5

PARTICLE FILTERING

Petar M. Djurić and Mónica F. Bugallo
Stony Brook University, Stony Brook, NY

Many problems in adaptive filtering are nonlinear and non-Gaussian. Of the many methods that have been proposed in the literature for solving such problems, particle filtering has become one of the most popular. In this chapter we provide the basics of particle filtering and review its most important implementations. In Section 5.1 we give a brief introduction to the area, and in Section 5.2 we motivate the use of particle filtering by examples from several disciplines in science and engineering. We then proceed with an introduction of the underlying idea of particle filtering and present a detailed explanation of its essential steps (Section 5.3). In Section 5.4 we address two important issues of particle filtering. Subsequently, in Section 5.5 we focus on some implementations of particle filtering and compare their performance on synthesized data. We continue by explaining the problem of estimating constant parameters with particle filtering (Section 5.6). This topic deserves special attention because constant parameters lack dynamics, which for particle filtering creates some serious problems. We can improve the accuracy of particle filtering by a method known as Rao–Blackwellization. It is basically a combination of Kalman and particle filtering, and is discussed in Section 5.7. Prediction and smoothing with particle filtering are described in Sections 5.8 and 5.9, respectively. In the last two sections of the chapter, we discuss the problems of convergence of particle filters (Section 5.10) and computational issues and hardware implementation of the filters (Section 5.11). At the end of the chapter we present a collection of exercises which should help the reader in getting additional insights into particle filtering.

Adaptive Signal Processing: Next Generation Solutions. Edited by Tülay Adalı and Simon Haykin

5.1 INTRODUCTION

Some of the most important problems in signal processing require the sequential processing of data. The data describe a system that is mathematically represented by equations that model its evolution with time, where the evolution contains a random component. The system is defined by its state variables of which some are dynamic and some are constant. A typical assumption about the state of the system is that it is Markovian, which means that given the state at a previous time instant, the distribution of the state at the current time instant does not depend on any other older state.

The state is not directly observable. Instead, we acquire measurements which are functions of the current state of the system. The measurements are degraded by noise or some other random perturbation, and they are obtained sequentially in time. The main objective of sequential signal processing is the recursive estimation of the state of the system based on the available measurements. The methods that are developed for estimation of the state are usually called filters. We may also be interested in estimates of the state in the past or would like to predict its values at future time instants.

In general, we can categorize the studied systems as linear and nonlinear and as Gaussian and non-Gaussian. For any of them, the complete information about the state is in the probability density functions (PDFs) of the state. These PDFs can be grouped as filtering, predictive, and smoothing densities (for their precise definitions, see the next section). If a system is linear and Gaussian, we can obtain all the relevant densities exactly, that is, without approximations. A filter that produces exact solutions is known as the Kalman filter (KF), and it obtains them by using analytical expressions [3, 43, 57].

When the description of the system deviates from linearity and Gaussianity, the solutions often represent approximations and therefore are not optimal. In fact, for some systems, such solutions can become poor to the point of becoming misleading. The traditional filter that deals with nonlinear systems is the extended Kalman filter (EKF) [3, 35, 40, 57]. The EKF implements the approximation by linearization of the system with a Taylor series expansion around the latest estimate of the state followed by application of the Kalman's linear recursive algorithm.

The unscented Kalman filter (UKF) [42] and the Gaussian quadrature Kalman filter (QKF) [39] form a group of methods that also assume Gaussian PDFs in the system but do not require the evaluation and computation of Jacobian matrices. Instead, they implement the necessary integration for filtering with different methods. The UKF uses the unscented transform, which represents a Gaussian posterior of the state with a set of deterministically chosen samples that capture the first two moments of the Gaussian. These samples are propagated through the system, and the obtained values contain information about the first two moments of the new Gaussian. The QKF exploits the Gauss–Hermite quadrature integration rule [29, 59].

For problems where the Gaussian assumption about the posterior is invalid, one can use the Gaussian sum filter. This filter approximates the *a posteriori* PDF by a sum of Gaussians, where the update of each Gaussian can be carried out by a separate EKF [2, 70], UKF [41], or QKF [5].

There is another group of approaches for nonlinear/non-Gaussian filtering, which is based on a different paradigm. These methods exploit the idea of evaluating the required PDFs over deterministic grids [14, 46, 49, 71]. Namely, the state space is represented by a fixed set of nodes and values associated with them, which represent the computed PDFs at the nodes. The final estimates of the densities are obtained by piecewise linear (first-order spline) interpolations. These methods can work well for low-dimensional state spaces; otherwise, they become quickly computationally prohibitive.

The particle filtering methodology[1] is very similar in spirit as the one that exploits deterministic grids. However, there is a subtle but very important difference between them. Particle filtering uses random grids, which means that the location of the nodes vary with time in a random way. In other words, at one time instant the grid is composed of one set of nodes, and at the next time instant, this set of nodes is completely different. The nodes are called particles, and they have assigned weights, which can be interpreted as probability masses. The particles and the weights form a discrete random measure,[2] which is used to approximate the densities of interest. In the generation of particles, each particle has a "parent", and the parent its own parent and so on. Such sequence of particles is called a particle stream, and it represents one possible evolution of the state with time. Filters based on this methodology are called particle filters (PFs).

A main challenge in implementing PFs is to place the nodes of the grids (i.e., generate particles) in regions of the state space over which the densities carry significant probability masses and to avoid placing nodes in parts of the state space with negligible probability masses. To that end, one exploits concepts from statistics known as importance sampling [48] and sampling-importance-resampling [68], which are explained in the sequel. For the computation of the particle weights, one follows the Bayesian theory. The sequential processing amounts to recursive updating of the discrete random measure with the arrival of new observations, where the updates correspond to the generation of new particles and computation of their weights. Thus, one can view particle filtering as a method that approximates evolving densities by generating streams of particles and assigning weights to them.

The roots of particle filtering were established about 50 years ago with the methods for estimating the mean squared extension of a self-avoiding random walk on lattice spaces [33, 67]. One of the first applications of the methodology was on the simulation of chain polymers. The control community produced interesting work on sequential Monte Carlo integration methods in the 1960s and 1970s, for example [1, 34]. The popularity of particle filtering in the last 15 years was triggered by [31], where the sampling-importance resampling filter was presented and applied to tracking problems. The timing of [31] was perfect because it came during a period when computing power started to become widely available. Ever since, the amount of work on particle filtering has proliferated and many important advances have been made.

[1] In the literature, particle filtering is also known as sequential Monte Carlo methodology [26].
[2] The random measure is basically a probability mass function.

One driving force for the advancement of particle filtering is the ever increasing range of its applications. We list some of them in the next section. Here we only reiterate the advantage of PFs over other filters and why they have become popular. PFs can be applied to any state space model where the likelihood and the prior are computable up to proportionality constants. The accuracy of the method depends on how well we generate the particles and how many particles we use to represent the random measure. From a mathematical point of view, it is important to have some understanding of the theoretical properties of PFs, and in particular the convergence of their estimates to the true states of the system. Some important results on this theory will also be presented in this chapter. At last, we have to keep in mind the practical issues related to PFs. Namely, it is well known that particle filtering is computationally intensive, which is an issue because in sequential signal processing any algorithm has to complete the processing of the latest measurement by a certain deadline. However, particle filtering allows for significant parallelization of its operations, which may speed up its time of execution by orders of magnitude. We also address some basic computational issues on this subject in the chapter.

5.2 MOTIVATION FOR USE OF PARTICLE FILTERING

In a typical setup, the observed data are modeled by

$$\boldsymbol{y}(n) = g_2(\boldsymbol{x}(n), \boldsymbol{v}_2(n)) \tag{5.1}$$

where $n = 1, 2, \ldots$ is a time index, $\boldsymbol{y}(n)$ is a vector of observations, $\boldsymbol{x}(n)$ is the state (or signal) that needs to be estimated, $\boldsymbol{v}_2(n)$ is an observation noise vector, and $g_2(\cdot)$ is a known function, which in general may change with time. We assume that all of the vectors in the above equation conform properly with their dimensions.

It is assumed that the state $\boldsymbol{x}(n)$ varies according to

$$\boldsymbol{x}(n) = g_1(\boldsymbol{x}(n-1), \boldsymbol{v}_1(n)) \tag{5.2}$$

where $\boldsymbol{v}_1(n)$ is a state noise vector, and $g_1(\cdot)$ is a known function (which also might vary with time). The expression (5.1) is known as an observation equation and (5.2) as a state equation, and the two are referred to as the state–space model. The objective is to estimate the unobserved signal $\boldsymbol{x}(n)$ from the observations $\boldsymbol{y}(n)$.

■ EXAMPLE 5.1

Consider the tracking of an object based on bearings-only measurements. The object moves in a two-dimensional sensor field according to [32]

$$\boldsymbol{x}(n) = \boldsymbol{A}\boldsymbol{x}(n-1) + \boldsymbol{B}\boldsymbol{v}_1(n) \tag{5.3}$$

where the state vector $\boldsymbol{x}(n)$ is defined by the position and velocity of the object, or

$$\boldsymbol{x}(n) = [x_1(n) \quad x_2(n) \quad \dot{x}_1(n) \quad \dot{x}_2(n)]^\top$$

with $x_1(n)$ and $x_2(n)$ being the coordinates of the object at time instant n and $\dot{x}_1(n)$ and $\dot{x}_2(n)$ the components of the target velocity. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are defined by

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{B} = \begin{pmatrix} \frac{T_s^2}{2} & 0 \\ 0 & \frac{T_s^2}{2} \\ T_s & 0 \\ 0 & T_s \end{pmatrix}$$

where T_s is the sampling period. The state noise $v_1(n)$ models small accelerations of the object and is assumed to have a known PDF.

The bearings-only range measurements are obtained by J sensors placed at known locations in the sensor field. The sensors measure the angle of the target with respect to a reference axis. At time instant n, the j-th sensor, which is located at $(l_{1,j}, l_{2,j})$, gets the measurement

$$y_j(n) = \arctan\left(\frac{x_2(n) - l_{2,j}}{x_1(n) - l_{1,j}}\right) + v_{2,j}(n) \tag{5.4}$$

where the noise samples $v_{2,j}(n)$ have a known joint PDF, and are independent from the state noise $\boldsymbol{v}_1(n)$.

Given the measurements of J sensors, $\boldsymbol{y}(n) = [y_1(n) \;\; y_2(n) \;\; \cdots \;\; y_J(n)]^\top$, and the movement model of the object defined by (5.3), we want to track the object in time, that is, estimate its position and velocity. In the literature this problem is known as target tracking [66].

In this model the state equation is (5.3), and it is clear that it is represented by a linear function. The observation equation is given by the set of expressions (5.4), and they are all nonlinear. So, in this example, optimal filtering cannot be achieved even though the noises in the system are Gaussian. Instead, one has to resort to sub-optimal methods like extended Kalman filtering or particle filtering.

An alternative characterization of the state–space system can be given in terms of PDFs

$$f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}(n-1)) \tag{5.5}$$

$$f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}(n)) \tag{5.6}$$

where obviously (5.5) is derived from (5.2), and (5.6) is obtained from (5.1). The forms of the PDFs in (5.5) and (5.6) depend on the functions $g_1(\cdot)$ and $g_2(\cdot)$ as well as on the PDFs of $\boldsymbol{v}_1(n)$ and $\boldsymbol{v}_2(n)$.

The problem of estimation can have various twists. Here, we distinguish three different problems, which we refer to as filtering, prediction, and smoothing. In filtering, the goal is to obtain the a *posteriori* PDF of $\boldsymbol{x}(n)$ given all of the measurements from time instant one to n, which we express by $\mathbf{y}(1:n)$. This density is accordingly called filtering density and is denoted by $f(\boldsymbol{x}(n) \,|\, \mathbf{y}(1:n))$, $n \geq 1$. All the information

about $\boldsymbol{x}(n)$ is in $f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n))$, and for example, if one wants to find point estimates of $\boldsymbol{x}(n)$, such as the minimum mean square error (MMSE) estimate or the maximum a *posteriori* (MAP) estimate, one can obtain them from the filtering density.

In prediction, the goal is to find the predictive PDF $f(\boldsymbol{x}(n+k) \mid \boldsymbol{y}(1:n))$, where $k > 0$, $n > 0$. Again, all the information about a future value of the state given the measurements $\boldsymbol{y}(1:n)$ is in the predictive PDF and various point estimates can be obtained from it.

Finally, the problem of smoothing amounts to obtaining $f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:N))$, where $0 \leq n < N$, and where the density is called the smoothing PDF.[3] We can expect that with smoothing, in general, we get more accurate estimates of $\boldsymbol{x}(n)$. It is also clear that with smoothing, we have a delay in processing of the data. For example, we first acquire N measurements and then obtain the smoothing densities. We distinguish several types of smoothing problems, and we describe them later in the chapter.

The objective of sequential signal processing in the context presented here consists of tracking the PDFs of interest by exploiting recursive relationships, that is,

- $f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n))$ from $f(\boldsymbol{x}(n-1) \mid \boldsymbol{y}(1:n-1))$ for the filtering problem, where $n \geq 1$;[4]
- $f(\boldsymbol{x}(n+k) \mid \boldsymbol{y}(1:n))$ from $f(\boldsymbol{x}(n+k-1) \mid \boldsymbol{y}(1:n))$ for the prediction problem where $k > 0$, $n \geq 0$; and
- $f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:N))$ from $f(\boldsymbol{x}(n+1) \mid \boldsymbol{y}(1:N))$ for the smoothing problem, where $0 \leq n < N$.

We reiterate that the complete information about the unknown values is in the respective densities of the unknowns. Many sequential methods provide only point estimates of these unknowns accompanied possibly with another metric that shows how variable the estimates are. By contrast, particle filtering has a much more ambitious aim than yielding point estimates. Its objective is to track in time the approximations of all the desired densities of the unknowns in the system. It is well known that when these densities are not Gaussian, there are not many available methods that can reach this goal. Thus, the motivation for using particle filtering is in its ability to estimate sequentially the densities of unknowns of non-Gaussian and/or nonlinear systems.

What are the forms of the recursions for filtering, prediction, and smoothing? Here we briefly explain only the recursion for obtaining the filtering PDF. The other two recursions are discussed in Sections 5.8 and 5.9. Suppose that at time $n-1$, we know the observations $\boldsymbol{y}(1:n-1)$ and the *a posteriori* PDF $f(\boldsymbol{x}(n-1) \mid \boldsymbol{y}(1:n-1))$. Once $\boldsymbol{y}(n)$ becomes available, we would like to update $f(\boldsymbol{x}(n-1) \mid \boldsymbol{y}(1:n-1))$ and modify it to $f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n))$. To achieve this, we formally write

$$f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n)) \propto f(\boldsymbol{y}(n) \mid \boldsymbol{x}(n)) f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n-1)) \tag{5.7}$$

[3] In the expression for the smoothing PDF, we have included the zero as a possible value of the time index, that is, $n = 0$. The PDF of $\boldsymbol{x}(0)$, $f(\boldsymbol{x}(0))$, is the *a priori* PDF of the state before the first measurement is taken.

[4] When $n = 1$, $f(\boldsymbol{x}(n-1) \mid \boldsymbol{y}(1:n-1))$ is simply the *a priori* PDF of $\boldsymbol{x}(0)$, that is, $f(\boldsymbol{x}(0))$.

where $\propto$ signifies proportionality. The first factor on the right of the proportionality sign is the likelihood function of the unknown state, and the second factor is the predictive density of the state. For the predictive density we have

$$f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n-1)) = \int f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}(n-1))\, f(\boldsymbol{x}(n-1) \,|\, \boldsymbol{y}(1:n-1))\, d\boldsymbol{x}(n-1). \quad (5.8)$$

In writing (5.8) we used the property of the state that given $\boldsymbol{x}(n-1)$, $\boldsymbol{x}(n)$ does not depend on $\boldsymbol{y}(1:n-1)$. Now, the required recursive equation for the update of the filtering density is obtained readily by combining (5.7) and (5.8), that is, we formally have

$$\begin{aligned} f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n)) &\propto f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}(n)) \\ &\quad \times \int f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}(n-1))\, f(\boldsymbol{x}(n-1) \,|\, \boldsymbol{y}(1:n-1))\, d\boldsymbol{x}(n-1). \end{aligned}$$

Thus, on the left of the proportionality sign we have the filtering PDF at time instant n, and on the right under the integral, we see the filtering PDF at time instant $n-1$.

There are at least two problems in carrying out the above recursion, and they may make the recursive estimation of the filtering density very challenging. The first one is the solving of the integral in (5.8) and obtaining the predictive density $f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n-1))$. In some cases it is possible to obtain the solution analytically, which considerably simplifies the recursive algorithm and makes it more accurate. The second problem is the combining of the likelihood and the predictive density in order to get the updated filtering density. These problems may mean that it is impossible to express the filtering PDF in a recursive form. For instance, in Example 5.1, we cannot obtain an analytical solution due to the nonlinearities in the observation equation. It will be seen in the sequel that the smoothing and the predictive densities suffer from analogous problems.

We can safely state that in many problems the recursive evaluation of the densities of the state–space model cannot be done analytically, and consequently we have to resort to numerical methods. As already discussed, an important class of systems which allows for exact analytical recursions is the one represented by linear state–space models with Gaussian noises. These recursions are known as Kalman filtering [3]. When analytical solutions cannot be obtained, particle filtering can be employed with elegance and with performance characterized by high accuracy.

Particle filtering has been used in many different disciplines. They include surveillance guidance, and obstacle avoidance systems, robotics, communications, speech processing, seismic signal processing, system engineering, computer vision, and econometrics. During the past decade, in practically all of these fields, the number of contributions has simply exploded. To give a perspective of how the subject has been vibrant with activities, we provide a few examples of developments in target tracking, positioning, and navigation.

In target tracking, there have been a large number of papers published that show the advantage of PFs over other filters in a variety of scenarios. For example, in [19], multiple targets were tracked by PFs using a combination of video and acoustic sensors while in [38], in a sonar application, the tracking is based on bearings-only measurements (as in Example 5.1). The tracking of the direction-of-arrival (DOA) of multiple moving targets using measurements of a passive sensor array was shown in [63]. PFs can also be used for simultaneous detection and tracking of multiple targets [62]. In [75], a moving acoustic source was tracked in a moderately reverberant room. A particle filtering-based method for the joint tracking of location and speaking activity of several speakers in a meeting room with a microphone array and multiple cameras was proposed in [64]. Tracking of multiaspect targets using image sequences with background clutter and where the target aspect changed were addressed in [13]. Mobility tracking in wireless communication networks based on received signal strength was explored in [60].

Positioning and navigation are related problems to target tracking [32]. In the former problem, an object estimates its own position, and in the latter, the object estimates more unknowns such as its velocity, attitude and heading, acceleration, and angular rates. For example, in [30] PFs were applied to navigation that uses a global positioning system whose measurements are distorted with multipath effects. In [45], PFs were used for maritime surface and underwater map-aided navigation. The surface navigation employed a radar sensor and a digital sea chart whereas the underwater navigation was based on a sonar sensor and a depth database. Land vehicle positioning by using synchronous and asynchronous sensor measurements was presented in [17]. Applications of positioning, navigation and tracking can, *inter alia*, be used for car collision avoidance in the car industry [32].

The above list is by no means exhaustive. The other areas where PFs are of interest are also rich with many interesting results and contributions. For more references on particle filtering applications, the reader should consult the review papers [6, 16, 23, 25] and the books [15, 26, 66].

5.3 THE BASIC IDEA

Consider the state of the system at time instant n, that is, $\boldsymbol{x}(n)$. Under the particle filtering framework, the *a posteriori* PDF of $\boldsymbol{x}(n)$, $f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))$, is approximated by a discrete random measure composed of M particles and weights

$$\chi(n) = \{\boldsymbol{x}^{(m)}(n),\, w^{(m)}(n)\}_{m=1}^{M} \tag{5.9}$$

where $\boldsymbol{x}^{(m)}(n)$ and $w^{(m)}(n)$ represent the m-th particle and weight, respectively. The particles are Monte Carlo samples of the system state, and the weights are nonnegative values that sum up to one and can be interpreted as probabilities of the particles. The previous measure allows for approximation of $f(\boldsymbol{x}(n) \,|\, \mathbf{y}(1:n))$ by

$$f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n)) \approx \sum_{m=1}^{M} w^{(m)}(n)\delta(\boldsymbol{x}(n) - \boldsymbol{x}^{(m)}(n)) \tag{5.10}$$

where $\delta(\cdot)$ denotes the Dirac delta function. With this approximation, computations of expectations of functions of the random process $\boldsymbol{X}(n)$ simplify to summations, that is,

$$E(h(\boldsymbol{X}(n))) = \int h(\boldsymbol{x}(n))\, f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))\, d\boldsymbol{x}(n)$$

$$\Downarrow$$

$$E(h(\boldsymbol{X}(n))) \approx \sum_{m=1}^{M} w^{(m)}(n) h(\boldsymbol{x}^{(m)}(n))$$

where $E(\cdot)$ denotes expectation, and $h(\cdot)$ is an arbitrary function of $\boldsymbol{X}(n)$.

