Learning of dynamical systems

Particle filters and Markov chain methods

Thomas B. Schön and Fredrik Lindsten

Contents

Chapter 1

Introduction

1.1 A few words for readers of the early manuscript

The notes that you have in your hand are very much ongoing work. Here are a few comments and disclaimers concerning this current version:

- This is a rough start and we are very interested in all sort of feedback (including "high level" comments regarding structure as well as more detailed comments) on this manuscript. Feel free to provide any feedback/comments/suggestions you might have to thomas.schon@it.uu.se and/or fredrik.lindsten@it.uu.se.
- Earlier versions of this manuscript have so far been used for courses/tutorials at the following universities; UC Berkeley (US), Universidad Técnica Federico Santa María (Valparaíso, Chile), UC Santa Barbara (US), Vrije Universiteit (Brussel, Belgium), University of Sydney (Australia) Royal Institute of Technology (Stockholm, Sweden) and Uppsala University (Sweden) and at the following companies; Sennheiser (US), Autoliv (Sweden) and Saab (Sweden).

Finally, we hope that you will learn interesting new things from these notes and that you forgive us for handing them to you at this early stage of the writing process.

Chapter 2

Probabilistic modelling

Probabilistic modelling provides the capability to represent and manipulate *uncertainty* in data, models, decisions and predictions. A *mathematical model* is a compact representation—set of assumptions—that in a precise mathematical form tries to capture the key features of the phenomenon we are interested in. The true value of measured data typically arises once it has been analyzed and some kind of tangible knowledge has been extracted from the data. By forming the link between the measured data and the underlying phenomenon, the model takes an important role in performing this analysis.

Dynamical phenomena give rise to temporal measurements (data) arriving as a sequence $Y_{1:T} = (Y_1, Y_2, \ldots, Y_T)$. These measurements are typically corrupted by noise due to various imperfections in the measurement process and knowledge of this noise is one source of uncertainty in the model. Uncertainty is also present in several other—often less obvious—forms in any model. Even if we know that there is noise in the measurement process, there is still uncertainty about its form and at which level it is present. Furthermore, the structure of the model can be uncertain and there can also be many *unobserved variables* Z present. These unobserved variables are sometimes referred to as hidden variables, missing data/variables or latent variables. The introduction and use of these uncertain unobserved variables in probabilistic models is one of the key components providing the models with interesting and powerful capabilities that would not be possible without them. Indeed, many of the most successful models owe its usefulness and expressiveness largely to the incorporation of unobserved variables. Yet another form of uncertainty in the model comes from the unknown static model *parameters* θ that are often present.

The capability to mathematically represent and manipulate uncertainty is provided by probability theory. It describes how to systematically combine observations with existing knowledge—in the form of a mathematical model—to solve a new task and hopefully generate new knowledge as a result. The goal of this chapter is to introduce the ideas underlying probabilistic modelling in general and also to explain the most commonly used models when it comes to representing dynamical phenomena.

2.1 Representing and modifying uncertainty

We represent all our knowledge about the model (both known and unknown variables) using probability distributions and we use probability as our fundamental measure of uncertainty. Hence, our mathematical models are mostly built up from various probability distributions or probability density functions (PDFs). We allow ourselves to use the words density (short form for probability density function) and distribution interchangeably. Larger models can be built for example by combining several simple distributions over a single or a small number of variables. Random variables and stochastic processes will be instrumental in our development. However, we will also make use of deterministic—but unknown—variables when we discuss the maximum likelihood approach to modelling.

The end result of most of our developments in this manuscript will be expressed in terms of probabilistic statements provided by distributions. For example, if we are given a set of t measurements $Y_{1:t}$ and are looking for the unknown parameters θ, the solution is represented by the conditional distribution of the parameters given the measured data $p(\theta \mid y_{1:t})$, which is an object that is referred to as the posterior distribution. Sometimes a point estimate of the unknown object is sufficient, but it is still useful to first compute a probabilistic representation. The reason being that it can be dangerous to extract a point estimate without knowledge of the underlying distribution, since there might be ambiguities if there is too much uncertainty present. For a concrete example of such a situation we refer to Figure 2.1, which also illustrates what an uncertain representation can look like in an indoor positioning application example. An indoor positioning system

Figure 2.1: The task is to compute the location x_t of a person as time progresses based on a map of the environment and measurements $Y_{1:t}$ from an inertial sensor attached to the person. The particle filter (introduced in Chapter 5) solves this problem using a representation including the position of the person. The filters uncertain representation of the position is visualized using blue crosses, and the true position is indicated using green circles. From the representation in the leftmost plot we see that the solution is too uncertain to extract a reliable point estimate. Over time, as more and more measurements are acquired the uncertainty decrease and in the rightmost plot we have computed a point estimate (the conditional mean) indicated using black circles.

locates people or objects as they move around indoors using measurements from sensors such as for example magnetometers (magnetic field), inertial sensors (acceleration and angular velocity) and cameras (images). These sensors are often available in a standard

smartphone device. This example indicates the importance of maintaining a solid representation of the uncertainty throughout all calculations from the sensor measurements to the persons' location.

When it comes to the notation used for densities, we let $p(\,\cdot\,)$ denote a generic distribution associated with a model. It is thus the arguments that explains which variable that is intended as implicitly hinted at above. We will in general not use different notation for continuous and discrete distributions.

In Appendix B we provide a brief introduction to the distributions that we will make use of the basic building blocks in our models. The reader is assumed to have basic experience in using these distributions. There are many different ways in which we can parameterize various distributions and another important reason for including this appendix is to clearly describe which parameterizations we make use of.

2.1.1 Marginalization and conditional distributions

The two basic facts of probability theory that are fundamental for most of the development in this manuscript are marginalization and conditional distributions. Consider two random variables $A \in \mathsf{A}$ and $B \in \mathsf{B}$. *Marginalization* allows us to compute the marginal distribution $p(a)$ by integrating the joint distribution $p(a, b)$ over the entire space B where the variable B is defined,

$$p(a) = \int_{\mathsf{B}} p(a, b) \mathrm{d}b. \tag{2.1}$$

In doing this we say that the discarded variable B has been *marginalized out*. If the random variables are discrete, the integral in the above equation is replaced by a sum. Marginalization is useful since it allows us to remove the influence of one variable simply by averaging over all its possible values according to (2.1).

The second basic fact of probability theory that we will make extensive use of is that the joint distribution of A and B can be factored according to

$$p(a, b) = p(a \,|\, b)p(b) = p(b \,|\, a)p(a), \tag{2.2}$$

where $p(a \,|\, b)$ is the *conditional* distribution of A given B and vice versa for $p(b \,|\, a)$. The conditional distribution $p(a \,|\, b)$ encodes what we know about A based on the fact that we know B. It is now straightforward to make combined use of the conditioning and marginalization to establish that

$$p(b \,|\, a) = \frac{p(a \,|\, b)p(b)}{p(a)} = \frac{p(a \,|\, b)p(b)}{\int_{\mathsf{B}} p(a, b) \mathrm{d}b}, \tag{2.3}$$

which is commonly referred to as *Bayes' rule*. One important application of conditional probabilities and Bayes' rule is to transfer information from one variable to another. We will make extensive use of this in order to transfer information from measurements $y_{1:t}$ onto various unobserved variables and parameters present in the model. One specific example of this was provided in Figure 2.1 where it opened up for computing the location of a person x_t based on all the available measurements $y_{1:t}$, i.e. $p(x_t \,|\, y_{1:t})$.

2.1.2 Basic variable classes

The most obvious model variable is probably the measured data y obtained from the phenomenon we are interested in. Most of the phenomena we are considering have measurements arriving in a sequential fashion, i.e. $y_{1:t} = (y_1, y_2, \ldots, y_t)$.

The model often contains unknown (static) parameters θ that are used to describe the phenomenon. There are quite often unknown model variables x_t (changing over time) that describe the particular state of the phenomenon at time t. In the indoor positioning example at the beginning of this section a few examples of such unknown model parameters are the position and velocity of the person at each point in time.

Another class of variables is the so-called explanatory variables u, denoting known variables that we do not bother to model as stochastic.

2.1.3 Key probabilistic objects

Let us now introduce the key probabilistic objects we are concerned with in models involving unknown parameters θ and measurements Y. First of all we refer to the joint distribution $p(\theta, y)$ as the *full probabilistic model*. It constitutes a *generative* model for all the variables involved, meaning that it can be used to generate samples from these model variables. Sampling variables from the full probabilistic model is often a good way of getting some understanding of what the model's capabilities are.

Making direct use of the factorization offered by conditional probabilities (2.2) we can factor the full probabilistic model according to

$$p(\theta, y) = p(y \mid \theta)p(\theta), \tag{2.4}$$

where $p(\theta)$ is referred to as the *prior distribution*. The prior encodes the assumptions that are made concerning the unknown parameters before any data has arrived. The distribution $p(y \mid \theta)$ is called the *data distribution* or the *likelihood*. Note that this should not be mistaken for the likelihood function, which is something different. In order to minimize the risk for confusion we will make the slightly non-standard choice of referring to $p(y \mid \theta)$ as the data distribution and reserve the name likelihood exclusively for discussions involving the likelihood function. The data distribution $p(y \mid \theta)$ describes how the available measurements $Y = y$ relate to the unknown variables θ.

The likelihood function builds on the assumption that we model the unknown parameter θ as a deterministic variable, rather than as a random variable. Similarly, the measurements are modelled as one particular realization of the underlying random variables, $Y = y$. The *likelihood function*—denoted by $\mathcal{L}(\theta; y)$—is defined as $\mathcal{L}(\theta; y) \triangleq p(Y = y \mid \theta)$. Hence, it is *not* a distribution. The likelihood function is the function of the unknown parameters θ that is obtained when we evaluate the data distribution using the obtained measurements y. The likelihood function is used to compute various point estimates by solving various optimization problems. For example, the classic maximum likelihood estimator is obtained by selecting the parameters for which the likelihood function attains its maximum. The likelihood function is also used as one component when we compute maximum a posteriori estimates and various other regularized point estimates.

Rather than conditioning $p(\theta, y)$ on the parameters as was done in (2.4), let us now instead condition on the measurements $p(y, \theta) = p(\theta \mid y)p(y)$, resulting in the following expression for the posterior distribution

$$p(\theta \mid y) = \frac{p(y, \theta)}{p(y)} = \frac{p(y \mid \theta)p(\theta)}{\int p(y \mid \theta)p(\theta)\mathrm{d}\theta}. \tag{2.5}$$

Since $p(y)$ is independent of θ it acts as a *normalization constant* ensuring that $p(\theta \mid y)$ is a proper density that integrates to one. We can compute $p(y)$ by averaging $p(\theta, y)$ over all possible parameter values, i.e.

$$p(y) = \int p(y \mid \theta)p(\theta)\mathrm{d}\theta. \tag{2.6}$$

In this way $p(y)$ can be interpreted as the support or evidence that a particular model provide for the observed data y, which explains why we will refer to $p(y)$ as the *model evidence*. It can for instance be used in comparing various competing model types ability to represent and explain the measured data. Since it does not depend on the parameters (they have all been marginalized out) it is the "next level of abstraction" (i.e. the model type) that it represents.

It is often sufficient to find and work with the *unnormalized posterior distribution*

$$p(\theta \mid y) \propto p(y \mid \theta)p(\theta), \tag{2.7}$$

where we have simply discarded the normalization constant. For most models the normalization constant is challenging to compute, since the integral in (2.6) is intractable. Due to its importance we will introduce the notation $\tilde{p}(\theta \mid y)$ for the unnormalized version of the $p(\theta \mid y)$.

A *prediction* or forecast is a statement about a future uncertain event that has not yet been observed. Making predictions is one of the most common applications of mathematical models. In making a prediction the model is used together with the available observed data to compute the *predictive distribution* $p(\bar{y} \mid y)$ representing what we can say about the currently unseen future measurement $\bar{Y}$ based on the available measurements $Y = y$. Prediction is thus about generalizing from the measurements we have already observed to what will happen in the future.

Let us first reason intuitively about what sources of information we have available to us in making a prediction of a future measurement $\bar{Y}$. First of all, we have the information that has been accumulated about the parameters θ based on the measured data y we already have available. This information is conveniently summarized in the posterior $p(\theta \mid y)$. However, in order to leverage the information in the posterior we need a way of connecting this information about the model parameters with the new unseen measurement $\bar{Y}$. This second source of information is provided to us by the model itself, in that it directly provide the distribution $p(\bar{y} \mid \theta)$ which establishes the link between $\bar{Y}$ and θ that we are looking for. These two sources of information can be combined simply by marginalizing $p(\bar{y}, \theta \mid y)$ w.r.t. θ,

$$p(\bar{y} \mid y) = \int p(\bar{y}, \theta \mid y)\mathrm{d}\theta = \int p(\bar{y} \mid \theta, y)p(\theta \mid y)\mathrm{d}\theta = \int p(\bar{y} \mid \theta)p(\theta \mid y)\mathrm{d}\theta, \tag{2.8}$$

where the second equality follows from the conditional independence of $\bar{Y}$ and Y given θ. Again it is worth reflecting upon the use of the two basic facts (2.1) and (2.2) from Section 2.1.1.

We are in this manuscript essentially concerned with various ways of computing conditional distributions of the full probabilistic model $p(\theta, z, y)$ or some of its marginals. These computations boils down to challenging integration or optimization problems, where we focus on formulations based on integrals.

2.2 Probabilistic autoregressive modelling

The development has so far been rather abstract. Let us now swing to the other end of the spectra and instead become very explicit by illustrating the use of a first simple probabilistic model. The aim of this section it to illustrate how we represent and manipulate uncertainty in mathematical models using a well-known representation that most readers will probably have seen in one form or another.

The problem we are interested in is that of building a mathematical model that is capable of describing the observed data $Y_{1:T} \triangleq \{Y_1, \ldots, Y_T\}$. Intuitively the information from recent observations should contain some information about the current observation. Based on this intuition we let the current observation Y_t be a linear combination of the previous n observations plus some normally distributed noise E_t,

$$Y_t = A_1 Y_{t-1} + A_2 Y_{t-2} + \cdots + A_n Y_{t-n} + E_t, \qquad E_t \sim \mathcal{N}(\mu, \tau^{-1}), \tag{2.9}$$

where $\sim$ denotes distributed according to. This gives rise to the *autoregressive (AR)* model of order n, denoted by AR(n). It is a linear regression model where the output variable Y_t is explained using a linear combination of its own previous values $Y_{t-n:t-1}$ and a stochastic noise term e_t explaining random fluctuations. As already indicated by the notation used in (2.9) we have chosen to model the noise properties μ and τ as known explanatory variables ($\mu = 0, \tau \neq 0$). There is of course nothing stopping us from assuming also these parameters to be unknown and include them into the parameter vector. However, for now we let the unknown parameters be $\theta = (A_1, \ldots, A_n)^\mathsf{T}$ and assign the following prior

$$\theta \sim \mathcal{N}(0, \rho^{-1} I_n), \tag{2.10}$$

where the precision ρ is assumed to be known and hence modelled as an explanatory variable. Based on T observations $y_{1:T}$ of $Y_{1:T}$ we want to learn the model (2.9) that explains the measured data, i.e. we are looking for the posterior distribution $p(\theta \,|\, y_{1:T})$ (recall (2.5)). The first thing to do is thus to formulate the model in terms of the *full probabilistic model*, which is the joint distribution over all observed $Y_{1:T}$ and unobserved θ model quantities $p(\theta, y_{1:T}) = p(y_{1:T} \,|\, \theta) p(\theta)$. The prior is already defined and the data distribution $p(y_{1:T} \,|\, \theta)$ can be derived by direct application of the conditional

probability (2.2)

$$p(y_{1:T} \,|\, \theta) = p(y_T \,|\, y_{1:T-1}, \theta) p(y_{1:T-1} \,|\, \theta) = \ldots = \prod_{t=1}^{T} p(y_t \,|\, y_{1:t-1}, \theta), \tag{2.11}$$

with the convention that $y_{1:0} = \emptyset$. According to (2.9) we have $p(y_t \,|\, y_{1:t-1}, \theta) = \mathcal{N}\left(y_t \,\middle|\, \theta^\mathsf{T} z_t,\, \tau^{-1}\right)$, where $Z_t = (Y_{t-1},\, Y_{t-2},\, \ldots,\, Y_{t-n})^\mathsf{T}$, which inserted into (2.11) gives us the data distribution

$$p(y_{1:T} \,|\, \theta) = \prod_{t=1}^{T} \mathcal{N}\left(y_t \,\middle|\, \theta^\mathsf{T} z_t,\, \tau^{-1}\right) = \mathcal{N}\left(\boldsymbol{y} \,\middle|\, \boldsymbol{z}\theta,\, \tau^{-1} I_T\right), \tag{2.12}$$

where we have made use of $\boldsymbol{y} = (y_1,\, y_2,\, \ldots,\, y_T)^\mathsf{T}$ and $\boldsymbol{z} = (z_1,\, z_2,\, \ldots,\, z_T)^\mathsf{T}$ for the second equality. The model can now be summarized in terms of the joint distribution of the parameters and the observations $p(\theta, y_{1:T})$. By making use of the expressions for the data distribution (2.12) and the prior (2.10), Theorem 9 reveals the following explicit formulation of the model

$$p(\theta, y_{1:T}) = \mathcal{N}\left(\begin{pmatrix}\theta \\ \boldsymbol{y}\end{pmatrix} \,\middle|\, \begin{pmatrix}0 \\ 0\end{pmatrix},\, \begin{pmatrix}\rho^{-1} I_2 & \rho^{-1}\boldsymbol{z}^\mathsf{T} \\ \rho^{-1}\boldsymbol{z} & \tau^{-1} I_T + \rho^{-1}\boldsymbol{z}\boldsymbol{z}^\mathsf{T}\end{pmatrix}\right). \tag{2.13}$$

The parameters θ are still unknown, but there is nothing preventing us from computing the model evidence $p(y_{1:T})$ by averaging over all values that the parameters can possibly attain. This averaging process amounts to integrating out θ from $p(\theta, y_{1:T})$, which can be done in closed-form using Corollary 1, resulting in

$$p(y_{1:T}) = \int p(\theta, y_{1:T}) \mathrm{d}\theta = \int p(y_{1:T} \,|\, \theta) p(\theta) \mathrm{d}\theta = \mathcal{N}\left(\boldsymbol{y} \,\middle|\, 0,\, \tau^{-1} I + \rho^{-1}\boldsymbol{z}\boldsymbol{z}^\mathsf{T}\right). \tag{2.14}$$

There are now several ways for us to compute the posterior, but perhaps the most straightforward way to once again make use of Corollary 1 to conclude that

$$p(\theta \,|\, y_{1:T}) = \mathcal{N}\left(\theta \,|\, m_T,\, S_T\right), \tag{2.15a}$$

where

$$m_T = \tau S_T \boldsymbol{z}^\mathsf{T} \boldsymbol{y}, \tag{2.15b}$$

$$S_T = \left(\rho^{-1} I_2 + \sigma \boldsymbol{z}^\mathsf{T} \boldsymbol{z}\right)^\mathsf{T}. \tag{2.15c}$$

Let us illustrate the development so far using a simulation in Example 2.1.

