these are exactly the same recursions that were derived in Section 4.10 for the state estimation; recall that under the Gaussian assumption, the posterior mean coincides with the least-squares estimate.

Here we have assumed that matrices F_n, H_n as well as the covariance matrices are known. This is most often the case. If not, these can be learned using similar arguments as those used in learning the hidden Markov model (HMM) parameters, which are discussed in Section 16.5.2 (see, e.g., [2]).

17.4 PARTICLE FILTERING

In Section 4.10, extended Kalman filtering (EKF) was discussed as one possibility to generalize Kalman filtering to nonlinear models. Particle filtering, to be discussed next, is a powerful alternative technique to EKF. The involved pdfs are approximated by *discrete random measures*. The underlying theory is that of sequential importance sampling (SIS); as a matter of fact, particle filtering is an instance of SIS.

Let us now consider the state-space model of the general form in Eqs. (17.12) and (17.13). From the specific form of these equations (and by the Bayesian network nature of such models, for the more familiar reader) we can write

$$p(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}, \boldsymbol{y}_{1:n-1}) = p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}) \tag{17.28}$$

and

$$p(\boldsymbol{y}_n|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n-1}) = p(\boldsymbol{y}_n|\boldsymbol{x}_n). \tag{17.29}$$

Our starting point is the *sequential* estimation of $p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$; the estimation of $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n})$, which comprises our main goal, will be obtained as a by-product. Note that [15]

$$\begin{aligned} p(\boldsymbol{x}_{1:n},\boldsymbol{y}_{1:n}) &= p(\boldsymbol{x}_n,\boldsymbol{x}_{1:n-1},\boldsymbol{y}_n,\boldsymbol{y}_{1:n-1}) \\ &= p(\boldsymbol{x}_n,\boldsymbol{y}_n|\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n-1})p(\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n-1}) \\ &= p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n-1}), \end{aligned} \tag{17.30}$$

where Eqs. (17.28) and (17.29) have been employed.

Our goal is to obtain an approximation, via the generation of particles, of the conditional pdf,

$$p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = \frac{p(\boldsymbol{x}_{1:n},\boldsymbol{y}_{1:n})}{\int p(\boldsymbol{x}_{1:n},\boldsymbol{y}_{1:n})\,\mathrm{d}\boldsymbol{x}_{1:n}} = \frac{p(\boldsymbol{x}_{1:n},\boldsymbol{y}_{1:n})}{Z_n}, \tag{17.31}$$

where

$$Z_n := \int p(\boldsymbol{x}_{1:n},\boldsymbol{y}_{1:n})\,\mathrm{d}\boldsymbol{x}_{1:n}.$$

To put the current discussion in the general framework of SIS, compare Eq. (17.31) with Eq. (17.6), which leads to the definition

$$\phi_n(\boldsymbol{x}_{1:n}) := p(\boldsymbol{x}_{1:n},\boldsymbol{y}_{1:n}). \tag{17.32}$$

Then, Eq. (17.9) becomes

$$w_n(\boldsymbol{x}_{1:n}) = w_{n-1}(\boldsymbol{x}_{1:n-1})\alpha_n(\boldsymbol{x}_{1:n}), \tag{17.33}$$

where now

$$\alpha_n(\boldsymbol{x}_{1:n}) = \frac{p(\boldsymbol{x}_{1:n},\boldsymbol{y}_{1:n})}{p(\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n-1})q_n(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n})},$$

which from Eq. (17.30) becomes

$$\alpha_n(\boldsymbol{x}_{1:n}) = \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})}{q_n(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n})}. \tag{17.34}$$

The final step is to select the proposal distribution. From Section 17.2, recall that the optimal proposal distribution is given from Eq. (17.11), which for our case takes the form

$$\begin{aligned} q_n^{opt}(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n}) &= p(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n}) \\ &= p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1},\boldsymbol{x}_{1:n-2},\boldsymbol{y}_n,\boldsymbol{y}_{1:n-1}), \end{aligned}$$

and exploiting the underlying independencies, as they are imposed by the Bayesian network structure of the state-space model, we finally get

$$\boxed{q^{opt}(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n}) = p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1},\boldsymbol{y}_n) : \text{ Optimal Proposal Distribution.}} \tag{17.35}$$

The use of the optimal proposal distribution leads to the following weight update recursion (Problem 17.6):

$$\boxed{w_n(\boldsymbol{x}_{1:n}) = w_{n-1}(\boldsymbol{x}_{1:n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_{n-1}) : \quad \text{Optimal Weights.}} \tag{17.36}$$

However, as is most often the case in practice, optimality is not always easy to obtain. Note that Eq. (17.36) requires the following integration,

$$p(\boldsymbol{y}_n|\boldsymbol{x}_{n-1}) = \int p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})\,\mathrm{d}\boldsymbol{x}_n,$$

which may not be tractable. Moreover, even if the integral can be computed, sampling from $p(\boldsymbol{y}_n|\boldsymbol{x}_{n-1})$ directly may not be feasible. In any case, even if the optimal proposal distribution cannot be used, we can still select the proposal distribution to be of the form

$$q_n(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1},\boldsymbol{y}_{1:n}) = q(\boldsymbol{x}_n|\boldsymbol{x}_{n-1},\boldsymbol{y}_n). \tag{17.37}$$

Note that such a choice is particularly convenient, because sampling at time, n, only depends on $\boldsymbol{x}_{n-1}$, and $\boldsymbol{y}_n$ and *not* on the entire history. If, in addition, the goal is to obtain estimates of $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n})$, then one need not keep in memory all previously generated samples, but only the most recent one, $\boldsymbol{x}_n$.

We are now ready to write the first particle-filtering algorithm.

Algorithm 17.4 (SIS particle filtering).

- Select a prior distribution, p, to generate the initial state $\boldsymbol{x}_0$.
- Select the number of particle streams, N.
- **For** $i = 1, 2, \ldots, N$, **Do**
  - Draw $\boldsymbol{x}_0^{(i)} \sim p(\boldsymbol{x})$; Initialize the N streams of particles.
  - Set $w_0^{(i)} = \frac{1}{N}$; Set all initial weights equal.
- **End For**
- **For** $n = 1, 2, \ldots,$ **Do**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - Draw $\boldsymbol{x}_n^{(i)} \sim q(\boldsymbol{x}|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_n)$
    - $w_n^{(i)} = w_{n-1}^{(i)} \dfrac{p(\boldsymbol{x}_n^{(i)}|\boldsymbol{x}_{n-1}^{(i)})p(\boldsymbol{y}_n|\boldsymbol{x}_n^{(i)})}{q(\boldsymbol{x}_n^{(i)}|\boldsymbol{x}_{n-1}^{(i)}\boldsymbol{y}_n)}$; formulae (17.33), (17.34), and (17.37).
  - **End For**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - Compute the normalized weights $W_n^{(i)}$
  - **End For**
- **End For**

Note that the generation of the N streams of particles can take place *concurrently*, by exploiting parallel processing capabilities, if they are available in the processor.