■ **EXAMPLE 5.2**

Assume that independent particles, $\boldsymbol{x}^{(m)}(n)$, can be drawn from $f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))$. In that case, all the particles have the same weights, $w^{(m)}(n) = 1/M$, and

$$E(h(\boldsymbol{X}(n))) = \int h(\boldsymbol{x}(n))\, f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))\, d\boldsymbol{x}(n)$$

$$\Downarrow$$

$$\hat{E}(h(\boldsymbol{X}(n))) = \frac{1}{M} \sum_{m=1}^{M} h(\boldsymbol{x}^{(m)}(n)) \tag{5.11}$$

where $\hat{E}(\cdot)$ is an unbiased estimator of the conditional expectation $E(\cdot)$. Note that in this case we intrinsically approximate $f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))$ by the random measure

$$\chi(n) = \{\boldsymbol{x}^{(m)}(n),\, w^{(m)}(n) = 1/M\}_{m=1}^{M}.$$

If the variance $\sigma_h^2 < \infty$,[5] then the variance of $\hat{E}(\cdot)$ is given by

$$\sigma^2_{\hat{E}(h(\cdot))} = \frac{\sigma_h^2}{M}. \tag{5.12}$$

As $M \to \infty$, from the strong law of large numbers, we get that $\hat{E}(h(\boldsymbol{X}(n)))$ converges to $E(h(\boldsymbol{X}(n)))$ almost surely [27], that is

$$\hat{E}(h(\boldsymbol{X}(n))) \xrightarrow{a.s.} E(h(\boldsymbol{X}(n))) \tag{5.13}$$

and from the central limit theorem, we obtain that $\hat{E}(h(\boldsymbol{X}(n)))$ converges in distribution to a Gaussian distribution [27]

$$\hat{E}(h(\boldsymbol{X}(n))) \xrightarrow{d} \mathcal{N}\left(E(h(\boldsymbol{X}(n)),\, \frac{\sigma_h^2}{M}\right). \tag{5.14}$$

[5]The notation σ_h^2 symbolizes the variance of $h(\boldsymbol{X}(n))$.

The example shows that if we sample from $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))$ a large number of particles M, we will be able to estimate $E(h(\boldsymbol{X}(n))$ with arbitrary accuracy. In practice, however, the problem is that we often cannot draw samples directly from the *a posteriori* PDF $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))$. An attractive alternative is to use the concept of importance sampling [58]. The idea behind it is based on the use of another function for drawing particles. This function is called importance sampling function or proposal distribution, and we denote it by $\pi(\boldsymbol{x}(n))$.

When the particles are drawn from $\pi(\boldsymbol{x}(n))$, the estimate of $E(h(\boldsymbol{X}(n)))$ in (5.11) can be obtained either by

$$E(h(\boldsymbol{X}(n))) \approx \frac{1}{M}\sum_{m=1}^{M} w^{*(m)}(n)h(\boldsymbol{x}^{(m)}(n)) \tag{5.15}$$

or by

$$E(h(\boldsymbol{X}(n))) \approx \sum_{m=1}^{M} w^{(m)}(n)h(\boldsymbol{x}^{(m)}(n)) \tag{5.16}$$

where

$$w^{*(m)}(n) = \frac{f(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{y}(1:n))}{\pi(\boldsymbol{x}^{(m)}(n))} \tag{5.17}$$

and

$$w^{(m)}(n) = \frac{\widetilde{w}^{(m)}(n)}{\sum_{i=1}^{M}\widetilde{w}^{(i)}(n)} \tag{5.18}$$

where

$$\widetilde{w}^{(m)}(n) = c\,w^{*(m)}(n)$$

with c being some unknown constant. The symbols $w^{*(m)}(n)$ and $w^{(m)}(n)$ are known as true and normalized importance weights of the particles $\boldsymbol{x}^{(m)}(n)$, respectively. They are introduced to correct for the bias that arises due to sampling from a different function than the one that is being approximated, $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))$. The estimate in (5.15) is unbiased whereas the one from (5.16) is with a small bias but often with a smaller mean-squared error than the one in (5.15) [55]. An advantage in using (5.16) over (5.15) is that we only need to know the ratio $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))/\pi(\boldsymbol{x}(n))$ up to a multiplicative constant and not the exact ratio in order to compute the estimate of the expectation of $h(\boldsymbol{X}(n))$.

How is (5.18) obtained? Suppose that the true weight cannot be found and instead we can only compute it up to a proportionality constant, that is

$$\begin{aligned}\widetilde{w}^{(m)}(n) &= c\frac{f(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{y}(1:n))}{\pi(\boldsymbol{x}^{(m)}(n))}\\ &= c\,w^{*(m)}(n)\end{aligned} \tag{5.19}$$

where the constant c is unknown. Since we must have

$$\begin{aligned} 1 &= \int \frac{f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))}{\pi(\boldsymbol{x}(n))} \pi(\boldsymbol{x}(n))\, d\boldsymbol{x}(n) \\ &\simeq \frac{1}{cM} \sum_{m=1}^{M} \widetilde{w}^{(m)}(n) \end{aligned}$$

from where we can estimate c by

$$c \approx \frac{1}{M} \sum_{m=1}^{M} \widetilde{w}^{(m)}(n). \tag{5.20}$$

Now, by using (5.19), we can express (5.15) in terms of $\widetilde{w}^{(m)}(n)$. We have

$$\begin{aligned} E(h(\boldsymbol{X}(n))) &\approx \frac{1}{M} \sum_{m=1}^{M} w^{*(m)}(n) h(\boldsymbol{x}^{(m)}(n)) \\ &= \frac{1}{cM} \sum_{m=1}^{M} \widetilde{w}^{(m)}(n) h(\boldsymbol{x}^{(m)}(n)) \\ &\approx \sum_{m=1}^{M} \frac{\widetilde{w}^{(m)}(n)}{\sum_{i=1}^{M} \widetilde{w}^{(i)}(n)} h(\boldsymbol{x}^{(m)}(n)) \\ &= \sum_{m=1}^{M} w^{(m)}(n) h(\boldsymbol{x}^{(m)}(n)) \end{aligned}$$

where $w^{(m)}(n)$ is the normalized weight from (5.18).

In summary, when we use a random measure to approximate a PDF, such as $f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))$, we can do it by

- drawing samples from a proposal distribution $\pi(\boldsymbol{x}(n))$, which needs to be known only up to a multiplicative constant, that is

$$\boldsymbol{x}^{(m)}(n) \sim \pi(\boldsymbol{x}(n)), \quad m = 1, 2, \ldots, M$$

- computing the weights of the particles $w^{(m)}(n)$ by (5.17) and (5.18).

The resulting random measure is of the form given by (5.9). As proposal distributions we choose ones that are easy to sample from and whose shape is close to the product of $h(\boldsymbol{x}(n))$ and $f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))$ [55]. A rule of thumb is that we would like to generate particles from regions of the support of $\boldsymbol{x}(n)$ where that product has large values.

In Figure 5.1 we see a posterior and a proposal distribution as well as particles and weights that form a random measure approximating the posterior. It is important to

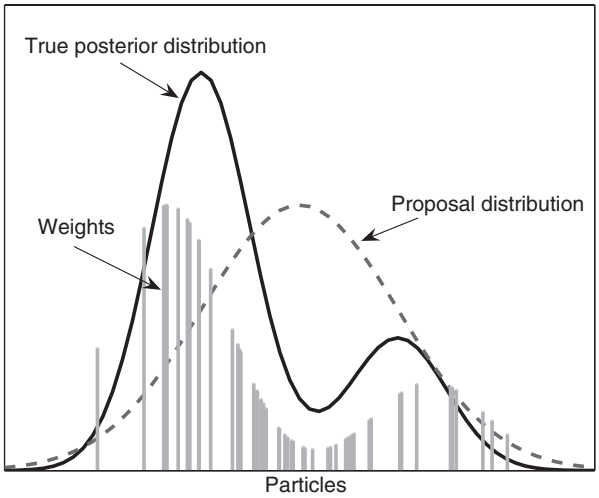

Figure 5.1 A posterior and a proposal distribution, and particles and weights that approximate the posterior.

note that the particle weights are *not* only functions of the posterior but also of the proposal distribution.

■ EXAMPLE 5.3

Suppose that we want to generate a random measure that approximates a Gaussian with mean one and variance one (target distribution). Let the proposal distribution be a standard Gaussian (that is, a Gaussian with mean zero and variance one). Figure 5.2 shows the target distribution, the drawn particles and their weights. From the figure, it may appear that there is a big discrepancy between the random measure and the Gaussian. However, when we compute the cumulative distribution functions (CDFs) of the Gaussian and the random measure, we can see that there is a good agreement between the two CDFs, as can be seen in Figure 5.3. Moreover, the figure also shows that there is better agreement between the two CDFs where there are more generated particles.

Since particle filtering exploits the concept of importance sampling heavily, we will address the choice of $\pi(\boldsymbol{x}(n))$ in more detail in Section 5.4. Now we turn our attention to the problem of recursive computation of the random measure $\chi(n)$.

How do we obtain $\chi(n)$ from $\chi(n-1)$? The random measures $\chi(n)$ and $\chi(n-1)$ approximate $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))$ and $f(\boldsymbol{x}(n-1)\,|\,\boldsymbol{y}(1:n-1))$, respectively. We write (see (5.10))

$$f(\boldsymbol{x}(n-1)\,|\,\boldsymbol{y}(1:n-1)) \simeq \sum_{m=1}^{M} w^{(m)}(n-1)\delta(\boldsymbol{x}(n-1)-\boldsymbol{x}^{(m)}(n-1)).$$

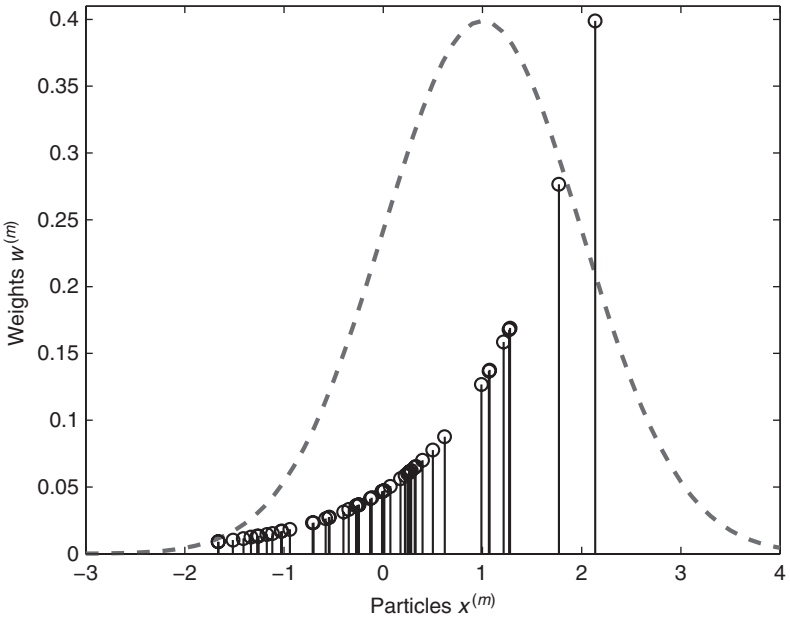

Figure 5.2 A target distribution $\mathcal{N}(1, 1)$, drawn particles from $\mathcal{N}(0, 1)$ and their weights.

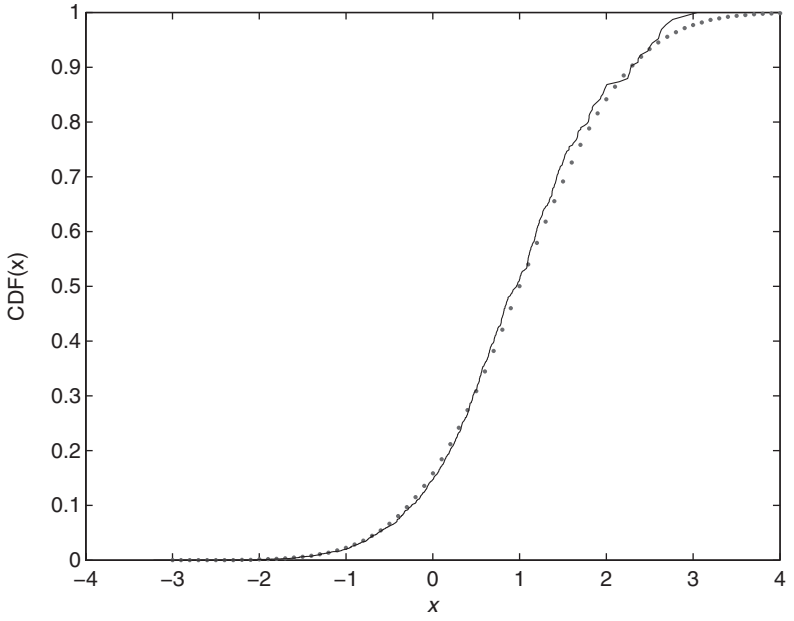

Figure 5.3 The CDF of $\mathcal{N}(1, 1)$ (dotted line) and its approximation (solid line).

For the filtering PDF $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))$, we have

$$\begin{aligned} f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n)) &\propto f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}(n)) \\ &\quad \times \int f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n-1)) f(\boldsymbol{x}(n-1)\,|\,\boldsymbol{y}(1:n-1))\, d\boldsymbol{x}(n-1) \\ &\simeq f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}(n)) \sum_{m=1}^{M} w^{(m)}(n-1) f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}^{(m)}(n-1)). \end{aligned}$$

We want to represent the new filtering density with $\chi(n)$, and to that end we need to generate particles $\boldsymbol{x}(n)$ and compute their weights. If for particle generation we use the proposal distribution $\pi(\boldsymbol{x}(n)\,|\,\boldsymbol{x}^{(m)}(n-1), \boldsymbol{y}(1:n))$, the newly generated particle $\boldsymbol{x}^{(m)}(n)$ is appended to the particle stream $\boldsymbol{x}^{(m)}(1:n-1)$, and we compute the value of its weight according to

$$\tilde{w}^{(m)}(n) = w^{(m)}(n-1) \frac{f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}^{(m)}(n)) f(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{x}^{(m)}(n-1))}{\pi(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{x}^{(m)}(n-1), \boldsymbol{y}(1:n))} \tag{5.21}$$

where $\tilde{w}^{(m)}(n)$ is a non-normalized weight of $\boldsymbol{x}^{(m)}(n)$.[6] We see that the weight of the m-th stream is obtained by updating its value at time instant $n-1$ with the factor

$$\frac{f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}^{(m)}(n)) f(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{x}^{(m)}(n-1))}{\pi(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{x}^{(m)}(n-1), \boldsymbol{y}(1:n))}.$$

The so obtained weights are then normalized so that they sum up to one.

In summary, the PF implements two steps. One is the generation of particles for the next time instant and the other is the computation of the weights of these particles. Figure 5.4 depicts the graphical flow and interpretation of the resulting particle filtering algorithm, and Table 5.1 summarizes the mathematical expressions needed for carrying out the recursive update (from $\chi(n-1)$ to $\chi(n)$).

■ EXAMPLE 5.4

Consider the state-space model given by

$$\begin{aligned} x(n) &= x(n-1) + v_1(n) \\ y(n) &= x(n) + v_2(n) \end{aligned}$$

where $v_1(n)$ and $v_2(n)$ are independent standard Gaussian random variables.[7] A particular realization of such system is given in Figure 5.5 and the values of the first four time instants are shown in Table 5.2.

[6]The nonnormalized weight $\tilde{w}^{(m)}(n)$ should be distinguished from the true weight defined by (5.17).

[7]We use this example only to explain how the particle filtering algorithm proceeds. Note that this model is linear and Gaussian and that the data can optimally be processed by the Kalman filter.

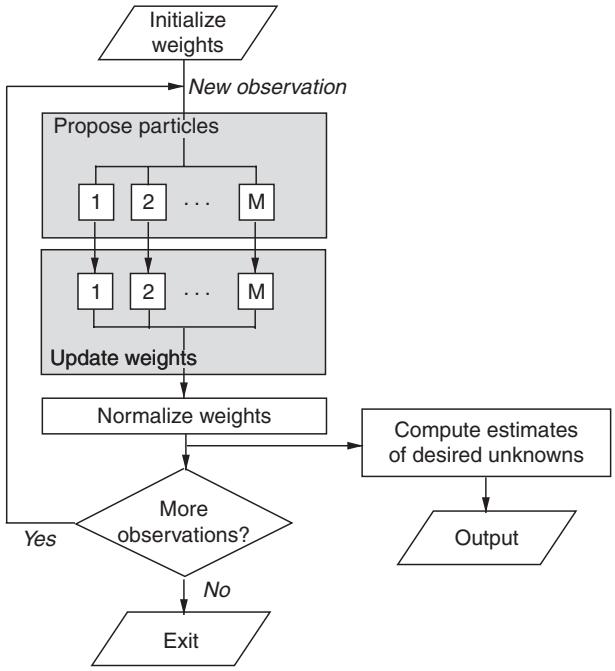

Figure 5.4 Flowchart of the sequential importance sampling algorithm.

Table 5.1 The sequential importance sampling algorithm

Particle generation

Basis

$$\pi(\boldsymbol{x}(0:n)\,|\,\boldsymbol{y}(1:n)) = \pi(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(0:n-1), \boldsymbol{y}(1:n))\,\pi(\boldsymbol{x}(0:n-1)\,|\,\boldsymbol{y}(1:n-1))$$

$$\boldsymbol{x}^{(m)}(0:n-1) \sim \pi(\boldsymbol{x}(0:n-1)\,|\,\boldsymbol{y}(1:n-1))$$

$$w^{(m)}(n-1) \propto \frac{f(\boldsymbol{x}^{(m)}(0:n-1)\,|\,\boldsymbol{y}(n-1))}{\pi(\boldsymbol{x}^{(m)}(0:n-1)\,|\,\boldsymbol{y}(1:n-1))}$$

Augmentation of the trajectory $\boldsymbol{x}^{(m)}(0:n-1)$ with $\boldsymbol{x}^{(m)}(n)$

$$\boldsymbol{x}^{(m)}(n) \sim \pi(\boldsymbol{x}(n)\,|\,\boldsymbol{x}^{(m)}(0:n-1), \boldsymbol{y}(1:n)),\ m = 1, \ldots, M$$

Weight update

$$w^{(m)}(n) \propto \frac{f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}^{(m)}(n))\,f(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{x}^{(m)}(n-1))}{\pi(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{x}^{(m)}(0:n-1), \boldsymbol{y}(1:n))} w^{(m)}(n-1) \quad m = 1, \ldots, M$$

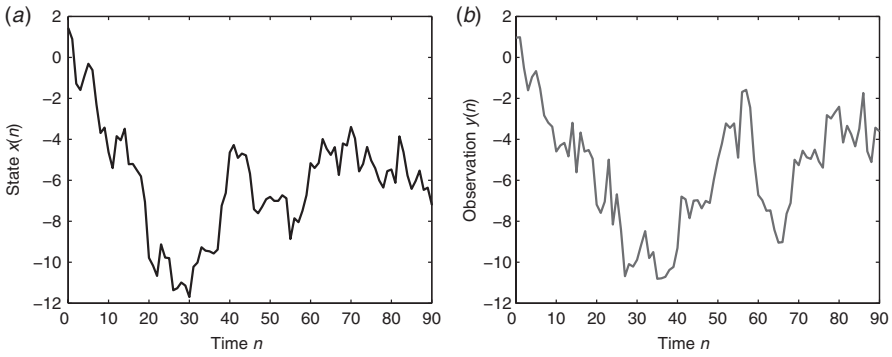

Figure 5.5 One realization of the state-space model of Example 5.4.

In this example, we choose to use the proposal distribution

$$\pi(x(n)) = f(x(n) \,|\, x(n-1)).$$

In other words, we generate the particles according to

$$x^{(m)} \sim f(x(n) \,|\, x^{(m)}(n-1)), \quad m = 1, 2, \ldots, M$$

that is, each particle stream has a separate proposal distribution. Since the proposal function is identical to the transition function $f(x(n) \,|\, x^{(m)}(n-1))$, from (5.21) we deduce that the update of the weights is according to

$$w^{(m)}(n) \propto f(y(n) \,|\, x^{(m)}(n)) w^{(m)}(n-1).$$

The step by step execution of the sequential importance sampling algorithm for the first three time instants proceeds as follows.

Initialization $n = 0$

$$x^{(m)}(0) \sim \mathcal{N}(0, 1) \quad m = 1, 2, \ldots, M.$$

Note that all the weights are equal (Fig. 5.6).