Example 2.1: Bayesian learning of an AR(2) model

Rather than continuing the discussion using the general AR(n) model we will limit ourself to a particular special case,

$$Y_t = A_1 Y_{t-1} + A_2 Y_{t-2} + E_t = \theta^T Z_t + E_t, \qquad E_t \sim \mathcal{N}(0, \tau^{-1}), \tag{2.16}$$

where $\theta = (A_1, A_2)^\mathsf{T}$ and $Z_t = (Y_{t-1}, Y_{t-2})^\mathsf{T}$. Let the true values for θ in the AR(2) model (2.16) be $\theta^\star = (0.6,\, 0.2)^\mathsf{T}$. Furthermore, we treat $\tau = 5$ and $\rho = 2$ as known explanatory variables. One advantage with using this low-dimensional example is that we can visualize the model. In the left panel of Figure 2.2 we see the prior $\mathcal{N}(0, 1/2I_2)$ before any measurements have used. The right panel shows 7 samples drawn from this prior.

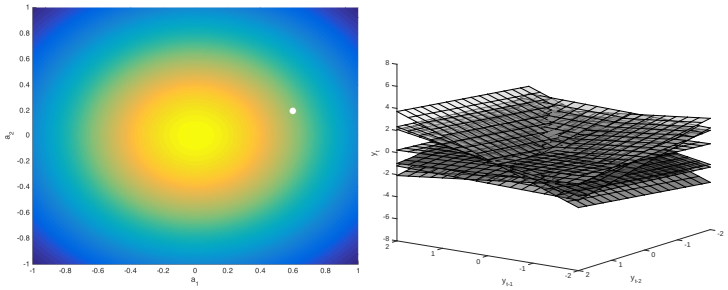

Figure 2.2: Illustration of the prior $\theta \sim \mathcal{N}(0, 1/2I_2)$ used for the Bayesian AR(2) model. The left plot shows the prior as a contour plot (the x-axis corresponds to a_1 and the y-axis corresponds to a_2). The right panel shows 7 samples drawn from this prior. The true parameter value is show by the white dot.

Figure 2.3 show the situations after the information available in the measurements $y_{1:t}$ has been used, for three different values for $t = 1, 2$ and $t = 20$.

2.2.1 Predictive distribution

Predicting the subsequent and yet unseen measurement Y_{T+1} amounts to combining the information we indirectly have available about Y_{T+1} via the posterior and the model by solving the following integral (recall (2.8))

$$p(y_{T+1} \,|\, y_{1:T}) = \int p(y_{T+1} \,|\, \theta) p(\theta \,|\, y_{1:T}) \mathrm{d}\theta. \tag{2.17}$$

This integral implements a weighted average, where the connection between the new observation and the model parameters provided by the model $p(y_{T+1} \,|\, \theta)$ is weighted with the information about the parameters that we have available in the posterior $p(\theta \,|\, y_{1:T})$. Recall that the posterior captures all information that is present in the measured data $y_{1:T}$ about the parameters θ, under the implicit assumption that the model is the one given in (2.16). This is the way in which the information from the old measurements

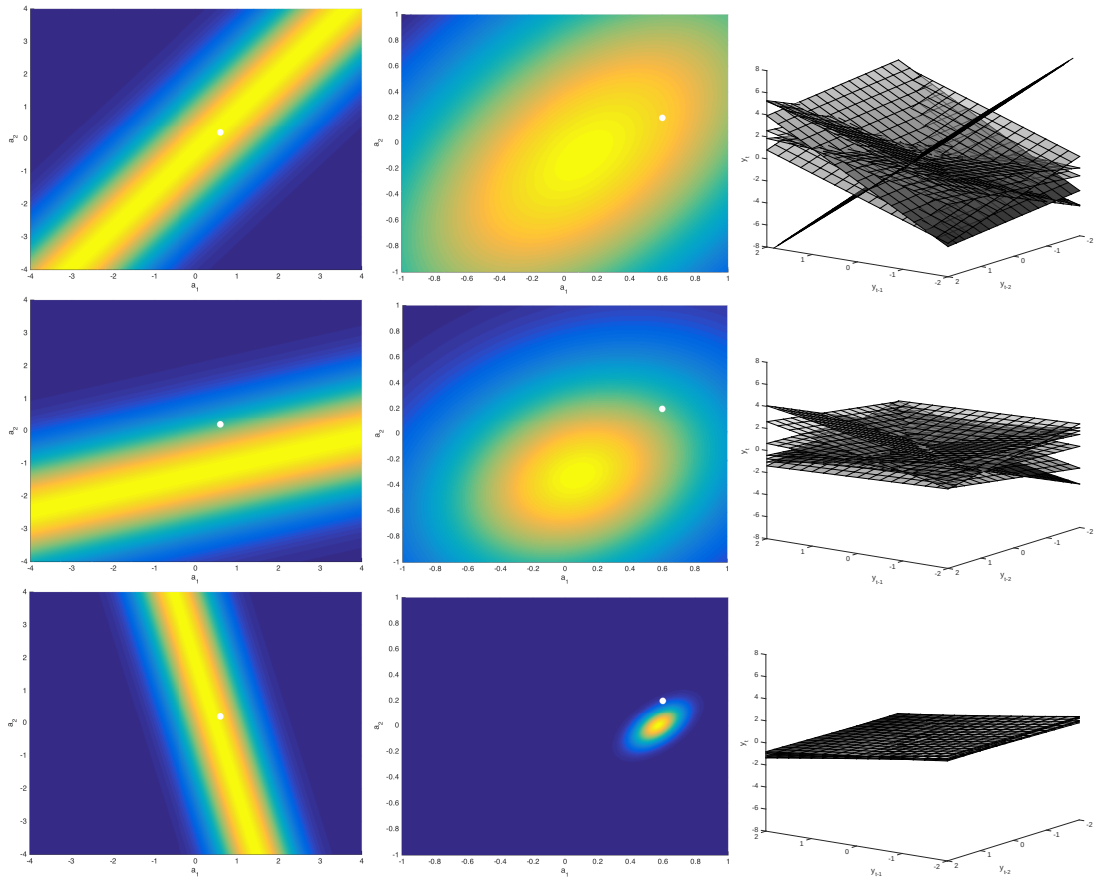

Figure 2.3: Illustration of the results from the Bayesian AR(2) model. The plots show the likelihood (left column), posterior (middle column) and 7 samples drawn from the posterior (right column). The x-axis corresponds to a_1 and the y-axis corresponds to a_2 (for the likelihood and posterior plots). The true parameter value is show by the white dot. The three rows shows the situation after 1 (top), 2 (middle) and 20 (bottom) measurements have been incorporated, respectively.

is used to make predictions about future observations. From the model (2.16), we have that

$$p(y_{T+1} \,|\, \theta) = \mathcal{N}\left(y_{T+1} \,\middle|\, \theta^{\mathsf{T}} z_{T+1},\, \tau^{-1}\right). \tag{2.18}$$

Inserting (2.18) and the posterior (2.15) into (2.17) allows us to make use of Corollary 1 to conclude that the predictive distribution is given by

$$p(y_{T+1} \,|\, y_{1:T}) = \mathcal{N}\left(y_{T+1} \,\middle|\, z_{T+1}^{\mathsf{T}} m_T,\, z_{T+1}^{\mathsf{T}} S_T z_{T+1} + \tau^{-1}\right). \tag{2.19}$$

The predictive mean is given by

$$f(z_{T+1}) = z_{T+1}^{\mathsf{T}} m_T = \tau z_{T+1}^{\mathsf{T}} S_T Z^{\mathsf{T}} Y = \tau z_{T+1}^{\mathsf{T}} S_T \sum_{t=1}^{T} y_t z_t$$
$$= \sum_{t=1}^{T} \tau z_{T+1}^{\mathsf{T}} S_T z_t y_t = \sum_{t=1}^{T} k(z_{T+1}, z_t) y_t, \tag{2.20}$$

where $k(z, z') = \tau z^{\mathsf{T}} S_T z'$ is referred to as the *equivalent kernel*. The expression (2.20) leads to the observation that rather than postulating a parametric model according to (2.16) (i.e. $f(z_{T+1}) = \theta^{\mathsf{T}} z_{T+1}$) we can *directly* make use of the model formulation offered by the equivalent kernel. This also motivates the term equivalent kernel, since starting from the formulation (2.20) we obtain exactly the same predictor as we obtain in starting from (2.16).

2.3 Latent variable models

We refer to model variables that are *not* observed as *latent*[1] variables. The idea of introducing latent variables into models is probably one of the most powerful concepts in probabilistic modelling. These latent variables provides more expressive models that can capture hidden structures in data that would otherwise not be possible. The price for the extra flexibility in the representation provided by the latent variables is that the problem of learning the model from measured data becomes significantly harder. This manuscript is to a large extent about how to solve makes inference in nonlinear latent variable models for dynamical phenomena. There will be plenty of examples later on, but to get an intuitive feeling for the way in which latent variables can provide more expressive models we give a first illustration in Example 2.2.

Example 2.2: AR model with a latent noise variance

The increased expressiveness that the latent variables offer for a model can be thought of as an internal memory. For dynamical models this is particularly natural.

The latent variables results in models with a much richer internal structure.

Use this as a basic building block and derive more expressive models. Ex. switching (combining a discrete and cont. SSM) “add more structure to it”. Provide a forward reference.

2.4 Markov chains

The Markov chain is a probabilistic model that is used for modelling a sequence of states $(X_0, X_1, \ldots, X_T)$, where the index variable t in our case refers to time, but it

[1] Other commonly used names for latent variables include hidden variables, missing variables and unobserved variables.

could also simply be thought of as an arbitrary indexing variable for the location within a sequence. The basic idea underlying a Markov model is that given the present value of the state X_t, its future evolution $\{X_{t+k}\}_{k>0}$ is conditionally independent of the past $\{X_s\}_{s<t}$. Hence, the current state acts as a memory in that it contains all there is to know about the phenomenon at this point in time based on what has happened in the past. The *Markov chain* is completely specified by an initial value X_0 and a transition model (kernel) $p(x_{t+1} \,|\, x_t)$ describing the transition from state X_t to state X_{t+1}. Let us properly define the corresponding stochastic process.

Definition 1 (Markov chain)**.** *A stochastic process $\{X_t\}_{t\geq 0}$ is referred to as a Markov chain if, for every $k > 0$ and t,*

$$p(x_{t+k} \,|\, x_1, x_2, \ldots, x_t) = p(x_{t+k} \,|\, x_t). \tag{2.21}$$

The Markov property (2.21) is highly desirable since it provides a memory efficient way of keeping track of the evolution of a dynamic phenomenon. Rather than keeping track of the growing full history of the process $\{X_s\}_{s=1}^{t}$, it is sufficient to keep track of the present state X_t of the process. Hence, in a Markov process the current state contains everything we need to know about the past and the present in order to predict the future. Hence, based on Definition 1 we can write the joint distribution of all states as

$$\begin{aligned} p(x_{0:T}) &= p(x_T \,|\, x_{0:T-1}) p(x_{0:T-1}) = p(x_T \,|\, x_{T-1}) p(x_{0:T-1}) = \ldots \\ &= p(x_0) \prod_{t=0}^{T-1} p(x_{t+1} \,|\, x_t). \end{aligned} \tag{2.22}$$

The Markov chain is a stochastic dynamical system and for some models there exist a so-called *stationary distribution* which is the distribution characterizing the long-term behaviour of the chain. If the Markov chain starts with its stationary distribution then the marginal distribution of all states will always be the stationary distribution.

Example 2.3: Stationary distribution of a simple Markov chain

Consider the Markov chain with initial state $X_0 = -40$ and state dynamics given by

$$X_{t+1} = 0.8 X_t + V_t, \qquad V_t \sim \mathcal{N}(0, 1), \tag{2.23}$$

which corresponds to the following transition model $p(x_{t+1} \,|\, x_t) = \mathcal{N}\left(x_{t+1} \,|\, 0.8 x_t,\, 1\right)$. First of all we note that since the Markov chain is defined by a linear transformation (2.23) of known ($X_0 = 0$) and normally distributed $\{V_t\}_{t\geq 1}$ random variables, its state X_t must remain Gaussian at all times. As such it is completely described by its mean value and variance. Let us now study what happens with the state distribution as $t \to \infty$. We start with the mean value

$$\mathbb{E}[X_t] = \mathbb{E}[0.8 X_{t-1} + V_{t-1}] = 0.8 \mathbb{E}[X_{t-1}] = \cdots = 0.8^t \mathbb{E}[X_0] \to 0, \tag{2.24}$$

as $t \to \infty$ independently of the initial value for that state. For the variance we have

$$\mathrm{Var}[X_t] = \mathbb{E}\Big[\big(X_t - \mathbb{E}[X_t]\big)^2\Big] = \mathbb{E}\big[X_t^2\big] - \big(\mathbb{E}[X_t]\big)^2, \tag{2.25}$$

where we already know that the second term will tend to zero as $t \to \infty$. For the first term we have

$$\mathbb{E}\big[X_t^2\big] = \mathbb{E}\big[0.8^2 X_t^2 + 1.6 X_{t-1} V_{t-1} + V_{t-1}^2\big] = 0.8^2 \mathbb{E}\big[X_{t-1}^2\big] + 1. \tag{2.26}$$

We can now find the stationary variance by introducing the notation $\bar{p} = \lim_{t\to\infty} \mathrm{Var}[X_t] = \lim_{t\to\infty} \mathbb{E}\big[X_t^2\big]$ into (2.26), resulting in $\bar{p} = 0.8^2\bar{p} + 1$ which means that $\bar{p} = \frac{1}{1-0.8^2}$. We have now proved that the stationary distribution of the Markov chain defined in (2.23) is given by

$$p^{\mathrm{s}}(x) = \mathcal{N}\left(x \,\middle|\, 0,\, \frac{1}{1-0.8^2}\right). \tag{2.27}$$

In Figure 2.4 we illustrate the result of the above calculations.

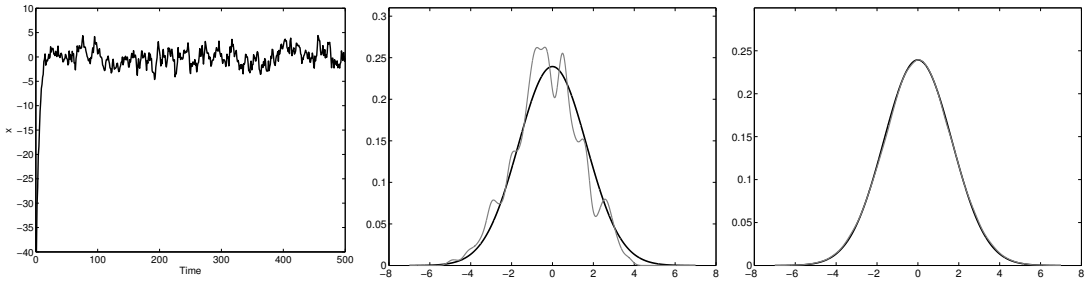

Figure 2.4: Left: Example realization showing 500 samples from (2.23). Middle: The stationary distribution (2.27) is shown in black and the empirical histogram obtained by simulating 10 000 samples from the Markov chain (2.23). The initial 1 000 samples were discarded. Right: Same as the middle plot, but here 100 000 samples are used.

The Markov chain constitutes a basic ingredient in the Markov chain Monte Carlo (MCMC) methods that we will develop in Chapter **??**, where we will also continue to more systematically study when there do exist a stationary distribution. In the subsequent section we will introduce a Markov model where we can only observe the state indirectly via a measurement that is related to the state.

2.5 State space models

The basic state space model (SSM)[2] is a Markov model that makes use of a latent variable representation to describe dynamical phenomena. It offers a practical representation not

[2]The state-space model travels under several names. It is sometimes referred as the hidden Markov model (HMM), clearly indicating that the state is unobserved (hidden) and modelled using a Markov process. It is worth noting that some authors have reserved the name HMM for the special case where the state variable is assumed to belong to a finite set of discrete variables. Another name sometimes used is latent Markov model.

only for modelling, but also for reasoning and doing inference. The SSM consists of two stochastic processes; an unobserved (the state) process $\{X_t\}_{t\geq 1}$ modelling the dynamics and an observed process $\{Y_t\}_{t\geq 1}$ modelling the measurements and their relationship to the unobserved state process. There are several different ways of representing the SSM. We start by describing it using the following functional form

$$X_{t+1} = f(X_t, \theta) + V_t, \tag{2.28a}$$
$$Y_t = g(X_t, \theta) + E_t, \tag{2.28b}$$

where the unknown variables are given by the latent state $X_t \in \mathsf{X}$ describing the system's evolution over time and the (static) parameters $\theta \in \Theta$. The initial state is modelled as $x_0 \sim p(x_0 \,|\, \theta)$. The known variable is give by the measurement $Y_t \in \mathsf{Y}$. The uncertain variables V_t and E_t denote noise terms, commonly referred to as the process noise and the measurement noise, respectively. Finally, the functions f and g denotes the dynamics and the measurement equation, respectively. Explanatory variables can straightforwardly be added to the model, but since they will not affect the basic principles we do not include then explicitly in the interest of a clean notation.

In Section 2.5.1 we will represent (2.28) using probability density functions instead and an Section 2.5.2 it will be represented as a graphical model. These three representations are all useful and interesting in their own rights, none of them is generally better or worse than the other. The choice of representation typically depends on taste and which task we are currently working with.