The particles generated along the ith stream $\boldsymbol{x}_n^{(i)}$, $n = 1, 2, \ldots$, represent a *path/trajectory* through the state-space. Once the particles have been drawn and the normalized weights computed, we obtain the estimate

$$\hat{p}(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta(\boldsymbol{x}_{1:n} - \boldsymbol{x}_{1:n}^{(i)}).$$

If, as commented earlier, our interest lies in keeping the terminal sample, $\boldsymbol{x}_n^{(i)}$, only, then discarding the path history, $\boldsymbol{x}_{1:n-1}^{(i)}$, we can write

$$\hat{p}(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}) = \sum_{i=1}^{N} W_n^{(i)} \delta(\boldsymbol{x}_n - \boldsymbol{x}_n^{(i)}).$$

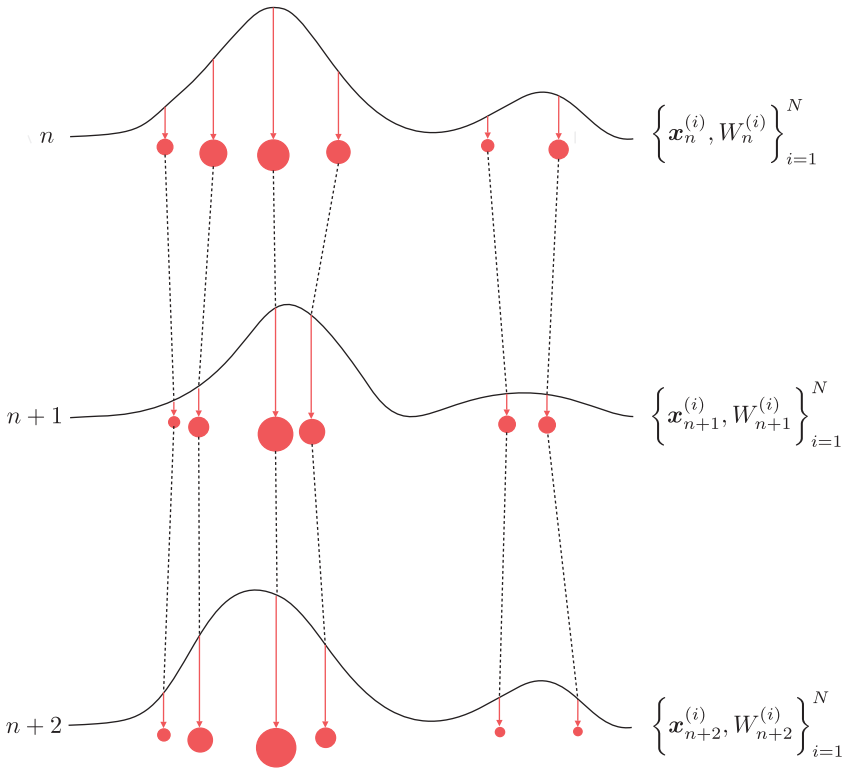

FIGURE 17.3

Three consecutive recursions, for the particle-filtering scheme given in Algorithm 17.4, with $N = 7$ streams of particles. The area of the circles corresponds to the size of the normalized weights of the respective particles drawn from the proposal distribution.

Note that as the number of particles, N, tends to infinity, the previous approximations tend to the true posterior densities. Figure 17.3 provides a graphical interpretation of the SIS algorithm 17.4.

Example 17.1. Consider the one-dimensional random walk model written as

$$\mathrm{x}_n = \mathrm{x}_{n-1} + \eta_n, \tag{17.38}$$

$$\mathrm{y}_n = \mathrm{x}_n + \mathrm{v}_n, \tag{17.39}$$

where $\eta_n \sim \mathcal{N}(\eta_n|0, \sigma_\eta^2)$, $\mathrm{v}_n \sim \mathcal{N}(v_n|0, \sigma_u^2)$, with $\sigma_\eta^2 = 1$, $\sigma_u^2 = 1$. Although this is a typical task for (linear) Kalman filtering, we will attack it here via the particle-filtering rationale in order to demonstrate some of the previously reported performance-related issues. The proposal distribution is selected to be

$$q(x_n|x_{n-1}, y_n) = p(x_n|x_{n-1}) = \mathcal{N}(x_n|x_{n-1}, \sigma_\eta^2).$$

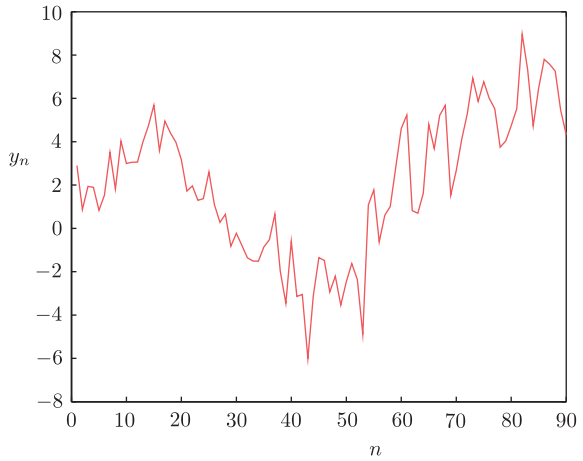

FIGURE 17.4

The observation sequence for Example 17.1.

1. Generate $T = 100$ observations, $y_n, n = 1, 2, \ldots, T$, to be used by the Algorithm 17.4. To this end, start with an arbitrary state value, for example, $x_0 = 0$ and generate a realization of the random walk, drawing samples from the Gaussians (we know how to generate Gaussian samples) $\mathcal{N}(\cdot|0, \sigma_n^2)$ and $\mathcal{N}(\cdot|0, \sigma_u^2)$, according to Eqs. (17.38) and (17.39). Figure 17.4 shows a realization for the output variable. Our goal is to use the sequence of the resulting observations to generate particles and demonstrate the increase of the variance of the associated weights as time goes by.
2. Use $\mathcal{N}(\cdot|0, 1)$ to initialize $N = 200$ particle streams, $x^{(i)}$, $i = 1, 2, \ldots, N$, and initialize the normalized weights to equal values, $W_0^{(i)} = \frac{1}{N}, i = 1, 2, \ldots, N$. Figure 17.5 provides the corresponding plot.
3. Perform Algorithm 17.4 and plot the resulting particles together with the respective weights at time instants, $n = 0$, $n = 1$, $n = 3$, and $n = 30$. Observe how the variance of the weights increases with time. At time $n = 30$ only a few particles have nonzero weights.
4. Repeat the experiment with $N = 1000$. Figure 17.6 is the counterpart of Figure 17.5 for the snapshots of $n = 3$ and $n = 30$. Observe that increasing the number of particles improves the performance with respect to the variance of weights. This is one path to obtain more particles with significant weight values. The other path is via resampling techniques.

17.4.1 DEGENERACY

Particle filtering is a special case of sequential importance sampling; hence, everything that has been said in Section 17.2 concerning the respective performance is also applied here.

A major problem is the *degeneracy* phenomenon. The variance of the importance weights increases in time, and after a few iterations only very few (or even only one) of the particles are assigned