Time instant $n = 1$

Generation of particles using the proposal distribution

$$x^{(m)}(1) \sim f(x(1) \,|\, x^{(m)}(0))$$
$$= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x(1) - x^{(m)}(0))^2}{2} \right).$$

Table 5.2 Values of the first four time instants for Example 5.4

	$n = 0$	$n = 1$	$n = 2$	$n = 3$
$\boldsymbol{x}(n)$	1.44	0.89	−1.28	−1.59
$y(n)$	0.96	0.98	−0.51	−1.61

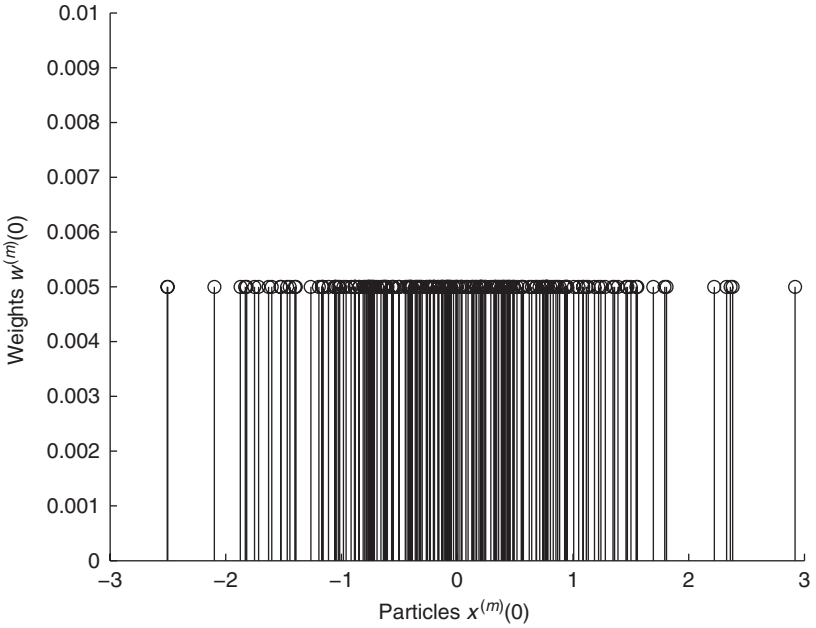

Figure 5.6 Particles and their weights at time instant $n = 0$.

Weight update

$$w^{(m)}(1) \propto \exp\left(-\frac{(y(1) - x^{(m)}(1))^2}{2}\right).$$

The particles and their weights are shown in Figure 5.7.

Time instant $n = 2$

Generation of particles using the proposal distribution

$$\begin{aligned}\mathbf{x}^{(m)}(2) &\sim f(x(2) \,|\, x^{(m)}(1)) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x(2) - x^{(m)}(1))^2}{2}\right).\end{aligned}$$

Weight update

$$w^{(m)}(2) \propto w^{(m)}(1) \exp\left(-\frac{(y(2) - x^{(m)}(2))^2}{2}\right).$$

The particles and their weights are shown in Figure 5.8.

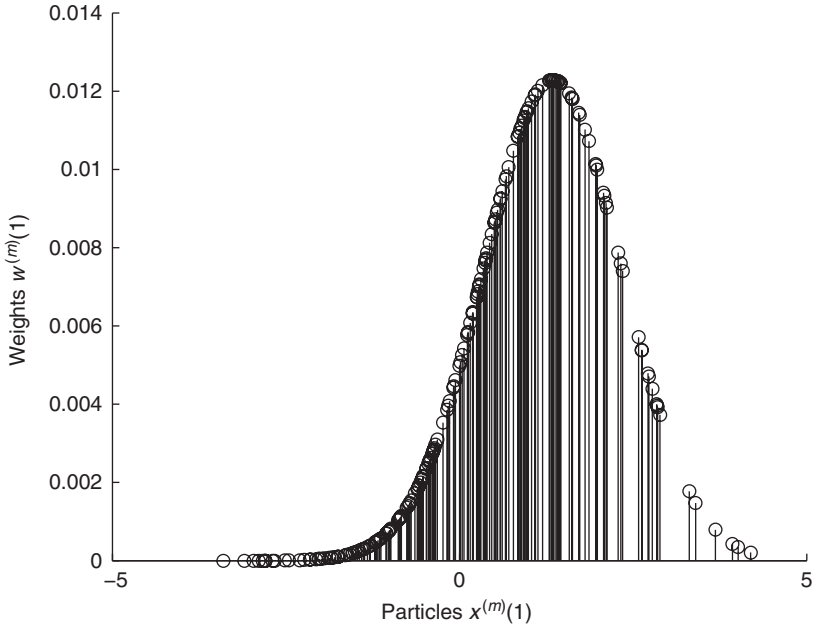

Figure 5.7 Particles and their weights at time instant $n = 1$.

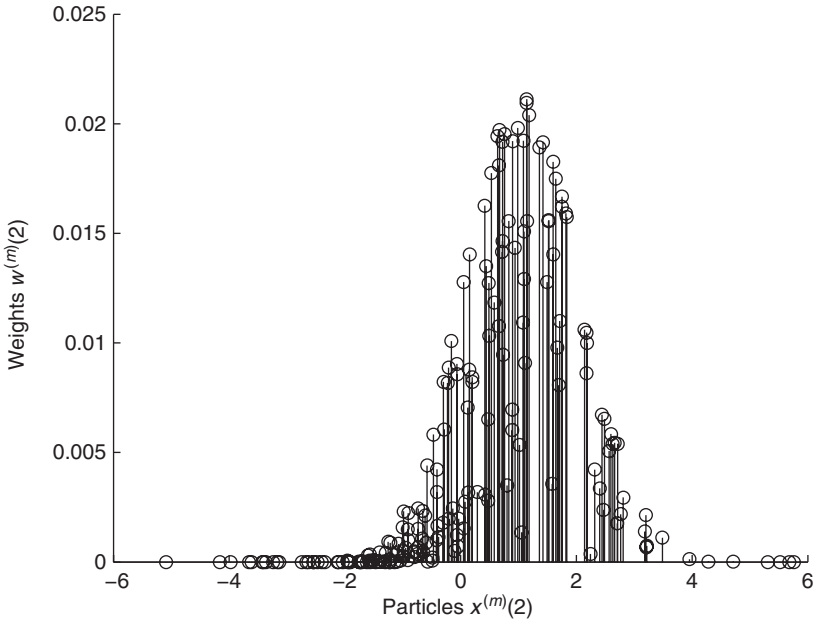

Figure 5.8 Particles and their weights at time instant $n = 2$.

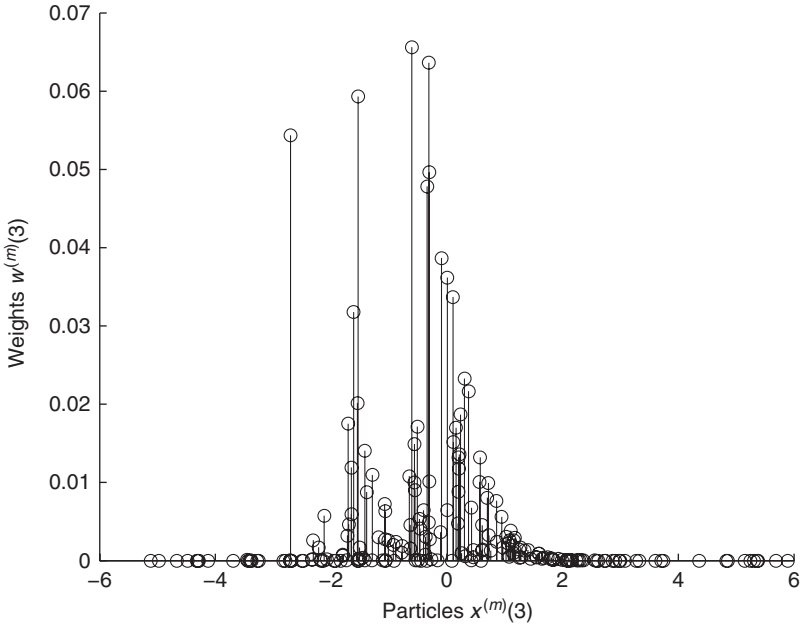

Figure 5.9 Particles and their weights at time instant $n = 3$.

Time instant $n = 3$

Generation of particles using the proposal distribution

$$x^{(m)}(3) \sim \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x(3) - x^{(m)}(2))^2}{2}\right).$$

Weight update

$$w^{(m)}(3) \propto w^{(m)}(2) \exp\left(-\frac{(y(3) - x^{(m)}(3))^2}{2}\right).$$

The particles and their weights are shown in Figure 5.9.

The process continues in the same way as new observations become available. Note that the variance of the weights increases with time, which is an unwanted effect. Even at time instant $n = 3$, a few particles have large weights and the rest have small weights.

5.4 THE CHOICE OF PROPOSAL DISTRIBUTION AND RESAMPLING

In this section two important issues that affect the performance and implementation of particle filtering algorithms are discussed. One is the choice of proposal distributions

and the other the concept of resampling, which turns out to be indispensable in the implementation of PFs.

5.4.1 Choice of Proposal Distribution

The proposal distribution (importance function) plays a crucial role in the performance of particle filtering. From a practical and intuitive point of view, it is desirable to use easy-to-sample proposal distributions that produce particles with a large enough variance in order to avoid exploration of the state space in too narrow regions and thereby contributing to losing the tracks of the state, but not too large to alleviate generation of too dispersed particles [55]. The support of the proposal distributions has to be the same as that of the targeted distribution.

As already pointed out, the approximation of the posterior distribution with the random measure obtained by the proposal distribution will improve if the proposal becomes very similar to the posterior. In fact, the optimal choice for the importance function is the posterior distribution, that is

$$\pi(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(0:n-1), \boldsymbol{y}(1:n)) = f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(n-1), \boldsymbol{y}(n)) \tag{5.22}$$

which corresponds to the following weight calculation

$$w^{(m)}(n) \propto w^{(m)}(n-1) f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}^{(m)}(n-1)).$$

This importance function minimizes the variance of the weights, $w^{(m)}(n)$, conditional on $\boldsymbol{x}^{(m)}(0:n-1)$ and $\boldsymbol{y}(1:n)$. However, the computation of the weights requires the integration of $\boldsymbol{x}(n)$, that is, solving

$$f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}^{(m)}(n-1)) = \int f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}(n)) f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(n-1)) d\boldsymbol{x}(n).$$

Thus, the implementation of the optimal importance function may be difficult for two reasons: First, direct sampling from the posterior (5.22) may not be easy, and second, the computation of the weights may require integration.

■ **EXAMPLE 5.5**

Consider the following decomposition of the posterior in (5.22)

$$f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(n-1), \boldsymbol{y}(n)) \propto f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}(n)) f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(n-1)).$$

If the distributions on the right-hand side of the previous expressions are Gaussians, their product will also be Gaussian. Thus, the proposal is a Gaussian and sampling from it can readily be performed. It is worth pointing out that if the noises in the system are additive and Gaussian and the observation is a linear

function of the state, we can sample from the optimal proposal distribution *even* if the function in the state equation is nonlinear.

A popular choice for the importance function is the prior

$$\pi(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(m)}(0:n-1), \boldsymbol{y}(1:n)) = f(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(m)}(n-1))$$

which yields importance weights proportional to the likelihood

$$w^{(m)}(n) \propto w^{(m)}(n-1) f(\boldsymbol{y}(n) \mid \boldsymbol{x}^{(m)}(n)).$$

The main advantage of this choice is the ease in the computation of the weights, which amounts to obtaining the likelihood function. However, the generation of the particles is implemented without the use of observations, and therefore not all of the available information is used to explore the state space. This may lead in some practical cases to poor estimation results. Strategies to improve this performance consist of the inclusion of a prediction step like that of the auxiliary PF [65] (see Subsection 5.5.2) or the use of a hybrid importance function if possible [37]. We refer to an importance function as a hybrid importance function if part of the state is proposed from a prior and the remaining state from the optimal importance function.

■ **EXAMPLE 5.6**

Consider a state space whose parameters can be divided in two groups $\boldsymbol{x}(n) = \{\boldsymbol{x}_1(n)\boldsymbol{x}_2(n)\}$ and where sampling can be carried out from $f(\boldsymbol{x}_2(n) \mid \boldsymbol{x}_1^{(m)}(n-1), \boldsymbol{x}_2^{(m)}(n-1))$ and $f(\boldsymbol{x}_1(n) \mid \boldsymbol{x}_2^{(m)}(n), \boldsymbol{x}_2^{(m)}(n-1), \boldsymbol{x}_1^{(m)}(n-1), \boldsymbol{y}(n))$. A hybrid proposal that combines the prior and the posterior importance functions is given by

$$\begin{aligned}\pi(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(m)}(n-1), \boldsymbol{y}(n)) = {} & f(\boldsymbol{x}_2(n) \mid \boldsymbol{x}_1^{(m)}(n-1), \boldsymbol{x}_2^{(m)}(n-1)) \\ & \times f(\boldsymbol{x}_1(n) \mid \boldsymbol{x}_2^{(m)}(n), \boldsymbol{x}_2^{(m)}(n-1), \boldsymbol{x}_1^{(m)}(n-1), \boldsymbol{y}(n))\end{aligned}$$

where $\boldsymbol{x}_2^{(m)}(n)$ is a sample from $f(\boldsymbol{x}_2(n) \mid \boldsymbol{x}_1^{(m)}(n-1), \boldsymbol{x}_2^{(m)}(n-1))$. The update of the weights can readily be obtained from the general expression given by (5.21).

5.4.2 Resampling

In particle filtering the discrete random measure degenerates quickly and only few particles are assigned meaningful weights. This degradation leads to a deteriorated functioning of particle filtering. Figure 5.10 illustrates three consecutive time instants of the operation of a PF which does not use resampling. Particles are represented by circles and their weights are reflected by the corresponding diameters. Initially, all the particles have the same weights, that is, all the diameters are equal, and at each time step the particles are propagated and assigned weights (note that the true posterior at each time instant is depicted in the figure.) As time evolves, all the particles except

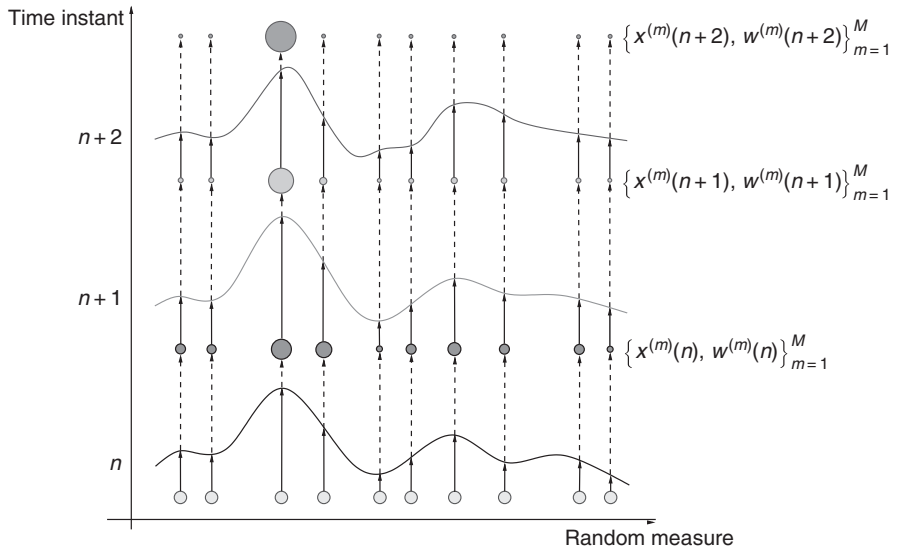

Figure 5.10 A pictorial description of particle filtering without resampling.

for very few are assigned negligible weights. In the last step there is only one particle with significant weight.

The mechanism of resampling as well as the use of good importance function can reduce this degeneracy. A measure of this degeneracy is the effective particle size defined by [54]

$$M_{\text{eff}} = \frac{M}{1 + \text{Var}(w^{*(m)}(n))}$$

where $w^{*(m)}(n) = \dfrac{f(\boldsymbol{x}(n))}{\pi(\boldsymbol{x}(n))}$ is the true particle weight. This metric can be estimated as

$$\hat{M}_{\text{eff}} = \frac{1}{\sum_{m=1}^{M} (w^{(m)}(n))^2}$$

with $w^{(m)}(n)$ being the normalized weight corresponding to the m-th particle at time instant n. If the effective particle size is below a predefined threshold, resampling is carried out. Clearly, when all the particles have the same weights, the variance of the weights is zero and the particle size is equal to the number of particles, M. The other extreme occurs when all the particles except one have negligible weights, and the particle size is equal to one.

Resampling eliminates particles with small weights and replicates particles with large weights. In general, if the random measure at time instant n is $\chi(n)$, it proceeds as follows.

1. Draw M particles, $\boldsymbol{x}^{(k_m)}(n)$, from the distribution given by $\chi(n)$, where the k_ms are the indexes of the drawn particles.
2. Let $\boldsymbol{x}^{(m)}(n) = \boldsymbol{x}^{(k_m)}(n)$, and assign equal weights $\frac{1}{M}$ to the particles.

The generation of the M resampled particles can be implemented in different ways. One possible scheme is illustrated in Figure 5.11 with $M = 5$ particles. There, the left column of circles depicts particles before resampling and the diameters of the circles are proportional to the weights of the particles. The dashed lines from the particles on the left to the middle line, which is equally divided in five areas, illustrate the mapping of the weights. The right column of circles are the particles after resampling. In general, the large particles are replicated and the small particles are removed. For example, the particle with the largest weight is replicated three times and the next two particles in size are preserved. The remaining two particles have small weights, and they are removed. After resampling all the circles have equal diameters, that is, all the weights are set to $\frac{1}{M}$.

Figure 5.12 illustrates the random measures and the actual probability distributions of interest in two consecutive time steps of the particle filtering algorithm that performs resampling after each cycle. In the figure, the solid curves represent the

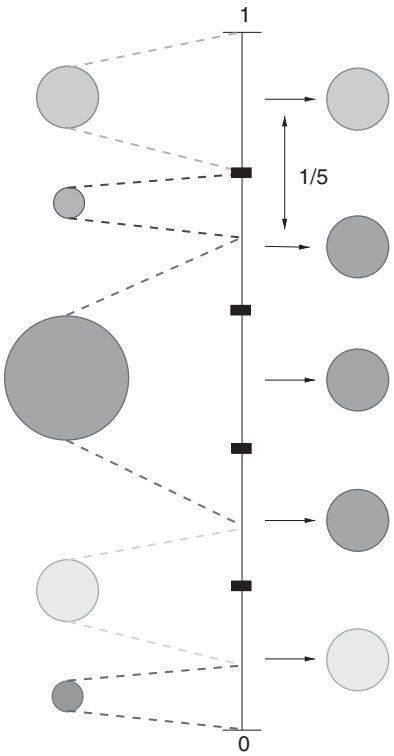

Figure 5.11 The concept of resampling.

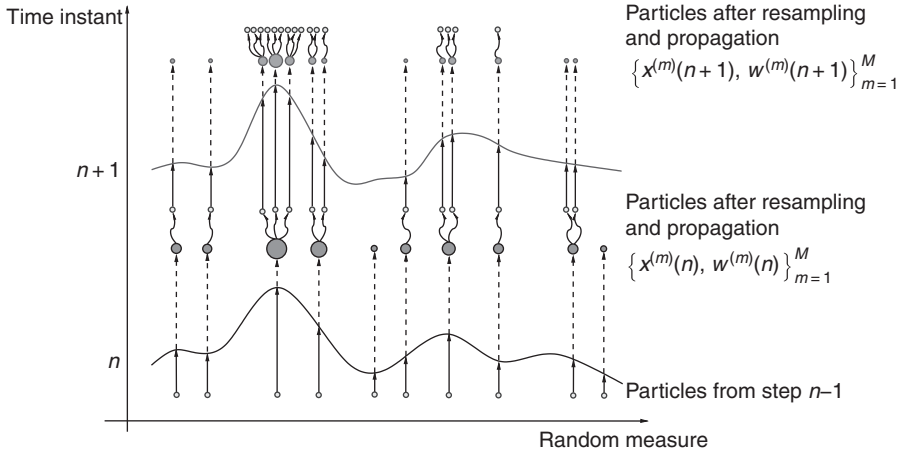

Figure 5.12 A pictorial description of particle filtering with resampling.

distributions of interest, which are approximated by the discrete measures. The sizes of the circles reflect the weights that are assigned to the particles after being generated from the proposal distribution. The resampling step eliminates the particles with the smallest weights and replicates those with large weights. The new set of particles have equal weights and are propagated in the next time step.

A summary of a particle filtering algorithm that includes resampling is displayed in the flowchart of Figure 5.13. At time instant n, a new set of particles is generated, and their weights are computed. Thereby the random measure $\chi(n)$ is generated and

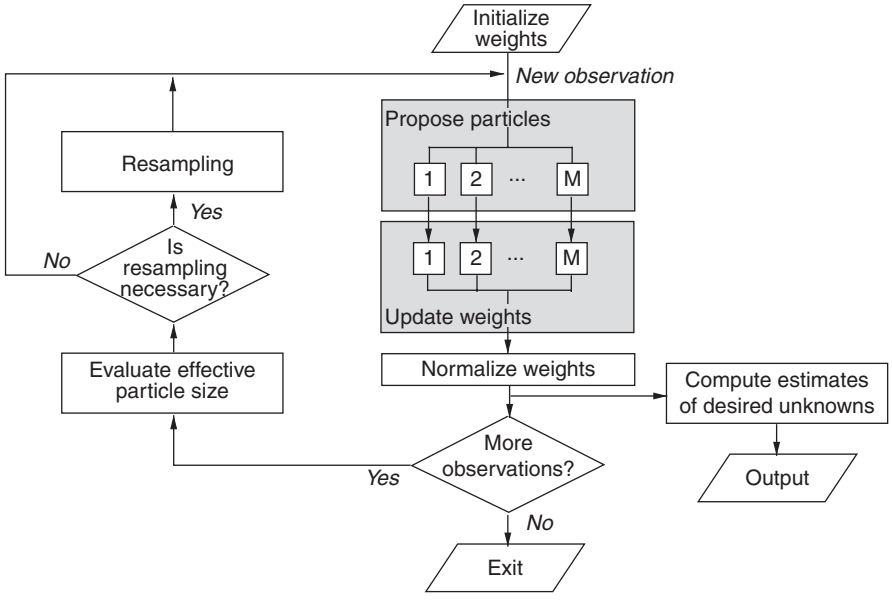

Figure 5.13 Flowchart of a particle filtering algorithm.

can be used for the estimation of the desired unknowns. Before the next step $n+1$, the effective particle size is estimated and resampling is carried out if necessary.