2.5.1 Representation using probability density functions

Let us start by noticing that we can express the distribution governing Markov chain's transition from X_t to the new state at the next time instant X_{t+1} according to

$$p(x_{t+1} \,|\, x_t, \theta) = p_{v_t}(x_{t+1} - f(x_t, \theta)) \tag{2.29}$$

where p_{v_t} denotes the distribution of the process noise v_t. Analogously (2.28b) defines the conditional distribution of the measurement Y_t given the current state X_t of the Markov chain and the parameters θ. Taken together we can express the SSM (2.28) as

$$X_{t+1} \,|\, (X_t = x_t, \theta = \theta) \sim p(x_{t+1} \,|\, x_t, \theta), \tag{2.30a}$$
$$Y_t \,|\, (X_t = x_t, \theta = \theta) \sim p(y_t \,|\, x_t, \theta), \tag{2.30b}$$
$$X_0 \sim p(x_0 \,|\, \theta). \tag{2.30c}$$

We have so far not made any specific assumptions about the unknown static model parameters θ other than that they might exist. Implicitly it is then commonly assumed that the parameters are modelled as deterministic, but unknown, leading us to various maximum likelihood formulations. On the other hand, when performing Bayesian inference we need to augment (2.30) with a fourth PDF $p(\theta)$ describing the prior stochastic nature of the static parameters,

$$\theta \sim p(\theta). \tag{2.31}$$

The reasoning below is made under this probabilistic assumption on the parameters. We can now factor the full probabilistic model $p(x_{0:T}, \theta, y_{1:T})$ according to

$$p(x_{0:T}, \theta, y_{1:T}) = \underbrace{p(y_{1:T} \mid x_{0:T}, \theta)}_{\text{data distribution}} \underbrace{p(x_{0:T} \mid \theta)p(\theta)}_{\text{prior distribution}}, \tag{2.32}$$

where $p(y_{1:T} \mid x_{0:T}, \theta)$ describes the distribution of the data and $p(x_{0:T}, \theta) = p(x_{0:T} \mid \theta)p(\theta)$ represents our initial—prior—assumptions about the unknown states and parameters. Using conditional probabilities we can factor the data distribution according to

$$\begin{aligned} p(y_{1:T} \mid x_{0:T}, \theta) &= p(y_T \mid y_{1:T-1}, x_{0:T}, \theta)p(y_{1:T-1} \mid x_{0:T}, \theta) \\ &= p(y_T \mid x_T, \theta)p(y_{1:T-1} \mid x_{0:T-1}, \theta), \end{aligned} \tag{2.33}$$

where we also made use of the conditional independence of the measurements given the current state. Similarly we can rewrite the prior distribution of the states by making use of conditional probabilities and the Markov property, resulting in

$$p(x_{0:T} \mid \theta) = p(x_T \mid x_{1:T-1}, \theta)p(x_{1:T-1} \mid \theta) = p(x_T \mid x_{T-1}, \theta)p(x_{1:T-1} \mid \theta). \tag{2.34}$$

From the above reasoning it is clear that that the data distribution is dictated by (2.30b) and that the prior distribution on the states is given by (2.30a). Repeated use of (2.33) and (2.34) results in

$$p(x_{0:T}, \theta, y_{1:T}) = \underbrace{\left(\prod_{t=1}^{T} \underbrace{p(y_t \mid x_t, \theta)}_{\text{observation}} \right)}_{\text{data distribution}} \underbrace{\left(\prod_{t=0}^{T-1} \underbrace{p(x_{t+1} \mid x_t, \theta)}_{\text{dynamics}} \right) \underbrace{p(x_0 \mid \theta)}_{\text{initial}} \underbrace{p(\theta)}_{\text{initial}}}_{\text{prior}} \tag{2.35}$$

It is useful to mention two of the most commonly used spaces X; the n_x-dimensional real space $\mathbb{R}^{n_x}$ and a finite number of positive integers $\{1, \ldots, K\}$. In Example 2.4 we provide an example where $\mathsf{X} = \{1, \ldots, K\}$, which results in the so-called Finite state space (FSS) model.

Example 2.4: Finite state space (FSS) model

Example 2.5: A nonlinear state space model

2.5.2 Graphical model representation

It is instructive to consider a third representation of the SSM, namely the use of *graphical models*. A graphical model is a graph encoding the probabilistic relationships among the involved random variables. Let us assume that we have T measurements from the SSM in (2.30), then the relevant variables are the states $X_{0:T} \triangleq \{X_0, \ldots, X_T\}$ and the measurements $Y_{1:T} \triangleq \{Y_1, \ldots, Y_T\}$. The graphical model can be viewed as a graph representing the joint distribution $p(x_{0:T}, y_{1:T})$ of all the involved random variables in a factorized form, where all the conditional independences inherent in the model are visualized. Each random variable is represented as a node. If the node is filled (gray), the corresponding variable is observed and if the node is not filled (white), the corresponding variable is hidden, i.e. not observed. The probabilistic relationships among the random variables are encoded using edges.

In representing the SSM as a graphical model we will make use of a specific instance of the graphical models, commonly referred to as *Bayesian networks* or *belief network*. Such a model corresponds to a directed acyclic graph, where the edges are constituted by arrows encoding the conditional independence structure among the random variables. In Figure 2.5 we provide the graphical model corresponding to the SSM. In a graph like the one in Figure 2.5 we say that node X_1 is the *child* of node X_0 and that node X_1 is the *parent* of node X_2 and Y_1. The arrows pointing to a certain node encodes which variables the corresponding node are conditioned upon, these variables constitute the set of parents to that particular variable.

A graphical model directly describes how the joint distribution of all the involved variables (here $p(x_{0:T}, y_{1:T})$) can be decomposed into a product of factors according to

$$p(x_{0:T}, y_{1:T}) = \prod_{t=0}^{T} p(x_t \,|\, \mathrm{pa}(x_t)) \prod_{t=1}^{T} p(y_t \,|\, \mathrm{pa}(y_t)), \tag{2.36}$$

where $\mathrm{pa}(X_t)$ denotes the set of parents to X_t. Hence, each factor in (2.36) consists of the PDF of a random variable conditioned on its parents. Using (2.36), we can decode the Bayesian network in Figure 2.5 resulting in the following joint distribution for the

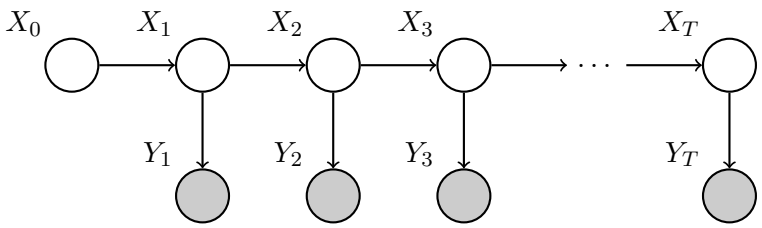

Figure 2.5: Graphical model for the SSM in (2.30). Each random variable is encoded using a node, where the nodes that are filled (gray) corresponds to variables that are observed and nodes that are not filled (white) are latent variables. The arrows encode the dependence among the variables.

states and the measurements,

$$p(x_{0:T}, y_{1:T}) = p(x_0) \prod_{t=1}^{T} p(x_t \,|\, x_{t-1}) \prod_{t=1}^{T} p(y_t \,|\, x_t). \tag{2.37}$$

Each arrow in Figure 2.5 corresponds to one of the factors in (2.37). The expression (2.37) for the joint distribution can of course also be derived directly from the underlying model assumptions.

Already for slightly more complex models, such as the conditionally linear Gaussian models (introduced in Section 2.7) we will start to see the beauty of making use of graphical models, in that they quickly reveals the underlying probabilistic structure inherent in a model. In considering Bayesian SSMs, graphical models are also very enlightening. Furthermore, the theory surrounding graphical models is well developed also when it comes to performing inference and this can be useful in considering more complex models.

2.6 Linear Gaussian state space models

The state is typically not visible (i.e. we cannot measure it directly) and it evolves by applying linear transformations to it and then adding Gaussian noise according to

$$X_{t+1} = AX_t + GV_t, \qquad V_t \sim \mathcal{N}(0, I), \tag{2.38a}$$

where $V_t \in \mathbb{R}^{n_v}$ and the initial state is modelled as $p(x_0 \,|\, \theta) = \mathcal{N}(x_0 \,|\, \mu, P_0)$. This equation is often referred to as the *transition dynamics* (or just dynamics) since it encodes the dynamic evolution of the memory (state) over time by encoding exactly how the state transitions from one time instance to the next. The uncertainty in the transition is represented by the *process noise* $\widetilde{V} \triangleq GV_t$ which is distributed according to $p(\tilde{v}) = \mathcal{N}\left(\tilde{v} \,\middle|\, 0, GG^{\mathsf{T}}\right)$.

Given the hidden state we get the visible observations Y_t by applying another linear transformation C to the state and adding Gaussian noise E_t according to

$$Y_t = CX_t + E_t, \qquad E_t \sim \mathcal{N}(0, R), \tag{2.38b}$$

where $Y_t \in \mathbb{R}^{n_y}$ and $E_t \in \mathbb{R}^{n_y}$. We will refer to (2.38b) as the *measurement equation*, since it describes how the measurements are obtained from the states. The fact that any measurement process is uncertain is modelled by the measurement noise E_t. Hence, each state X_t generates a noisy observation Y_t according to (2.38b), where the observations a conditionally independent given the state.

The linear dynamical model described by (2.38) is called the linear Gaussian state space model (LG-SSM). It only involves linear transformations of Gaussian random variables, implying that all random variables involved in the model will be Gaussian. The LG-SSM has a direct generative interpretation.

From the above model description it is far from obvious which variables that should be considered as being latent. We need more information in order to make a wise choice

on this subtle matter. Let us make this more precise by considering a few common situations.

In the situation where $G = 0$ the transition equation (2.38a) is given by $X_{t+1} = AX_t$. This means that if we know the initial state X_0, then all other states are known as well. Hence, the latent variable is given by the initial state X_0. In models of this kind, without any process noise (i.e. a deterministic transition equation) are sometimes referred to as *output error (OE)* models. The full probability model is given by

$$p(x_1, \theta, y_{1:T}) = \left(\prod_{t=1}^{T} p(y_t \mid x_t, \theta) \right) p(x_0 \mid \theta) p(\theta), \tag{2.39}$$

where the state at time t is a deterministic mapping from the initial state $x_t = A^t x_0$.

Let us now consider the case where the dimension of the process noise is the same as the state dimension, $n_v = n_x$ and that the matrix G is full rank and hence invertible. For this case we need to keep track of all state variables and the latent variable is thus given by $X_{1:T}$.

$$p(z, \theta, y) = p(x_{1:T}, \theta, y_{1:T}) = \left(\prod_{t=1}^{T} p(y_t \mid x_t, \theta) \right) \left(\prod_{t=1}^{T} p(x_{t+1} \mid x_t, \theta) \right) p(x_1 \mid \theta) p(\theta), \tag{2.40}$$

where accoding to (2.38a) we have that $p(x_{t+1} \mid x_t, \theta) = \mathcal{N}\left(x_{t+1} \,\middle|\, Ax_t,\, GG^{\mathsf{T}}\right)$.

As a third case we consider the situation where $n_w < n_x$. This corresponds to a situations where the covariance matrix of the process noise $\widetilde{V}_t = GV_t$ is singular, i.e. $\operatorname{rank}\left(GG^{\mathsf{T}}\right) < n_x$.

In this situation it often makes sense to choose the initial state and the process noise as latent variables, $Z_{0:T} = \{X_0, V_{1:T}\}$.

$$p(z, \theta, y) = p(x_1, w_{1:T}, \theta, y_{1:T}) = \prod_{t=1}^{T} p(y_t \mid x_t, \theta) p(v_{1:T} \mid \theta) p(\theta), \tag{2.41}$$

where the states are given by a deterministic mapping from the initial state and the process noise, $X_{t+1} = AX_t + GV_t$. An alternative choice of latent variables is provided by the part of the state vector that is stochastic.

The linear Gaussian state space model is probably the most well-studied and well-used class of latent variable models when it comes to dynamical phenomena. There are at least three good reasons for this. First, it is mathematically simple to work with—it is one of the few model classes that allows for an analytical treatment. Second, it provides a sufficiently accurate description of many interesting dynamical systems. Third, it is often used as a component in more complex models. Some concrete examples of this are provided in Section 2.7, where the so called conditionally linear state space model is introduced.

2.7 Conditionally linear Gaussian state space model

A conditionally linear Gaussian state space (CLGSS) model is an SSM with the property that conditioned on one part of the state, the remaining part constitutes an LGSS model. This is an interesting model, since its structure allows for exact inference of the conditionally linear Gaussian part of the state vector. This is a property that we wish to exploit when constructing algorithms for this type of models.

Definition 2 (Conditionally linear Gaussian state space (CLGSS) model)**.** *Assume that the state X_t of an SSM can be partitioned according to $X_t = (S_t, Z_t)$. The SSM is a CLGSS model if the conditional process $\{(Z_t, Y_t) \,|\, S_{1:t} = s_{1:t}\}_{t\geq 1}$ is described by an LGSS model.*

Note that in the above definition we made use of the list notation for the state, i.e. $X_t = (S_t, Z_t)$ allowing us to work with combinations of discrete and continuous state spaces. The reason for this rather abstract definition is that there are several different functional forms (some of which are explicitly introduced below), that all share the same fundamental property; conditioned on one part of the state, the remaining part behaves as an LGSS model. The Z_t-process is conditionally linear, motivating the name *linear state* for Z_t whereas S_t will be referred to as the *nonlinear state*. An important detail is that it is necessary to condition on the entire nonlinear trajectory $S_{1:t}$ for the conditional process Z_t to be linear Gaussian, i.e. to condition on just S_t is not sufficient. This is thoroughly explained in the following example, where we also introduce one commonly used instantiation of the CLGSS model, the so called *hierarchical CLGSS* model.

Example 2.6: Hierarchical CLGSS model

The hierarchical CLGSS model carries its name due to the hierarchical relationship between the variables S_t and Z_t, where S_t is at the top of the hierarchy, evolving according to a Markov kernel $p(s_{t+1} \,|\, s_t)$ independently of Z_t. The linear state Z_t evolves according to an LGSS model (parameterized by S_t). Hence, the model is given by

$$S_{t+1} \,|\, (S_t = s_t) \sim p(s_{t+1} \,|\, s_t), \tag{2.42a}$$

$$Z_{t+1} = f_t(S_t) + A(S_t)Z_t + V_t(S_t), \tag{2.42b}$$

$$Y_t = h_t(S_t) + C(S_t)Z_t + E_t(S_t), \tag{2.42c}$$

with Gaussian noise sources $V_t(S_t)$ and $E_t(S_t)$. The initial density is defined by $S_1 \sim \mu^s(s_1)$ and $Z_1 \,|\, (S_1 = s_1) \sim \mathcal{N}(\bar{z}_1(S_1), P(S_1))$. The corresponding graphical model is provided in Figure 2.6. We have assumed that the matrices A, C, etc. are functions of the nonlinear state S_t. However, for any fixed time $t \geq 1$ and conditioned on $S_{1:t}$, the sequences $\{A(S_k)\}_{k=1}^t$, $\{C(S_k)\}_{k=1}^t$, etc. are known. Hence, conditioned on $S_{1:t}$, (2.42b) - (2.42c) constitutes an LGSS model with state Z_t. As previously pointed out, to condition on just S_t is not sufficient. In that case, the sequence $\{A(S_k)\}_{k=1}^t$ (for instance) would consist of random elements, where only the final element is known.

Conditioning the model (2.42) on $S_{1:t}$ means that we can in fact remove the linear state Z_t from the model by marginalizing out the variables $Z_{1:t}$, effectively reduc-

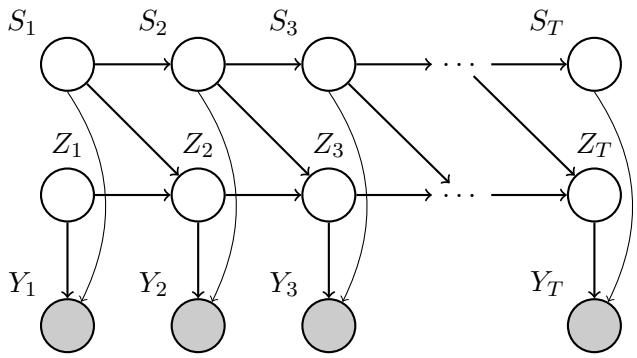

Figure 2.6: Graphical model for the hierarchical CLGSS model (2.42).

ing (2.42) to,

$$S_{t+1} \,|\, (S_t = s_t) \sim p(s_{t+1} \,|\, s_t), \tag{2.43a}$$
$$Y_t \,|\, (S_{1:t} = s_{1:t}, Y_{1:t} = y_{1:t}) \sim p(y_t \,|\, s_{1:t}, y_{1:t-1}). \tag{2.43b}$$

This reduced SSM is similar to the "full" SSM (**??**), save for the fact that the marginalization of the z-process resulted in a measurement model (2.43b) that depends on the complete history $s_{1:t}, y_{1:t-1}$.

In Section 2.7.1 and Section 2.7.2 we introduce two commonly studied members of the CLGSS model family, namely the switching linear Gaussian state space model (which is a special case of the hierarchical CLGSS model) and the mixed Gaussian state space model, respectively.

2.7.1 Switching linear Gaussian state space model

A popular special case of the CLGSS model is the so called switching linear Gaussian state space (SLGSS) model. The SLGSS model consists of a combination of the FSS model (Example 2.4) and the LG-SSM. Hence, the SLGSS model is constituted by a finite number (K) of LGSS models and a discrete switching variable $S_t \in (1, \ldots, K)$ indicating which of the K LGSS models that is to be used at time t. The switching variable itself is modelled as a Markov process on this finite state space, i.e. an FSS model. Alternatively the SLGSS model can be thought of as a generalization of the LGSS model, where we rather than making use of one global model instead make use of several LGSS models. Each model will then capture a specific aspect of the underlying dynamical system and the switching variable reveals how to switch between the different models.

Definition 3 (Switching linear Gaussian state space (SLGSS) model)**.** *The SLGSS*

model is defined according to

$$S_{t+1} \,|\, (S_t = s_t) \sim p(s_{t+1} \,|\, s_t), \tag{2.44a}$$
$$Z_{t+1} = A(S_t)Z_t + V(S_t), \tag{2.44b}$$
$$Y_t = C(S_t)Z_t + E(S_t), \tag{2.44c}$$

where $X_t = (S_t, Z_t)$ *and* $\mathsf{X} = (1, \ldots, K) \times \mathbb{R}^{n_z}$*, i.e.* $S_t \in (1, \ldots, K)$ *and* $Z_t \in \mathbb{R}^{n_z}$*. Furthermore,* $Y_t \in \mathbb{R}^{n_y}$ *denotes the observed measurement and the initial mode variable* S_1 *is distributed according to* $p(s_1)$*. The initial* Z_1 *and the two white noise sequences* $V_t \in \mathbb{R}^{n_x}$ *and* $E_t \in \mathbb{R}^{n_y}$ *are assumed Gaussian distributed according to* $Z_1 \sim \mathcal{N}(\mu, P_1)$ *and*

$$\begin{pmatrix} V(S_t) \\ E(S_t) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \bar{v}(S_t) \\ \bar{e}(S_t) \end{pmatrix}, \begin{pmatrix} Q(S_t) & S(S_t) \\ S(S_t)^{\mathsf{T}} & R(S_t) \end{pmatrix} \right). \tag{2.44d}$$

The state vector (using the list notation) $X_t = (S_t, Z_t)$ consists of both the discrete valued switching variable S_t and the continuous valued Z_t. The graphical model for the SLGSS model is the same as the one for the hierarchical CLGSS model provided in Figure 2.6, which results in the following expression for the joint PDF of all the states and all the measurements,

$$\begin{aligned} p(x_{1:T}, y_{1:T}) &= p(s_{1:T}, z_{1:T}, y_{1:T}) \\ &= p(z_1)p(s_1) \prod_{t=1}^{T-1} p(z_{t+1} \,|\, z_t, s_t) p(s_{t+1} \,|\, s_t) \prod_{t=1}^{T} p(y_t \,|\, s_t, z_t) \prod_{t=1}^{T}. \end{aligned} \tag{2.45}$$

The SLGSS model is one instance of a more general family of models commonly referred to as *hybrid* models. A hybrid model is a model where the state variable consists of both discrete states and continuous states. Other names used for the SLGSS model and slight variants thereof are jump Markov linear model and linear jump model.