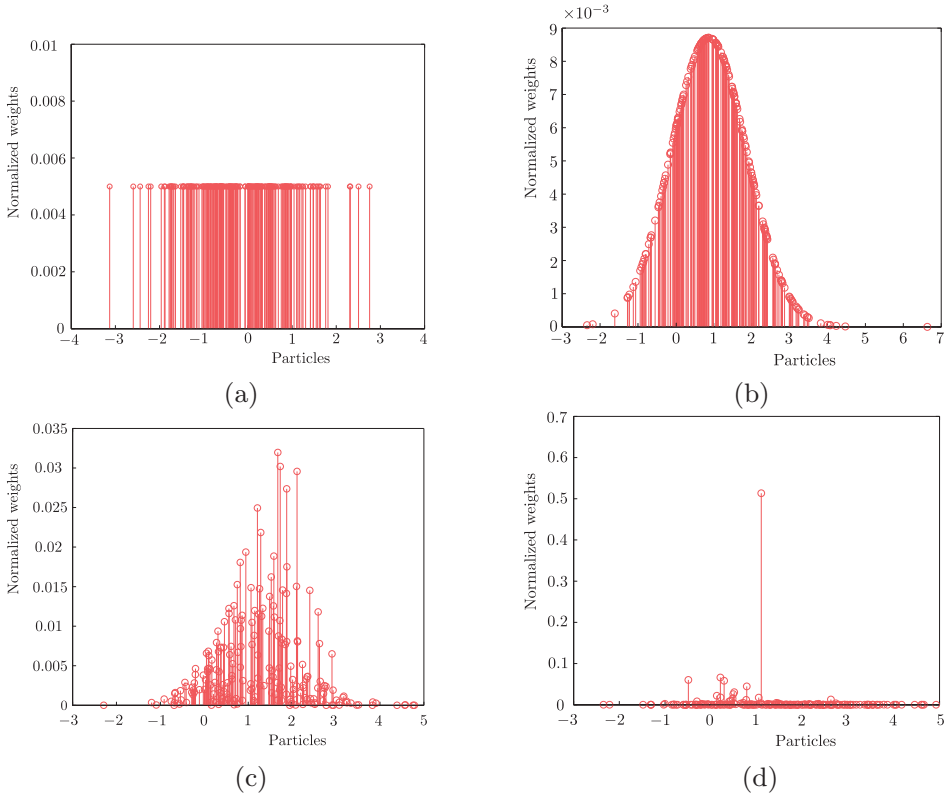

FIGURE 17.5

Plot of $N = 200$ generated particles with the corresponding (normalized) weights, for Example 17.1, at time instants (a) $n = 0$, (b) $n = 1$, (c) $n = 3$, and (d) $n = 30$. Observe that as time goes by, the variance of the weights increases. At time $n = 30$, only very few particles have a nonzero weight value.

nonnegligible weights, and the discrete random measure degenerates quickly. There are two methods for reducing degeneracy: one is selecting a good proposal distribution and the other is resampling.

We know the optimal choice for the proposal distribution is

$$q(\cdot|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_n) = p(\cdot|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_n).$$

There are cases where this is available in analytic form. For example, this happens if the noise sources are Gaussian and the observation equation is linear (e.g., [11]). If analytic forms are not available and direct sampling is not possible, approximations of $p(\cdot|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_n)$ are mobilized. Our familiar (from Chapter 12) Gaussian approximation via local linearization of $\ln p(\cdot|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_n)$ is a possibility [11]. The use of suboptimal filtering techniques such as the extended/unscented Kalman filter have also been advocated [37]. In general, it must be kept in mind that the choice of the proposal distribution plays a *crucial* role in the performance of particle filtering. Resampling is the other path that has been discussed

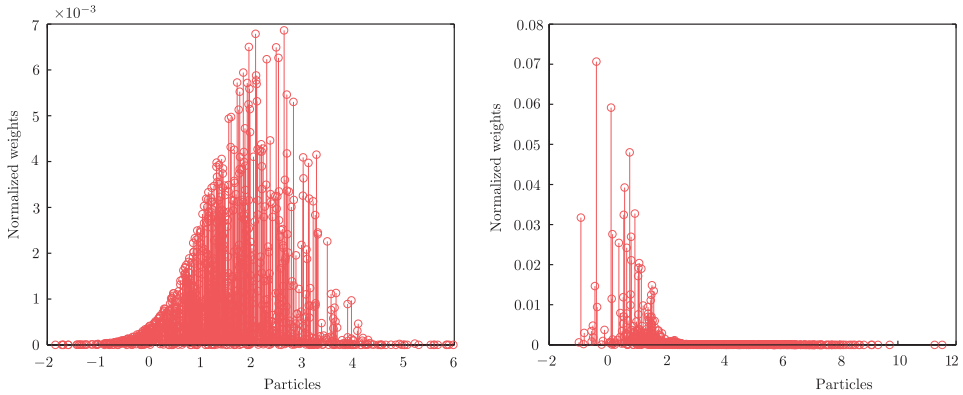

FIGURE 17.6

Plot of $N = 1000$ generated particles with the corresponding (normalized) weights, for Example 17.1, at time instants (a) $n = 3$, (b) $n = 30$. As expected, compared to Figure 17.5, more particles with significant weights survive.

in Section 17.2.2. The counterpart of Algorithm 17.1 can also be adopted for the case of particle filtering. However, we are going to give a slightly modified version of it.

17.4.2 GENERIC PARTICLE FILTERING

Resampling has a number of advantages. It discards, with high probability, particles of low weights; that is, only particles corresponding to regions of high-probability mass are propagated. Of course, resampling has its own limitations. For example, a particle of low weight at time, n, will not necessarily have a low weight at later time instants. In such a case, resampling is rather wasteful. Moreover, resampling limits the potential of parallelizing the computational process, because particles along the different streams have to be "combined" at each time instant. However, some efforts for enhancing parallelism have been reported (see, e.g., [21]). Also, particles corresponding to high values of weights are drawn many times and lead to a set of samples of low diversity; this phenomenon is also known as *sample impoverishment*. The effects of this phenomenon become more severe in cases of low state/process noise, $\boldsymbol{\eta}_n$, in Eq. (17.12), where the set of the sampling points may end up comprising a single point (e.g., [1]).

Hence, avoiding resampling can be beneficial. In practice, resampling is performed only if a related metric of the variance of the weights is below a threshold. In [28, 29], the *effective number* of samples is approximated by

$$N_{eff} \approx \frac{1}{\sum_{i=1}^{N} \left(W_n^{(i)}\right)^2}. \tag{17.40}$$

The value of this index ranges from 1 to N. Resampling is performed if $N_{eff} \leq N_T$, typically with $N_T = \frac{N}{2}$.

Algorithm 17.5 (Generic particle filtering).

- Select a prior distribution, p, to generate particles for the initial state $\mathbf{x}_0$.
- Select the number of particle streams, N.
- **For** $i = 1, 2, \ldots, N$, **Do**
  - Draw $\boldsymbol{x}_0^{(i)} \sim p(\boldsymbol{x})$; Initialize N streams.
  - set $W_0^{(i)} = \frac{1}{N}$; All initial normalized weights are equal.
- **End For**
- **For** $n = 1, 2, 3, \ldots,$ **Do**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - Draw $\boldsymbol{x}_n^{(i)} \sim q(\boldsymbol{x}|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_n)$
    - $w_n^{(i)} = w_{n-1}^{(i)} \dfrac{p(\boldsymbol{x}_n^{(i)}|\boldsymbol{x}_{n-1}^{(i)}) p(\boldsymbol{y}_n|\boldsymbol{x}_n^{(i)})}{q(\boldsymbol{x}_n^{(i)}|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_n)}$
  - **End For**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - Compute the normalized $W_n^{(i)}$.
  - **End For**
  - Compute N_{eff}; Eq. (17.40).
  - **If** $N_{eff} \leq N_T$; preselected value N_T.
    - Resample $\{\boldsymbol{x}_n^{(i)}, W_n^{(i)}\}_{i=1}^N$ to obtain $\{\bar{\boldsymbol{x}}_n^{(i)}, \frac{1}{N}\}_{i=1}^N$
    - $\boldsymbol{x}_n^{(i)} = \bar{\boldsymbol{x}}_n^{(i)}$, $w_n^{(i)} = \frac{1}{N}$
  - **End If**
- **End For**

Figure 17.7 presents a graphical illustration of the time evolution of the algorithm.

Remarks 17.2.

- A popular choice for the proposal distribution is the prior,

$$q(\boldsymbol{x}|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_n) = p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(i)}),$$

which yields the following weights' update recursion,

$$w_n^{(i)} = w_{n-1}^{(i)} p(\boldsymbol{y}_n|\boldsymbol{x}_n^{(i)}).$$

The resulting algorithm is known as *sampling-importance-resampling* (SIR). The great advantage of such a choice is its simplicity. However, the generation mechanism of particles ignores important information that resides in the observation sequence; the proposal distribution is independent of the observations. This may lead to poor results. A remedy can be offered by the use of auxiliary particle filtering, to be reviewed next. Another possibility is discussed in [22], via a combination of the prior and the optimal proposal distributions.