It is important to note that resampling becomes a major obstacle for efficient implementation of particle filtering algorithms in parallel very large scale integration (VLSI) hardware devices, because it creates full data dependencies among processing units [11]. Although some methods have been recently proposed [12], parallelization of resampling algorithms remains an open area of research.

5.5 SOME PARTICLE FILTERING METHODS

In this section we present three different particle filtering methods. They are also known as sampling-importance-resampling (SIR), auxiliary particle filtering (APF), and Gaussian particle filtering (GPF). The common feature of each of these methods is that at time instant n, they are represented by a discrete random measure given by $\chi(n) = \{\boldsymbol{x}^{(m)}(n),\ w^{(m)}(n)\}_{m=1}^{M}$, where, as before, $\boldsymbol{x}^{(m)}(n)$ is the m-th particle of the state vector at time instant n, $w^{(m)}(n)$ is the weight of that particle, and M is the number of particles. For each of these filters, we show how this random measure is obtained from $\chi(n-1)$ by using the observation vector $\boldsymbol{y}(n)$.

5.5.1 SIR particle filtering

The SIR method is the simplest of all of the particle filtering methods. It was proposed in [31] and was named bootstrap filter. Namely, the SIR method employs the prior density for drawing particles, which implies that the weights are only proportional to the likelihood of the drawn particles.[8] We now explain this in more detail.

Recall that if the particles are generated from a density function $\pi(\boldsymbol{x}(n))$, and the weights of the particles in the previous time step were $w^{(m)}(n-1)$, upon the reception of the measurement $\boldsymbol{y}(n)$, the weights are updated by

$$\widetilde{w}^{(m)}(n) = w^{(m)}(n-1)\frac{f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}^{(m)}(n))f(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{x}^{(m)}(n-1))}{\pi(\boldsymbol{x}^{(m)}(n)\,|\,\boldsymbol{x}^{(m)}(n-1),\boldsymbol{y}(1:n))}$$

where $\widetilde{w}^{(m)}(n)$ denotes a nonnormalized weight. If the proposal distribution is equal to the prior, that is

$$\pi(\boldsymbol{x}(n)\,|\,\boldsymbol{x}^{(m)}(n-1),\boldsymbol{y}(1:n)) = f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}^{(m)}(n-1))$$

the computation of the weights simplifies to

$$\widetilde{w}^{(m)}(n) = w^{(m)}(n-1)f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}^{(m)}(n)).$$

[8]Here we assume that the weights from the previous time instant are all equal due to resampling.

Furthermore, if the weights from the previous time step were all equal (because of resampling), the previous update equation becomes even simpler

$$\widetilde{w}^{(m)}(n) = f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}^{(m)}(n))$$

that is, the weight of the particle $\boldsymbol{x}^{(m)}(n)$ is only proportional to the likelihood of that particle.

Next, we explain in more detail the steps of the SIR method. In the first step we draw candidate particles $\boldsymbol{x}_c^{(m)}(n)$ according to

$$\boldsymbol{x}_c^{(m)}(n) \sim f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(n-1)), \;\; m = 1, 2, \ldots M.$$

In the second step, we compute the non-normalized weights of these particles by

$$\widetilde{w}^{(m)}(n) = f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}_c^{(m)}(n)), \;\; m = 1, 2, \; \ldots \; M$$

and then normalize them so that they sum up to one, that is, we use

$$w^{(m)}(n) = \frac{\widetilde{w}^{(m)}(n)}{\sum_{j=1}^{M} \widetilde{w}^{(j)}(n)}.$$

With the completion of this step, we have the random measure $\{\boldsymbol{x}_c^{(m)}(n), w^{(m)}(n)\}_{m=1}^{M}$, which should be used for computations of desired estimates.

In the third and final step we perform resampling of the particles $\boldsymbol{x}_c^{(m)}(n)$ according to the multinomial probability mass function defined by the weights $w^{(m)}(n)$. Namely, we draw indices k_m, for $m = 1, 2, \ldots, M$, where $Pr(k_m = i) = w^{(i)}(n)$. Once the indexes are drawn, we set

$$\boldsymbol{x}^{(m)}(n) = \boldsymbol{x}_c^{(k_m)}(n).$$

The weights of the particles $\boldsymbol{x}^{(m)}(n)$ are set to $w^{(m)}(n) = 1/M$. The whole SIR procedure is summarized in Table 5.3.

If resampling was not implemented as a third step, we have

$$\begin{aligned} \boldsymbol{x}^{(m)}(n) &= \boldsymbol{x}_c^{(m)}(n) \\ \widetilde{w}^{(m)}(n) &= w^{(m)}(n-1) f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}_c^{(m)}(n)) \end{aligned}$$

for $m = 1, 2, \ldots M$, and then we normalize the weights. After normalization, we would typically perform a test to decide if resampling is needed. If it was necessary, we would implement it as described; if not, we would proceed with the next time step. Recall that with resampling some particles are removed and the ones that are preserved may be replicated. We reiterate that, in general, resampling does not have to be implemented at every time step, but here we adopt to do so. If resampling is not performed at the end

Table 5.3 The SIR algorithm

Initialization

For $m = 1, \ldots, M$

Sample $\boldsymbol{x}^{(m)}(0) \sim f(\boldsymbol{x}(0))$

$w^{(m)}(0) = \dfrac{1}{M}$

Recursions

For $n = 1, 2, \ldots$

For $m = 1, 2, \ldots, M$

Proposal of candidate particles

Sample $\boldsymbol{x}_c^{(m)}(n) \sim f(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(m)}(n-1))$

Computation of weights

Evaluate the weights, $\tilde{w}^{(m)}(n) = f(\boldsymbol{y}(n) \mid \boldsymbol{x}_c^{(m)}(n))$

Normalize the weights, $w^{(m)}(n) = \dfrac{\tilde{w}^{(m)}(n)}{\sum_{j=1}^{M} \tilde{w}^{(j)}(n)}$

Resampling

Sample k_m, where $Pr(k_m = i) = w^{(i)}(n)$

Set $\boldsymbol{x}^{(m)}(n) = \boldsymbol{x}_c^{(k_m)}(n)$ and

$w^{(m)}(n) = \dfrac{1}{M}$

of the recursion, the generated particles in the first step represent the support of the random measure used for the next time instant.

A clear advantage of the SIR method is that it is very simple for implementation. Its disadvantage is that in drawing the particles $\boldsymbol{x}^{(m)}(n)$ for exploring the state space, we do not use the observation $\boldsymbol{y}(n)$. In other words, once the particles are generated, the only thing we can do to steer the particles towards the region of the state space with large probability density is by resampling. If all of the generated particles are already far away from such regions, resampling will not help.

5.5.2 Auxiliary Particle Filtering

The APF attempts to improve the ability of the PF in exploring the state space by using the latest measurements. We know that with particle filtering, we approximate the

filtering density $f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n))$ by the mixture density

$$f(\boldsymbol{y}(n) \mid \boldsymbol{x}(n)) \sum_{m=1}^{M} w^{(m)}(n-1) f(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(m)}(n-1)). \tag{5.23}$$

The underlying idea behind APF is to propose samples $\boldsymbol{x}^{(m)}(n)$ from this density. In order to do so, we introduce an auxiliary variable, which is an index variable. We denote it by k and we index it by m, so that we write it as k_m. We draw it from the set $\{1, 2, \ldots, M\}$,[9] and it denotes the particle stream which we want to update. Thus, if we draw $k_m = 5$, we work with the 5th stream, if we have $k_m = 11$, it is the 11th stream and so on.

First we describe the basic APF method. This method makes easy the problem of drawing $\boldsymbol{x}(n)$ from (5.23) by using estimates of $\boldsymbol{x}(n)$ for each stream of particles. If we denote the estimates by $\hat{\boldsymbol{x}}^{(m)}(n)$, we modify (5.23) and create a proposal distribution given by

$$\sum_{m=1}^{M} w^{(m)}(n-1) f(\boldsymbol{y}(n) \mid \hat{\boldsymbol{x}}^{(m)}(n)) f(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(m)}(n-1)). \tag{5.24}$$

An estimate of $\boldsymbol{x}^{(m)}(n)$ can be any value of $\boldsymbol{x}(n)$ that is a good representative, which means that it should be a value that can easily be computed and has high likelihood. For example, if the state equation is

$$\boldsymbol{x}(n) = g_1(\boldsymbol{x}(n)) + \boldsymbol{v}_1(n)$$

and the noise vector $\boldsymbol{v}_1(n)$ is zero mean, an estimate of $\boldsymbol{x}^{(m)}(n)$ could be

$$\hat{\boldsymbol{x}}^{(m)}(n) = g_1(\boldsymbol{x}^{(m)}(n-1)).$$

With the estimates $\hat{\boldsymbol{x}}^{(m)}(n)$ and the new form of the proposal distribution (5.24), it is much easier to propose new particles $\boldsymbol{x}^{(m)}(n)$. The idea has a subtle point: we use (5.24) as a joint distribution of the auxiliary variable and the state. The implications is that first we draw the auxiliary variable (index) k_m from a multinomial distribution, where $Pr(k_m = i) \propto w^{(i)}(n-1)\, f(\boldsymbol{y}(n) \mid \hat{\boldsymbol{x}}^{(i)}(n))$.[10] The drawn index, say i, identifies the distribution from which we draw $\boldsymbol{x}^{(m)}(n)$, $f(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(k_m=i)}(n-1))$, and so we proceed by drawing a particle from $f(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(k_m=i)}(n-1))$. Once the particles are drawn, as with SIR, the last step is the computation of the weights.

Before we derive the formula for the update of the weights, we express the proposal distribution in a form that will make the derivation easy. First, we rewrite

[9] In fact, the number of drawn auxiliary variables can be different from M [65].
[10] This is basically the same procedure applied for resampling.

the proposal as

$$\pi(\boldsymbol{x}(n),\, k_m \,|\, \boldsymbol{y}(1:n)) = f(\boldsymbol{x}(n) \,|\, k_m,\, \boldsymbol{y}(1:n)) f(k_m \,|\, \boldsymbol{y}(1:n)).$$

The first factor on the right is

$$f(\boldsymbol{x}(n) \,|\, k_m,\, \boldsymbol{y}(1:n)) = f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(k_m)}(n-1))$$

because according to (5.24), given k_m, $\boldsymbol{y}(n)$ does not affect the density of $\boldsymbol{x}(n)$ and furthermore, given k_m, the density of $\boldsymbol{x}(n)$ is not a function of $\boldsymbol{y}(1:n-1)$ either, and instead it is simply the prior. This follows from the Markovian nature of the state variable. Recall that k_m is an index that points to the k_m-th stream and all its particles $\boldsymbol{x}^{(k_m)}(0:n-1)$. Thus, once, k_m is known, so is $\boldsymbol{x}^{(k_m)}(n-1)$, implying that all of the measurements $\boldsymbol{y}(1:n-1)$ become irrelevant in expressing the density of $\boldsymbol{x}(n)$.

For the second factor we can write

$$f(k_m \,|\, \boldsymbol{y}(1:n)) \propto w^{(k_m)}(n-1) f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(k_m)}(n))$$

where the weight $w^{(k_m)}(n-1)$ is a function of $\boldsymbol{y}(1:n-1)$ and, clearly, $f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(k_m)}(n))$ is a function of $\boldsymbol{y}(n)$. It is obvious that $f(k_m \,|\, \boldsymbol{y}(1:n))$ represents the probability of drawing k_m.

Therefore, for the update of the weight, we have the following

$$\begin{aligned}
\widetilde{w}^{(m)}(n) &= w^{(k_m)}(n-1) \frac{f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}^{(m)}(n)) f(\boldsymbol{x}^{(m)}(n) \,|\, \boldsymbol{x}^{(k_m)}(n-1))}{\pi(\boldsymbol{x}^{(m)}(n),\, k_m \,|\, \boldsymbol{y}(1:n))} \\
&= w^{(k_m)}(n-1) \frac{f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}^{(m)}(n)) f(\boldsymbol{x}^{(m)}(n) \,|\, \boldsymbol{x}^{(k_m)}(n-1))}{w^{(k_m)}(n-1) f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(k_m)}(n)) f(\boldsymbol{x}^{(m)}(n) \,|\, \boldsymbol{x}^{(k_m)}(n-1))} \\
&= \frac{f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}^{(m)}(n))}{f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(k_m)}(n))}.
\end{aligned}$$

In summary, the procedure is rather straightforward. Given the particles and their weights at time instant $n-1$, $\boldsymbol{x}^{(m)}(n-1)$ and $w^{(m)}(n-1)$, respectively, first we compute estimates $\hat{\boldsymbol{x}}^{(m)}(n)$. For these estimates we evaluate the weights by

$$\hat{w}^{(m)}(n) \propto w^{(m)}(n-1) f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(m)}(n)). \tag{5.25}$$

Next we draw the indexes of the particle streams that we will continue to append. The indexes are drawn from the multinomial distribution with parameters $\hat{w}^{(m)}(n)$. With this, we effectively perform resampling. After this step, we draw the particles of $\boldsymbol{x}(n)$ according to

$$\boldsymbol{x}^{(m)}(n) \sim f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(k_m)}(n-1)),\ \ m = 1, 2, \ldots, M.$$

The last step amounts to computing the weights of these particles by

$$w^{(m)}(n) \propto \frac{f(\boldsymbol{y}(n) \mid \boldsymbol{x}^{(m)}(n))}{f(\boldsymbol{y}(n) \mid \hat{\boldsymbol{x}}^{(k_m)}(n))}. \tag{5.26}$$

The algorithm is summarized in Table 5.4.

So, what do we gain by computing estimates of $\boldsymbol{x}(n)$ and implementing the drawing of particles by auxiliary variables? By using the estimates of $\boldsymbol{x}^{(m)}(n)$, we look ahead to how good the particle streams may be. Rather than resampling from samples obtained from the prior, we first resample by using the latest measurement and then propagate from the surviving streams. Thereby, at the end of the recursion instead of having particles propagated without the use of $\boldsymbol{y}(n)$, we have particles moved in directions preferred by $\boldsymbol{y}(n)$. With SIR, the data $\boldsymbol{y}(n)$ affect the direction of particle propagation later than they do with APF.

Table 5.4 Auxiliary particle filter

Initialization

For $m = 1, 2, \ldots, M$

Sample $\boldsymbol{x}(0)^{(m)} \sim f(\boldsymbol{x}(0))$

$w^{(m)}(0) = \dfrac{1}{M}$

Recursions

For $n = 1, 2, \ldots$

For $m = 1, 2, \ldots, M$

Estimation of next particles

Compute $\hat{\boldsymbol{x}}^{(m)}(n) = E(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(m)}(n-1))$

Sample the indexes k_m of the streams that survive

Sample $k_m = i$ with probability

$w^{(i)}(n-1) f(\mathbf{y}(n) \mid \hat{\boldsymbol{x}}^{(i)}(n))$

Sample the new particles for time instant n

$\boldsymbol{x}^{(m)}(n) \sim f(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(k_m)}(n-1))$

Computation of weights

Evaluate the weights $\tilde{w}^{(m)}(n) = \dfrac{f(\boldsymbol{y}(n) \mid \boldsymbol{x}^{(m)}(n))}{f(\boldsymbol{y}(n) \mid \hat{\boldsymbol{x}}^{(k_m)}(n))}$

Normalize the weights $w^{(m)}(n) = \dfrac{\tilde{w}^{(m)}(n)}{\sum_{j=1}^{M} \tilde{w}^{(j)}(n)}$

It is important to note that one cannot guarantee that the basic APF method will perform better than the SIR algorithm, the help of the latest observation notwithstanding. The reason for this is that the new particles are still generated by the prior only.

Now we describe a more general version of the APF method. As with the basic APF, one preselects the streams that are propagated. Instead of a preselection based on the weights (5.25), we use weights that we denote by $w_a^{(m)}(n)$. After the k_m is selected, for propagation, we use the proposal distribution $\pi(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(k_m)}(n-1), \boldsymbol{y}(n))$, that is

$$\boldsymbol{x}^{(m)}(n) \sim \pi(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(k_m)}(n-1), \boldsymbol{y}(n)). \tag{5.27}$$

The new weights are then computed according to

$$w^{(m)}(n) \propto \frac{w^{(k_m)}(n-1)}{w_a^{(k_m)}(n)} \frac{f(\boldsymbol{y}(n) \mid \boldsymbol{x}^{(m)}(n)) f(\boldsymbol{x}^{(m)}(n) \mid \boldsymbol{x}^{(k_m)}(n-1))}{\pi(\boldsymbol{x}^{(m)}(n) \mid \boldsymbol{x}^{(k_m)}(n-1), \boldsymbol{y}(n))}. \tag{5.28}$$

Note that for the basic APF we have

$$w_a^{(k_m)}(n) \propto w^{(k_m)}(n-1) f(\boldsymbol{y}(n) \mid \hat{\boldsymbol{x}}^{(k_m)}(n))$$

and

$$\pi(\boldsymbol{x}^{(m)}(n) \mid \boldsymbol{x}^{(k_m)}(n-1), \boldsymbol{y}(n)) = f(\boldsymbol{x}^{(m)}(n) \mid \boldsymbol{x}^{(k_m)}(n-1)).$$

As with the standard PFs, the performance of the APF depends strongly on the proposal distribution $\pi(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(k_m)}(n-1), \boldsymbol{y}(n))$. However, its performance also depends on the choice of the weights $w_a^{(m)}(n)$.

5.5.3 Gaussian Particle Filtering

With Gaussian particle filtering, we approximate the posterior and the predictive densities of the states with Gaussians [50] as is done by the EKF [3]. However, unlike with the EKF, we do not linearize any of the nonlinearities in the system. The method is very simple to implement, and its advantage over other particle filtering methods is that it does not require resampling. Another advantage is that the estimation of constant parameters is not a problem as is the case with other methods (see Section 5.6). In brief, the treatment of dynamic and constant states is identical. On the other hand, its disadvantage is that if the approximated densities are not Gaussians, the estimates may be inaccurate and the filter may diverge as any other method that uses Gaussian approximations. In that case, an alternative could be the use of Gaussian sum particle filtering where the densities in the system are approximated by mixture Gaussians [51].

The method proceeds as follows. Let the random measure at time instant $n-1$ be given by $\chi(n-1) = \{\boldsymbol{x}^{(m)}(n-1),\, 1/M\}_{m=1}^{M}$. In the first step, we draw samples $\boldsymbol{x}_c^{(m)}(n)$ from the prior, that is

$$\boldsymbol{x}_c^{(m)}(n) \sim f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(n-1)).$$

For the obtained samples, we compute their weights according to

$$\widetilde{w}^{(m)}(n) = f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}_c^{(m)}(n))$$

and then we normalize them. Now, the assumption is that the particles $\boldsymbol{x}_c^{(m)}(n)$ and their weights $w^{(m)}(n)$ approximate a normal distribution whose moments are estimated by

$$\boldsymbol{\mu}(n) = \sum_{m=1}^{M} w^{(m)}(n)\, \boldsymbol{x}_c^{(m)}(n)$$

$$\boldsymbol{\Sigma}(n) = \sum_{m=1}^{M} w^{(m)}(n)(\boldsymbol{x}_c^{(m)}(n) - \boldsymbol{\mu}(n))(\boldsymbol{x}_c^{(m)}(n) - \boldsymbol{\mu}(n))^{\top}.$$

Finally, the particles that are used for propagation at the next time instant $n+1$ are generated by

$$\boldsymbol{x}^{(m)}(n) \sim \mathcal{N}(\boldsymbol{\mu}(n), \boldsymbol{\Sigma}(n))$$

and they are all assigned the same weights. The method is summarized by Table 5.5.

5.5.4 Comparison of the Methods

In this Subsection, we show the performances of the presented methods (SIR, APF, and GPF) on simulated data.

■ **EXAMPLE 5.7**

The data are considered to be generated according to a stochastic volatility model, which is defined by [6, 16, 31, 47]

$$x(n) = \alpha x(n-1) + v_1(n) \tag{5.29}$$

$$y(n) = \beta v_2(n) e^{\boldsymbol{x}(n)/2} \tag{5.30}$$

where the unknown state variable $x(n)$ is called log-volatility, $v_1(n) \sim \mathcal{N}(0, \sigma^2)$, $v_2(n) \sim \mathcal{N}(0, 1)$, and α and β are parameters known as persistence in volatility shocks and modal volatility, respectively. The Gaussian processes $v_1(n)$ and $v_2(n)$ are assumed independent. This model is very much researched in the econometrics literature. Given the observations $y(n)$ and the model parameters, we want to estimate the unobserved state $x(n)$.