2.7.2 Mixed Gaussian state space model

The mixed Gaussian state space (MGSS) model is given by

$$s_{t+1} = f_t^s(s_t) + A_t^s(s_t)z_t + V_t^s(s_t), \tag{2.46a}$$
$$z_{t+1} = f_t^z(s_t) + A_t^z(s_t)z_t + V_t^z(s_t), \tag{2.46b}$$
$$Y_t = h_t(s_t) + C_t(s_t)z_t + E_t(s_t), \tag{2.46c}$$

where the process noise $V_t(s_t) = \begin{pmatrix} V_t^s(s_t)^{\mathsf{T}} & V_t^z(s_t)^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}}$ and the measurement noise $E_t(s_t)$ are mutually independent and they are both white and Gaussian distributed according to

$$V_t(s_t) \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} Q^s(s_t) & Q^{sz}(s_t) \\ (Q^{sz}(s_t))^{\mathsf{T}} & Q^z(s_t) \end{pmatrix} \right) = \mathcal{N}(0, Q(s_t)), \tag{2.46d}$$
$$e_t(s_t) \sim \mathcal{N}(0, R(s_t)). \tag{2.46e}$$

In contrast to the SLGSS model, the MGSS model allows for an intricate cross-dependence between the linear and the nonlinear parts of the state vector. The result is that the nonlinear state process $\{s_t\}_{t\geq 1}$ alone is non-Markovian. This class of models arise, for instance, when the observations depend nonlinearly on a subset of the states in a system with linear dynamics.

In Definition 4 we formally define the MGSS model using a more compact notation compared to (2.46), which will result in clearer equations in the upcoming inference algorithms, since some of the details are hidden in this form.

Definition 4 (Mixed Gaussian state space (MGSS) model)**.** *The MGSS model is defined according to*

$$X_{t+1} = f_t(s_t) + A_t(s_t)z_t + V_t(s_t), \tag{2.47a}$$
$$Y_t = h_t(s_t) + C_t(s_t)z_t + E_t(s_t), \tag{2.47b}$$

where

$$X_t = \begin{pmatrix} s_t \\ z_t \end{pmatrix}, \qquad f_t(s_t) = \begin{pmatrix} f_t^s(s_t) \\ f_t^z(s_t) \end{pmatrix}, \qquad A_t(s_t) = \begin{pmatrix} A_t^s(s_t) \\ A_t^z(s_t) \end{pmatrix}. \tag{2.47c}$$

The process noise $V_t(s_t)$ and the measurement noise $E_t(s_t)$ are assumed to be mutually independent, white and Gaussian distributed according to (2.46d)-(2.46e)*. The initial distribution of the state is given by $s_t \sim p(s_t)$ and $s_1 \,|\, (s_1 = s_1) \sim \mathcal{N}(\mu(s_1), P_1(s_1))$.*

The graphical model for the MGSS model is given in Figure 2.7. The more intricate cross dependence between the linear parts and the nonlinear parts of the state that is present in the MGSS model compared to the SLGSS model is graphically illustrated by comparing Figure 2.6 and Figure 2.7.

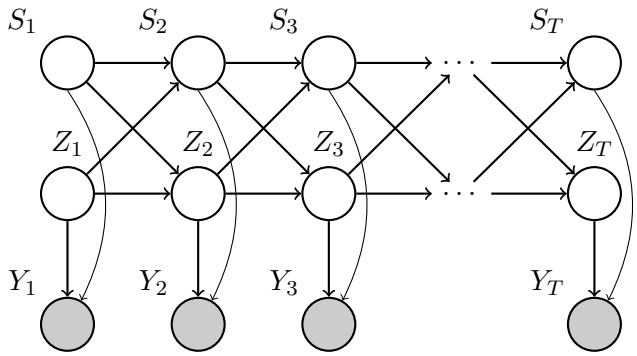

Figure 2.7: Graphical model for the mixed Gaussian state space model in Definition 4.

2.8 History and further reading

(Jazwinski, 1970) (Cappé et al., 2005) (Kailath et al., 2000) (Verhaegen and Verdult, 2007). Gelman et al. (2003), (Denison et al., 2002) Koller and Friedman (2009) Jordan (2004) Cappé et al. (2005), Douc et al. (2014) Rabiner (1989) Fox et al. (2011)

Chapter 3

Inference and learning strategies

The aim in this chapter is to provide the basic strategies that will be used throughout the rest of the manuscript in solving various inference problems in dynamical systems. These strategies will identify certain key problems that have to be solved. While there are no analytical solutions to these problems (save for some restrictive special cases), there are computational methods available allowing us to implement the strategies by approximately solving these problems. The inference problems we are interested in can broadly be classified as either state inference or (static) parameter inference.

The *state inference problem* deals with finding information about the states using the information available in the measurements. We will throughout this chapter make use of the SSM introduced in (2.30) to model the dynamical system. The information about the state is represented using PDFs on the form $p(x_{r:t} \mid y_{1:s})$, where $y_{1:s} \triangleq \{y_1, y_2, \ldots, y_s\}$ denotes the set of available measurements. Depending on the relationship between the time indices r, t and s this problem falls into one of the following three main categories; filtering, prediction or smoothing. A more detailed description of the most commonly encountered state inference problems is provided in Table 3.1. Note that whenever we discuss state inference we will without loss of generality suppress the model's possible dependence on an unknown parameter θ in the interest of readability.

We start out in Section 3.2 by explaining how to compute these PDFs by only traversing the data once in the forward direction, i.e. from $t = 1$ to $t = T$, commonly

Table 3.1: Filtering and smoothing densities of particular interest.

	PDF
Marginal filtering	$p(x_t \mid y_{1:t})$
Joint filtering	$p(x_{0:t} \mid y_{1:t})$, $t = 0, 1, \ldots, T-1$
Prediction	$p(x_{t+1} \mid y_{1:t})$
k-step prediction	$p(x_{t+k} \mid y_{1:t})$
Joint smoothing	$p(x_{1:T} \mid y_{1:T})$
Marginal smoothing ($t \leq T$)	$p(x_t \mid y_{1:T})$
Fixed-interval smoothing ($s < t \leq T$)	$p(x_{s:t} \mid y_{1:T})$
Fixed-lag smoothing (l fixed)[a]	$p(x_{t-l+1:t} \mid y_{1:t})$

referred to as forward computations. In dynamical systems the future holds information about the past, which makes it interesting to perform backwards computations, i.e. traversing the data from $t = T$ to $t = 1$, which is performed in various ways in Section 3.3. The strategies derived in Section 3.2 and Section 3.3 are then used throughout the manuscript as targets for the approximative algorithms that will be derived and analyzed. Throughout this chapter we will provide several different strategies for computing the same object. The reason is that different strategies typically opens up for different algorithms to be derived, which we will see in later chapters.

The task in *parameter inference* is to find information about the static parameters θ of the model based on the information provided by the measurements. We place equal relevance to both the maximum likelihood (Section 3.5.2) approach and the Bayesian approach (Section 3.5.3) when it comes to inferring static parameters and various mixes of the two are indeed becoming more and more interesting.

3.1 State inference

3.2 Forward computations

The forward computations amount to computing the densities mentioned in Table 3.1 sequentially in time from $t = 0$ to $t = T$.

3.2.1 Forward filtering

The filtering problem amounts to recovering information about the state X_t given the information present in the measurements $y_{1:t}$ by computing the filtering PDF $p(x_t \mid y_{1:t})$. From application of Bayes' theorem we have

$$p(x_t \mid y_{1:t}) = p(x_t \mid y_t, y_{1:t-1}) = \frac{p(y_t \mid x_t, y_{1:t-1})p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}. \tag{3.1}$$

The measurements from an SSM are conditionally independent, which results in

$$p(x_t \mid y_{1:t}) = \frac{p(y_t \mid x_t)p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}. \tag{3.2}$$

which is commonly referred to as the *measurement update*. It requires the prediction PDF $p(x_t \mid y_{1:t-1})$, which we can find based on objects we have. To see this we start by considering the joint PDF $p(x_t, x_{t-1} \mid y_{1:t-1})$, which can be written as,

$$p(x_t, x_{t-1} \mid y_{1:t-1}) = p(x_t \mid x_{t-1}, y_{1:t-1})p(x_{t-1} \mid y_{1:t-1}). \tag{3.3}$$

The Markov property results in

$$p(x_t, x_{t-1} \mid y_{1:t-1}) = p(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1}). \tag{3.4}$$

The prediction PDF $p(x_t \,|\, y_{1:t-1})$ is now obtained simply by marginalizing (3.4) w.r.t. X_{t-1},

$$p(x_t \,|\, y_{1:t-1}) = \int p(x_t, x_{t-1} \,|\, y_{1:t-1}) \mathrm{d}x_{t-1} = \int p(x_t \,|\, x_{t-1}) p(x_{t-1} \,|\, y_{1:t-1}) \mathrm{d}x_{t-1}, \quad (3.5)$$

which is referred to as the *time update*. Summarizing the above development we have

$$p(x_t \,|\, y_{1:t}) = \frac{\overbrace{p(y_t \,|\, x_t)}^{\text{measurement}} \overbrace{p(x_t \,|\, y_{1:t-1})}^{\text{prediction}}}{p(y_t \,|\, y_{1:t-1})}, \qquad \text{(Measurement update)} \quad (3.6a)$$

$$p(x_t \,|\, y_{1:t-1}) = \int \underbrace{p(x_t \,|\, x_{t-1})}_{\text{dynamics}} \underbrace{p(x_{t-1} \,|\, y_{1:t-1})}_{\text{filtering}} \mathrm{d}x_{t-1}, \qquad \text{(Time update)} \quad (3.6b)$$

Hence, the filtering density at time t can be computed from the filtering density at time $t-1$ via a two-step procedure (3.6). The time update performs a prediction of the state one step ahead in time. The functional form (3.6b) makes intuitive sense for the following reasons. The information we have about the state at time t is encapsulated in the filtering PDF $p(x_t \,|\, y_{1:t})$. Furthermore, in the absence of future measurements, the dynamics $p(x_t \,|\, x_{t-1})$ is the only source of information we have about what happens to the state in the future. Hence, it is natural that these objects are used in computing the time update.

In order to get some intuition for the measurement update let us first note that the denominator in (3.6a) is a normalizing factor. By marginalizing $p(y_t, x_t \,|\, y_{1:t-1})$ w.r.t. X_t this normalizing factor can be computed according to

$$p(y_t \,|\, y_{1:t-1}) = \int p(y_t \,|\, x_t) p(x_t \,|\, y_{1:t-1}) \mathrm{d}x_t. \quad (3.7)$$

Hence, in the light of (3.6a) and (3.7) we see that the measurement update is constituted by a combination of two objects, the measurement model $p(y_t \,|\, x_t)$ and the prediction PDF $p(x_t \,|\, y_{1:t-1})$. Thinking about what we know about the state at time t, this seems reasonable, since there are two sources of information; the current measurement y_t and all the previous measurements $y_{1:t-1}$. The relationship between the new measurement and the state is encoded via the measurement model. Finally, the prediction PDF $p(x_t \,|\, y_{1:t-1})$ efficiently encapsulates everything that is known about the state X_t based on the information in all the previous measurements $y_{1:t-1}$.

Let us also note that the k-step prediction PDF $p(x_{t+k} \,|\, y_{1:t})$ is obtained by integrating $p(x_{t:t+k} \,|\, y_{1:t})$ w.r.t. $x_{t:t+k-1}$, resulting in

$$\begin{aligned} p(x_{t+k} \,|\, y_{1:t}) &= \int \prod_{l=t}^{t+k} p(x_{l+1} \,|\, x_l) p(x_{t:t+k} \,|\, y_{1:t}) \mathrm{d}x_{t:t+k-1} \\ &= \int \prod_{l=t}^{t+k} p(x_{l+1} \,|\, x_l) p(x_t \,|\, y_{1:t}) \mathrm{d}x_{t:t+k-1}. \end{aligned} \quad (3.8)$$

The above expression corresponds to iteratively making use of the dynamics in order to compute the prediction. As mentioned above, without any new measurements, the only information available about what might happen in the future is provided by the dynamics. In the subsequent example the general expressions derived above are employed to solve a particular special case.

Example 3.1: Kalman filter

Let us consider the linear Gaussian special case (2.38), with $S = 0$ for simplicity. All densities associated to the LG-SSM are Gaussian, due to the fact that an affine transformation of a Gaussian variable is still Gaussian. In this example it will be shown how to compute the filtering PDF $p(x_t \mid y_{1:t})$ and prediction PDF $p(x_{t+1} \mid y_{1:t})$ simply by inserting the model (2.38) into (3.6). We will make use of the results on how to manipulate Gaussian variables provided in Appendix B.3.1.

Let us assume that

$$p(x_t \mid y_{1:t-1}) = \mathcal{N}\left(x_t \mid \widehat{x}_{t\mid t-1},\, P_{t\mid t-1}\right). \tag{3.9}$$

The measurement model (2.38b) involves an affine transformation

$$p(y_t \mid x_t, y_{1:t-1}) = \mathcal{N}\left(y_t \mid Cx_t + Du_t,\, R\right). \tag{3.10}$$

The measurement update (3.6a) can now be realized by direct use of Corollary 1 (with $x_a = x_t$ and $x_b = y_t$), resulting in

$$p(x_t \mid y_{1:t}) = \mathcal{N}\left(x_t \,\middle|\, \widehat{x}_{t\mid t-1} + K_t(y_t - C\widehat{x}_{t\mid t-1} - Du_t),\, P_{t\mid t-1} - K_t L_t K_t^{\mathsf{T}}\right) \tag{3.11a}$$

$$\triangleq \mathcal{N}\left(x_t \mid \widehat{x}_{t\mid t},\, P_{t\mid t}\right), \tag{3.11b}$$

where $K_t = P_{t\mid t-1}C^{\mathsf{T}}L_t$ and $L_t = (CP_{t\mid t-1}C^{\mathsf{T}} + R)^{-1}$.

The time update amounts to computing the integral (3.6b) for the LG-SSM. Let us start by noting that the dynamics (2.38a) involve an affine transformation

$$p(x_{t+1} \mid x_t) = \mathcal{N}\left(x_{t+1} \mid Ax_t + Bu_t,\, Q\right), \tag{3.12}$$

which together with Theorem 9 allows us to conclude

$$p(x_t, x_{t+1} \mid y_{1:t}) = \mathcal{N}\left(\begin{pmatrix} x_t \\ x_{t+1} \end{pmatrix} \,\middle|\, \begin{pmatrix} \widehat{x}_{t\mid t} \\ A\widehat{x}_{t\mid t} + Bu_t \end{pmatrix}, \begin{pmatrix} P_{t\mid t} & P_{t\mid t}A^{\mathsf{T}} \\ AP_{t\mid t} & AP_{t\mid t}A^{\mathsf{T}} + Q \end{pmatrix}\right). \tag{3.13}$$

Recall that $p(x_t, x_{t+1} \mid y_{1:t}) = p(x_{t+1} \mid x_t)p(x_t \mid y_{1:t})$, which explains why the time update (3.6b) is realized by marginalizing (3.13) w.r.t. x_t. Direct use of Corollary 1 results in

$$p(x_{t+1} \mid y_{1:t}) = \mathcal{N}\left(x_{t+1} \,\middle|\, A\widehat{x}_{t\mid t} + Bu_t,\, AP_{t\mid t}A^{\mathsf{T}} + Q\right) \triangleq \mathcal{N}\left(x_{t+1} \mid \widehat{x}_{t+1\mid t},\, P_{t+1\mid t}\right). \tag{3.14}$$

The measurement update (3.11) and the time update (3.14) are collectively referred to as Kalman filter (KF) providing the recursions explaining how to compute the filtering and the prediction PDFs as new measurements arrives.

As we saw in Example 3.1 it is very natural to structure the derivation of the state inference algorithm (i.e. the Kalman filter) for the LG-SSM according to the time update (3.6b) and measurement update (3.6a) equations. There are several other models where it also makes sense to start the derivation of the inference algorithm from (3.6), but for an arbitrary SSM it can be unnecessarily restrictive to enforce the explicit structure of a time update and a measurement update in deriving an inference algorithm. Instead it can for example be beneficial to combine the two equations into one, according to

$$p(x_t \mid y_{1:t}) = \frac{p(y_t \mid x_t)}{p(y_t \mid y_{1:t-1})} \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) \mathrm{d}x_{t-1}. \tag{3.15}$$

This will for example allow for a more efficient use of the current measurement y_t in deriving sequential Monte Carlo algorithms, as we will see in Chapter 5.

3.2.2 Forward smoothing

The JSD $p(x_{0:t} \mid y_{1:t})$ can be computed sequentially according to the following two-step procedure (obtained using Bayes' theorem),

$$p(x_{0:t} \mid y_{1:t}) = \frac{p(y_t \mid x_t) p(x_{0:t} \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}, \tag{3.16a}$$

$$p(x_{0:t+1} \mid y_{1:t}) = p(x_{t+1} \mid x_t) p(x_{0:t} \mid y_{1:t}). \tag{3.16b}$$

The above equations will be denoted the forward recursion for the JSD, since they evolve forward in time. The filtering PDF (3.6a) is the marginal of (3.16a) that is obtained by integrating over $x_{0:t-1}$.

3.3 Backward computations

State inference via backward computations amounts to propagating information backwards in time from $t = T$ to $t = 0$. In this section we provide the strategies used in structuring these computations.

3.3.1 The JSD and the backward kernel

Repeated use of conditional probabilities results in the following representation of the JSD,

$$\begin{aligned} p(x_{0:T} \mid y_{1:T}) &= p(x_0 \mid x_{1:T},\ y_{1:T}) p(x_{1:T} \mid y_{1:T}) = \dots \\ &= p(x_T \mid y_{1:T}) \prod_{t=0}^{T-1} p(x_t \mid x_{t+1:T},\ y_{1:T}). \end{aligned} \tag{3.17}$$

To clearly see what this product entails, let us study the PDF $p(x_t \mid x_{t+1:T},\ y_{1:T})$ in more detail. First of all, the Markov property allows us to conclude that $p(x_t \mid x_{t+1:T}, y_{1:T}) =$

$p(x_t \,|\, x_{t+1}, y_{1:T})$. Secondly, the measurements are conditionally independent, given the state, resulting in $p(x_t|x_{t+1}, y_{1:t}, y_{t+1:T}) = p(x_t|x_{t+1}, y_{1:t})$. An interpretation of this is that given the state at time $t+1$, there is no additional information about the state x_t available in the measurements $y_{t+1:T}$, nor in the states $x_{t+2:T}$. To summarize we have now shown that

$$p(x_t \,|\, x_{t+1:T},\ y_{1:T}) = p(x_t \,|\, x_{t+1},\ y_{1:t}), \tag{3.18}$$

which inserted into (3.17) results in the following expression for the JSD

$$p(x_{0:T} \,|\, y_{1:T}) = \left(\prod_{t=0}^{T-1} p(x_t \,|\, x_{t+1},\ y_{1:t}) \right) p(x_T \,|\, y_{1:T}). \tag{3.19}$$

The PDF $p(x_t \,|\, x_{t+1},\ y_{1:t})$ is often referred to as the *backward kernel*, since it works backwards in time. The backward kernel can be computed by noting that

$$p(x_t \,|\, x_{t+1},\ y_{1:t}) = \frac{p(x_t, x_{t+1} \,|\, y_{1:t})}{p(x_{t+1} \,|\, y_{1:t})} = \frac{p(x_{t+1} \,|\, x_t) p(x_t \,|\, y_{1:t})}{\int p(x_{t+1} \,|\, x_t) p(x_t \,|\, y_{1:t}) \mathrm{d}x_t}, \tag{3.20}$$

where all densities on the right hand side are familiar. Inserting (3.20) into (3.19) gives

$$p(x_{0:T} \,|\, y_{1:T}) = \prod_{t=0}^{T-1} \frac{p(x_{t+1} \,|\, x_t) p(x_t \,|\, y_{1:t})}{p(x_{t+1} \,|\, y_{1:t})} p(x_T \,|\, y_{1:T}). \tag{3.21}$$

Alternatively we can make use of the backward kernel to obtain the following recursion for the JSD, evolving backward in time,

$$p(x_{t:T} \,|\, y_{1:T}) = p(x_t \,|\, x_{t+1}, y_{1:t}) p(x_{t+1:T} \,|\, y_{1:T}), \tag{3.22}$$

starting with the filtering density at time T, $p(x_T \,|\, y_{1:T})$. This is known as the backward recursion. At time $t = 0$, the JSD for the time interval $0, \ldots, T$ is obtained.