Example 17.2. Repeat example 17.1, using $N = 200$ particles, for Algorithm 17.5. Use the threshold value $N_T = 100$. Observe in Figure 17.8 that, for the corresponding time instants, more particles with significant weights are generated compared to Figure 17.5.

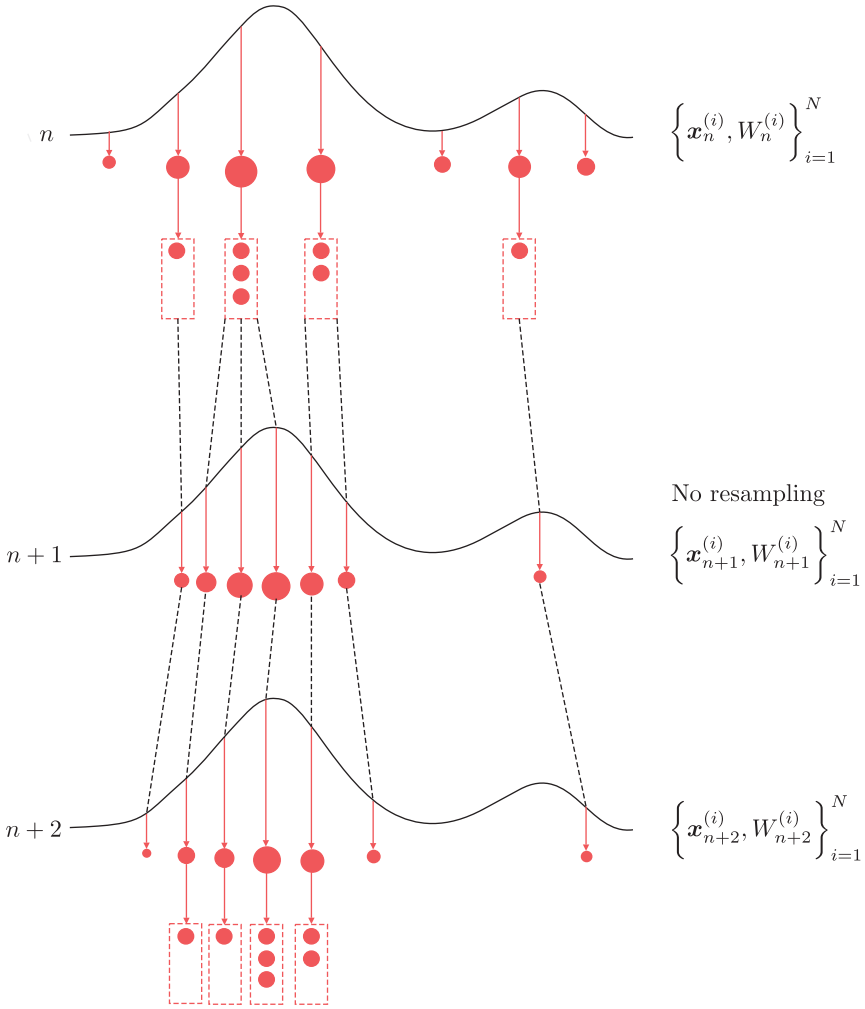

FIGURE 17.7

Three successive time iterations for $N = 7$ streams of particles corresponding to Algorithm 17.5. At steps n and $n+2$ resampling is performed. At step $n+1$ no resampling is needed.

17.4.3 AUXILIARY PARTICLE FILTERING

Auxiliary particle filters were introduced in [34] in order to improve performance when dealing with heavy-tailed distributions. The method introduces an auxiliary variable; this is the index of a particle at the previous time instant. We allow for a particle in the ith stream at time n to be drawn using a particle from a different stream at time $n-1$. Let the ith particle at time n be $\boldsymbol{x}_n^{(i)}$ and the index of its "parent"

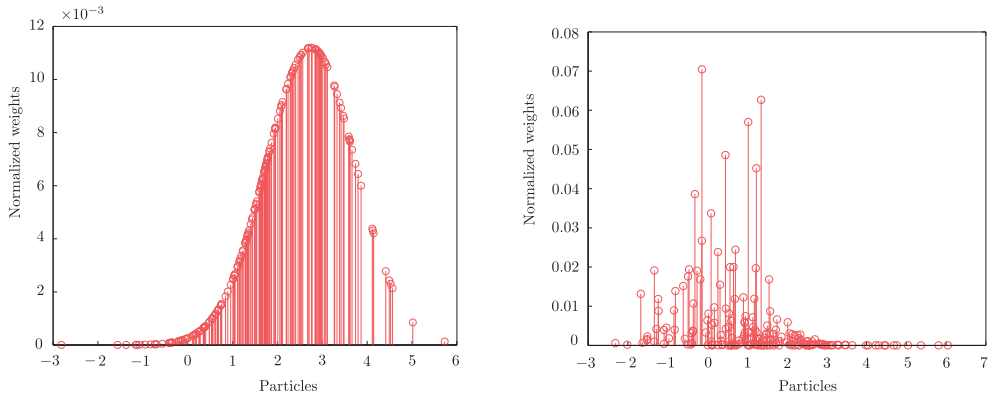

FIGURE 17.8

Plot of $N = 200$ generated particles with the corresponding (normalized) weights, for Example 17.2 using resampling, at time instants (a) $n = 3$, (b) $n = 30$. Compared to Figure 17.5c, d, more particles with significant weights survive.

particle at time $n-1$ be i_{n-1}. The idea is to sample for the pair $(\boldsymbol{x}_n^{(i)}, i_{n-1})$, $i = 1, 2, \ldots, N$. Employing Bayes rule, we obtain

$$\begin{aligned} p(\boldsymbol{x}_n, i|\boldsymbol{y}_{1:n}) &\propto p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n, i|\boldsymbol{y}_{1:n-1}) \\ &= p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|i, \boldsymbol{y}_{1:n-1})P(i|\boldsymbol{y}_{1:n-1}), \end{aligned} \tag{17.41}$$

where the conditional independencies underlying the state-space model have been used. To unclutter notation, we have used $\boldsymbol{x}_n$ in place of $\boldsymbol{x}_n^{(i)}$, and the subscript $n-1$ has been dropped from i_{n-1} and we use i instead. Note that by the definition of the index i_{n-1}, we have

$$p(\boldsymbol{x}_n|i, \boldsymbol{y}_{1:n-1}) = p\left(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(i)}, \boldsymbol{y}_{1:n-1}\right) = p\left(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(i)}\right), \tag{17.42}$$

and also

$$P(i|\boldsymbol{y}_{1:n-1}) = W_{n-1}^{(i)}. \tag{17.43}$$

Thus, we can write

$$p\left(\boldsymbol{x}_n, i|\boldsymbol{y}_{1:n}\right) \propto p(\boldsymbol{y}_n|\boldsymbol{x}_n)p\left(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(i)}\right) W_{n-1}^{(i)}. \tag{17.44}$$

The proposal distribution is chosen as

$$q\left(\boldsymbol{x}_n, i|\boldsymbol{y}_{1:n}\right) \propto p\left(\boldsymbol{y}_n|\boldsymbol{\mu}_n^{(i)}\right) p\left(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(i)}\right) W_{n-1}^{(i)}. \tag{17.45}$$

Note that we have used $\boldsymbol{\mu}_n^{(i)}$ in place of $\boldsymbol{x}_n$ in $p(\boldsymbol{y}_n|\boldsymbol{x}_n)$, because $\boldsymbol{x}_n$ is still to be drawn. The estimate $\boldsymbol{\mu}_n^{(i)}$ is chosen in order to be easily computed and at the same time to be a good representative of $\boldsymbol{x}_n$. Typically, $\boldsymbol{\mu}_n^{(i)}$ can be the mean, the mode, a draw, or another value associated with the distribution $p\left(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(i)}\right)$.