Table 5.5 The GPF algorithm

Initialization

For $n = 1, 2, \ldots, M$

Sample $\boldsymbol{x}_c^{(m)}(0) \sim f(\boldsymbol{x}(0))$

$$w^{(m)}(0) = \frac{1}{M}$$

Recursions

For $n = 1, 2, \ldots$

Drawing particles from the predictive density

$$\boldsymbol{x}_c^{(m)}(n) \sim f(\boldsymbol{x}(n) \mid \boldsymbol{x}^{(m)}(n-1)), \quad m = 1, 2, \ldots, M$$

Computation of the weights

$$\tilde{w}^{(m)}(n) = f(\boldsymbol{y}(n) \mid \boldsymbol{x}_c^{(m)}(n))$$

$$w^{(m)}(n) = \frac{\tilde{w}^{(m)}(n)}{\sum_{j=1}^{M} \tilde{w}^{(j)}(n)}$$

Computation of the moments of the filtering density

$$\boldsymbol{\mu}(n) = \sum_{m=1}^{M} w^{(m)}(n) \boldsymbol{x}_c^{(m)}(n)$$

$$\boldsymbol{\Sigma}(n) = \sum_{m=1}^{M} w^{(m)}(n) (\boldsymbol{x}_c^{(m)}(n) - \boldsymbol{\mu}(n)) (\boldsymbol{x}_c^{(m)}(n) - \boldsymbol{\mu}(n))^{\top}$$

Drawing particles for propagation at the next time instant

$$\boldsymbol{x}^{(m)}(n) \sim \mathcal{N}(\boldsymbol{\mu}(n), \boldsymbol{\Sigma}(n)), \quad m = 1, 2, \ldots, M.$$

In Figure 5.14, we can see one realization of the observation $y(n)$ generated with model parameters $\alpha = 0.9702$, $\beta = 0.5992$ and $\sigma^2 = 0.178$. The figure also shows the hidden process $x(n)$ and its MSE estimate obtained by SIR with $M = 2000$ particles. The estimates obtained by the other two filters are not shown because they basically yielded the same estimates.

In Figure 5.15 we can see the computed MSEs per sample in 50 different realizations when the number of particles was $M = 500$. The results suggest that the three methods have comparable performance.[11]

[11]The performances of the APF and GPF may be improved if the proposal distributions provide better particles. For an application of an APF with improved performance, see [15].

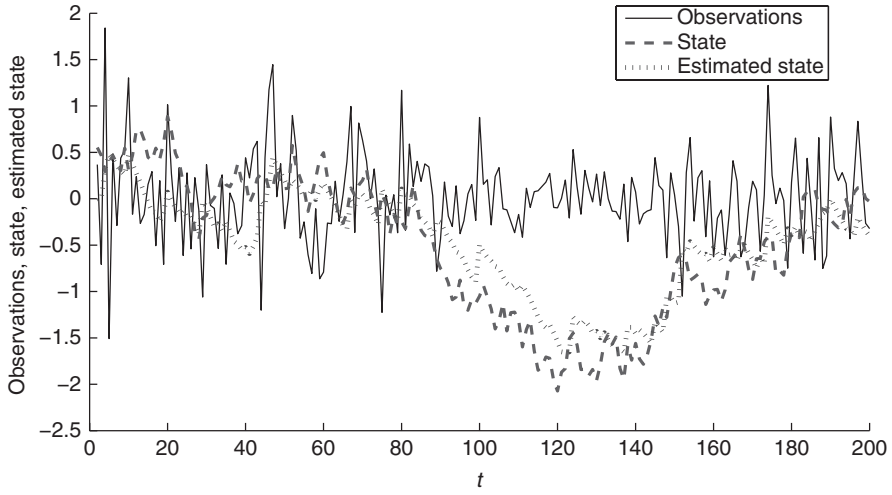

Figure 5.14 Stochastic volatility model: One realization of the observations, the state and its estimate.

One can expect that with the number of particles, the performance of the PFs improves. Indeed this is the case. In Figure 5.16, we can see the MSE per sample of the three filters as a function of the used number of particles. The number of generated realizations in the sample with a fixed number of particles was 1000. The results also show that for large enough number of particles the methods yield indistinguishable results.

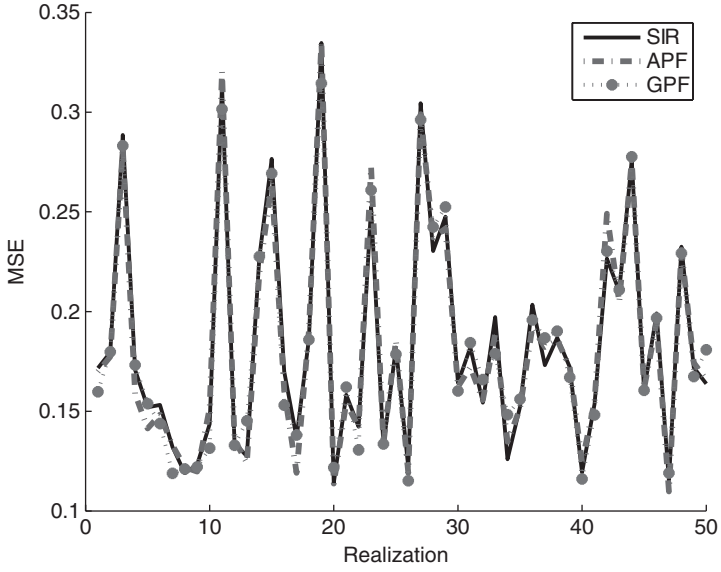

Figure 5.15 Average MSEs (obtained with $M = 500$ particles in 50 different realizations.

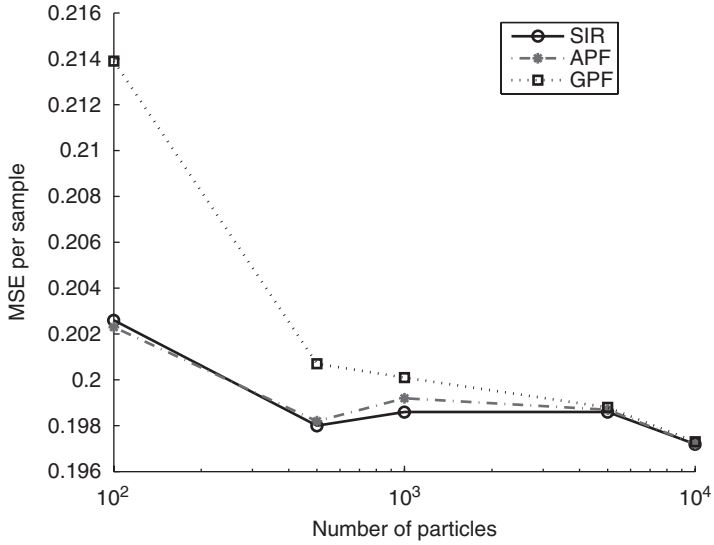

Figure 5.16 MSEs per sample as a function of the number of used particles.

5.6 HANDLING CONSTANT PARAMETERS

The particle filtering methodology was originally devised for the estimation of dynamic signals rather than static parameters. The most efficient way to address the problem is to integrate out the unknown parameters when possible, either analytically (see next section) [20] or by Monte Carlo procedures [72]. The former methods, however, depend on the feasibility of integration that is, on the mathematical model of the system. Generic solutions, useful for any model, are scant and limited in performance. A common feature of most of the approaches is that they introduce artificial evolution of the fixed parameters and thereby treat them in a similar way as the dynamic states of the model [31, 56]. Some methods insert the use of Markov chain Monte Carlo sampling to preserve diversity of the particles [10, 28]. A recent work [24] introduces a special class of PFs called density-assisted PFs that approximate the filtering density with a predefined parametric density by generalizing the concepts of Gaussian PFs and Gaussian sum PFs. These new filters can cope with constant parameters more naturally than previously proposed methods. In this section the problem of handling static parameters by PFs is reviewed under a kernel-based auxiliary PF method [56] and the density-assisted particle filtering technique [24].

We can reformulate the state-space model introduced in Section 1.2 to explicitly incorporate fixed parameters as

$$\boldsymbol{x}(n) = g_1(\boldsymbol{x}(n-1),\, \boldsymbol{\theta},\, \boldsymbol{v}_1(n))$$
$$\boldsymbol{y}(n) = g_2(\boldsymbol{x}(n),\, \boldsymbol{\theta},\, \boldsymbol{v}_2(n))$$

where all the symbols have the same meaning as before and $\boldsymbol{\theta}$ is a vector of fixed parameters. Based on the observations $\boldsymbol{y}(n)$ and the assumptions, the objective is to estimate $\boldsymbol{x}(n)$ and $\boldsymbol{\theta}$ recursively. In the particle filtering context, this amounts to obtaining the approximation of $f(\boldsymbol{x}(n), \boldsymbol{\theta} \,|\, \boldsymbol{y}(1:n))$ by updating the approximation for $f(\boldsymbol{x}(n-1), \boldsymbol{\theta} \,|\, \boldsymbol{y}(1:n-1))$.

5.6.1 Kernel-based Auxiliary Particle Filter

The inclusion of fixed parameters in the model implies extending the random measure to the form

$$\chi(n) = \{\boldsymbol{x}^{(m)}(n), \boldsymbol{\theta}^{(m)}(n), w^{(m)}(n)\}_{m=1}^{M}$$

where the index n in the samples of $\boldsymbol{\theta}$ indicates the approximation of the posterior at time n and *not* time-variation of the parameter vector. The random measure approximates the density of interest $f(\boldsymbol{x}(n), \boldsymbol{\theta} \,|\, \boldsymbol{y}(n))$, which can be decomposed as

$$f(\boldsymbol{x}(n), \boldsymbol{\theta} \,|\, \boldsymbol{y}(1:n)) \propto f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}(n), \boldsymbol{\theta}) f(\boldsymbol{x}(n) \,|\, \boldsymbol{\theta}, \boldsymbol{y}(1:n-1)) f(\boldsymbol{\theta} \,|\, \boldsymbol{y}(1:n-1)).$$

From the previous expression it is clear that there is a need for approximation of the density $f(\boldsymbol{\theta} \,|\, \boldsymbol{y}(1:n-1))$. In [56], a density of the form

$$f(\boldsymbol{\theta} \,|\, \boldsymbol{y}(1:n-1)) \approx \sum_{m=1}^{M} w^{(m)}(n) \mathcal{N}(\boldsymbol{\theta} \,|\, \bar{\boldsymbol{\theta}}^{(m)}(n), h^2 \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(n))$$

is used, and it represents a mixture of Gaussian distributions, where the mixands $\mathcal{N}(\boldsymbol{\theta} \,|\, \bar{\boldsymbol{\theta}}^{(m)}(n), h^2 \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(n))$ are weighted by the particle weights $w^{(m)}(n)$. The parameters of the mixands are obtained using the previous time instant particles and weights, that is

$$\bar{\boldsymbol{\theta}}^{(m)}(n) = \sqrt{1-h^2}\boldsymbol{\theta}^{(m)}(n-1) + (1-\sqrt{1-h^2}) \sum_{i=1}^{M} w^{(i)}(n-1)\boldsymbol{\theta}^{(i)}(n-1)$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(n) = \sum_{m=1}^{M} w^{(m)}(n)(\boldsymbol{\theta}^{(m)}(n) - \bar{\boldsymbol{\theta}}(n))(\boldsymbol{\theta}^{(m)}(n) - \bar{\boldsymbol{\theta}}(n))^{\top}$$

where h^2 is a smoothing parameter computed by $h^2 = 1 - \left(\dfrac{3\gamma - 1}{2\gamma}\right)^2$, and $\gamma \in (0,1)$ represents a discount factor, typically around 0.95–0.99.

We now describe the implementation of this idea in the context of auxiliary particle filtering. Assume that at time instant $n-1$, we have the random measure $\chi(n-1) = \{\boldsymbol{x}^{(m)}(n-1), \boldsymbol{\theta}^{(m)}(n-1), w^{(m)}(n-1)\}$. Then, we proceed as follows.

1. Estimate the next particle by $\hat{\boldsymbol{x}}^{(m)}(n) = E(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(n-1), \boldsymbol{\theta}^{(m)}(n-1))$. This step is identical to the APF.

2. Sample the indexes k_m of the streams that survive, where $k_m = i$ with probability $w_a^{(i)} \propto w^{(i)}(n-1) f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(i)}(n),\, \bar{\boldsymbol{\theta}}^{(i)}(n-1))$. Here we basically resample so that the most promising streams are kept in the random measure and the less promising are removed.
3. Draw particles of the fixed parameter vector according to

$$\boldsymbol{\theta}^{(m)}(n) \sim \mathcal{N}(\bar{\boldsymbol{\theta}}^{(k_m)}(n-1),\, h^2 \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(n-1))$$

   where the means and the covariance of the mixands are obtained at the end of the cycle of computations for time instant $n-1$. This step takes care of the vector of constant parameters.
4. Draw particles of the state according to

$$\boldsymbol{x}^{(m)}(n) \sim f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(k_m)}(n-1),\, \boldsymbol{\theta}^{(m)}(n)).$$

   With this step we complete the drawing of new particles.
5. Update and normalize the weights, where

$$w^{(m)}(n) \propto \frac{f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}^{(m)}(n),\, \boldsymbol{\theta}^{(m)}(n))}{f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(k_m)}(n),\, \bar{\boldsymbol{\theta}}^{(k_m)}(n-1))}.$$

   With this step we associate weights to the particles.
6. Compute the parameters of the kernels used in constructing the mixture Gaussian according to

$$\bar{\boldsymbol{\theta}}^{(m)}(n) = \sqrt{1-h^2}\,\boldsymbol{\theta}^{(m)}(n) + (1-\sqrt{1-h^2}) \sum_{i=1}^{M} w^{(i)}(n) \boldsymbol{\theta}^{(i)}(n)$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(n) = \sum_{m=1}^{M} w^{(m)}(n) (\boldsymbol{\theta}^{(m)}(n) - \bar{\boldsymbol{\theta}}(n))(\boldsymbol{\theta}^{(m)}(n) - \bar{\boldsymbol{\theta}}(n))^{\top}.$$

■ EXAMPLE 5.8

We go back to the problem described in Example 5.7 where we want to estimate the hidden stochastic volatility. Suppose that in addition to the unknown $\boldsymbol{x}(n)$ we also do not know the parameters $\boldsymbol{\theta} = [\alpha\ \beta\ \sigma^2]^{\top}$. If we were to apply the kernel-based APF, we would first generate particles of $\boldsymbol{x}^{(m)}(0)$, and $\boldsymbol{\theta}^{(m)}(0)$ from the priors and implement the method shown in Table 5.5. As a first step in the recursion, we estimate $\hat{\boldsymbol{x}}^{(m)}(n),\ m = 1, 2, \cdots, M$ by

$$\hat{\boldsymbol{x}}^{(m)}(n) = \alpha^{(m)}(n-1) \boldsymbol{x}^{(m)}(n-1).$$

Then we obtain the weights $w_a^{(i)} \propto w^{(i)}(n-1) f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(i)}(n),\, \bar{\boldsymbol{\theta}}^{(i)}(n-1))$, where

$$f(\boldsymbol{y}(n) \,|\, \hat{\boldsymbol{x}}^{(i)}(n),\, \bar{\boldsymbol{\theta}}^{(i)}(n-1)) = \mathcal{N}(0,\, \bar{\beta}^{2^{(i)}}(n-1) e^{\hat{\boldsymbol{x}}^{(i)}(n)})$$

and sample the indexes k_m according the computed weights. We follow with generating the new particles of the constant parameters

$$\boldsymbol{\theta}^{(m)}(n) \sim \mathcal{N}(\bar{\boldsymbol{\theta}}^{(k_m)}(n-1),\, h^2\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(n-1))$$

and the unknown states

$$\boldsymbol{x}^{(m)}(n) \sim \mathcal{N}(\alpha^{(k_m)}(n)\boldsymbol{x}^{(k_m)}(n-1),\, \sigma^{2^{(k_m)}}(n)).$$

The weights are computed by

$$w^{(m)}(n) \propto \frac{\mathcal{N}(0,\, \beta^{2^{(m)}}(n)e^{\boldsymbol{x}^{(m)}(n)})}{\mathcal{N}(0,\, \bar{\beta}^{2^{(k_m)}}(n-1)e^{\hat{\boldsymbol{x}}^{(k_m)}(n)})}.$$

Finally the kernel parameters $\bar{\boldsymbol{\theta}}^{(m)}(n)$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(n)$ are found as shown in Table 5.5.

■ **EXAMPLE 5.9**

Consider as in Example 5.1 the problem of tracking one target using two static sensors which collected bearings-only measurements. We extend the problem to the case where the measurements are biased. Then, the mathematical formulation of the problem is given by

$$\begin{aligned}\boldsymbol{x}(n) &= \boldsymbol{A}\boldsymbol{x}(n-1) + \boldsymbol{B}\boldsymbol{v}_1(n)\\ \boldsymbol{y}(n) &= g_2(\boldsymbol{x}(n)) + \boldsymbol{\theta} + \boldsymbol{v}_2(n)\end{aligned}$$

where all the parameters have the same meaning as before and $\boldsymbol{\theta} = [b_1\ b_2]^\top$ represents a vector of unknown constant biases.

In step 1, we generate $\hat{\boldsymbol{x}}^{(m)}(n)$ by

$$\hat{\boldsymbol{x}}^{(m)}(n) = \boldsymbol{A}\boldsymbol{x}^{(m)}(n).$$

The resampling is carried out by using weights that are proportional to $w^{(i)}(n-1)\mathcal{N}(g_2(\hat{\boldsymbol{x}}^{(i)}(n)) + \bar{\boldsymbol{\theta}}^{(i)}(n-1), \boldsymbol{\Sigma}_v)$, where the Gaussian is computed at $\boldsymbol{y}(n)$. The remaining steps can readily be deduced from the scheme described by Table 5.5.

5.6.2 Density-assisted Particle Filter

As seen in the previous section, the GPFs approximate the predictive and filtering densities by Gaussian densities whose parameters are estimated from the particles and their weights. Similarly, the Gaussian sum PFs [51] approximate these densities with mixtures of Gaussians. The approximating densities can be other than Gaussians or mixtures of Gaussian, and therefore, we refer to the general class of filters

of this type as density-assisted PFs (DAPFs) [24], which are a generalization of the Gaussian and Gaussian sum PFs. The main advantages of DAPFs is that they do not necessarily use resampling in the sense carried out by standard PFs and they do not share the limitation regarding the estimation of constant model parameters. Here we explain how we can use DAPFs when we have constant parameters in the model.

Let $\chi(n-1) = \{\boldsymbol{x}^{(m)}(n-1), \boldsymbol{\theta}^{(m)}(n-1), w^{(m)}(n-1)\}_{m=1}^{M}$ be the random measure at time instant $n-1$, $\boldsymbol{x}^{(m)}(n-1)$ and $\boldsymbol{\theta}^{(m)}(n-1)$ the particles of $\boldsymbol{x}(n-1)$ and $\boldsymbol{\theta}$, respectively, and $w^{(m)}(n-1)$ their associated weights. If we denote the approximating density of $f(\boldsymbol{x}(n-1), \boldsymbol{\theta} \,|\, \boldsymbol{y}(1:n-1))$ by $\pi(\boldsymbol{\phi}(n-1))$, where $\boldsymbol{\phi}(n-1)$ are the parameters of the approximating density, the steps of the density assisted particle filtering are the following.

1. Draw particles according to

$$\{\boldsymbol{x}^{(m)}(n-1), \boldsymbol{\theta}^{(m)}(n-1)\} \sim \pi(\boldsymbol{\phi}(n-1)).$$

2. Draw particles according to

$$\boldsymbol{x}^{(m)}(n) \sim f(\boldsymbol{x}(n) \,|\, \boldsymbol{x}^{(m)}(n-1), \boldsymbol{\theta}^{(m)}(n-1)).$$

3. Set $\boldsymbol{\theta}^{(m)}(n) = \boldsymbol{\theta}^{(m)}(n-1)$.
4. Update and normalize the weights

$$w^{(m)}(n) \propto f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}^{(m)}(n), \boldsymbol{\theta}^{(m)}(n)).$$

5. Estimate the parameters, $\phi(n)$, of the density from

$$\chi(n) = \{\boldsymbol{x}^{(m)}(n), \boldsymbol{\theta}^{(m)}(n), w^{(m)}(n)\}_{m=1}^{M}.$$

The problem of standard PFs regarding constant parameters is avoided in the first step by drawing particles of the constants from an approximation of the posterior. It is important to note that one can combine standard PFs and DAPFs, in that we apply the standard PFs for the dynamic variables and DAPFs for the constant parameters. We show this in the next example.