3.3.2 Marginal smoothing densities

The marginal smoothing density $p(x_t \,|\, y_{1:T})$ can of course be obtained by marginalizing (3.21). Yet another strategy for computing the marginal smoothing density is obtained by starting from the observation that

$$p(x_t \,|\, y_{1:T}) = \int p(x_t, x_{t+1} \,|\, y_{1:T}) \mathrm{d}x_{t+1}, \tag{3.23}$$

where $p(x_t, x_{t+1} \,|\, y_{1:T}) = p(x_t \,|\, x_{t+1}, y_{1:T}) p(x_{t+1} \,|\, y_{1:T})$. Furthermore, from (3.18) we know that $p(x_t \,|\, x_{t+1}, y_{1:T}) = p(x_t \,|\, x_{t+1}, y_{1:t})$, which allows us to conclude that

$$p(x_t, x_{t+1} \,|\, y_{1:T}) = p(x_t \,|\, x_{t+1}, y_{1:t}) p(x_{t+1} \,|\, y_{1:T}). \tag{3.24}$$

Inserting (3.24) and (3.20) into (3.23) results in the following expression for the marginal smoothing density

$$p(x_t \,|\, y_{1:T}) = p(x_t \,|\, y_{1:t}) \int \frac{p(x_{t+1} \,|\, x_t) p(x_{t+1} \,|\, y_{1:T})}{p(x_{t+1} \,|\, y_{1:t})} \mathrm{d}x_{t+1}. \tag{3.25}$$

3.4 Forward and backward computations

The backward kernel density (3.20) at time t depends only on the transition density $p(x_{t+1} \mid x_t)$ and on the filtering density $p(x_t \mid y_{1:t})$, a property which is of key relevance. Hence, to utilise the backward recursion (3.22) for computing the JSD, the filtering densities must first be computed for $t = 1, \ldots, T$.

3.4.1 Forward filtering backward smoothing

Consequently, this procedure is generally called forward filtering/backward smoothing.

Example 3.2: RTS smoother

The LGSS model only involves linear transformations of Gaussian variables, which implies that all manipulations can be efficiently performed using the properties of Gaussian variables provided in Appendix B.3.1. More specifically, the strategy we employ is to marginalise (3.24) w.r.t. x_{t+1} according to (3.23). In order to do this we start by finding an expression for the backward kernel $p(x_t \mid x_{t+1}, y_{1:t})$. Note that

$$p(x_{t+1} \mid x_t) = \mathcal{N}\left(x_{t+1} \mid Ax_t + Bu_t,\, Q\right), \tag{3.26a}$$
$$p(x_t \mid y_{1:t}) = \mathcal{N}\left(x_t \mid \widehat{x}_{t\mid t},\, P_{t\mid t}\right), \tag{3.26b}$$

where the last equality is provided by the Kalman filter (recall Example 3.1). Using (3.26) together with Corollary 1 (with $x_a = x_t, x_b = x_{t+1}$ and the fact that all densities are conditioned w.r.t. $y_{1:t}$) shows that the backward kernel is given by

$$\begin{aligned} p(x_t \mid x_{t+1}, y_{1:t}) &= \mathcal{N}\left(x_t \,\middle|\, \widehat{x}_{t\mid t} + J_t\left(x_{t+1} - Bu_t - A\widehat{x}_{t\mid t}\right),\, P_{t\mid t} - J_t A P_{t\mid t}\right) \\ &= \mathcal{N}\left(x_t \,\middle|\, J_t x_{t+1} - J_t\widehat{x}_{t+1\mid t} + \widehat{x}_{t\mid t},\, P_{t\mid t} - J_t A P_{t\mid t}\right), \end{aligned} \tag{3.27}$$

where $J_t = P_{t\mid t}A^{\mathsf{T}}(AP_{t\mid t}A^{\mathsf{T}}+Q)^{-1}$. Using (3.27) together with $p(x_{t+1} \mid y_{1:T}) = \mathcal{N}(x_{t+1} \mid \widehat{x}_{t+1\mid T}, P_{t+1\mid T})$ and Corollary 1 (this time with $x_a = x_{t+1}, x_b = x_t$) results in

$$p(x_t \mid y_{1:T}) = \mathcal{N}\left(x_t \,\middle|\, J_t\widehat{x}_{t+1\mid T} - J_t\widehat{x}_{t+1\mid t} + \widehat{x}_{t\mid t},\, P_{t\mid t} - J_t A P_{t\mid t} + J_t P_{t+1\mid T} J_t^{\mathsf{T}}\right). \tag{3.28}$$

This shows that $\widehat{x}_{t\mid T} = \widehat{x}_{t\mid t} + J_t\left(\widehat{x}_{t+1\mid T} - \widehat{x}_{t+1\mid t}\right)$. Furthermore, from (3.28) we have (assuming that J_t is invertible)

$$P_{t\mid T} = P_{t\mid t} + J_t P_{t+1\mid T} J_t^{\mathsf{T}} - J_t A P_{t\mid t} J_t^{-\mathsf{T}} J_t^{\mathsf{T}} = \cdots = P_{t\mid t} + J_t\left(P_{t+1\mid T} - P_{t+1\mid t}\right) J_t^{\mathsf{T}}, \tag{3.29}$$

which concludes our derivation. To summarize, we have now showed that the marginal smoothing PDF for the LGSS model is given by $p(x_t \mid y_{1:T}) = \mathcal{N}\left(x_t \mid \widehat{x}_{t\mid T}, P_{t\mid T}\right)$, where

$$\widehat{x}_{t\mid T} = \widehat{x}_{t\mid t} + J_t\left(\widehat{x}_{t+1\mid T} - \widehat{x}_{t+1\mid t}\right), \tag{3.30a}$$
$$P_{t\mid T} = P_{t\mid t} + J_t\left(P_{t+1\mid T} - P_{t+1\mid t}\right) J_t^{\mathsf{T}}, \tag{3.30b}$$
$$J_t = P_{t\mid t}A^{\mathsf{T}}(AP_{t\mid t}A^{\mathsf{T}} + Q)^{-1} = P_{t\mid t}A^{\mathsf{T}}P_{t+1\mid t}^{-1}. \tag{3.30c}$$

This smoother is commonly referred to as the RTS smoother after Rauch, Tung and Striebel, who back in 1965 were the first to derive this solution.

3.4.2 Forward filtering backward simulation

Backward simulation is a strategy for generating realizations of latent variables in probabilistic models. The strategy is based on the forward-backward idea introduced above. In the backward pass, the state process is simulated backward in time, i.e. by first simulating x_T, then x_{T-1} etc., until a complete state trajectory $x_{1:T}$ is generated. This procedure gives us a tool to address the state smoothing problem in models for which no closed form solution is available. This is done by simulating multiple backward trajectories from the smoothing distribution, i.e. conditionally on the observations $y_{1:T}$. These samples can then be used for example in Monte Carlo integration.

The backward simulation strategy suggets itself from the following factorization of the JSD,

$$p(x_{0:T} \mid y_{1:T}) = \left(\prod_{t=0}^{T-1} p(x_t \mid x_{t+1},\ y_{1:t}) \right) p(x_T \mid y_{1:T}). \tag{3.31}$$

previously derived in (3.19). Initially, we generate a sample from the filtering density at time T,

$$\widetilde{x}_T \sim p(x_T \mid y_{1:T}). \tag{3.32a}$$

We then, successively, augment this *backward trajectory* by generating samples from the backward kernel,

$$\widetilde{x}_t \sim p(x_t \mid \widetilde{x}_{t+1}, y_{1:t}), \tag{3.32b}$$

for $t = T-1, \ldots, 1$. After a complete backward sweep, the backward trajectory $\widetilde{x}_{1:T}$ is (by construction) a realization from the JSD (3.31). The strategy given by (3.32), i.e. to sequentially sample (either exactly or approximately) from the backward kernel to generate a realization from the JSD, is what we collectively refer to as *backward simulation*.

An important question that remains to be answered in order to take this from a strategy to an implementable algorithm is how to generate samples from the backward kernel. For reasons of illustration let us who how this is done for the LG-SSM, where the backward kernel is Gaussian enabling exact backward simulation.

Example 3.3: Exact backward simulation in LGSS models

For an LG-SSM (2.38) it is possible to generate an exact realization from the JSD using backward simulation (3.32). To compute the backward kernel, we first run a forward filter (i.e. the Kalman filter according to Example 3.1) to find the filtering densities $p(x_t \mid y_{1:t})$ for $t = 1, \ldots, T$. According to (3.27) the backward kernel is given by

$$p(x_t \mid x_{t+1}, y_{1:t}) = \mathcal{N}(x_t \mid \mu_t, L_t), \tag{3.33a}$$

with

$$\mu_t = \hat{x}_{t \mid t} + P_{t \mid t} A^{\mathsf{T}} (A P_{t \mid t} A^{\mathsf{T}} + Q)^{-1} (x_{t+1} - A\hat{x}_{t \mid t}), \tag{3.33b}$$

$$L_t = P_{t \mid t} - P_{t \mid t} A^{\mathsf{T}} (A P_{t \mid t} A^{\mathsf{T}} + Q)^{-1} A P_{t \mid t}. \tag{3.33c}$$

Note that, if more than one sample is desired, multiple backward trajectories can be generated independently, without having to rerun the forward Kalman filter. To illustrate the possibility of generating samples from the JSD using backward simulation, we consider a first order LG-SSM,

$$x_{t+1} = 0.9x_t + v_t, \qquad v_t \sim \mathcal{N}(0, 0.1), \tag{3.34a}$$

$$y_t = x_t + e_t, \qquad e_t \sim \mathcal{N}(0, 1), \tag{3.34b}$$

and $x_1 \sim \mathcal{N}(0, 10)$. We simulate $T = 50$ samples $y_{1:T}$ from the model. Since the model is linear Gaussian, the marginal smoothing densities $p(x_t \mid y_{1:T})$ can be computed exactly according to the RTS smoother in Example 3.2. However, we can also generate samples from the JSD $p(x_{1:T} \mid y_{1:T})$ by running a backward simulator. We simulate $M = 5\,000$ independent trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$, according to the above development. Histograms over the simulated states at three specific time points, $t = 1$, $t = 25$ and $t = 50$, are given in Figure 3.1. As expected, the histograms are in close agreement with the true marginal smoothing distributions.

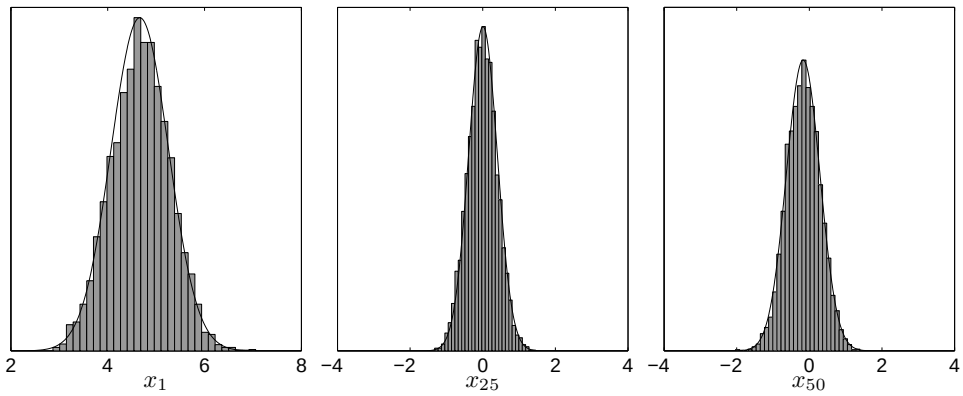

Figure 3.1: Histograms of $\{\widetilde{x}_t^j\}_{j=1}^M$ for $t = 1$, $t = 25$ and $t = 50$ (from left to right). The true marginal smoothing densities $p(x_t \mid y_{1:T})$ are shown as black lines.

Backward simulation is a powerful strategy when it comes to state inference in general nonlinear/non-Gaussian SSMs and also for more general non-Markovian latent variable models. For these models it is no longer possible to perform exact backward simulation. However, as we will see in later chapters we can make use of SMC and MCMC samplers to construct interesting and useful approximate backwards simulators.

3.4.3 Two-filter smoothing

The two-filter smoothing strategy is based on one filter working forward in time and one filter working backwards in time. The smoothing density the then computed by merging the result from the two (independent) filters. We use the following factorization of the marginal smoothing density to reveal the structure that is underlying the two-filter smoothing strategy,

$$\begin{aligned} p(x_t \mid y_{1:T}) = p(x_t \mid y_{1:t-1}, y_{t:T}) &= \frac{p(x_t, y_{t:T} \mid y_{1:t-1})}{p(y_{t:T} \mid y_{1:t-1})} = \frac{p(y_{t:T} \mid x_t, y_{1:t-1})p(x_t \mid y_{1:t-1})}{p(y_{t:T} \mid y_{1:t-1})} \\ &= \frac{p(y_{t:T} \mid x_t)p(x_t \mid y_{1:t-1})}{p(y_{t:T} \mid y_{1:t-1})} \propto \underbrace{p(x_t \mid y_{1:t-1})}_{\text{filer one}} \underbrace{p(y_{t:T} \mid x_t)}_{\text{filer two}}, \end{aligned} \tag{3.35}$$

where the proportionality is valid since $p(y_{t:T} \mid y_{1:t-1})$ does not depend on x_t. Filter one is provided according to the standard forward filtering recursion (3.6). The backwards recursion for filter two (sometimes referred to as the *backward information filter*) is given by

$$p(y_{t:T} \mid x_t) = p(y_t \mid x_t) \int p(y_{t+1:T} \mid x_{t+1}) p(x_{t+1} \mid x_t) \mathrm{d}x_{t+1}, \tag{3.36}$$

which advantageously may be decomposed into a two-step procedure according to

$$p(y_{t:T} \mid x_t) = p(y_{t+1:T}, y_t \mid x_t) = p(y_{t+1:T} \mid x_t) p(y_t \mid x_t). \tag{3.37a}$$

where

$$p(y_{t+1:T} \mid x_t) = \int p(y_{t+1:T}, x_{t+1} \mid x_t) \mathrm{d}x_{t+1} = \int p(y_{t+1:T} \mid x_{t+1}) p(x_{t+1} \mid x_t) \mathrm{d}x_{t+1}. \tag{3.37b}$$

The object $p(y_{t:T} \mid x_t)$ computed by filter two is not necessarily a PDF in the argument x_t, implying that the integral over x_t is not necessarily finite, which is important to acknowledge in trying to realize this strategy. For intuition it is worth noting that the object $p(y_{t:T} \mid x_t)$ is similar to the likelihood with respect to the state.

3.5 Parameter learning

Two commonly used parameter learning formalisms are grounded in frequentistic and Bayesian statistics, respectively. We will treat both of these formulations in this manuscript, without making any individual ranking among them.

The frequentistic, or *maximum likelihood* method is based on the rather natural idea that the unknown parameters should be chosen in such a way that the observed measurements becomes *as likely as possible*. This is achieved by maximizing the observed data likelihood function w.r.t. the unknown parameters,

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} p(Y_{1:T} = y_{1:T} \,|\, \theta) = \arg\max_{\theta \in \Theta} \log p(Y_{1:T} = y_{1:T} \,|\, \theta). \tag{3.38}$$

Hence, the maximum likelihood method amounts to computing *point estimates* $\widehat{\theta}$ by making use of a particular realization of the measurements $\{y_t\}_{t=1}^T$. Note that an underlying assumption is that the unknown parameters θ are modelled as deterministic variables.

In the *Bayesian* formulation, the unknown parameters θ are modeled as a random variable. The parameter learning problem thus amounts to computing the *posterior* distribution of θ condition on the observed measurements. According to Bayes' theorem, the posterior distribution is given by,

$$p(\theta \,|\, y_{1:T}) = \frac{p(y_{1:T} \,|\, \theta) p(\theta)}{p(y_{1:T})}. \tag{3.39}$$

The *central object* in both the frequentistic and Bayesian formulations above is the data distribution $p(y_{1:T} \,|\, \theta)$, motivating Section 3.5.1, which examines this object further. The maximum likelihood formulation is then described in Section 3.5.2 and the Bayesian formulation in Section 3.5.3.

3.5.1 Data distribution

The data distribution is the PDF of the observed measurements conditioned on the model parameters θ, i.e. $p(y_{1:T} \,|\, \theta)$. The data distribution for the general SSM can be computed by marginalizing the joint distribution of the latent states and the observed measurements

$$p(x_{0:T}, y_{1:T} \,|\, \theta) = p(x_0 \,|\, \theta) \prod_{t=1}^{T} p_\theta(y_t \,|\, x_t) \prod_{t=1}^{T} p(x_t \,|\, x_{t-1} \,|\, \theta), \tag{3.40}$$

w.r.t. the latent state sequence $x_{1:T}$ according to

$$p(y_{1:T} \,|\, \theta) = \int p(x_{0:T}, y_{1:T} \,|\, \theta) \mathrm{d}x_{0:T}. \tag{3.41}$$

One interpretation of (3.41) is that the data distribution $p(y_{1:T} \,|\, \theta)$ is obtained by averaging the joint PDF of the states and the measurements $p(x_{0:T}, y_{1:T} \,|\, \theta)$ over all possible state trajectories $x_{0:T}$.