For example, $\boldsymbol{\mu}_n^{(i)} \sim p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(i)})$. Also, if the state equation is $\boldsymbol{x}_n = \boldsymbol{f}(\boldsymbol{x}_{n-1}) + \boldsymbol{\eta}_n$, a good choice would be $\boldsymbol{\mu}_n^{(i)} = \boldsymbol{f}(\boldsymbol{x}_{n-1}^{(i)})$.

Applying the Bayes rule in Eq. (17.45) and adopting

$$q\left(\boldsymbol{x}_n|i,\boldsymbol{y}_{1:n}\right) = p\left(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^{(i)}\right),$$

we obtain

$$q\left(i|\boldsymbol{y}_{1:n}\right) \propto p\left(\boldsymbol{y}_n|\boldsymbol{\mu}_n^{(i)}\right) W_{n-1}^{(i)}. \tag{17.46}$$

Hence, we draw the value of the index i_{n-1} from a multinomial distribution, that is,

$$i_{n-1} \sim q\left(i|\boldsymbol{y}_{1:n}\right) \propto p\left(\boldsymbol{y}_n|\boldsymbol{\mu}_n^{(i)}\right) W_{n-1}^{(i)}, \quad i = 1, 2, \ldots, N. \tag{17.47}$$

The index i_{n-1} identifies the distribution from which $\boldsymbol{x}_n^{(i)}$ will be drawn, that is,

$$\boldsymbol{x}_n^{(i)} \sim p\left(\boldsymbol{x}_n \left| \boldsymbol{x}_{n-1}^{(i_{n-1})} \right.\right), \quad i = 1, 2, \ldots, N. \tag{17.48}$$

Note that Eq. (17.47) actually performs a resampling. However, now, the resampling at time $n-1$ takes into consideration information that becomes available at time n, via the observation $\boldsymbol{y}_n$. This information is exploited in order to determine which particles are to survive, after resampling at a given time instant, so that their "offsprings" are likely to land in regions of high-probability mass. Once sample $\boldsymbol{x}_n^{(i)}$ has been drawn, the index i_{n-1} is discarded, which is equivalent to marginalizing $p(\boldsymbol{x}_n, i|\boldsymbol{y}_{1:n})$ to obtain $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n})$. Each sample $\boldsymbol{x}_n^{(i)}$ is finally assigned a weight according to

$$w_n^{(i)} \propto \frac{p\left(\boldsymbol{x}_n^{(i)}, i_{n-1}|\boldsymbol{y}_{1:n}\right)}{q\left(\boldsymbol{x}_n^{(i)}, i_{n-1}|\boldsymbol{y}_{1:n}\right)} = \frac{p\left(\boldsymbol{y}_n|\boldsymbol{x}_n^{(i)}\right)}{p\left(\boldsymbol{y}_n|\boldsymbol{\mu}_n^{(i)}\right)},$$

which results by dividing the right-hand sides of Eqs. (17.44) and (17.45). Note that the weight accounts for the mismatch between the likelihood $p(\boldsymbol{y}_n|\cdot)$ at the actual sample and at the predicted point, $\boldsymbol{\mu}_n^{(i)}$. The resulting algorithm is summarized next.

Algorithm 17.6 (Auxiliary particle filtering).

- Initialization: Select a prior distribution, p, to generate the initial state $\boldsymbol{x}_0$.
- Select N.
- **For** $i = 1, 2, \ldots, N$, **Do**
  - Draw $\boldsymbol{x}_0^{(i)} \sim p(\boldsymbol{x})$; Initialize N streams of particles.
  - Set $W_0^{(i)} = \frac{1}{N}$; Set all normalized weights to equal values.
- **End For**
- **For** $n = 1, 2, \ldots,$ **Do**
  - **For** $i = 1, 2, \ldots, N$, **Do**
    - Draw/compute $\boldsymbol{\mu}_n^{(i)}$
    - $Q_i = p\left(\boldsymbol{y}_n|\boldsymbol{\mu}_n^{(i)}\right) W_{n-1}^{(i)}$; This corresponds to $q(i|\boldsymbol{y}_{1:n})$ in Eq. (17.46).
  - **End For**

- **For** $i = 1, 2, \ldots, N$, **Do**
  - Compute normalized Q_i
- **End For**
- **For** $i = 1, 2, \ldots, N$, **Do**
  - $i_{n-1} \sim Q_i$; Eq. (17.47).
  - Draw $\boldsymbol{x}_n^{(i)} \sim p\left(\boldsymbol{x}|\boldsymbol{x}_{n-1}^{(i_{n-1})}\right)$
  - Compute $w_n^{(i)} = \dfrac{p\left(\boldsymbol{y}_n|\boldsymbol{x}_n^{(i)}\right)}{p\left(\boldsymbol{y}_n|\boldsymbol{\mu}_n^{(i)}\right)}$
- **End For**
- **For** $i = 1, 2, \ldots, N$, **Do**
  - Compute normalized $W_n^{(i)}$
- **End For**

- **End For**

Figure 17.9 shows $N = 200$ particles and their respective normalized weights, generated by Algorithm 17.6 for the observation sequence of Example 17.1 and using the same proposal distribution. Observe that compared to the corresponding Figures 17.5 and 17.8, a substantially larger number of particles with significant weights survive.

The previous algorithm is sometimes called the *single-stage* auxiliary particle filter as opposed to the *two-stage* one, which was originally proposed in [34]. The latter involved an extra resampling step to obtain samples with equal weights. It has been experimentally verified that the single-stage version leads to enhanced performance, and it is the one that is widely used. It has been reported that the auxiliary particle filter may lead to enhanced performance compared to Algorithm 17.5, for high signal-to-noise ratios. However, for high-noise terrains its performance degrades (see, e.g., [1]). More results concerning the performance and analysis of the auxiliary filter can be found in, for example, [13, 23, 35].

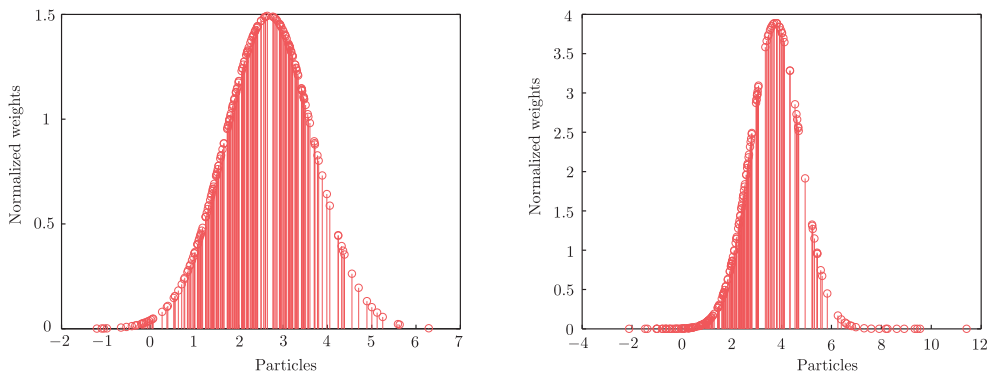

FIGURE 17.9

Plot of $N = 200$ generated particles with the corresponding (normalized) weights, for the same observation sequence as that in Example 17.1, using the auxiliary particle-filtering algorithm, at time instants (a) $n = 3$ and (b) $n = 30$. Compared to Figures 17.5c, d, and Figure 17.8, more particles with significant weights survive.