■ EXAMPLE 5.10

Again we have the problem from Example 5.9, where we track a target in a two-dimensional plane. We assume that the marginal posterior density of the bias vector $\boldsymbol{\theta}$ is a Gaussian density. Suppose that at time instant $n-1$, we have the random measure $\chi(n-1) = \{\boldsymbol{x}^{(m)}(n-1), \boldsymbol{\theta}^{(m)}(n-1), w^{(m)}(n-1)\}_{m=1}^{M}$. From the random measure, we can compute the parameters of the approximating

Gaussian of the bias vector by

$$\boldsymbol{\mu}(n-1) = \sum_{m=1}^{M} w^{(m)}(n-1)\boldsymbol{\theta}^{(m)}(n-1)$$

$$\boldsymbol{\Sigma}(n-1) = \sum_{m=1}^{M} w^{(m)}(n-1)(\boldsymbol{\theta}^{(m)}(n-1) - \boldsymbol{\mu}(n-1))(\boldsymbol{\theta}^{(m)}(n-1) - \boldsymbol{\mu}(n-1))^{\top}.$$

Then we draw the particles of $\boldsymbol{\theta}$ for the next time step, that is, $\boldsymbol{\theta}^{(m)}(n) \sim \mathcal{N}(\boldsymbol{\mu}(n-1), \boldsymbol{\Sigma}(n-1))$. Once we have the particles of the biases, we proceed by using the favorite PF. For example, if it is the APF, first we project the dynamic particles as in the previous example, that is

$$\hat{\boldsymbol{x}}^{(m)}(n) = \hat{\boldsymbol{A}}\boldsymbol{x}^{(m)}(n).$$

This is followed by resampling of the streams whose weights are given by $w^{(k_m)}(n-1)\mathcal{N}(g_2(\hat{\boldsymbol{x}}^{(k_m)}(n)) + \boldsymbol{\theta}^{(k_m)}(n), \boldsymbol{\Sigma}_v)$, and where the Gaussian is computed at $\boldsymbol{y}(n)$. Once the indexes of the streams for propagation are known, we draw the particles of the dynamic variables, $\boldsymbol{x}^{(m)}(n)$. Next, the new weights of the streams are computed by

$$w^{(m)}(n) \propto \frac{f(\boldsymbol{y}(n) \mid \boldsymbol{x}^{(m)}(n),\ \boldsymbol{\theta}^{(k_m)}(n))}{f(\boldsymbol{y}(n) \mid \hat{\boldsymbol{x}}^{(k_m)}(n),\ \boldsymbol{\theta}^{(k_m)}(n))}.$$

5.7 RAO–BLACKWELLIZATION

In many practical problems, the considered dynamic nonlinear system may have some states that are conditionally linear given the nonlinear states of the system. When the applied methodology is particle filtering, this conditional linearity can be exploited using the concept of Rao–Blackwellization [20, 25]. Rao–Blackwellization is a statistical procedure that is used for reducing variance of estimates obtained by Monte Carlo sampling methods [18], and by employing it, we can have improved filtering of the unknown states.

The main idea consists of tracking the linear states differently from the nonlinear states by treating the linear parameters as nuisance parameters and marginalizing them out of the estimation problem. This strategy allows for more accurate estimates of the unknowns because the dimension of the space that is explored with particles is reduced and therefore it is much better searched. At every time instant the particles of the nonlinear states are propagated randomly, and once they are known, the problem is linear in the rest of the states. Therefore, one can find their 'optimal' values by employing Kalman filtering and associate them with the sampled nonlinear states. Some recent applications of Rao–Blackwellized PFs include tracking of maneuvering

targets in clutter [61] and joint target tracking using kinematic radar information [4]. In [44], a computational complexity analysis of Rao–Blackwellized PFs is provided.

For the scenario of a nonlinear system with conditionally linear states, we write the state space model as

$$\begin{aligned}\boldsymbol{x}_n(n) &= g_{1,n}(\boldsymbol{x}_n(n-1)) + \boldsymbol{A}_{1,n}(\boldsymbol{x}_n(n-1))\boldsymbol{x}_l(n-1) + \boldsymbol{v}_{1,n}(n)\\ \boldsymbol{x}_l(n) &= g_{1,l}(\boldsymbol{x}_n(n-1)) + \boldsymbol{A}_{1,l}(\boldsymbol{x}_n(n-1))\boldsymbol{x}_l(n-1) + \boldsymbol{v}_{1,l}(n)\\ \boldsymbol{y}_t &= g_2(\boldsymbol{x}_n(n)) + \boldsymbol{A}_2(\boldsymbol{x}_n(n))\boldsymbol{x}_l(n) + \boldsymbol{v}_2(n)\end{aligned}$$

where the system state, $\boldsymbol{x}(n)$, includes nonlinear and conditionally linear components, that is, $\boldsymbol{x}^\top(n) = [\boldsymbol{x}_n^\top(n)\,\boldsymbol{x}_l^\top(n)]$, with $\boldsymbol{x}_n(n)$ and $\boldsymbol{x}_1(n)$ being the nonlinear and conditionally linear states, respectively; $\boldsymbol{v}_{1,n}(n)$ and $\boldsymbol{v}_{1,l}(n)$ are state noise vectors at time instant n which are assumed to be Gaussian; $g_{1,n}(\cdot)$ and $g_{1,l}(\cdot)$ are nonlinear state transition functions; $\boldsymbol{A}_{1,n}$ and $\boldsymbol{A}_{1,l}$ are matrices whose entries may be functions of the nonlinear states; $g_2(\cdot)$ is a nonlinear measurement function of the nonlinear states; $\boldsymbol{A}_2$ is another matrix whose entries may be functions of the nonlinear states; and $\boldsymbol{v}_2(n)$ is observation noise vector at time n which is also assumed to be Gaussian and independent from the state noises.

Suppose that at time instant $n-1$, the random measure composed of M streams is given by

$$\chi(n-1) = \{\boldsymbol{x}_n^{(m)}(n-1),\, w^{(m)}(n-1)\}_{m=1}^{M}$$

and that the linear state $\boldsymbol{x}_l(n-1)$ in the m-th stream is Gaussian distributed, that is

$$\boldsymbol{x}_l(n-1) \sim \mathcal{N}(\hat{\boldsymbol{x}}_l^{(m)}(n-1),\, \hat{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n-1))$$

where $\hat{\boldsymbol{x}}_l^{(m)}(n-1)$ is the estimate of $\boldsymbol{x}_l$ in the m-th stream at time instant $n-1$ and $\hat{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n-1)$ is the covariance matrix of that estimate. The scheme runs as follows.

1. Generate the nonlinear state particles according to the marginalized prior PDF

$$\boldsymbol{x}_n^{(m)}(n) \sim f(\boldsymbol{x}_n(n) \,|\, \boldsymbol{x}_n^{(m)}(n-1), \boldsymbol{y}(1:n-1))$$

where

$$\begin{aligned}f(\boldsymbol{x}_n(n) \,|\, \boldsymbol{x}_n^{(m)}(n-1), \boldsymbol{y}(1:n-1)) = \int & f(\boldsymbol{x}_n(n) \,|\, \boldsymbol{x}_n^{(m)}(n-1), \boldsymbol{x}_l^{(m)}(n-1))\\ \times\; & f(\boldsymbol{x}_l^{(m)}(n-1) \,|\, \boldsymbol{x}_n^{(m)}(n-1), \boldsymbol{y}(1:n-1))\, d\boldsymbol{x}_l^{(m)}(n-1).\end{aligned}$$

If the distributions inside the integral are Gaussians, $f(\boldsymbol{x}_n(n)\,|\,\boldsymbol{x}_n^{(m)}(n-1), \boldsymbol{y}(1:n-1))$ is also a Gaussian distribution. It can be shown that

$$f(\boldsymbol{x}_n(n)\,|\,\boldsymbol{x}_n^{(m)}(n-1), \boldsymbol{y}(1:n-1)) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}_n}^{(m)}(n), \boldsymbol{\Sigma}_{\boldsymbol{x}_n}^{(m)}(n))$$

where

$$\boldsymbol{\mu}_{\boldsymbol{x}_n}^{(m)}(n) = g_{1,n}(\boldsymbol{x}_n^{(m)}(n-1)) + \boldsymbol{A}_{1,n}^{(m)}(n-1)\hat{\boldsymbol{x}}_l^{(m)}(n-1) \tag{5.31}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{x}_n}^{(m)}(n) = \boldsymbol{A}_{1,n}^{(m)}(n-1)\hat{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n-1)\boldsymbol{A}_{1,n}^{(m)^\top}(n-1) + \boldsymbol{C}_{\boldsymbol{v}1,n} \tag{5.32}$$

where we have dropped in the notation that the matrix $\boldsymbol{A}_{1,n}^{(m)}(n-1)$ may be a function of $\boldsymbol{x}_n^{(m)}(n-1)$. The symbol $\hat{\boldsymbol{x}}_l^{(m)}(n-1)$ is the estimate of the linear state at time instant $n-1$ from the m-th stream, and $\hat{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n-1)$ is the covariance matrix of that estimate.

2. Update the linear states with quasi measurements. We define the quasi measurements by

$$\begin{aligned}\boldsymbol{z}^{(m)}(n) &= \boldsymbol{x}_n^{(m)}(n) - g_{1,n}\left(\boldsymbol{x}_n^{(m)}(n-1)\right)\\ &= \boldsymbol{A}_{1,n}^{(m)}(n-1)\hat{\boldsymbol{x}}_l^{(m)}(n-1) + \boldsymbol{v}_{1,n}(n)\end{aligned}$$

and use them to improve the estimate $\hat{\boldsymbol{x}}_l^{(m)}(n-1)$. The new estimate $\hat{\hat{\boldsymbol{x}}}_l^{(m)}(n-1)$ is obtained by applying the measurement step of the KF, that is,

$$\begin{aligned}\hat{\hat{\boldsymbol{x}}}_l^{(m)}(n-1) &= \hat{\boldsymbol{x}}_l^{(m)}(n-1) + \boldsymbol{L}^{(m)}(n-1)(\boldsymbol{z}^{(m)}(n) - \boldsymbol{A}_{1,n}^{(m)}(n-1)\hat{\boldsymbol{x}}_l^{(m)}(n-1))\\ \boldsymbol{L}^{(m)}(n-1) &= \hat{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n-1)\\ &\quad\times(\boldsymbol{A}_{1,n}^{(m)}(n-1)\boldsymbol{A}_{1,n}^{(m)}(n-1)\hat{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n-1)\boldsymbol{A}_{1,n}^{(m)^\top}(n-1) + \boldsymbol{C}_{\boldsymbol{v}_{1,n}})^{-1}\\ \hat{\hat{\boldsymbol{C}}}_{\boldsymbol{x}_l}^{(m)}(n-1) &= (\boldsymbol{I} - \boldsymbol{L}^{(m)}(n-1)\boldsymbol{A}_{1,n}^{(m)}(n-1))\,\hat{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n-1).\end{aligned}$$

3. Perform time update of the linear states according to

$$\begin{aligned}\tilde{\boldsymbol{x}}_l^{(m)}(n) &= g_{1,l}(\boldsymbol{x}_n^{(m)}(n-1)) + \boldsymbol{A}_{1,l}^{(m)}(n-1)\hat{\hat{\boldsymbol{x}}}_l^{(m)}(n-1)\\ \tilde{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n) &= \boldsymbol{A}_{1,l}^{(m)}(n-1)\hat{\hat{\boldsymbol{C}}}_{\boldsymbol{x}_l}^{(m)}(n-1)\boldsymbol{A}_{1,l}^{(m)^\top}(n-1) + \boldsymbol{C}_{\boldsymbol{v}_{1,l}}.\end{aligned}$$

4. Compute the weights by

$$w^{(m)}(n) \propto w^{(m)}(n-1)\, f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}_n^{(m)}(0:n), \boldsymbol{y}(1:n-1))$$

where

$$f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}_n^{(m)}(0:n), \boldsymbol{y}(1:n-1)) = \int f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}_n^{(m)}(n), \boldsymbol{x}_l^{(m)}(n)) \times f(\boldsymbol{x}_l^{(m)}(n) \,|\, \boldsymbol{x}_n^{(m)}(0:n), \boldsymbol{y}(1:n-1)) d\boldsymbol{x}_l^{(m)}(n).$$

Again, if the two densities of $\boldsymbol{x}_l^{(m)}(n)$ inside the integral are Gaussian densities, the integral can be solved analytically. We obtain

$$f(\boldsymbol{y}(n) \,|\, \boldsymbol{x}_n^{(m)}(0:n), \boldsymbol{y}(1:n-1)) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{y}}^{(m)}(n), \boldsymbol{\Sigma}_{\boldsymbol{y}}^{(m)}(n))$$

where

$$\boldsymbol{\mu}_{\boldsymbol{y}}^{(m)}(n) = g_2(\boldsymbol{x}_n^{(m)}(n)) + \boldsymbol{A}_2^{(m)}(n)\tilde{\boldsymbol{x}}_l^{(m)}(n)$$
$$\boldsymbol{\Sigma}_{\boldsymbol{y}}^{(m)}(n) = \boldsymbol{A}_2^{(m)}(n)\tilde{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n)\boldsymbol{A}_2^{(m)^\top}(n) + \boldsymbol{C}_{\boldsymbol{v}_2}.$$

5. Finally we carry out the measurement update of the linear states

$$\hat{\boldsymbol{x}}_l^{(m)}(n) = \tilde{\boldsymbol{x}}_l^{(m)}(n) + \boldsymbol{K}^{(m)}(n)(\boldsymbol{y}(n) - g_2(\boldsymbol{x}_n^{(m)}(n)) - \boldsymbol{A}_2^{(m)}(n)\tilde{\boldsymbol{x}}_l^{(m)}(n))$$
$$\boldsymbol{K}^{(m)}(n) = \tilde{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n)\boldsymbol{A}_2^{(m)}(n)(\boldsymbol{A}_2^{(m)}(n)\tilde{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n)\boldsymbol{A}_2^{(m)^\top}(n) + \boldsymbol{C}_{\boldsymbol{v}_2})^{-1}$$
$$\hat{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n) = (\boldsymbol{I} - \boldsymbol{K}^{(m)}(n)\boldsymbol{A}_2^{(m)}(n))\tilde{\boldsymbol{C}}_{\boldsymbol{x}_l}^{(m)}(n).$$

In general, the Rao–Blackwellization amounts to the use of a bank of KFs, one for each particle stream. So, with M particle streams, we will have M KFs.

■ EXAMPLE 5.11

The problem of target tracking using biased measurements presented in Example 5.9 can be addressed using a Rao–Blackwellized scheme. Here we outline the steps of the scheme. First, we assume that at time instant $n-1$ we have the random measure $\chi(n-1) = \{\boldsymbol{x}^{(m)}(n-1), w^{(m)}(n-1)\}_{m=1}^{M}$ and the estimates of the bias in each particle stream $\hat{\boldsymbol{\theta}}^{(m)}(n-1)$ and the covariances of the estimates $\hat{\boldsymbol{C}}_{\boldsymbol{\theta}}^{(m)}(n-1)$.

At time instant n, we first generate the particles of the dynamic variables

$$\boldsymbol{x}^{(m)}(n) \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{x}^{(m)}(n-1), \boldsymbol{B}\boldsymbol{C}_{\boldsymbol{v}_1}\boldsymbol{B}^\top).$$

The second and the third steps are skipped because $\boldsymbol{\theta}$ only appears in the measurement equation. The computation of the weights proceeds by

$$w^{(m)}(n) \propto w^{(m)}(n-1)\, \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{y}}^{(m)}(n), \boldsymbol{\Sigma}_{\boldsymbol{y}}^{(m)}(n))$$

where

$$\boldsymbol{\mu}_{\boldsymbol{y}}^{(m)}(n) = g_2(\boldsymbol{x}^{(m)}(n)) + \hat{\boldsymbol{\theta}}^{(m)}(n)$$
$$\boldsymbol{\Sigma}_{\boldsymbol{y}}^{(m)}(n) = \hat{\boldsymbol{C}}_{\boldsymbol{\theta}}^{(m)}(n-1) + \boldsymbol{C}_{v_2}.$$

In the last step we update the estimates of the biases and their covariances by

$$\hat{\boldsymbol{\theta}}^{(m)}(n) = \hat{\boldsymbol{\theta}}^{(m)}(n-1) + \boldsymbol{K}^{(m)}(n)(\boldsymbol{y}(n) - g_2(\boldsymbol{x}^{(m)}(n)) - \hat{\boldsymbol{\theta}}^{(m)}(n-1))$$
$$\boldsymbol{K}^{(m)}(n) = \hat{\boldsymbol{C}}_{\boldsymbol{\theta}}^{(m)}(n-1)(\hat{\boldsymbol{C}}_{\boldsymbol{\theta}}^{(m)}(n-1) + \boldsymbol{C}_{v_2})^{-1}$$
$$\hat{\boldsymbol{C}}_{\boldsymbol{\theta}}^{(m)}(n) = (\boldsymbol{I} - \boldsymbol{K}^{(m)}(n))\hat{\boldsymbol{C}}_{\boldsymbol{\theta}}^{(m)}(n).$$

5.8 PREDICTION

Recall that the prediction problem revolves around the estimation of the predictive density $f(\boldsymbol{x}(n+k)\ \boldsymbol{y}(1:n))$, where $k > 0$. The prediction of the state is important in many applications. One of them is for model selection, where from a set of models $\mathcal{M}_l$, $l = 1, 2, \ldots, L$, one has to choose the best model according to a given criterion, for example the one based on MAP [9]. Then, one has to work with predictive probability distributions of the form $f(\boldsymbol{y}(n+1)\ \boldsymbol{y}(n), \mathcal{M}_l)$, where

$$f(\boldsymbol{y}(n+1) \,|\, \boldsymbol{y}(1:n), \mathcal{M}_l) = \int f(\boldsymbol{y}(n+1) \,|\, \boldsymbol{x}(n+1), \mathcal{M}_l) \times f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{y}(1:n), \mathcal{M}_l)\, d\boldsymbol{x}(n+1)$$

where the second factor of the integrand is the predictive density of the state. The above integral often cannot be solved analytically, which is why we have interest in the prediction problem.

First we address the case $k = 1$, that is, the approximation of the predictive density $f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{y}(1:n))$. Theoretically, we can obtain it from the filtering PDF $f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{y}(1:n))$ by using the following integral

$$f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{y}(1:n)) = \int f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{x}(n)) f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))\, d\boldsymbol{x}(n). \tag{5.33}$$

Note that we already saw this expression in (5.7). Recall also that the PF provides an estimate of the filtering density in the form

$$f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n)) \simeq \sum_{m=1}^{M} w^{(m)}(n)\ \delta(\boldsymbol{x}(n) - \boldsymbol{x}^{(m)}(n)) \tag{5.34}$$

which when used in (5.33) yields an estimate of the predictive density in the form of a mixture density given by

$$f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{y}(1:n)) \simeq \sum_{m=1}^{M} w^{(m)}(n)\, f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{x}^{(m)}(n)). \tag{5.35}$$

If we express this PDF by using a discrete random measure, then a natural way of expressing this measure is by $\{\boldsymbol{x}^{(m)}(n+1),\, w^{(m)}(n)\}_{m=1}^{M}$, where

$$\boldsymbol{x}^{(m)}(n+1) \sim f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{x}^{(m)}(n))$$

and

$$f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{y}(1:n)) \simeq \sum_{m=1}^{M} w^{(m)}(n)\ \delta(\boldsymbol{x}(n+1) - \boldsymbol{x}^{(m)}(n+1)).$$

In summary, we obtain an estimate of the predictive PDF $f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{y}(1:n))$ from the random measure $\chi(n) = \{\boldsymbol{x}^{(m)}(n),\, w^{(m)}(n)\}_{m=1}^{M}$ by simply (a) drawing particles $\boldsymbol{x}^{(m)}(n+1)$ from $f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{x}^{(m)}(n))$, for $m = 1, 2, \ldots, M$ and (b) assigning the weights $w^{(m)}(n)$ to the drawn particles $\boldsymbol{x}^{(m)}(n+1)$.[12]

Next we do the case $k = 2$. The strategy for creating the random measure that approximates $f(\boldsymbol{x}(n+2) \,|\, \boldsymbol{y}(1:n))$ is analogous. Now

$$\begin{aligned} f(\boldsymbol{x}(n+2) \,|\, \boldsymbol{y}(1:n)) = \int\int & f(\boldsymbol{x}(n+2) \,|\, \boldsymbol{x}(n+1))\, f(\boldsymbol{x}(n+1) \,|\, \boldsymbol{x}(n)) \\ & \times f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))\, d\boldsymbol{x}(n)\, d\boldsymbol{x}(n+1). \end{aligned} \tag{5.36}$$

Here we have two integrals, where the first involves the integration of $\boldsymbol{x}(n)$ carried out in the same way as above. We reiterate that we can approximate $f(\boldsymbol{x}(n) \,|\, \boldsymbol{y}(1:n))$ as in

[12] Another approximation would be by drawing the particles from the mixture in (5.35), that is, by first sampling the mixand and then drawing the particle from it. In that case all the drawn particles have the same weight.

(5.34), and after the integration we obtain

$$\begin{aligned} f(\boldsymbol{x}(n+2)\,|\,\boldsymbol{y}(1:n)) &= \int f(\boldsymbol{x}(n+2)\,|\,\boldsymbol{x}(n+1)) f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:n))\,d\boldsymbol{x}(n+1) \\ &\simeq \int f(\boldsymbol{x}(n+2)\,|\,\boldsymbol{x}(n+1)) \sum_{m=1}^{M} w^{(m)}(n)\delta(\boldsymbol{x}(n+1) \\ &\quad - \boldsymbol{x}^{(m)}(n+1))\,d\boldsymbol{x}\,(n+1) \\ &= \sum_{m=1}^{M} w^{(m)}(n) f(\boldsymbol{x}(n+2)\,|\,\boldsymbol{x}^{(m)}(n+1)). \end{aligned}$$

Further, we approximate this PDF by

$$f(\boldsymbol{x}(n+2)\,|\,\boldsymbol{y}(1:n)) \simeq \sum_{m=1}^{M} w^{(m)}(n)\delta(\boldsymbol{x}(n+2) - \boldsymbol{x}^{(m)}(n+2))$$

where as before we obtain $\boldsymbol{x}^{(m)}(n+2)$ by drawing it from $f(\boldsymbol{x}(n+2)\,|\,\boldsymbol{x}^{(m)}(n+1))$.