Alternatively we can note that the data distribution $p(y_{1:T} \,|\, \theta)$ can be computed via repeated use of conditional probabilities according to $p(y_T, y_{1:T-1} \,|\, \theta) = p(y_T \,|\, y_{1:T-1}, \theta) p(y_{1:T-1} \,|\, \theta)$, resulting in,

$$p(y_{1:T} \,|\, \theta) = \prod_{t=1}^{T} p(y_t \,|\, y_{1:t-1} \,|\, \theta), \tag{3.42}$$

where we have made use of the convention $y_{1:0} \triangleq \emptyset$. Hence, the data distribution is obtained as the product of the one step ahead predictors of the measurements. These one step ahead predictors can be computed by marginalizing the joint PDF $p_\theta(x_t, y_{1:t}) = p_\theta(y_t \,|\, x_t)p_\theta(x_t \,|\, y_{1:t-1})$ w.r.t. the state x_t,

$$p_\theta(y_{1:T}) = \prod_{t=1}^{T} \int p_\theta(y_t \,|\, x_t)p_\theta(x_t \,|\, y_{1:t-1})\mathrm{d}x_t. \tag{3.43}$$

From the above development we can see that in order to compute the observed data likelihood we have to solve a state inference problem in order to find information about the latent states. This also highligts the tight relationship between the parameter learning problem and the state inference problem. As a further consequence of this we can also conclude that in general the observed data likelihood is not available in closed form. One simple special case where there exists a closed form expression for the observed data likelihood is provided by the LG-SSM.

Example 3.4: The observed data likelihood in an LG-SSM

This example will show how to compute the observed data likelihood for the LG-SSM (2.38) (with $S = 0$ for simplicity). Let us start by recalling that according to Example 3.1 we have that $p(x_t \,|\, y_{1:t-1}) = \mathcal{N}\left(x_t \,\middle|\, \widehat{x}_{t\,|\,t-1},\, P_{t\,|\,t-1}\right)$, where $\widehat{x}_{t\,|\,t-1}$ and $P_{t\,|\,t-1}$ are provided by the Kalman filter. Inserting this expression into (3.7) provides an explicit expression for the one step ahead predictor of the measurement

$$p(y_t \,|\, y_{1:t-1}.\theta) = \mathcal{N}\left(y_t \,\middle|\, C\widehat{x}_{t\,|\,t-1},\, CP_{t\,|\,t-1}C^{\mathsf{T}} + R\right), \tag{3.44}$$

which inserted into (3.42) provides the following expression

$$p(y_{1:t} \,|\, \theta) = \prod_{t=1}^{T} \mathcal{N}\left(y_t \,\middle|\, C\widehat{x}_{t\,|\,t-1},\, CP_{t\,|\,t-1}C^{\mathsf{T}} + R\right), \tag{3.45}$$

for the observed data likelihood. Note that the dependence on the matrices A and Q is implicit via the Kalman filter.

We conclude the example by noting that in case we are interested in the computing the ML estimate it is often convenient to work with the logarithm of the likelihood. This is especially true for members of the exponential family, but for numerical reasons it often holds true for other distributions as well. From (3.45) we have that the log-likelihood is given by

$$\begin{aligned}
\log p_\theta(y_{1:t}) &= \sum_{t=1}^{T} \log \mathcal{N}\left(y_t \,\middle|\, C\widehat{x}_{t\,|\,t-1},\, CP_{t\,|\,t-1}C^{\mathsf{T}} + R\right) \\
&= \frac{Tn_x}{2}\log 2\pi + \sum_{t=1}^{T}\left(-\frac{1}{2}\log\det(\Lambda_t) - \frac{1}{2}\|y_t - C\widehat{x}_{t\,|\,t-1}\|^2_{\Lambda_t^{-1}}\right),
\end{aligned} \tag{3.46}$$

where $\Lambda_t = CP_{t\,|\,t-1}C^{\mathsf{T}} + R$.

For a general SSM it is not possible to derive a closed form expression for the observed data likelihood as was done for the LG-SSM special case in Example 3.4. This forces us to numerical approximations, which will be developed throughout this manuscript.

3.5.2 Maximum likelihood learning

In maximum likelihood learning it is assumed that the observed measurements $y_{1:T}$ are modelled as one particular realization from the random variables $Y_{1:T}$. This implies that once the numerical values $y_{1:T}$ are inserted into the data distribution $p(y_{1:T} \,|\, \theta)$, it is not a probability distribution anymore, but rather a *deterministic function* of the unknown parameters θ. The resulting deterministic object is commonly referred to as the *likelihood function*.

Definition 5 ((log-) likelihood function)**.** *The likelihood function $\mathcal{L}(\theta; y_{1:T})$ is obtained by inserting the observed measurements $y_{1:T}$ into the PDF modelling the corresponding random variables $Y_{1:T}$,*

$$\mathcal{L}(\theta; y_{1:T}) \triangleq p(Y_{1:T} = y_{1:T} \,|\, \theta). \tag{3.47}$$

The corresponding log-likelihood function is give by

$$\ell_\theta = \log \mathcal{L}(\theta; y_{1:T}). \tag{3.48}$$

Note that the dependence on the specific realisation $y_{1:T}$ has been suppressed in the definition of the log-likelihood in the interest of a compact notation. We talk about the likelihood function as the likelihood of the parameters given a particular realization of data or just the likelihood of the parameters. The end product from maximum likelihood learning is a point estimate $\widehat{\theta}$ that is selected such that it maximizes the joint distribution of the measurements $p(Y_{1:T} = y_{1:T} \,|\, \theta)$ w.r.t. θ, i.e. maximizing the likelihood function according to (3.43).

The logarithm is a monotonically increasing function, which implies that maximizing the log-likelihood function results in a problem that is equivalent to (3.38). When the model is assumed to belong to the exponential family, the optimisation problem obtained in using the log-likelihood rather than the likelihood function is typically simpler (recall that $\log e^a = a$). The exponential family is rather rich and it contains many of the "standard" densities. Another important reason for why it is convenient to work with the log-likelihood is that the use of the logarithm results in algorithms that are numerically more well-behaved.

From (3.38) it is hard to see what maximum likelihood learning entails, since the problem structure to a large extent is "hidden" within the cost function (i.e. the likelihood function) of the optimization problem. The fact that we have constrained ourselves to SSMs means that we have an expression for the likelihood function according to (3.43), which inserted into (3.38) results in

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} \int p(x_0 \,|\, \theta) \prod_{t=1}^{T} p_\theta(y_t \,|\, x_t) \prod_{t=1}^{T} p_\theta(x_t \,|\, y_{1:t-1}) \mathrm{d}x_{0:T}. \tag{3.49}$$

From (3.49) it is clear that ML learning of SSMs also involves the state inference problem, since the one step ahead predictor $p_\theta(x_t \mid y_{1:t-1})$ is part of the cost function.

3.5.3 Bayesian learning

In Bayesian modelling the unknown parameters are modelled as random variables. Hence, *all* quantities are modelled as random variables with an associated PDF. The aim in Bayesian learning of unknown parameters is to compute the posterior distribution of the parameters conditioned on the observed measurements $p(\theta \mid y_{1:T})$. This posterior distribution constitutes an object that provide a description of the knowledge that is present in the observed measurements about those parameters. The description is interesting in the sense that it includes information about the uncertainty inherent in the observed data, not just a point estimate.

The posterior distribution depends on the model and the observed measurements according to (3.39).

3.6 History and further reading

Fisher (1912, 1922) Rauch et al. (1965) Kalman (1960) (Jazwinski, 1970) (Cappé et al., 2005) (Kailath et al., 2000). (Ljung, 1999) Bernado and Smith (2000), Box and Tiao (1992), Robert (2001) Carter and Kohn (1994), Frühwirth-Schnatter (1994) De Jong and Shephard (1995), Durbin and Koopman (2002), Wilkinson and Yeung (2002) Bresler (1986). Peterka (1981) Ninness and Henriksen (2010) Ljung (1999), Söderström and Stoica (1989).

Chapter 4

Monte Carlo

Assume that you are engaged in a game of solitaire. When laying out the playing cards on the table, you ask yourself what the probability is of finishing the game successfully. Rather than engaging in tedious combinatorial calculations, a more practical way to answer this questions might be to lay out the cards, say, 100 times and observe the number of successes. The proportion of successes is then a natural *estimator* of the success probability.

This is an example of a Monte Carlo method. In using stochastic simulations (here, shuffling and laying out the deck of cards) we can estimate some quantity which is otherwise tedious, or intractable, to compute. Clearly, the result of this approach will be affected by the randomness of the experiment. Indeed, the answer that we provide will be a random variable in itself! However, as we do more and more trials, we expect the answer to become increasingly more accurate. This property lies at the heart of Monte Carlo methods.

Computing probabilities, or more generally[1] expected values, is essentially a problem of integration. Indeed, as noted in Chapter 3, in solving inference problems we are often faced with various integration problems, typically in high-dimensional spaces. This holds for both state inference and parameter inference, and for both maximum likelihood and Bayesian approaches. In order to clarify this, two examples of integration problems that often arise are given below.

- **Expectation**. An expected value often provides an interesting and interpretable point estimate. If the random variable X is distributed according to $\pi(x)$, then computing the expectation of $\varphi(X)$ amounts to solving the integral

$$\mathbb{E}_\pi[\varphi(X)] = \int \varphi(x)\pi(x)\mathrm{d}x, \tag{4.1}$$

  where $\varphi : \mathsf{X} \to \mathbb{R}^{n_\varphi}$ is some function of interest (commonly referred to as a test function). Computing expectations is an important capability in working

[1] The probability of an event A can be written as $\mathbb{P}(A) = \mathbb{E}[\mathbb{1}(A)]$, with $\mathbb{1}(\cdot)$ being an indicator function.

with probabilistic models and a common example of (4.1) is when $X = X_t$. More specifically, we want to compute a point estimate of the state X_t based on measurements obtained from a nonlinear SSM on the form (2.30), $\hat{x}_{t\,|\,T} = \mathbb{E}[X_t \,|\, y_{1:T}] = \int x_t p(x_t \,|\, y_{1:T}) \mathrm{d}x_t$. Another example is to compute the variance of the posterior mean estimate of the parameter θ in a dynamic system, given the measurements $y_{1:T}$. This provides a useful measure of uncertainty of the estimate. With $\widehat{\theta} = \int \theta p(\theta \,|\, y_{1:T}) \mathrm{d}\theta$ being the posterior mean, the variance is given by $\sigma_\theta^2 = \int (\theta - \widehat{\theta})^2 p(\theta \,|\, y_{1:T}) \mathrm{d}\theta$.

- **Marginalization**. Given the joint density $\pi(x_1, x_2)$ of two random variables X_1 and X_2, we can compute the marginal density of, say, X_1 according to

$$\pi(x_1) = \int \pi(x_1, x_2) \mathrm{d}x_2. \tag{4.2}$$

  Marginalization amounts to averaging over one variable, here X_2, and focusing the attention on the remaining variable. Note that X_1 and X_2 are commonly multivariate and this is indeed the case in many of the marginalization examples that we will see in this manuscript. Marginalization often arises when working with probabilistic models and, indeed, it accounts for one of the main challenges when doing inference in these models. One example is to compute the marginal likelihood $p(y_{1:T})$, i.e. the normalization factor. The integral is in this case given by

$$p(y_{1:T}) = \int p(y_{1:T} \,|\, \theta) p(\theta) \mathrm{d}\theta. \tag{4.3}$$

The above examples serve as a motivation for why it is highly relevant to understand and be able to solve integration problems when doing inference in dynamical systems. For the special cases of the LG-SSM and the FSS model there are explicit expressions available, but for most other cases we are forced to approximate the above integrals in some way. Methods for computing these approximations can be broadly categorized into one of the following two classes, where Monte Carlo methods fall in the second class.

1. **Deterministic analytical approximation:** The methods in this class performs the approximation by assuming that the posterior density has a particular form or that it factors in a specific manner. Hence, they somehow assume a certain parameterized analytical form for the posterior. The parameters are then inferred from data, typically in an iterative fashion. Examples of methods falling in this class are the Laplace approximation, variational Bayes (VB), belief propagation (BP),and expectation propagation (EP). These are all very useful inference methods, but we will not work with them in this manuscript.
2. **Stochastic simulation approximation:**The methods in this class aim at generating samples from the posterior. These samples can then be used to generate estimates of any relevant quantity of interest. The developments in this manuscript are to a large extent geared toward this type of methods.

Monte Carlo methods provide a computational solution to the problems of computing expectations and performing marginalization, where the obtained accuracy is only limited to our computational resources. Monte Carlo methods respect the model and the general solution – the approximation stems from the fact that we are trying to approximate the solution itself and not the model. It is also worth noting that Monte Carlo methods are more general than this and are indeed used for solving other types of problems in different fields. Balancing this, it is important to recognize that even though the approximations underlying the deterministic analytical methods might seem very restrictive, these methods can provide very good solutions for hard problems. The manageable computational complexity of these algorithms makes them interesting e.g. for large data sets. There is no single approximation method that is going to outperform all other methods on all problems. Hence, it is important to understand and be able to use as many of these methods, both stochastic and deterministic, as possible. Indeed solutions based on combinations of Monte Carlo methods and analytical approximation methods are likely to become increasingly important in the future.

We will in this manuscript show how the above problems (expectation and marginalization), and many more, can be solved in the context of dynamical systems using algorithms based on various Monte Carlo methods. The *aim* in the present chapter is to introduce the Monte Carlo idea and the most fundamental Monte Carlo methods. We will also provide some examples of their use in solving inference problems in dynamical systems. However, these problems will be more thoroughly discussed in Chapter 5 – Chapter **??**, which are devoted specifically to state and parameter inference in dynamical systems using Monte Carlo methods. For this reason we will only consider the LG-SSM in this chapter. As we shall see, however, already for linear models, Monte Carlo samplers open up for interesting inference which is hard to perform without this theory.

4.1 The Monte Carlo idea

As before, let $\pi(x)$ be a PDF on X, referred to as the target density. Assume that we want to compute the expected value of some function $\varphi(X)$ where X is distributed according to $\pi(x)$,

$$I(\varphi) \triangleq \mathbb{E}_\pi[\varphi(X)] = \int \varphi(x)\pi(x)\mathrm{d}x. \tag{4.4}$$

Let us start by making the assumption that we can generate independent samples $\{x^i\}_{i=1}^N$, distributed according to $\pi(x)$. This is in fact a very restrictive assumption and a large part of this manuscript is concerned with strategies for generating realizations from random variables with complicated distributions. Nevertheless it is instructive to make this assumption in order to be able to focus on the *key idea* underlying all Monte

Carlo methods. Based on these samples, we approximate (4.4) by the sample average,

$$I(\varphi) \approx \frac{1}{N} \sum_{i=1}^{N} \varphi(x^i). \tag{4.5}$$

This is referred to as the Monte Carlo estimator of (4.4). An equivalent way to arrive at the Monte Carlo estimator, is to let the samples $\{x^i\}_{i=1}^N$ define an empirical approximation of the target distribution,

$$\widehat{\pi}_{\mathrm{MC}}(x) = \sum_{i=1}^{N} \frac{1}{N} \delta_{x^i}(x), \tag{4.6}$$

where $\delta_{x'}(x)$ denotes a Dirac point-mass located at the point $x' \in \mathsf{X}$. Hence, we approximate the target distribution (which may be continuous) with a discrete probability distribution, by placing a point-mass probability of $1/N$ at each of the generated samples. Inserting the approximation (4.6) into (4.4) results in

$$\widehat{I}^N(\varphi) = \int \varphi(x) \sum_{i=1}^{N} \frac{1}{N} \delta_{x^i}(x) \mathrm{d}x = \frac{1}{N} \sum_{i=1}^{N} \varphi(x^i), \tag{4.7}$$

i.e. we indeed obtain the Monte Carlo estimator (4.5). The idea of letting a collection of samples define an empirical point-mass distribution as in (4.6) is very convenient and it will be frequently used in the sequel. By inserting such a point-mass distribution into, e.g. (4.1) or (4.2), a previously intractable integral is transformed into a tractable finite sum.

The Monte Carlo estimator (4.7) comes with many desirable properties, which to a large extent explains the popularity of the Monte Carlo method. First, the estimator is unbiased, $\mathbb{E}[\widehat{I}^N(\varphi)] = I(\varphi)$, where the expectation is w.r.t. the random realizations $\{x^i\}_{i=1}^N$ in (4.7). Second, the strong law of large numbers (SLLN) implies almost sure convergence to the true expectation,

$$\widehat{I}^N(\varphi) \xrightarrow{\text{a.s.}} I(\varphi), \qquad \text{as } N \to \infty. \tag{4.8}$$

Additionally, if the variance of $\varphi(X)$ is finite, i.e. $\sigma_\varphi^2 = \mathrm{Var}[\varphi(X)] \triangleq \mathbb{E}\big[\varphi^2(X)\big] - I^2(\varphi) < \infty$, then the central limit theorem (CLT) implies

$$\frac{\sqrt{N}\left(\widehat{I}^N(\varphi) - I(\varphi)\right)}{\sigma_\varphi} \xrightarrow{\text{d}} \mathcal{N}(0,1), \qquad \text{as } N \to \infty. \tag{4.9}$$

In fact, the variance of the estimator (4.7) is explicitly given by, $\mathrm{Var}[\widehat{I}^N(\varphi)] = \sigma_\varphi^2/N$. From (4.9) it follows that the error in the Monte Carlo estimate decreases as $\mathcal{O}(N^{-\frac{1}{2}})$. This is a very interesting result, since it means that the convergence rate is independent of the dimension of the space X. This clearly distinguishes Monte Carlo based integration methods from deterministic integration methods, where the latter typically have

approximation errors that grow with the dimension of the space in a devastating fashion. We say that these methods suffer from the *curse of dimensionality*. This is not the case for Monte Carlo methods, making them particularly suitable for high-dimensional integration problems. We provide an intuitive explanation of how Monte Carlo can avoid the curse of dimensionality in Example 4.1 below.

Example 4.1

The assumption that it is possible to generate i.i.d. samples from the target distribution $\pi(X)$ is what made all of the above possible. As already stated, this is a very unrealistic assumption, since for most practical problems $\pi(x)$ is a complicated distribution of which we only have partial knowledge. This motivates the rest of this chapter, which is devoted to strategies for generating samples from complicated target distributions, effectively rendering the use of the Monte Carlo idea introduced above possible.

4.2 Rejection sampling

An often encountered difficulty is that the target density $\pi(x)$ can be evaluated only up to proportionality. That is, we can write the target density as $\pi(x) = \widetilde{\pi}(x)/Z$, where $\widetilde{\pi}(x)$ can be evaluated point-wise, but where the normalization constant Z (which is independent of x) is unknown. The typical setting is when Bayes' theorem is used to express a posterior PDF in terms of the prior, the likelihood and the (unknown) normalizing constant. For instance, consider the measurement update (3.6a) of the filtering recursion for a general SSM. The unknown normalizing constant is here given by the density $p(y_t \,|\, y_{1:t-1})$. As another example, in Bayesian parameter inference, we seek the posterior density $p(\theta \,|\, y_{1:T}) = p(y_{1:T} \,|\, \theta)p(\theta)/p(y_{1:T})$. Here, the prior and the data distribution are often directly available from the model, but the normalizing constant $Z = p(y_{1:T})$ is typically unknown and intractable.