Remarks 17.3.

- Besides the algorithms presented earlier, a number of variants have been proposed over the years in order to overcome the main limitations of particle filters, associated with the increasing variance and the sample impoverishment problem. In *resample - move* [17] and *block sampling* [14], instead of just sampling for $\boldsymbol{x}_n^{(i)}$ at time instant, n, one also tries to modify past values, over a window $[n-1, n-L+1]$ of fixed size L, in light of the newly arrived observation $\boldsymbol{y}_n$. In the *regularized particle filter* [32], in the resampling stage of Algorithm 17.5, instead of sampling from a discrete distribution, samples are drawn from a smooth approximation,

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}) \simeq \sum_{i=1}^{N} W_n^{(i)} K(\boldsymbol{x}_n - \boldsymbol{x}_n^{(i)}),$$

  where $K(\cdot)$ is a smooth kernel density function. In [26, 27], the posteriors are approximated by Gaussians; as opposed to the more classical extended Kalman filters, the updating and filtering is accomplished via the propagation of particles.
  The interested reader may find more information concerning particle filtering in the tutorial papers [1, 9, 15].
- *Rao - Blackwellization* is a technique used to reduce the variance of estimates that are obtained via Monte Carlo sampling methods (e.g., [4]). To this end, this technique has also been employed in particle filtering of dynamic systems. It turns out that, often in practice, some of the states are conditionally linear given the nonlinear ones. The main idea consists of treating the linear states differently by viewing them as nuisance parameters and *marginalizing* them out of the estimation process. The particles of the nonlinear states are propagated randomly, and then the task is treated linearly via the use of a Kalman filter (see, e.g., [10, 12, 24]).
- *Smoothing* is closely related to filtering processing. In filtering, the goal lies in obtaining estimates of $\boldsymbol{x}_{1:n}$ ($\boldsymbol{x}_n$) based on observations taken in the interval $[1, n]$, that is, on $\boldsymbol{y}_{1:n}$. In smoothing, one obtains estimates of $\boldsymbol{x}_n$ based on an observation set $\boldsymbol{y}_{1:n+k}$, $k > 0$. There are two paths to smoothing. One is known as *fixed lag* smoothing, where k is a fixed lag. The other is known as *fixed interval*, where one is interested in obtaining estimates based on observations taken over an interval $[1, T]$, that is, based on a fixed set of measurements $\boldsymbol{y}_{1:T}$.
  There are different algorithmic approaches to smoothing. The naive one is to run the particle filtering up to time k or T and use the obtained weights for weighting the particles at time n, in order to form the random measure, that is,

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n+k}) \simeq \sum_{i=1}^{N} W_{n+k}^{(i)} \delta(\boldsymbol{x}_n - \boldsymbol{x}_n^{(i)}).$$

  This can be a reasonable approximation for small values of k (or $T-n$). Other, more refined, techniques adopt a two-pass rationale. First, a particle filtering is run, and then a backward set of recursions is used to modify the weights (see, e.g., [10]).
- A summary concerning convergence results related to particle filtering can be found in, for example, [6].
- A survey on applications of particle filtering in signal processing-related tasks is given in [8, 9].

- Following the general trend for developing algorithms for distributed learning, a major research effort has been dedicated in this direction in the context of particle filtering. For a review on such schemes, see, for example, [20].
- One of the main difficulties of the particle-filtering methods is that the number of particles required to approximate the underlying distributions increases exponentially with the state dimension. To overcome this problem, several methods have been proposed. In [30], the authors propose to partition the state and estimate each partition independently. In [16], the annealed particle filter is proposed, which implements a coarse-to-fine strategy by using a series of smoothed weighting functions. The unscented particle filter, [37], proposes to use the unscented transform for each particle to avoid wasting resources in low likelihood regions. In [31], a hierarchical search strategy is proposed that uses auxiliary lower dimension models to guide the search in the higher dimensional one.

Example 17.3. *Stochastic Volatility Model*. Consider the following state-space model for generating the observations

$$\begin{aligned} \mathrm{x}_n &= \alpha \mathrm{x}_{n-1} + \eta_n \\ \mathrm{y}_n &= \beta \mathrm{v}_n \exp\left(\frac{\mathrm{x}_n}{2}\right). \end{aligned}$$

This model belongs to a more general class known as *stochastic volatility* models, where the variance of a process is itself randomly distributed. Such models are used in financial mathematics to model derivative securities, such as options. The state variable is known as the *log-volatility*. We assume the two noise sequences to be i.i.d. and mutually independent Gaussians with zero mean and variances σ_η^2 and σ_v^2, respectively. The model parameters, α and β, are known as the *persistence in volatility shocks* and *modal volatility*, respectively. The adopted values for the parameters are $\sigma_\eta^2 = 0.178$, $\sigma_v^2 = 1$, $\alpha = 0.97$, and $\beta = 0.69$.

The goal of the example is to generate a sequence of observations and then, based on these measurements, to predict the state, which is assumed to be unknown. To this end, we generate a sequence of $N = 2000$ particles, and the state variable at each time instant is estimated as the weighted average of the generated particles, that is

$$\hat{x}_n = \sum_{i=1}^{N} W_n^{(i)} x_n^{(i)}.$$

Both the SIR Algorithm 17.5 and the auxiliary filter method of Algorithm 17.6 were used. The proposal distribution was

$$q(x_n|x_{n-1}) = \mathcal{N}(x_n|\alpha x_{n-1}, \sigma_\eta^2).$$

Figure 17.10 shows the observation sequence together with the obtained estimate. For comparison reasons, the corresponding true-state value is also shown. Both methods for generating particles gave almost identical results, and we only show one of them. Observe how closely the estimates follow the true values.

Example 17.4. *Visual Tracking*. Consider the problem of visual tracking of a circle, which has a constant and known radius. We seek to track its position, that is, the coordinates of its center, $\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2]^T$. This vector will comprise the state variable. The model for generating the observations is given by

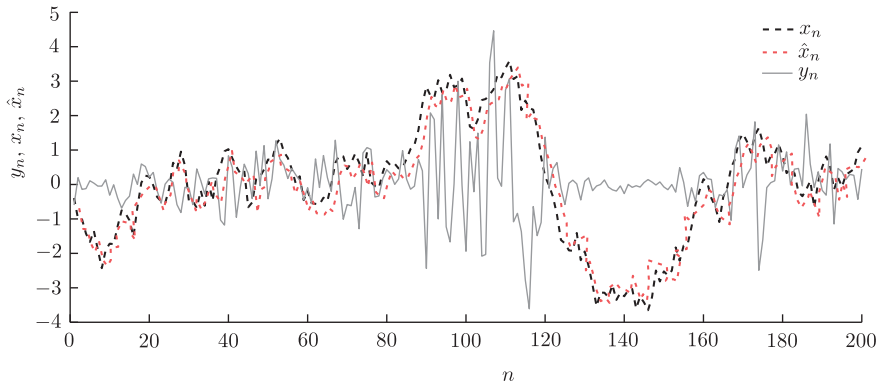

FIGURE 17.10

The observation sequence generated by the volatility model together with the true and estimated values of the state variable.

$$\begin{aligned} \mathbf{x}_n &= \mathbf{x}_{n-1} + \boldsymbol{\eta}_n, \\ \mathbf{y}_n &= \mathbf{x}_n + \mathbf{v}_n, \end{aligned} \tag{17.49}$$

where $\boldsymbol{\eta}_n$ is a uniform noise in the interval $[-10, 10]$ pixels, for each dimension. Note that, due to the uniform nature of the noise, Kalman filtering, in its standard formulation, is no longer the optimal choice, in spite of the linearity of the model. The noise $\mathbf{v}_n$ follows a Gaussian pdf $\mathcal{N}(\mathbf{0}, \Sigma_{\mathrm{v}})$, where

$$\Sigma_{\mathrm{v}} = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix}.$$

Initially, the target circle is located in the image center. The particle filter employs N=50 particles and the SIS sampling method was used, (see, also, MATLAB Exercise 17.12)

Figure 17.11 shows the circle and the generated particles, which attempt to track the center of the circle from the noisy observations, for different time instants. Observe how closely the particles track the center of the circle as it moves around. A related video is available from the companion site of this book.