Thus, the approximation of the predictive density $f(\boldsymbol{x}(n+2)\,|\,\boldsymbol{y}(1:n))$ is obtained by

1. drawing particles $\boldsymbol{x}^{(m)}(n+1)$ from $f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{x}^{(m)}(n))$
2. generating $\boldsymbol{x}^{(m)}(n+2)$ from $f(\boldsymbol{x}(n+2)\,|\,\boldsymbol{x}^{(m)}\,(n+1))$; and
3. associating with the samples $\boldsymbol{x}^{(m)}(n+2)$ the weights $w^{(m)}(n)$ and thereby forming the random measure $\{\boldsymbol{x}^{(m)}(n+2), w^{(m)}(n)\}_{m=1}^{M}$.

At this point, it is not difficult to generalize the procedure for obtaining the approximation of $f(\boldsymbol{x}(n+k)\,|\,\boldsymbol{y}(1:n))$ for any $k > 0$. We propagate the particle streams from time instant n to time instant $n+k$ by using the transition PDFs $f(\boldsymbol{x}(n+i)\,|\,\boldsymbol{x}(n+i-1))$, $i = 1, 2, \ldots, k$, and associating with the last set of generated particles $\boldsymbol{x}^{(m)}(n+k)$ the weights $w^{(m)}(n)$ to obtain the random measure $\{\boldsymbol{x}^{(m)}(n+k), w^{(m)}(n)\}_{m=1}^{M}$.

5.9 SMOOTHING

In Section 5.2, we pointed out that sometimes we may prefer to estimate $\boldsymbol{x}(n)$ based on data $\boldsymbol{y}(1:n+k)$, where $k > 0$. All the information about $\boldsymbol{x}(n)$ in that case is in the smoothing PDF $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(n+k))$. With particle filtering, we approximate this density with a random measure in a similar way as we approximated filtering PDFs.

When we deal with the problem of smoothing, we discriminate between two types of smoothing. One of them is called fixed-lag smoothing and the interest there is in the PDF $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n+k))$, where k is a fixed lag. The other is called fixed interval smoothing, where the objective is to find the PDF $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:N))$ for any n where $0 \leq n < N$.

A naive approach to these two problems would be to conduct particle filtering in the usual way and store all the random measures, and for obtaining approximations of the smoothing densities use the set of particles at time instant n and the weights of these streams at time instants $n+k$ or N, respectively. In other words, we use

$$f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n+k)) \simeq \sum_{m=1}^{M} w^{(m)}(n+k)\delta(\boldsymbol{x}(n)-\boldsymbol{x}^{(m)}(n))$$

and

$$f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:N)) \simeq \sum_{m=1}^{M} w^{(m)}(N)\delta(\boldsymbol{x}(n)-\boldsymbol{x}^{(m)}(n)).$$

These approximations may be good if k or $N-n$ are small numbers. For even moderate values of k or $N-n$, the approximations may become quite inaccurate, especially when resampling is implemented at each time instant. Namely, with resampling, we deplete the number of surviving streams k or $N-n$ lags later, which may make the representation at time instant n rather poor.

In the sequel, we show two approaches for improved smoothing. They are both based on backward smooth recursions, where the first recursion uses only the set of weights generated during the forward filtering pass [52] and the second one generates an additional set of weights during the backward filtering pass [25]. Here we address the fixed-interval smoothing problem because modifications for fixed-lag smoothing are straightforward.

First we write the joint a posteriori PDF of the states as follows

$$f(\boldsymbol{x}(0:N)\,|\,\boldsymbol{y}(1:N)) = f(\boldsymbol{x}(N)\,|\,\boldsymbol{y}(1:N))\prod_{n=0}^{N-1} f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n+1),\boldsymbol{y}(1:N)).$$

We note that

$$\begin{aligned} f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n+1),\boldsymbol{y}(1:N)) &= f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n+1),\boldsymbol{y}(1:n)) \\ &\propto f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{x}(n))f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n)). \end{aligned}$$

From the above two expressions, we obtain the recursion that takes us from the smoothing PDF $f(\boldsymbol{x}(n+1:N)\,|\,\boldsymbol{y}(1:N))$ to $f(\boldsymbol{x}(n:N)\,|\,\boldsymbol{y}(1:N))$, that is,

$$\begin{aligned} f(\boldsymbol{x}(n:N)\,|\,\boldsymbol{y}(1:N)) &= f(\boldsymbol{x}(n+1:N)\,|\,\boldsymbol{y}(1:N))f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n+1),\boldsymbol{y}(1:N)) \\ &= f(\boldsymbol{x}(n+1:N)\,|\,\boldsymbol{y}(1:N))f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n+1),\boldsymbol{y}(1:n)) \\ &\propto f(\boldsymbol{x}(n+1:N)\,|\,\boldsymbol{y}(1:N))f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{x}(n))f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n)). \end{aligned} \tag{5.37}$$

If we approximate $f(\boldsymbol{x}(n+1:N)\,|\,\boldsymbol{y}(1:N))$ by the smoothing random measure $\chi_s(n+1) = \{\boldsymbol{x}^{(m)}(n+1), w_s^{(m)}(n+1)\}_{m=1}^{M}$, where the weights $w_s^{(m)}(n+1)$ are the

smoothing weights of the particles, then the problem is in obtaining the smoothing weights from the above recursion.

From (5.37), it is clear that we have to run the PF in the usual way from time instant 0 to time instant N. The random measure at time instant N, $\chi(N)$ is identical to the smoothing random measure $\chi_s(N)$, which means that $w_s^{(m)}(N) = w^{(m)}(N)$. We obtain the random measure $\chi_s(N-1)$ from $\chi_s(N)$ and $\chi(N-1)$ by exploiting (5.37) as follows. First we draw a particle from the random measure $\chi_s(N)$. Let this particle be $\boldsymbol{x}^{(k)}(N)$. Then the nonnormalized smoothing weights are computed by

$$\widetilde{w}_s^{(m)}(N-1) = w^{(m)}(N-1) f(\boldsymbol{x}^{(k)}(N) \,|\, \boldsymbol{x}^{(m)}(N-1)), \;\; m = 1, 2, \ldots, M.$$

Subsequently the weights $\widetilde{w}_s^{(m)}(N-1)$ are normalized to $w_s^{(m)}(N-1)$, and the smoothing random measure $\chi_s(N-1) = \{\boldsymbol{x}^{(m)}(N-1), w_s^{(m)}(N-1)\}_{m=1}^{M}$ is formed. With this computation completed, we proceed with finding $\chi_s(N-2)$. We repeat the above steps by first drawing $\boldsymbol{x}(N-1)$ from $\chi_s(N-1)$, and then computing the weights $w_s(N-2)$ and so on. The method is summarized in Table 5.6. The procedure can be repeated to obtain independent realizations of the smoothing random measures.

Table 5.6 Smoothing algorithm I

Forward particle filtering

Run particle filtering in the forward direction and store all the random measures

$\chi(n) = \{\boldsymbol{x}^{(m)}(n), w^{(m)}(n)\}_{m=1}^{M}, \; n = 1, 2, \ldots, N$

Backward recursions

Set the smoothing weights

$w_s^{(m)}(N) = w^{(m)}(N)$ and

$\chi_s(N) = \{\boldsymbol{x}^{(m)}(N), w_s^{(m)}(N)\}_{m=1}^{M}$

For $n = N-1, \ldots, 1, 0$

Drawing a particle from $\chi_s(n+1)$

Draw $\boldsymbol{x}(n+1)$ from $\chi_s(n+1)$ and let the drawn particle be $\boldsymbol{x}^{(k)}(n+1)$

Computation of the smoothing weights $w_s(n)$

Compute the smoothing non-normalized weights of $\boldsymbol{x}^{(m)}(n)$ by

$\widetilde{w}_s^{(m)}(n) = w^{(m)}(n) f(\boldsymbol{x}^{(k)}(n+1) \,|\, \boldsymbol{x}^{(m)}(n)), \; m = 1, 2, \ldots, M$

Normalize the smoothing weights by computing

$$w_s^{(m)}(n) = \frac{\widetilde{w}_s^{(m)}(n)}{\sum_{j=1}^{M} \widetilde{w}_s^{(j)}(n)}$$

Construction of the smoothing random measure $\chi_s(n)$

Set $\chi_s(n) = \{\boldsymbol{x}^{(m)}(n), w_s^{(m)}(n)\}_{m=1}^{M}$

The second algorithm is based on an identity from [46], which connects the smoothing density $f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:N))$ with $f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{y}\ (1:N))$ through an integral and which is given by

$$f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:N)) = f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n)) \times \int \frac{f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{x}(n))f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:N))}{f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:n))}\,d\boldsymbol{x}(n+1). \quad (5.38)$$

The identity can readily be proved by noting that

$$f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:N)) = \int f(\boldsymbol{x}(n), \boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:N))\,d\boldsymbol{x}(n+1) \quad (5.39)$$

and

$$\begin{aligned} f(\boldsymbol{x}(n), \boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:N)) &= f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:N))f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n+1), \boldsymbol{y}(1:n)) \\ &= f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:N)) \\ &\quad \times \frac{f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{x}(n))f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))}{f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:n))}. \end{aligned} \quad (5.40)$$

We use the following approximations

$$f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:N)) \simeq \sum_{m=1}^{M} w_s^{(m)}(n)\delta(\boldsymbol{x}(n) - \boldsymbol{x}^{(m)}(n))$$

$$f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n)) \simeq \sum_{m=1}^{M} w^{(m)}(n)\delta(\boldsymbol{x}(n) - \boldsymbol{x}^{(m)}(n))$$

and

$$\begin{aligned} f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{y}(1:n)) &= \int f(\boldsymbol{x}(n+1)\,|\,\boldsymbol{x}(n))f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))\,d\boldsymbol{x}(n) \\ &\approx \sum_{m=1}^{M} w^{(m)}(n)f(\boldsymbol{x}(n+1)|\boldsymbol{x}^{(m)}(n)) \end{aligned}$$

to obtain

$$w_s^{(m)}(n) = \sum_{j=1}^{M} w_s^{(j)}(n+1)\frac{w^{(m)}(n)f(\boldsymbol{x}^{(j)}(n+1)\,|\,\boldsymbol{x}^{(m)}(n))}{\sum_{l=1}^{M} w^{(l)}(n)f(\boldsymbol{x}^{(j)}(n+1)\,|\,\boldsymbol{x}^{(l)}(n))}.$$

The algorithm is summarized in Table 5.7. The second algorithm is more complex than the first.

Table 5.7 Smoothing algorithm II

Forward PF

Run PF in the forward direction and store all the random measures $\chi(n) = \{\boldsymbol{x}^{(m)}(n), w^{(m)}(n)\}_{m=1}^{M}, \quad n = 1, 2, \cdots, N$

Backward recursions

Set the smoothing weights

$w_s^{(m)}(N) = w^{(m)}(N)$ and

$\chi_s(N) = \{\boldsymbol{x}^{(m)}(N), w_s^{(m)}(N)\}_{m=1}^{M}$

For $n=N-1,\ldots, 1, 0$

Computation of the smoothing weights $w_s(n)$

Compute the smoothing weights of $\boldsymbol{x}^{(m)}(n)$ by

$$w_s^{(m)}(n) = \sum_{j=1}^{M} w_s^{(j)}(n+1) \frac{w^{(m)}(n) f(\boldsymbol{x}^{(j)}(n+1) | \boldsymbol{x}^{(m)}(n))}{\sum_{l=1}^{M} w^{(l)}(n) f(\boldsymbol{x}^{(j)}(n+1) | \boldsymbol{x}^{(l)}(n))}$$

Construction of the smoothing random measure $\chi_s(n)$

Set $\chi_s(n) = \{\boldsymbol{x}^{(m)}(n), w_s^{(m)}(n)\}_{m=1}^{M}$

5.10 CONVERGENCE ISSUES

Whenever we develop algorithms for estimation, we study their performance and often express it in some probabilistic/statistical terms. For example, when we deal with parameter estimation, a benchmark for performance comparison of unbiased estimators is the Cramér–Rao bound. In particle filtering, where we basically rely on the Bayesian methodology, we use the concept of posterior Cramér–Rao bounds (PCRBs) and in obtaining them we combine the information about the unknowns extracted from observed data and prior information [74]. The PCRBs represent the MSEs that can be achieved by an estimator. The computation of the PCRB in the context of sequential signal processing is presented in [73]. Thus, often we find that particle filtering algorithms are compared with PCRBs of the estimated states, where the PCRBs themselves are computed by using Monte Carlo integrations. The difference between the MSEs and the PCRBs provides us with quantitative information about how far the algorithm is from optimal performance. An important issue that has not been successfully resolved yet is the required number of particles that are needed for achieving a desired performance. A related problem is the curse of dimensionality, that is, that for satisfactory accuracy the number of particles most likely will go up steeply as the dimension of the state increases [22].

An important type of performance issue is related to the convergence of particle filtering methods. For example, one would like to know if the methods converge, and if they do, in what sense and with what rate. We know that particle filtering is based on approximations of PDFs by discrete random measures and we expect that the approximations deteriorate as the dimension of the state space increases but

hope that as the number of particles $M \to \infty$, the approximation of the density will improve. We also know that the approximations are based on particle streams and that they are dependent because of the necessity of resampling. Therefore classic limit theorems for studying convergence cannot be applied, which makes the analysis more difficult.

The study of the convergence of particle filtering is outside the scope of this chapter. Here we only state some results without any proofs. For detailed technical details, the reader is referred to [21]. Here we summarize three results: (a) almost sure (weak) convergence of the empirical distributions to the true ones, (b) the set of sufficient conditions that ensure asymptotic convergence of the mean square error to zero, and (c) conditions for uniform convergence in time.

First we define the concept of weak convergence in the context of particle filtering. We say that a sequence of random measures

$$\begin{aligned}\chi^M(n) &= \{\boldsymbol{x}^{(m)}(n),\, w^{(m)}(n)\}_{m=1}^{M} \\ &= \sum_{m=1}^{M} w^{(m)}(n)\delta(\boldsymbol{x}(n) - \boldsymbol{x}^{(m)}(n))\end{aligned}$$

approximating the *a posteriori* PDF $f(\boldsymbol{x}(n)|\ \boldsymbol{y}(1:n))$ converges weekly to $f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:\mathrm{n}))$ if for any continuous bounded function $h(\boldsymbol{x}(n))$, we have

$$\lim_{M\to\infty} \int h(\boldsymbol{x}(n))\chi^M(n)\, d\boldsymbol{x}(n) = \int h(\boldsymbol{x}(n)) f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n))\, d\boldsymbol{x}(n).$$

Now we quote a theorem from [21].

Theorem 1 *If $f(\boldsymbol{x}(n) \mid \boldsymbol{x}(n-1))$ is Feller and the likelihood function $f(\boldsymbol{y}(n) \mid \boldsymbol{x}(n))$ is bounded, continuous and strictly positive, then*

$$\lim_{M\to\infty} \chi(n)^M = f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n))$$

almost surely.

The next result is about the convergence of the MSE. We define the error by

$$e^M(n) = \int h(\boldsymbol{x}(n))\chi^M(n)\, d\boldsymbol{x}(n) - \int h(\boldsymbol{x}(n)) f(\boldsymbol{x}(n) \mid \boldsymbol{y}(1:n))\, d\boldsymbol{x}(n).$$

Theorem 2 *If the likelihood function $f(\boldsymbol{y}(n) \mid \boldsymbol{x}(n))$ is bounded, for all $n \geq 1$ there exists a constant $c(n)$ independent of M such that for any bounded function $h(\boldsymbol{x}(n))$*

$$E(e^2(n)) \leq c(n)\frac{\|h(\boldsymbol{x}(n))\|^2}{M} \tag{5.41}$$

where $E(\cdot)$ is an expectation operator, $h(\boldsymbol{x}(n))$ has the same meaning as before and $\|h(\boldsymbol{x}(n))\|$ denotes the supremum norm, that is,

$$\|h(\boldsymbol{x}(n))\| = \sup_{\boldsymbol{x}(n)} |\, h(\boldsymbol{x}(n))\,|.$$

For a given precision on the MSE in (5.41), the number of particles depends on $c(n)$, which itself can depend on the dimension of the state space. The theorem holds for bounded functions of $\boldsymbol{x}(n)$, which means that it does not hold for $h(\boldsymbol{x}(n)) = \boldsymbol{x}(n)$. In other words, the result does not hold for the minimum MSE estimate of the state. In summary, the last theorem ensures that

$$\lim_{M\to\infty} \int h(\boldsymbol{x}(n))\chi^{M}(n)\, d\boldsymbol{x}(n) \to \int h(\boldsymbol{x}(n)) f(\boldsymbol{x}(n)\,|\,\boldsymbol{y}(1:n))\, d\boldsymbol{x}(n)$$

in the mean square sense and that the rate of convergence of the approximation error $E(e^2(n))$ is in $1/M$.

Clearly, the upper bound on the right hand side of the inequality in (5.41) depends on $c(n)$, and it can be shown that it increases with n. This implies that to have a certain precision, one has to increase the number of particles with n. To avoid the increase of $c(n)$ with time, we must have some mixing assumptions about the dynamic model that will allow for exponential forgetting of the error with time [21].

Before we proceed, we define a mixing kernel $r(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n-1))$ by

$$r(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n-1)) = f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}(n)) f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n-1)).$$

We now make the assumption that the mixing kernel is weakly dependent on $\boldsymbol{x}(n-1)$, and we express it by

$$\varepsilon\lambda(\boldsymbol{x}(n)) \le r(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n-1)) \le \varepsilon^{-1}\lambda(\boldsymbol{x}(n))$$

where $\lambda(\boldsymbol{x}(n))$ is some PDF. If in addition for

$$\rho(n) = \frac{\sup_{x} f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}(n))}{\inf_{\phi} \int f(\boldsymbol{y}(n)\,|\,\boldsymbol{x}(n)) \int \phi(\boldsymbol{x}(n-1)) f(\boldsymbol{x}(n)\,|\,\boldsymbol{x}(n-1))\, d\boldsymbol{x}(n-1)\, d\boldsymbol{x}(n)}$$

where $\phi(\boldsymbol{x}(n-1))$ is some PDF, we assume that

$$\rho(n) < \rho < \infty$$

we can show that the following theorem holds true:

Theorem 3 *For any bounded function $h(\boldsymbol{x}(n))$ and for all n, there exists a constant $c(\varepsilon)$ which is independent of M such that*

$$E(e^2(n)) \le c(\varepsilon)\frac{\|h(\boldsymbol{x}(n))\|^2}{M}. \tag{5.42}$$

In summary, if the optimal filter is quickly mixing, which means that the filter has a short memory, then uniform convergence of the PF is ensured. If the state vector contains a constant parameter, the dynamic model is not ergodic and uniform convergence cannot be achieved.

The material in this section is based on [21]. An alternative approach to studying the theoretical properties of PFs can be found in [53].

5.11 COMPUTATIONAL ISSUES AND HARDWARE IMPLEMENTATION

The particle filtering methodology is computationally very intensive. However, many of the required computations can be performed in parallel, which may produce significant savings in computing time. Particle filtering is intended for applications where real time processing is required and where deadlines for processing the incoming samples are imposed. Therefore, it is important to study the computational requirements of the methodology and the possibilities for its hardware implementations.

It is clear that particle filtering algorithms coded for a personal computer must often be modified so that when they are implemented in hardware they can have higher speeds. On the other hand, implementation in hardware usually requires fixed point arithmetics which can affect the performance of PFs. Even implementations on digital signal processors (DSPs) that are highly pipelined and support some concurrency may not be befitting for high-speed real-time particle filtering. In working on modifications of algorithms so that they are more suitable for hardware implementation, one has to address issues that include the reduction of computational complexity, high throughput designs, scalability of parallel implementation, and reducing memory requirements. Good general approaches to resolving them are based on joint algorithmic and architectural designs. Some designs that are tractable for VLSI implementations are discussed in [69].

One of the first efforts on development of hardware architectures for supporting high speed particle filtering is presented in [11]. The achieved speed improvements on a field-programmable gate array (FPGA) platform over implementations on (at that time) state-of-the-art DSPs was about 50 times. An FPGA prototype of a PF was presented in [8].

We recall that the three basic operations of particle filtering are particle generation, weight computations, and resampling. Of the three steps, the most computationally intensive are the particle generation and weight computation. Resampling is not computationally intensive but poses a different set of problems.

The speed of particle filtering on an algorithmic level can be increased by reducing the number of operations, using operational concurrency between the particle generation and weight computation. A designer is often faced with various problems including ways of implementing the required mathematical operations and choosing the level of pipelining for the hardware blocks. It has been found that for pipelined hardware implementation which is not parallel, the maximum achievable speed is proportional to $1/(2MT_{clk})$, where, as before, M represents the number of particles and T_{clk} is the clock period [11].

In [11], a hardware designed with processing elements (PEs) and a central unit (CU) was proposed. Each PE processes fractions of the particles (meaning particle propagation and their weight computations) whereas the CU controls the PEs and performs resampling. The resampling step represents a bottleneck in the parallel implementation of the filter, and it introduces other disadvantages. It increases the sampling period of the filter, that is, it slows down its operation, and it increases the memory requirements of the filter. The resampling also entails intensive data exchange within the filter, where large sets of data have to be exchanged through the interconnection network. In brief, besides being a bottleneck, the resampling step increases the complexity of the CU.