Rejection sampling is a Monte Carlo method which, under these conditions, can be used to generate independent samples *exactly* distributed according to the target density $\pi(x)$. Let us introduce the idea underlying rejection sampling by considering a specific example. Consider the (rather complicated) PDF shown by the gray area in Figure 4.1 (left), and assume that we want to generate samples that are distributed according to this density. Let (X, U) be a two-dimensional random vector, distributed uniformly over the gray area; see Figure 4.1 (right). It then holds that X marginally is distributed according to $\pi(x)$ (this claim will be verified below). The problem is, of course, that sampling uniformly over the gray area is just as hard as the original problem, but it leads us to the following idea. First, let $q(x)$ be a user-chosen PDF which is easy to sample from. Such a distribution is referred to as a *proposal* distribution. Second, assume that there exists a constant B such that

$$\widetilde{\pi}(x) \leq Bq(x), \forall x \in \mathsf{X}. \tag{4.10}$$

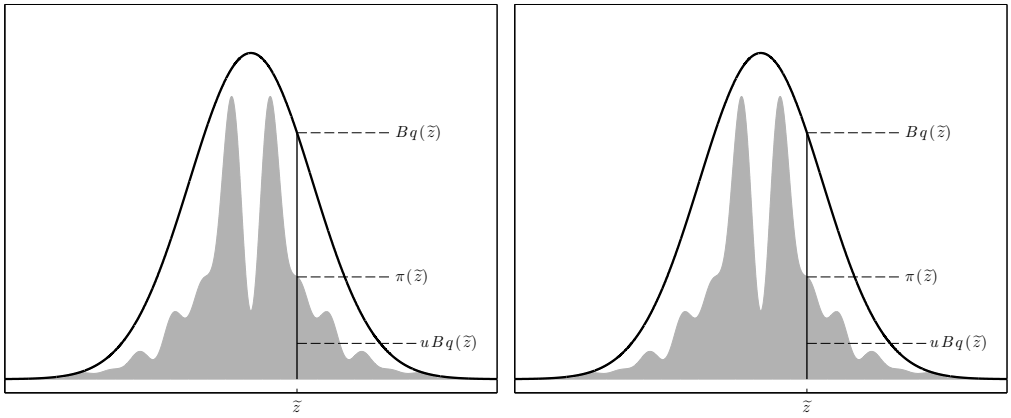

Figure 4.1: Illustration of rejection sampling. (Left) The graph of $\widetilde{\pi}(x)$ (gray area) is bounded by the graph of $Bq(x)$ (black curve). A sample is generated uniformly over the area under the black curve. If the sample falls in the gray area, it is accepted as a draw from $\pi(x)$, otherwise it is rejected. (Right) PDF of the two-dimensional random vector (X, U).

Now, if we sample independently and uniformly under the graph of $Bq(x)$, but only keep the samples that falls under the graph of $\widetilde{\pi}(x)$ (i.e. in the gray area in Figure 4.1), then the surviving draws are i.i.d. samples from the target distribution. The idea outlined above is formalized in Algorithm 4.1, which allows us to generate N i.i.d. samples from the target distribution $\pi(x)$.

Algorithm 4.1: Rejection Sampler (RS)

for $i = 1$ **to** N **do**
  notfound $\leftarrow 1$
  while *notfound* **do**
    Sample $\widetilde{x} \sim q(x)$.
    Sample $u \sim \mathcal{U}[0, 1]$.
    if $u \leq \frac{\widetilde{\pi}(\widetilde{x})}{Bq(\widetilde{x})}$ **then**
      $x^i = \widetilde{x}$
      notfound $\leftarrow 0$
    end
  end
end

This algorithm is sometimes also referred to as acceptance-rejection sampling or accept-reject sampling. To formally show that Algorithm 4.1 indeed generates i.i.d. samples from $\pi(z)$, we first note that, by construction, the samples $\{x^i\}_{i=1}^N$ are independent and identically distributed (since they are all generated in the same way). Now, let

$A \subset \mathsf{X}$ and consider the distribution of one of the samples, say X^1,

$$\mathbb{P}\big(X^1 \in A\big) = \mathbb{P}\Big(\widetilde{X} \in A \,|\, \widetilde{X} \text{ is accepted}\Big) = \frac{\mathbb{P}\Big(\widetilde{X} \in A \text{ and } \widetilde{X} \text{ is accepted}\Big)}{\mathbb{P}\Big(\widetilde{Z} \text{ is accepted}\Big)}. \tag{4.11}$$

Since $\widetilde{X}$ is generated from the proposal density $q(x)$ and U is uniform on $[0, 1]$, the numerator can be expressed as

$$\begin{aligned}&\mathbb{P}\Big(\widetilde{X} \in A \text{ and } U \leq \widetilde{\pi}(\widetilde{X})/Bq(\widetilde{X})\Big) \\ &\qquad = \int_A \left(\int_0^{\frac{\widetilde{\pi}(x)}{Bq(x)}} 1 \mathrm{d}u \right) q(x) \mathrm{d}x = \int_A \frac{\widetilde{\pi}(x)}{Bq(x)} q(x) \mathrm{d}x = \frac{Z_\pi}{B} \int_A \pi(x) \mathrm{d}x. \end{aligned} \tag{4.12}$$

Similarly, the denominator in (4.11) is given by $\mathbb{P}\Big(\widetilde{X} \text{ is accepted}\Big) = Z_\pi / B$. Inserting these expressions into (4.11) results in

$$\mathbb{P}\big(X^1 \in A\big) = \int_A \pi(x) \mathrm{d}x. \tag{4.13}$$

Since (4.13) holds for any subset $A \subset \mathsf{X}$, if follows that x^1 (and thus all the samples produced by Algorithm 4.1) is indeed distributed according to the target density $\pi(x)$. It is worth reflecting upon the fact that, due to the rejection procedure, the output from Algorithm 4.1 are samples drawn from the target distribution, despite the fact that we internally sample from the proposal distribution.

Example 4.2: Rejection sampling

Let us assume that we wish to generate N i.i.d. samples from the PDF given by,

$$\pi(x) = \frac{1}{Z_\pi} e^{-\frac{1}{2}x^2} \left(\sin(6x)^2 + 3\cos(x)^2 \sin(4x)^2 + 1 \right), \tag{4.14}$$

which in Figure 4.1 is visualized as the gray area. This task will be solved using rejection sampling implemented according to Algorithm 4.1. The proposal distribution is chosen as $q(x) = \mathcal{N}(x \,|\, 0,\, 1)$ and the constant as $B = 12$, which is visualized using a black line in Figure 4.1. The resulting empirical distributions

$$\widehat{\pi}_{\mathrm{MC}}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^i}(x) \tag{4.15}$$

for $N = 10, 100, 10\,000$ are shown in Figure 4.2 (top left, top right and bottom left, respectively). The same figure (bottom right) also illustrates Algorithm 4.1 by plotting the rejected samples in red and the accepted samples in blue for the case when $N = 1\,000$. This illustrates that using rejection sampling we are indeed capable of sampling uniformly from the area under the graph of the target distribution.

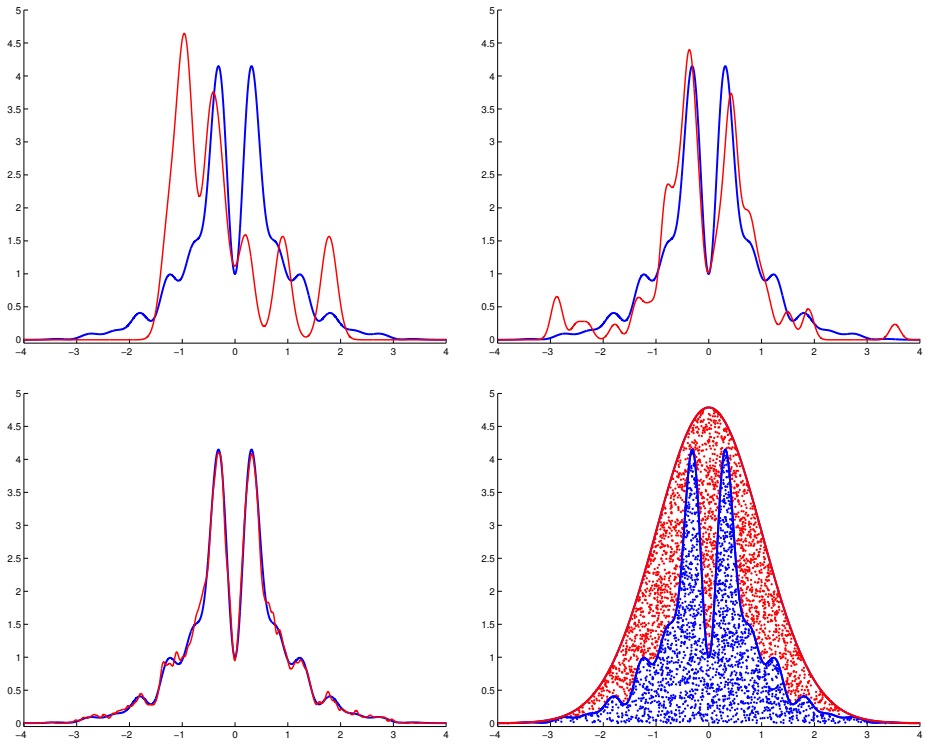

Figure 4.2: Convergence of the rejection sampler as more and more samples are used. The blue curve is the target density and the red curve corresponds to the estimate from the rejection sampler using 10 (upper left), 100 (upper right) and 10 000 (lower left) samples, respectively. In the lower right figure we visualise the rejected samples in red and the accepted samples in blue for the case when $N = 1\,000$.

The choice of the proposal density $q(x)$ and the constant B are very important from a practical point of view. As noted above, the *acceptance probability* is given by $\mathbb{P}\Big(\widetilde{X} \text{ is accepted}\Big) = Z_\pi/B$. Consequently, the average number of times that we need to execute the while-loop in Algorithm 4.1, to generate one sample x^i, is B/Z_π. It is thus imperative that B is not too large. This can be intuitively understood by looking at Figure 4.2 (lower right) and consider how many red samples there would be if the area between the proposal density and the target density grows larger. However, for the algorithm to work, we also require that the graph of $\widetilde{\pi}(x)$ is completely below the graph of $Bq(x)$, i.e. B is at least as large as the largest discrepancy between the proposal $q(x)$ and the (unnormalized) target $\widetilde{\pi}(x)$. Consider, for instance, the same target density as in (4.14), but assume that we use as proposal density $q(x) = \mathcal{N}\left(x \,\middle|\, 0,\, 10^2\right)$, i.e. we increase the standard deviation to 10. To ensure $\widetilde{\pi}(x) \leq Bq(x)$, $\forall x \in \mathsf{X}$ we then need to take $B \gtrsim 105$, leading to many wasted simulations.

Finding a proposal density in close agreement with the target density is easy for the toy problem considered in Example 4.2. However, as the target density becomes more complicated and, in particular, as the dimension of X increases, this becomes

harder. For the sake of illustration, assume that we wish to draw samples from the d-dimensional, standard normal distribution using rejection sampling. As proposal, we use a d-dimensional, zero-mean normal distribution with covariance matrix $\sigma_q^2 I_d$. For the ratio between the target and the proposal densities to be bounded, we require that $\sigma_q \geq 1$. Then, the smallest bound on this ratio is given by $B = \sigma_q^d$. Hence, the acceptance probability decays exponentially as we increase the dimension of the problem and the algorithm is indirectly affected by the curse of dimensionality. In high dimensions, what appears to be a small discrepancy between the proposal and the target densities, can in fact have a huge impact, rendering the method impractical.

4.3 Importance sampling

Importance sampling offers a solution to the problem of evaluating integrals of the form (4.4), but it does not generate exact draws from the target distribution. In the rejection sampler introduced above, we first generate candidate samples from some proposal density $q(x)$. These samples are then either accepted or rejected with certain probabilities, depending on how well they "fit" the target distribution. Importance sampling proceeds similarly, by generating draws from some proposal distribution. However, rather than discarding some of the simulated values, all samples are used, but they are assigned individual weights depending on how well they "fit" the target distribution.

After introducing the importance sampling algorithm in Section 4.3.1 we will justify the above statement by providing some basic analysis of the importance sampler in Section 4.3.3. In Section 4.3.4 we thereafter provide a first solution to the nonlinear filtering problem using the importance sampler.

4.3.1 Derivation and algorithm

Consider the general problem of computing the integral (4.4). Importance sampling can be used to construct a Monte Carlo estimator of this quantity, without requiring exact draws from the target density $\pi(x)$. Similarly to the rejection sampler, the importance sampler relies on the use of a proposal distribution $q(x)$ (also referred to as the importance distribution or instrumental distribution), which is easy to generate samples from. Let $X' \sim q(x)$ be an instrumental random variable, distributed according to the proposal. We can then express (4.4) as,

$$\begin{aligned}\mathbb{E}_\pi[\varphi(X)] &= \int \varphi(x)\pi(x)\mathrm{d}x = \int \varphi(x)\frac{\pi(x)}{q(x)}q(x)\mathrm{d}x = \int \varphi(x)\omega(x)q(x)\mathrm{d}x \\ &= \mathbb{E}_q\left[\varphi(X')\omega(X')\right], \end{aligned} \tag{4.16}$$

where we have introduced the *weight function* $\omega(x) \triangleq \pi(x)/q(x)$ and where we have assumed that $q(x) > 0$ for all x where $\pi(x) > 0$ (that is, $\operatorname{supp}(\pi) \subset \operatorname{supp}(q)$). Hence, we have rewritten the expectation of $\varphi(X)$, as the expectation of the instrumental variable X' mapped through a different function. This provides a feasible way of approximating $\mathbb{E}_\pi[\varphi(X)]$. By construction, it is easy to generate samples from $q(x)$ and we can thus

construct a Monte Carlo estimator for (4.16) by sampling independently $x^i \sim q(x)$ for $i = 1, \ldots, N$ and setting,

$$\widehat{I}_{\mathrm{IS}}^N(\varphi) = \frac{1}{N} \sum_{i=1}^{N} \omega(x^i)\varphi(x^i). \tag{4.17}$$

This estimator is similar to (4.7), but we see that the samples are weighted by so-called *importance weights*, accounting for the discrepancy between the proposal and the target densities. Intuitively speaking, the importance weights contain information about how useful each proposed value x^i is for computing the integral in (4.4).

Example 4.3: The importance sampling idea

As mentioned in the previous section, it is very common that it is only possible to evaluate the target distribution up to an unknown normalization constant. That is, we can write $\pi(x) = \widetilde{\pi}(x)/Z_\pi$, where $\widetilde{\pi}(x)$ can be evaluated for any $x \in \mathsf{X}$ but the constant Z_π is unknown[2]. We then have,

$$I(\varphi) = \mathbb{E}[\varphi(X)] = \int \varphi(x) \frac{\widetilde{\pi}(x)}{Z_\pi q(x)} q(x) \mathrm{d}x = \frac{1}{Z_\pi} \int \varphi(x)\omega(x)q(x)\mathrm{d}x, \tag{4.18}$$

where (with abuse of notation) we have redefined the weight function as

$$\omega(x) \triangleq \frac{\widetilde{\pi}(x)}{q(x)}. \tag{4.19}$$

Hence, the importance sampling estimator (4.17) is given by,

$$\widehat{I}_{\mathrm{IS}}^N(\varphi) = \frac{1}{NZ_\pi} \sum_{i=1}^{N} \bar{w}^i \varphi(x^i), \tag{4.20}$$

where we have explicitly introduced the weights $\bar{w}^i = \omega(x^i)$ for $i = 1, \ldots, N$. Note that, since $\bar{w}^i$ is given by a transformation of a random variable, it is itself a random variable. From the above expression it appears as if we have just moved the problem with the unknown normalization constants from one place to another. However, we can make use of the samples $\{x^i\}_{i=1}^N$ generated from the proposal distribution to compute an approximation of the unknown constant. Indeed, since $\pi(x)$ is a PDF and thus integrates to one, we have

$$Z_\pi = \int \widetilde{\pi}(x)\mathrm{d}x = \int \frac{\widetilde{\pi}(x)}{q(x)} q(x)\mathrm{d}x \approx \frac{1}{N} \sum_{i=1}^{N} \bar{w}^i. \tag{4.21}$$

[2]Similarly, we may assume that the proposal density can only be evaluated up to proportionality, but that is less common in practice.

Note that the expression on the right-hand-side of (4.21) is a vanilla Monte Carlo estimator of the normalizing constant Z_π and the properties of such estimators discussed in Section 4.1 thus apply. Of specific interest is the fact that the estimator is unbiased: $\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\bar{w}^i\right] = Z_\pi$.

By inserting the approximation (4.21) into (4.20), we obtain the *self-normalized* importance sampling estimator,

$$\widehat{I}_{\mathrm{IS}}^{N}(\varphi) \triangleq \sum_{i=1}^{N} w^i \varphi(x^i), \tag{4.22}$$

where $\{w^i\}_{i=1}^N$ denote the normalized importance weights, defined according to

$$w^i \triangleq \frac{\bar{w}^i}{\sum_{j=1}^{N}\bar{w}^j}. \tag{4.23}$$

Analogously to (4.6), an alternative interpretation of the above is that the importance sampler provides an empirical point-mass approximation of the target distribution, according to,

$$\widehat{\pi}_{\mathrm{IS}}(x) = \sum_{i=1}^{N} w^i \delta_{x^i}(x). \tag{4.24}$$

Even though the importance sampler does *not* provide samples from the target distribution, the weighted samples $\{x^i, w^i\}_{i=1}^N$ define an empirical distribution (4.24), which approximates the target distribution. Furthermore, inserting this empirical distribution into $I(\varphi) = \int \varphi(x)\pi(x)\mathrm{d}x$ straightforwardly results in (4.22). Note that, even if the constant Z_π is known, the importance weights must be normalized according to (4.23) for the point-mass approximation (4.24) to be a probability distribution[3]. Furthermore, there is a bias-variance trade-off in normalizing the weights, where normalization can reduce the variance of the estimator, but introduces a bias. .

The above development is summarized in Algorithm 4.2, where each step is carried out for $i = 1, \ldots, N$. We will make use of this convention throughout the manuscript in the interest of a clean notation.

Algorithm 4.2: Importance sampler

1 Sample $x^i \sim q(x)$.
2 Compute the importance weights $\bar{w}^i = \widetilde{\pi}(x^i)/q(x^i)$.
3 Normalize the importance weights $w^i = \bar{w}^i / \sum_{j=1}^{N}\bar{w}^j$.

From the discussion on the rejection sampler in Section 4.2, we recall that the choice of proposal distribution was important in order to obtain a practical algorithm. This

[3]If the weights are not normalized, we can still construct a point-mass measure as in (4.24), but it will not be a probability distribution.

holds true also for the importance sampler. In Example 4.4 we illustrate this (among other things) for an importance sampler that is used to solve a parameter estimation problem in an LG-SSM.