PROBLEMS

17.1 Let

$$\mu := \mathbb{E}[f(\mathbf{x})] = \int f(\boldsymbol{x}) p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$$

and $q(\boldsymbol{x})$ be the proposal distribution. Show that if

$$w(\boldsymbol{x}) := \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})},$$

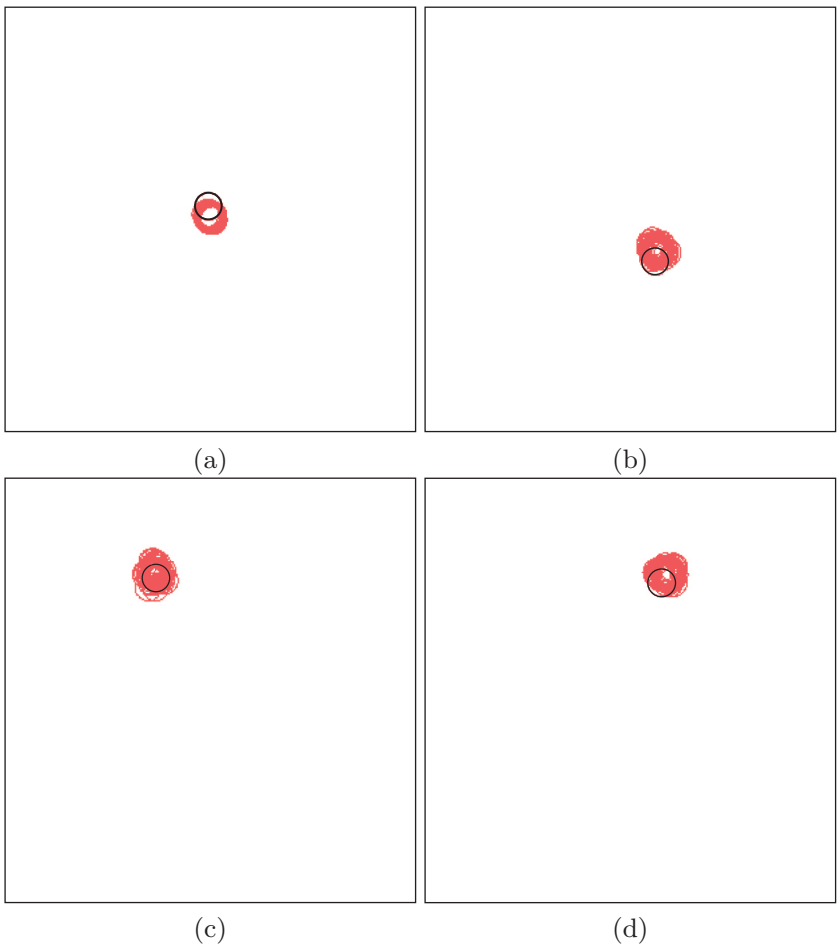

FIGURE 17.11

The circle (in gray) and the generated particles (in red) for time instants $n = 1$, $n = 30$, $n = 60$, and $n = 120$.

and

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_i) f(\boldsymbol{x}_i),$$

then the variance

$$\sigma_f^2 = \mathbb{E}\left[\left(\hat{\mu} - \mathbb{E}\left[\hat{\mu}\right]\right)^2\right] = \frac{1}{N}\left(\int \frac{f^2(\boldsymbol{x})p^2(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x} - \mu^2\right).$$

Observe that if $f^2(\boldsymbol{x})p^2(\boldsymbol{x})$ goes to zero slower than $q(\boldsymbol{x})$, then for fixed N, $\sigma_f^2 \longrightarrow \infty$.

17.2 In importance sampling, with weights defined as

$$w(\boldsymbol{x}) = \frac{\phi(\boldsymbol{x})}{q(\boldsymbol{x})},$$

where

$$p(\boldsymbol{x}) = \frac{1}{Z}\phi(\boldsymbol{x}),$$

we know from Problem 14.6 that the estimate

$$\hat{Z} = \frac{1}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_i)$$

defines an unbiased estimator of the normalizing constant, Z. Show that the respective variance is given by

$$\text{var}[\hat{Z}] = \frac{Z^2}{N}\left(\int \frac{p^2(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x} - 1\right).$$

17.3 Show that using resampling in importance sampling, then as the number of particles tends to infinity, the approximating, by the respective discrete random measure, distribution, $\bar{p}$, tends to the true (desired) one, p.
Hint: Consider the one-dimensional case.

17.4 Show that in sequential importance sampling, the proposal distribution that minimizes the variance of the weight at time n, conditioned on $\boldsymbol{x}_{1:n-1}$, is given by

$$q_n^{opt}(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}) = p_n(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}).$$

17.5 In a sequential importance sampling task, let

$$p_n(\boldsymbol{x}_{1:n}) = \prod_{k=1}^{n} \mathcal{N}(x_k|0,1)$$

$$\phi_n(\boldsymbol{x}_{1:n}) = \prod_{k=1}^{n} \exp\left(-\frac{x_k^2}{2}\right),$$

and let the proposal distribution be

$$q_n(\boldsymbol{x}_{1:n}) = \prod_{k=1}^{n} \mathcal{N}(x_k|0,\sigma^2).$$

Let the estimate of $Z_n = (2\pi)^{\frac{n}{2}}$, be

$$\hat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w(\boldsymbol{x}_{1:n}^{(i)}).$$

Show that the variance of the estimator is given by

$$\text{var}[\hat{Z}_n] = \frac{Z_n^2}{N}\left(\left(\frac{\sigma^4}{2\sigma^2 - 1}\right)^{\frac{n}{2}} - 1\right).$$

Observe that for $\sigma^2 > 1/2$, which is the range of values for the above formula makes sense and guarantees a finite value for the variance, the variance exhibits an exponential increase with respect to n. To keep the variance small, one has to make N very large, that is, to generate a very large number of particles [15].

17.6 Prove that the use of the optimal proposal distribution in particle filtering leads to

$$w_n(\boldsymbol{x}_{1:n}) = w_{n-1}(\boldsymbol{x}_{1:n-1})p(\boldsymbol{y}_n|\boldsymbol{x}_{n-1}).$$

MATLAB Exercises

17.7 For the state-space model of Example 17.1, implement the generic particle filtering algorithm for different numbers of particle streams N and different thresholds of effective particle sizes N_{eff}.

Hint. Start by selecting a distribution (the normal should be a good start) and initialize. Then, update the particles in each step according to the algorithm. Finally, check whether the N_{eff} is lower than the threshold and if it is, continue with the resampling process.

17.8 For the same example as before, implement the SIS particle filtering algorithm and plot the resulting particles together with the normalized weights for various time instances n. Observe the degeneracy phenomenon of the weights as time evolves.

17.9 For Example 17.1, implement the SIR particle filtering algorithm for different numbers of particle streams N and for various time instances n. Use $N_T = N/2$. Compare the performance of SIR and SIS algorithms.

17.10 Repeat the previous exercise, implement the auxiliary particle filtering (APF) algorithm and compare the particle-weight histogram with the ones obtained from SIS and SIR algorithms. Observe that the number of particles with significant weights that survive is substantially larger.