As pointed out, there are many implementation issues with particle filtering. One of them is the development of resampling schemes which are better for hardware implementation. For example, schemes that require less operations, less memory and less memory access are desirable. It is also preferable to have a resampling scheme with a deterministic processing time. Designs of algorithms that allow for overlap of the resampling step with the remaining steps are particularly desirable. Some generic architectures for sampling and resampling are presented in [7]. GPFs do not require resampling in the usual sense and they are better for parallel implementation. The data exchange that is required for them is deterministic and is much lower than for standard PFs, and they do not require the storage of particles between sampling instants. Some of the implementations of GPFs proposed in [11] achieve speeds twice that of standard PFs. The computation requirements of the GPFs are, however, higher in that they need more random number generators and more multipliers.

Other issues include the use of fixed and floating point arithmetic and scalability. The area and speed of the filter design depend on various factors including the number of used particles and the levels of implemented parallelism.

Reconfigurable architectures for PFs have also been explored [8]. Since PFs can be applied to many different problems, the idea is to develop architectures that can be used for different problems and thereby allow for flexibility and generality. Namely, designs on adaptable platforms can be configured for various types of PFs by modifying minimal sets of control parameters. We can view this effort as the development of programmable particle filtering hardware. One idea is to be able to select a PF from a set of possibilities where all of the available filters have maximum resource sharing in that the processing blocks are maximally reused, and the interconnection and interface between them is carried out by distributed buffers, controllers, and multiplexers [8]. An example of a reconfigurable PF for real-time bearings-only tracking is presented in [36]. Reconfigurability can include the type of used PF, the dimension of the state spaces, and the number of employed particles.

5.12 ACKNOWLEDGMENTS

The work of the authors on particle filtering has been supported by the National Science Foundation (Awards CCR-0220011 and CCF-0515246) and the Office of Naval Research (Award N00014-09-1-1154).

5.13 EXERCISES

5.1 Consider the following dynamic system

$$x(n) = x(n-1) + v_1(n)$$
$$y(n) = x(n) + v_2(n)$$

where $\boldsymbol{x}(n)$ and $y(n)$ are scalars and $v_1(n)$ and $v_2(n)$ are independent Gaussian noises of parameters μ_1 and σ_1^2, and μ_2 and σ_2^2, respectively. This system evolves for $n = 0{:}500\ s$.

(a) If the prior proposal distribution is chosen for generation of particles, derive the weight update expression.

(b) If the posterior proposal distribution is chosen for generation of particles, derive the weight update expression.

(c) Program a particle filter in MATLAB based on Table 5.3 for the previous system and using as proposal distribution the prior distribution.

(d) Extend the previous program to plot the evolution of the weights. Justify the degeneracy of the weights as time evolves.

(e) Implement a particle filter that uses the posterior proposal distribution and compare its performance with the previous one in terms of mean square error (MSE). The MSE is defined as

$$\text{MSE}(n) = \frac{1}{J}\sum_{j=1}^{J} (\hat{x}^j(n) - x^j(n))^2$$

where J denotes the number of simulation runs, and $x(n)$ and $\hat{x}^j(n)$ are the true and estimated (in the j-th run) values of the state at time instant n, respectively.

5.2 For the model from the previous example, implement the SIR PF with two possible resampling schemes: one that performs the resampling at each time step, and another one where the resampling is carried out depending on the effective particle size measure. Compare the performances of the filters for different numbers of particles M and different thresholds of effective particle sizes.

5.3 A model for univariate nonstationary growth can be written as

$$x(n) = \alpha x(n-1) + \beta\frac{x(n-1)}{1 + x^2(n-1)} + \gamma\cos(1.2(n-1)) + v_1(n)$$
$$y(n) = \frac{x^2(n)}{20} + v_2(n),\ n = 1, \ldots, N$$

where $\boldsymbol{x}(0) = 0.1$, $\alpha = 0.5$, $\beta = 25$, $\gamma = 8$, $N = 500$, $v_1(n) \sim N(0, 1)$ and $v_2(n) \sim \mathcal{N}(0, 1)$. The term in the process equation that is independent of $x(n)$

but varies with time can be interpreted as time-varying noise. The likelihood $f(y(n) \mid x(n))$ has bimodal nature due to the form of the measurement equation. Implement the SIR PF, APF, and GPF and compare their performances in terms of MSE.

5.4 Consider the recursive estimation of a target trajectory in a two-dimensional plane. The dynamic state space model is given by

$$\boldsymbol{x}(n) = \boldsymbol{x}(n-1) + T(\boldsymbol{s}(n) + \boldsymbol{v}_1(n))$$
$$y_i(n) = 10 \log_{10}\left(\frac{P_0}{|\boldsymbol{x}(n) - \boldsymbol{r}_i|^{\theta}}\right) + v_{2,i}(n)$$

where $\boldsymbol{x}(n) \in \Re^2$ is the vehicle position at time n, $T = 5\,\boldsymbol{s}$ is the observation period, $\boldsymbol{s}(n) \in \Re^2$ is the measured vehicle speed at time n, $\boldsymbol{v}_1(n) \sim \mathcal{N}(0, \boldsymbol{I}_2)$ is a 2×1 vector of a Gaussian error in the measurement of the speed, $y_i(n)$ is the measurement of the signal power arriving from the i-th sensor, $\boldsymbol{r}_i \in \Re^2$ is the (known) position of the i-th sensor, P_0 is the transmitted power which attenuates exponentially with the distance according to the unknown parameter $\theta \in [1, \theta_{max})$, and $v_{2,i}(n) \sim \mathcal{N}(0, 5)$ is a Gaussian error in the measurement of $y_i(n)$. All the noise processes are statistically independent.
Implement and compare two different particle filters that estimate jointly $\boldsymbol{x}(0\!:\!n)$ and θ from the sequence of observations $\boldsymbol{y}(1\!:\!n)$, where $\boldsymbol{y}(n) = [y_1(n)\ y_2(n)\ y_3(n)]^{\top}$. Assume that the prior distribution of the state is $\boldsymbol{x}(0) \sim f(\boldsymbol{x}(0)) = \mathcal{N}(0, 10\boldsymbol{I}_2)$, $\theta_{max} = 6$ and the prior distribution of the fixed parameter is uniform, that is, $f(\theta) \sim \mathcal{U}(1, 6)$.

5.5 The state-space model for detection of digital signals in flat fading channels can be represented as

$$\boldsymbol{h}(n) = \boldsymbol{D}\boldsymbol{h}(n-1) + \boldsymbol{g}\ v_1(n)$$
$$y(n) = \boldsymbol{g}^{\top}\boldsymbol{h}(n)s(n) + v_2(n)$$

where

$$\boldsymbol{h}(n) = [h(n)\ h(n-1)]^{\top}$$

is a vector of complex Rayleigh fading coefficient whose temporal correlation is modeled by an AR(2) process with known parameters a_1 and a_2, that is

$$\boldsymbol{D} = \begin{bmatrix} -a_1 & -a_2 \\ 1 & 0 \end{bmatrix}$$

where $a_1 = -1.993$ and $a_2 = 0.996$, and $\boldsymbol{g} = [\mathbf{1}\ \mathbf{0}]^{\top}$, $v_1(n)$ is a complex Gaussian noise with zero mean and unit variance, $s(n)$ is an M-ary transmitted

signal, $v_2(n)$ is a complex Gaussian noise with zero mean and variance σ^2. At any time instant n, the unknowns of the problem are $s(n)$ and $\boldsymbol{h}(n)$. (Note that this is also a nonlinear problem but where $\boldsymbol{h}(n)$ is a continuous variable and $s(n)$ is a discrete variable.)
Implement a marginalized PF that detects the transmitted signal $s(n)$ (note that given $s(n)$, the system is linear in $\boldsymbol{h}(n)$). Compare the obtained PF with a one that solves the problem without marginalizing out the conditionally linear parameter.

5.6 Consider the model

$$x(n) = \alpha + \beta x(n-1) + v_1(n)$$
$$y(n) = e^{x(n)/2} v_2(n)$$

where $x(n)$ is the state of the system, $v_1 \sim \mathcal{N}(0, \sigma^2)$, $y(n)$ is an observation and $v_2(n) \sim \mathcal{N}(0, 1)$. The parameters α and β and the variance σ^2 are all unknown. The objective is to track the dynamic state of the system $x(n)$. Develop a PF for tracking the unknowns in the system. (Hint: For modeling the prior of the variance σ^2, use the inverse Gamma PDF.)

5.7 A frequent problem in signal processing is the tracking of the instantaneous frequency of a sinusoid. In other words, let $x(n)$ represent the unknown frequency of a sinusoid modeled by a random walk

$$x(n) = x(n-1) + v_1(n)$$

where $v_1(n) \sim N(0, \sigma_{v_1}^2)$ and where the noise variance $\sigma_{v_1}^2$ is known and small. We observe $y(n)$ given by

$$y(n) = a_1 \cos(2\pi x(n)n) + a_2 \sin(2\pi x(n)n) + v_2(n)$$

with the amplitudes a_1 and a_2 being unknown, and where $v_2(n) \sim N(0, \sigma_{v_2}^2)$, where $\sigma_{v_2}^2$ is assumed known. Develop a PF that applies Rao–Blackwellization (the amplitudes a_1 and a_2 are conditionally linear parameters). Extend the PF so that it can track the signal frequency when the noise variances in the system are unknown.

REFERENCES

1. H. Akashi and H. Kumamoto, Construction of discrete-time nonlinear filter by Monte Carlo methods with variance reducing techniques, *Systems and Control*, 19, 211, (1975).
2. D. L. Alspach and H. W. Sorenson, Nonlinear Bayesian estimation using Gaussian sum approximation, *IEEE Trans. on Automatic Control*, 17, 439, (1972).

3. B. D. O. Anderson and J. B. Moore, *Optimal filtering*, Mineola, NY: Dover Publications, Inc., 1979.
4. D. Angelova and L. Mihaylova, Joint target tracking and classification with particle filtering and mixture Kalman filtering using kinematic radar information, *Digital Signal Processing*, 16, 180, (2006).
5. I. Arasaratnam, S. Haykin, and R. J. Elliott, Discrete-time nonlinear filtering algorithms using Gauss-Hermite quadrature, *Proc. of the IEEE*, 95, 953, 2007.
6. M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, *IEEE Trans. on Signal Processing*, 50, 174, (2002).
7. A. Athalye, Miodrag Bolić, S. Hong, and P. M. Djurić, Generic hardware architectures for sampling and resampling in particle filters, *EURASIP Journal on Applied Signal Processing*, 2005, 2888 (2005).
8. A. Athalye, "Design and Implementation of Reconfigurable Hardware for Real-Time Particle Filtering," Unpublished doctoral dissertation, Stony Brook University, (2007).
9. J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*, John Wiley & Sons, New York, (1994).
10. C. Berzuini, N. G. Best, W. R. Gilks, and C. Larizza, Dynamic conditional independence models and Markov chain Monte Carlo methods, *Journal of the American Statistical Association*, 92, 1403, (1997).
11. M. Bolić, "Architectures for Efficient Implementation of Particle Filters", Unpublished doctoral dissertation, Stony Brook University, (2004).
12. M. Bolić, P. M. Djurić, and S. Hong, Resampling algorithms and architectures for distributed particle filters, *IEEE Trans. Signal Processing*, 53, 2442, (2005).
13. M. G. S. Bruno, Bayesian methods for multiaspect target tracking in image sequences, *IEEE Trans. on Signal Processing*, 52, 1848, (2004).
14. R. S. Bucy, Bayes theorem and digital realization for nonlinear filters, *Journal of Astronautical Sciences*, 80, 73, (1969).
15. O. Cappé, E. Moulines, and T. Ryden, *Inference in Hidden Markov Models*, Springer, New York, (2005).
16. O. Cappé, S. J. Godsill, and E. Moulines, An overview of existing methods and recent advances in sequential Monte Carlo, *Proc. of the IEEE*, 95, 899, (2007).
17. F. Caron, M. Davy, E. Duflos, and P. Vanheeghe, Particle filtering for multisensor data fusion with switching observation models: Application to land vehicle positioning, *IEEE Trans. on Signal Processing*, 55, 2703, (2007).
18. G. Casella and C. P. Robert, Rao-Blackwellization of sampling schemes, *Biometrika*, 84, 81, (1996).
19. V. Cevher, A. C. Sankaranarayanan, J. H. McClellan, and R. Chellappa, Target tracking using a joint acoustic video system, *IEEE Trans. on Multimedia*, 9, 715 (2007).
20. R. Chen and J. S. Liu, Mixture Kalman filters, *Journal of the Royal Statistical Society*, 62, 493, 2000.
21. D. Crisan and A. Doucet, A survey of convergence results on particle filtering methods for practitioners, *IEEE Trans. on Signal Processing*, 50, 736, (2002).
22. F. Daum and J. Huang, "Curse of dimensionality and particle filters," *Proc. of IEEE Aerospace Conference*, IV, 1979, (2003).

23. P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez, Particle filtering, *IEEE Signal Processing Magazine*, 20, 19, (2003).
24. P. M. Djurić, M. F. Bugallo, and J. Míguez, "Density assisted particle filters for state and parameter estimation", *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Canada, (2004).
25. A. Doucet, S. J. Godsill, and C. Andrieu, On sequential Monte Carlo sampling methods for Bayesian filtering, *Statistics and Computing*, 10, 197, (2000).
26. A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*, Springer, New York, (2001).
27. A. Doucet, N. de Freitas, and N. Gordon, "An introduction to sequential Monte Carlo methods", In A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*, Springer, New York, (2001).
28. P. Fearnhead, Markov chain Monte Carlo, sufficient statistics and particle filters, *Journal of Computational and Graphical Statistics*, 11, 848, (2002).
29. S. Frühwirth-Schnatter, Applied state space modeling of non-Gaussian time series using integration-based Kalman filtering, *Statistics and Computing*, 4, 259, (1994).
30. A. Giremus, J.-Y. Tourneret, and V. Calmettes, A particle filtering approach for joint detection/estimation of multipath effects on GPS measurements, *IEEE Trans. on Signal Processing*, 55, 1275 (2007).
31. N. J. Gordon, D. J. Salmond, and A. F. M. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proc.-F*, 140, 107, (1993).
32. F. Gustaffson, F. Gunnarsson, N. Bergman, U. Forssel, J. Jansson, R. Karlsson, and P.-J. Nordlund, Particle filtering for positioning, navigation, and tracking, *IEEE Trans. on Signal Processing*, 50, 425, (2002).
33. J. M. Hammersley and K. W. Morton, Poor man's Monte Carlo, *Journal of the Royal Statistical Society, Series B*, 16, 23, (1954).
34. J. E. Handshin, Monte Carlo techniques for prediction and filtering of nonlinear stochastic processes, *Automatica*, 6, 555, (1970).
35. A. C. Harvey, *Forecasting, Structural Time Series Models and the Kalman Filter*, Cambridge: Cambridge University Press, (1989).
36. S. Hong, J. Lee, A. Athalye, P. M. Djurić, and W. D. Cho, Design methodology for domain-specific particle filter realization, *IEEE Trans. on Circuits and Systems – I: Regular Papers*, 54, 1987, (2007).
37. Y. Huang and P. M. Djurić, A blind particle filtering detector of signals transmitted over flat fading channels, *IEEE Trans. on Signal Processing*, 52, 1891, (2004).
38. C. Hue, J.-P. Le Cadre and P. Perez, Sequential Monte Carlo methods for multiple target tracking and data fusion, *IEEE Trans. on Signal Processing*, 50, 309, (2002).
39. K. Ito and K. Xiong, Gaussian filters for nonlinear filtering problems, *IEEE Trans. on Automatic Control*, 45, 910, (2000).
40. A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic Press, (1970).
41. S. J. Julier and J. K. Uhlmann, "A New extension of the Kalman filter to nonlinear systems", *Proc. of AeroSense: The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls* (1997).

42. S. Julier, J. Uhlmann, and H. F. Durante-White, A new method for nonlinear transformation of means and covariances in filters and estimators, *IEEE Trans. on Automatic Control*, 45, 477 (2000).
43. R. E. Kalman, A new approach to linear filtering and prediction Pproblems, *Trans. of the ASME-Journal of Basic Engineering*, 82, 35, (1960).
44. R. Karlsson and F. Gustafsson, Complexity analysis of the marginalized particle filter, *IEEE Trans. on Signal Processing*, 53, 4408, (2005).
45. R. Karlsson and F. Gustafsson, Bayesian surface and underwater navigation, *IEEE Trans. on Signal Processing*, 54, 4204, (2006).
46. G. Kitagawa, Non-Gaussian state-space modeling of nonstationary time series, *Journal of the American Statistical Association*, 82, 1032, (1987).
47. G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, *Journal of Computational and Graphical Statistics*, 1, 25, (1996).
48. T. Kloek and H. K. van Dijk, Bayesian estimates of system equation parameters: An application of integration by Monte Carlo, *Econometrica*, 46, 1, (1978).
49. S. C. Kramer and H. W. Sorenson, Recursive Bayesian estimation using piece-wise constant approximations, *Automatica*, 24, 789, (1988).
50. J. Kotecha and P. M. Djurić, Gaussian particle filtering, *IEEE Trans. on Signal Processing*, 51, 2592, (2003).
51. J. Kotecha and P. M. Djurić, Gaussian sum particle filtering, *IEEE Trans. on Signal Processing*, 51, 2602, (2003).
52. H. R. Künsch, "State space and hidden Markov models", in O. E. Barndorff-Nielsen, D. R. Cox, and C. Klueppelberg, Eds., *In Complex Stochastic Systems*, 109, Boca Raton: CRC Publishers, (2001).
53. H. R. Künsch, Recursive Monte Carlo filters: Algorithms and theoretical analysis, *The Annals of Statistics*, 33, 1983, (2005).
54. J. S. Liu and R. Chen, Sequential Monte Carlo methods for dynamical systems, *Journal of the American Statistical Association*, 93, 1032, 1998.
55. J. S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer, New York, 2001.
56. J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in A. Doucet, N. de Freitas, and N. Gordon, Eds. *Sequential Monte Carlo Methods in Practice*, 197, Springer, New York, (2001).
57. L. Ljung and T. Södereström, *Theory and Practice of Recursive Identification*, Cambridge, MA: The MIT Press, (1987).
58. A. Marshall, "The use of multi-stage sampling schemes in Monte Carlo computations", in M. Meyer, Ed., *Symposium on Monte Carlo Methods*, Wiley, pp. 123–140, (1956).
59. C. J. Masreliez, Approximate non-Gaussian filtering with linear state and observation relations, *IEEE Trans. on Automatic Control*, 20, 107, (1975).
60. L. Mihaylova, D. Angelova, S. Honary, D. R. Bull, C.N. Canagarajah, and B. Ristic, Mobility tracking in cellular networks using particle filtering, *IEEE Trans. on Wireless Communications*, 6, 3589 (2007).
61. M. R. Morelande and S. Challa, Manoeuvering target tracking in clutter using particle filters, *IEEE Trans. on Aerospace and Electronic Systems*, 41, 252, (2005).

62. M. R. Morelande, C. M. Kreucher, and K. Kastella, A Bayesian approach to multiple target detection and tracking, *IEEE Trans. on Acoustics, Speech and Signal Processing*, 55, 1589 (2007).
63. M. Orton and W. Fitzgerald, A Bayesian approach to tracking multiple targets using sensor arrays and particle filters, *IEEE Trans. on Signal Processing*, 50, 216, (2002).
64. D. Gatica-Perez, G. Lathoud, J.-M. Odobez, and I. McCowan, Audiovisual probabilistic tracking of multiple speakers in meetings, *IEEE Trans. on Audio, Speech, and Language Processing*, 15, 601, (2007).
65. M. Pitt and N. Shepard, Filtering via simulation: auxiliary particle filters, *Journal of the American Statistical Association*, 94, 590, 1999.
66. B. Ristić, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter*, Boston, MA: Artech House (2004).
67. M. N. Rosenbluth and A. W. Rosenbluth, Monte Carlo calculation of the average extension of molecular chains, *Journal of Chemical Physics*, 23, 356, (1956).
68. D. B. Rubin, Comment on 'The calculation of posterior distributions by data augmentation', by M. A Tanner and W. H. Wong, *Journal of the American Statistical Association*, 82, 543, (1987).
69. A. C. Sankaranarayanan, R. Chellappa, and A. Srivastava, "Algorithmic and architectural design methodology for particle filters in hardware," *ICCD '05: Proc. of the 2005 International Conference on Computer Design*, (2005).
70. H. W. Sorenson and D. L. Alspach, Nonlinear Bayesian estimation using Gaussian sums, *Automatica*, 7, 465 (1971).
71. H. W. Sorenson, "Recursive estimation for nonlinear dynamic systems", in J. C. Spall, Ed. *Bayesian Analysis of Time Series and Dynamic Models*, New York: Dekker, (1988).
72. G. Storvik, Particle filters for state-space models with presence of unknown static parameters, *IEEE Trans. on Signal Processing*, 50, 281, (2002).
73. P. Tichavský and C. H. Muravchik and A. Nehorai, Posterior Cramér-Rao bounds for discrete-time nonlinear filtering, *IEEE Trans. on Signal Processing*, 46, 1386, (1998).
74. H. L. Van Trees, *Detection, Estimation, and Modulation Theory*, John Wiley & Sons, New York, (1968).
75. D. B. Ward, E. A. Lehmann, and R. C. Williamson, Particle filtering algorithms for tracking an acoustic source in a reverberant environment, *IEEE Trans. Speech Audio Processing*, 11, 826, (2003).