17.11 Reproduce Figure 17.10 for the stochastic volatility model of Example 17.3 and observe how the estimated sequence $\hat{x}_n$ follows the true sequence x_n based on the observations y_n.

17.12 Develop the MATLAB code to reproduce the visual tracking of the circle of Example 17.4. Because, at each time instant, we are only interested in the $\boldsymbol{x}_n$ and not on the whole sequence, modify the SIS sampling in Algorithm 17.4 to care for this case.

Specifically, given Eqs. (17.28)–(17.29), then in order to estimate $\mathbf{x}_n$ instead of $\mathbf{x}_{1:n}$, Eq. (17.31) is simplified to

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}) = \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})}{p(\boldsymbol{y}_n|\boldsymbol{y}_{1:n-1})}, \tag{17.50}$$

where

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1}) = \int_{\boldsymbol{x}_{n-1}} p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{x}_{n-1}|\boldsymbol{y}_{1:n-1})\,\mathrm{d}\boldsymbol{x}_{n-1}. \tag{17.51}$$

The samples are now weighted as

$$w_n^{(i)} = \frac{p(\boldsymbol{x}_n^{(i)}|\boldsymbol{y}_{1:n})}{q(\boldsymbol{x}_n^{(i)}|\boldsymbol{y}_{1:n})}, \tag{17.52}$$

and a popular selection for the proposal distribution is

$$q(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}) \equiv p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1}). \tag{17.53}$$

Substituting Eqs. (17.50) and (17.53) into Eq. (17.52), we get the following rule for the weights:

$$w_n^{(i)} \propto p(\boldsymbol{y}_n|\boldsymbol{x}_n^{(i)}). \tag{17.54}$$

REFERENCES

[1] M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Process. 50 (2) (2002) 174-188.

[2] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, New York, 2006.

[3] J. Carpenter, P. Clifford, P. Fearnhead, Improved particle filter for nonlinear problems, in: Proceedings IEE, Radar, Sonar and Navigation, vol. 146, 1999, pp. 2-7.

[4] G. Casella, C.P. Robert, Rao-Blackwellisation of sampling schemes, Biometrika 83 (1) (1996) 81-94.

[5] N. Chopin, Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference, Ann. Stat. 32 (2004).

[6] D. Crisan, A. Doucet, A survey of convergence results on particle filtering methods for practitioners, IEEE Trans. Signal Process. 50 (3) (2002) 736-746.

[7] P. Del Moral, Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications, Springer-Verlag, New York, 2004.

[8] P.M. Djuric, Y. Huang, T. Ghirmai, Perfect sampling: a review and applications to signal processing, IEEE Trans. Signal Process. 50 (2002) 345-356.

[9] P.M. Djuric, J.H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M.F. Bugallo, J. Miguez, Particle filtering, IEEE Signal Process. Mag. 20 (2003) 19-38.

[10] P.M. Djuric, M. Bugallo, Particle filtering, in: T. Adali, S. Haykin (Eds.), Adaptive Signal Processing: Next Generation Solutions, John Wiley & Sons, Inc., New York, 2010.

[11] A. Doucet, S. Godsill, C. Andrieu, On sequential Monte Carlo sampling methods for Bayesian filtering, Stat Comput 10 (2000) 197-208.

[12] R. Douc, O. Cappe, E. Moulines, Comparison of resampling schemes for particle filtering, in: 4th International Symposium on Image and Signal Processing and Analysis (ISPA), 2005.

[13] R. Douc, E. Moulines, J. Olsson, On the auxiliary particle filter, 2010, arXiv:0709.3448v1 [math.ST].

[14] A. Doucet, M. Briers, S. Sénécal, Efficient block sampling strategies for sequential Monte Carlo methods, J. Comput. Graph. Stat. 15 (2006) 693-711.

[15] A. Doucet, A.M. Johansen, A tutorial on particle filtering and smoothing: Fifteen years later, in: Handbook of Nonlinear Filtering, Oxford University Press, Oxford, 2011.

[16] J. Deutscher, A. Blake, I. Reid, Articulated body motion capture by annealed particle filtering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2000, pp. 126-133.

[17] W.R. Gilks, C. Berzuini, Following a moving target—Monte Carlo inference for dynamic Bayesian models, J. R. Stat. Soc. B 63 (2001) 127-146.

[18] N.J. Gordon, D.J. Salmond, A.F.M. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, Proc IEEE F 140 (2) (1993) 107-113.

[19] J.M. Hammersley, K.W. Morton, Poor man's Monte Carlo, J. R. Stat. Soc. B 16 (1) (1954) 23-38.

[20] O. Hinka, F. Hlawatz, P.M. Djuric, Distributed particle filtering in agent networks, IEEE Signal Process. Mag. 30 (1) (2013) 61-81.

[21] S. Hong, S.S. Chin, P.M. Djurić, M. Bolić, Design and implementation of flexible resampling mechanism for high-speed parallel particle filters, J. VLSI Signal Process. 44 (1-2) (2006) 47-62.

[22] Y. Huang, P.M. Djurić, A blind particle filtering detector of signals transmitted over flat fading channels, IEEE Trans. Signal Process. 52 (7) (2004) 1891-1900.

[23] A.M. Johansen, A. Doucet, A note on auxiliary particle filters, Stat. Probab. Lett. 78 (12) (2008) 1498-1504.
[24] R. Karlsson, F. Gustafsson, Complexity analysis of the marginalized particle filter, IEEE Trans. Signal Process. 53 (11) (2005) 4408-4411.
[25] G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, J. Comput. Graph. Stat. 5 (1996) 1-25.
[26] J.H. Kotecha, P.M. Djurić, Gaussian particle filtering, IEEE Trans. Signal Process. 51 (2003) 2592-2601.
[27] J.H. Kotecha, P.M. Djurić, Gaussian sum particle filtering, IEEE Trans. Signal Process. 51 (2003) 2602-2612.
[28] J.S. Liu, R. Chen, Sequential Monte Carlo methods for dynamical systems, J. Am. Stat. Assoc. 93 (1998) 1032-1044.
[29] J.S. Liu, Monte Carlo Strategies in Scientific Computing, Springer, New York, 2001.
[30] J. MacCormick, M. Isard, Partitioned sampling, articulated objects, and interface-quality hand tracking, in: Proceedings of the 6th European Conference on Computer Vision, Part II, ECCV, Springer-Verlag, London, UK, 2000, pp. 3-19.
[31] A. Makris, D. Kosmopoulos, S. Perantonis, S. Theodoridis, A hierarchical feature fusion framework for adaptive visual tracking, Image Vis. Comput. 29 (9) (2011) 594-606.
[32] C. Musso, N. Oudjane, F. Le Gland, Improving regularised particle filters, in: A. Doucet, N. de Freitas, N.J. Gordon (Eds.), Sequential Monte Carlo Methods in Practice, Springer-Verlag, New York, 2001.
[33] A. Owen, Y. Zhou, Safe and effective importance sampling, J. Am. Stat. Assoc. 95 (2000) 135-143.
[34] M.K. Pitt, N. Shephard, Filtering via simulation: auxiliary particle filters, J. Am. Stat. Assoc. 94 (1999) 590-599.
[35] M.K. Pitt, R.S. Silva, P. Giordani, R. Kohn, Auxiliary particle filtering within adaptive Metropolis-Hastings sampling, 2010, http://arxiv.org/abs/1006.1914[stat.Me].
[36] M.N. Rosenbluth, A.W. Rosenbluth, Monte Carlo calculation of the average extension of molecular chains, J. Chem. Phys. 23 (2) (1956) 356-359.
[37] R. van der Merwe, N. de Freitas, A. Doucet, E. Wan, The unscented particle filter, in: Proceedings Advances in Neural Information Processing Systems (NIPS), 2000.