Example 4.4: Learning an LG-SSM using importance sampling

We will in this example show how to make use of the importance sampler provided in Algorithm 4.2 in order to compute the posterior distribution $p(\theta \mid y_{1:T})$, where θ is the unknown parameter in the dynamics of the following Bayesian LG-SSM,

$$X_{t+1} = \theta X_t + V_t, \qquad V_t \sim \mathcal{N}(0, 0.1), \tag{4.25a}$$
$$Y_t = 0.5X_t + E_t, \qquad E_t \sim \mathcal{N}(0, 0.1), \tag{4.25b}$$
$$X_1 \sim \mathcal{N}(0, 0.1), \tag{4.25c}$$
$$\theta \sim \mathcal{N}\left(\mu_\theta, \sigma_\theta^2\right). \tag{4.25d}$$

Hence, our importance sampler should target

$$\pi(\theta) = p(\theta \mid y_{1:T}) = \frac{p(y_{1:T} \mid \theta)p(\theta)}{p(y_{1:T})}. \tag{4.26}$$

In order to make use of Algorithm 4.2 we have to choose a proposal distribution. In this example we choose it to be the same as the prior distribution, i.e. $q(\theta) = \mathcal{N}\left(\theta \,\middle|\, \mu_\theta, \sigma_\theta^2\right)$. The unnormalized target density can be evaluated as $\widetilde{\pi}(\theta) = p(y_{1:T} \mid \theta)p(\theta)$, for any realization of θ. As usual, we do not have to worry about the normalization constant $Z_\pi = p(y_{1:T})$. Finally, the fact that the proposal distribution and the prior distribution are the same in this example allows us to compute the importance weights according to (for $i = 1, \ldots, N$)

$$\bar{w}^i = \frac{\widetilde{\pi}(\theta^i)}{q(\theta^i)} = p(y_{1:T} \mid \theta^i). \tag{4.27}$$

Hence, the importance weights are given by the likelihood evaluated for a specific parameter value. Intuitively this makes a lot of sense, since the likelihood reveals how likely the obtained measurements are for a specific θ^i. For a specific realization of θ the model (4.25) is a standard LG-SSM and the likelihood is provided by the Kalman filter according to,

$$p(y_{1:T} \mid \theta) = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \theta) = \prod_{t=1}^{T} \mathcal{N}\left(y_t \mid \widehat{y}_{t \mid t-1}(\theta),\, S_{t \mid t-1}(\theta)\right), \tag{4.28a}$$
$$\widehat{y}_{t \mid t-1}(\theta) = 0.5\widehat{x}_{t \mid t-1}(\theta), \tag{4.28b}$$
$$S_{t \mid t-1}(\theta) = 0.5^2 P_{t \mid t-1}(\theta) + 0.1, \tag{4.28c}$$

initialized at $\widehat{y}_{1 \mid 0} = 0$, $P_{1 \mid 0} = 0.1$. Summarizing the above development we obtain the

importance sampler in Algorithm 4.3.

Algorithm 4.3: Importance sampler targeting $p(\theta \mid y_{1:T})$ in (4.25)

1 Sample $\theta^i \sim \mathcal{N}\left(\theta \mid \mu_\theta, \sigma_\theta^2\right)$.
2 Compute the importance weights $\bar{w}^i = p(y_{1:T} \mid \theta^i)$ according to (4.28).
3 Normalize the importance weights: $w^i = \bar{w}^i / \sum_{j=1}^N \bar{w}^j$.

To illustrate Algorithm 4.3 we generate $T = 15$ realizations of the measurements $y_{1:15}$ from the LG-SSM (4.25). The parameters used for the prior distribution in (4.25d) are $\mu_\theta = 0$ and $\sigma_\theta = 1.5$. The result is shown in Figure 4.3, where the blue line is the estimate from the importance sampler using 10 (upper left), 100 (upper right) and 50 000 (lower left) samples, respectively. This clearly shows that the importance sampling estimate

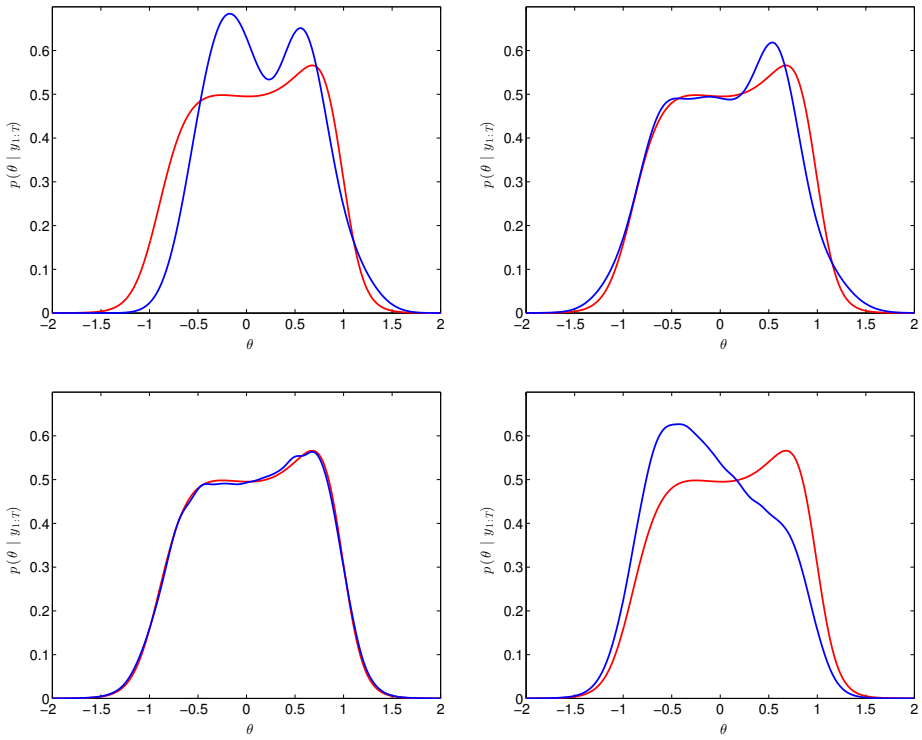

Figure 4.3: Convergence of the importance sampler (Algorithm 4.3) as more and more samples are used. The red curve is the true density and the blue curves corresponds to the estimate from the importance sampler using 10 (upper left), 100 (upper right) and 50 000 (lower left) samples, respectively. In the lower right figure we provide the result of using 100 000 samples generated from a different proposal distribution $q_2(\theta) = \mathcal{N}(\theta \mid -0.5, 1)$. This clearly illustrates the importance of having a good proposal distribution.

converges towards the truth (red curve) as more and more samples are used. The true PDF is calculated using deterministic numerical integration, which is possible here since θ is one-dimensional. In the lower right figure we have made use of a different proposal distribution, $q_2(\theta) = \mathcal{N}\left(\theta \mid -0.5, 1^2\right)$. Even though the estimate visualized in the figure

is made up of 100 000 samples, it is a very poor approximation of the truth. This clearly illustrates the importance of having a good proposal distribution.

When it comes to dynamical systems, a very interesting use of the importance sampling idea arises when we consider inference in nonlinear systems. The reason is simple: the importance sampling idea provides a concrete and mathematically solid way forward in approximating the underlying integrals, which do not admit any analytical solutions. This is the foundation for the particle filter, which is discussed in detail in Chapter 5.

4.3.2 A note on practical implementation

In practical implementations it is not uncommon that direct evaluation of the importance weights as in (4.19) results in numerical imprecision or execution errors due to division by zero when normalising the weights. To avoid such issues it is good practice to compute and work with the log-importance-weights, as described in this section.

Assume that $\omega(x) = \exp(\tau(x))$ where $\tau(x)$ can be robustly evaluated numerically. We can then compute the log-weights as

$$t^i = \tau(x^i), \qquad i = 1, \ldots, N. \tag{4.29}$$

From these we find the maximum $m := \max_i t^i$ and subtract this from all the log-weights: $\widetilde{t}^i = t^i - m$ for $i = 1, \ldots, N$. A simple calculation now shows that the normalized importance weights (4.23) can be computed as

$$w^i = \frac{\exp(\widetilde{t}^i)}{\sum_{j=1}^N \exp(\widetilde{t}^j)} \tag{4.30}$$

where division by zero is avoided since the sum in the denominator is lower bounded by 1.

Furthermore, if the estimate of the normalising constant for $\widetilde{\pi}$ is of interest we can compute the logarithm of the estimator (4.21) directly in a numerically robust way, since

$$\log\left\{\frac{1}{N}\sum_{i=1}^N \bar{w}^i\right\} = \log\left\{\sum_{i=1}^N \exp(\widetilde{t}^i)\right\} + m - \log N, \tag{4.31}$$

where, again, the term in the brackets is bounded away from zero.

4.3.3 Convergence and diagnostic tools

The self-normalized importance sampler differs from a vanilla Monte Carlo method in the sense the normalized importance weights used to compute the estimator (4.22) are not independent. Indeed, the normalization depends on all the weights, so it is clear that the ith normalized weight w^i will depend on the values of all the random variables $\{\bar{w}^i\}_{i=1}^N$. A relevant question to ask is therefore: how will this affect the theoretical properties of the estimator?

To answer this question it is instructive to write the estimator (4.22) as a ratio of two estimators,

$$\widehat{I}_{\mathrm{IS}}^N(\varphi) = \frac{\frac{1}{N}\sum_{i=1}^N \omega(x^i)\varphi(x^i)}{\frac{1}{N}\sum_{j=1}^N \omega(x^j)} =: \frac{\widehat{S}^N}{\widehat{Z}_\pi^N}. \tag{4.32}$$

Since $\{x^i\}_{i=1}^N$ are i.i.d. draws from the proposal distribution $q(x)$, it follows from (4.8) and (4.9) and the unbiasedness of vanilla Monte Carlo estimators that the following properties hold for the numerator of this expression:

$$\mathbb{E}\left[\widehat{S}^N\right] = Z_\pi I(\varphi), \tag{4.33a}$$

$$\widehat{S}^N \xrightarrow{\text{a.s.}} Z_\pi I(\varphi) \qquad \text{as } N \to \infty, \tag{4.33b}$$

$$\sqrt{N}\left(\widehat{S}^N - Z_\pi I(\varphi)\right) \xrightarrow{\text{d}} \mathcal{N}\left(0, \mathrm{Var}_q[\omega(x)\varphi(x)]\right) \qquad \text{as } N \to \infty, \tag{4.33c}$$

where $\mathrm{Var}_q[\,\cdot\,]$ denotes the variance when $x \sim q(x)$. Analogously, we have for the denominator:

$$\mathbb{E}\left[\widehat{Z}_\pi^N\right] = Z_\pi, \tag{4.34a}$$

$$\widehat{Z}_\pi^N \xrightarrow{\text{a.s.}} Z_\pi \qquad \text{as } N \to \infty, \tag{4.34b}$$

$$\sqrt{N}\left(\widehat{Z}_\pi^N - C_\pi\right) \xrightarrow{\text{d}} \mathcal{N}\left(0, \mathrm{Var}_q[\omega(x)]\right) \qquad \text{as } N \to \infty. \tag{4.34c}$$

Consequently, from (4.33a) and (4.34a) we see that the numerator and the denominator, respectively, are unbiased. Indeed, the fact that $\widehat{Z}_\pi^N$ is an unbiased estimator of the normalizing constant Z_π we noted already in Section 4.3.1. However, this does *not* imply unbiasedness of $\widehat{I}_{\mathrm{IS}}^N(\varphi)$ since, in general, $\mathbb{E}[X/Y] \neq \mathbb{E}[X]\,/\mathbb{E}[Y]$. Hence, the self-normalized importance sampling estimator is typically *biased*, contrary to the vanilla Monte Carlo estimator.

From (4.33b) and (4.34b), however, we can conclude that $\widehat{I}_{\mathrm{IS}}^N(\varphi)$ nevertheless is strongly consistent (follows the SLLN), i.e.

$$\widehat{I}_{\mathrm{IS}}^N(\varphi) \xrightarrow{\text{a.s.}} I(\varphi) \qquad \text{as } N \to \infty. \tag{4.35}$$

Furthermore, by using the so-called delta method, equations (4.33c) and (4.34c) can be used to establish a CLT for $\widehat{I}_{\mathrm{IS}}^N(\varphi)$, as is done below.

Theorem 1 (CLT for the self-normalized importance sampler)**.** *Assume that*

$$\sigma_{IS}^2 := \mathbb{E}_\pi\left[\frac{\omega(X)}{Z_\pi}\left(\varphi(X) - I(\varphi)\right)^2\right] < \infty, \tag{4.36}$$

where $\mathbb{E}_\pi[\,\cdot\,]$ denotes expectation when $X \sim \pi(x)$. Then, the self-normalized importance sampling estimator is asymptotically normal:

$$\sqrt{N}\left(\widehat{I}_{IS}^N(\varphi) - I(\varphi)\right) \xrightarrow{\text{d}} \mathcal{N}\left(0, \sigma_{IS}^2\right) \qquad \textit{as } N \to \infty. \tag{4.37}$$

Proof. Define $V^i = \omega(X^i)\varphi(X^i)$ and $Y^i = \omega(X^i)$ and note that $\widehat{S}^N = \frac{1}{N}\sum_{i=1}^N x^i$ and $\widehat{Z}_\pi^N = \frac{1}{N}\sum_{i=1}^N Y^i$. A bivariate CLT for the vector $(\widehat{S}^N, \widehat{Z}_\pi^N)^\mathsf{T}$ reads

$$\sqrt{N}\left(\begin{pmatrix}\widehat{S}^N \\ \widehat{Z}_\pi^N\end{pmatrix} - \begin{pmatrix}\mu_x \\ \mu_y\end{pmatrix}\right) \xrightarrow{\mathrm{d}} \mathcal{N}(0, \Sigma), \qquad \text{as } N \to \infty,$$

where we have defined $\mu_x = Z_\pi I(\varphi)$, $\mu_y = Z_\pi$ and

$$\Sigma = \begin{pmatrix}\sigma_x^2 & \rho_{xy}\sigma_x\sigma_y \\ \rho_{xy}\sigma_x\sigma_y & \sigma_y^2\end{pmatrix} = \begin{pmatrix}\mathrm{Var}[V] & \mathrm{Cov}[V,Y] \\ \mathrm{Cov}[V,Y] & \mathrm{Var}[Y]\end{pmatrix}$$

for convenience. Define $g(x,y) = x/y$ and note that $\widehat{I}_{\mathrm{IS}}^N(\varphi) = g(\widehat{S}^N, \widehat{Z}_\pi^N)$. The delta method can now be used to translate the bivariate CLT above to a CLT for $\widehat{I}_{\mathrm{IS}}^N(\varphi)$. Specifically, it follows that

$$\sqrt{N}\left(\widehat{I}_{\mathrm{IS}}^N(\varphi) - g(\mu_x, \mu_y)\right) \xrightarrow{\mathrm{d}} \mathcal{N}\left(0, \nabla g(\mu_x,\mu_y)\cdot\Sigma\cdot\nabla g(\mu_x,\mu_y)^\mathsf{T}\right) \qquad \text{as } N \to \infty,$$

where

$$\nabla g(x,y) = \begin{pmatrix}\frac{\partial g}{\partial x} & \frac{\partial g}{\partial y}\end{pmatrix} = \begin{pmatrix}\frac{1}{y} & -\frac{x}{y^2}\end{pmatrix}.$$

It remains to show that we obtain the correct mean and variance. For the mean we have, $g(\mu_x,\mu_y) = \mu_x/\mu_y = I(\varphi)$. For the variance we have

$$\begin{aligned}\nabla g(\mu_x,\mu_y)\cdot\Sigma\cdot\nabla g(\mu_x,\mu_y)^\mathsf{T} &= \frac{\sigma_x^2}{\mu_y^2} + \frac{\mu_x^2\sigma_y^2}{\mu_y^4} - 2\frac{\mu_x\rho_{xy}\sigma_x\sigma_y}{\mu_y^3} \\ &= \frac{1}{\mu_y^2}\left\{\sigma_x^2 + \left(\frac{\mu_x}{\mu_y}\right)^2\sigma_y^2 - 2\frac{\mu_x}{\mu_y}\rho_{xy}\sigma_x\sigma_y\right\} = \frac{1}{\mu_y^2}\mathbb{E}\left[\left(V - \frac{\mu_x}{\mu_y}Y\right)^2\right] \\ &= \frac{1}{Z_\pi^2}\,\mathbb{E}_q\left[\left(\omega(X)\varphi(X) - I(\varphi)\omega(X)\right)^2\right] = \mathbb{E}_\pi\left[\frac{\omega(X)}{Z_\pi}\left(\varphi(X) - I(\varphi)\right)^2\right].\end{aligned}$$

□

Comparing the asymptotic variance of the self-normalized importance sampler, Equation (4.36), with the asymptotic variance of the vanilla Monte Carlo estimator which is given by $\mathbb{E}_\pi[(\varphi(X) - I(\varphi))^2]$ we see that the two expressions differ only by the inclusion of the factor $\omega(\,\cdot\,)/Z_\pi$ in the former. This reveals an interesting property of importance sampling; if $\omega(x)/Z_\pi$ is small when $(\varphi(x) - I(\varphi))^2$ is large, then the importance sampler can attain lower asymptotic variance than vanilla Monte Carlo estimation. At first this might seem counterintuitive, but the explanation lies in the fact that variance reduction is only possible if we have a specific test function $\varphi(x)$ (or a restricted family of test functions) in mind.

To give a concrete example, assume that we want to estimate the probability of a rare event, i.e. we seek $\mathbb{P}_\pi(X \in A) = \mathbb{E}_\pi[\mathbb{1}_A(X)]$ for some subset $A \subset \mathsf{X}$. This probability

can be estimated by vanilla Monte Carlo by simply sampling N i.i.d. draws $\{x^i\}_{i=1}^N$ from $\pi(x)$ and count the proportion which ends up in the set A. However, by assumption the probability of any x^i falling in A is close to zero (we are trying to estimate a rare event) so the resulting Monte Carlo estimator will have high variance. An importance sampler can remedy this issue by sampling $\{x^i\}_{i=1}^N$ from a proposal distribution which assigns a larger probability to the set A, thus ensuring that the N sampled points are more likely to be sampled in the region of the space which is most relevant for computing the sought probability.

In the present manuscript we are primarily interested in methods which are generic in the sense that they are not specifically targeting a given test function $\varphi(x)$, but rather the distribution $\pi(x)$ itself. Specifically, we seek a distributional approximation (4.24) which can be used to compute expectations of a wide range of test functions. This *function-free* viewpoint on Monte Carlo sampling is useful, not only because it avoids the requirement for specifying the test function *a priori*, but also because an accurate approximation of the full distribution π is key in enabling a Sequential Monte Carlo implementation for inference in dynamical systems (see Chapter 5).

Another interesting property revealed by the asymptotic variance expression (4.36) is that, if we assume that $\mathrm{Var}_\pi[\varphi(X)] < \infty$, then a *sufficient condition* for finiteness of σ^2_{IS} is that the weight function $\omega(x)$ is upper bounded by some constant B. In fact, this is precisely the same condition that was required for rejection sampling to be applicable; see Equation (4.10). As was discussed in the context of rejection sampling in Section 4.2, the bound B will typically grow exponentially with the dimension of the space X. The aforementioned connection suggests that the *curse of dimensionality* will therefore also affect importance sampling

Generally, a mismatch between the proposal distribution and the target distribution can result in poor estimates (recall Example 4.4). Thus, it is important to be able to quantify the accuracy of an importance sampling approximation. One possibility is to estimate the asymptotic variance (4.36) which can be done using the weighted samples (Exercise **??**). However, as mentioned above we are often interested in approximating the target distribution π itself rather than a specific test function, and it is therefore of interest to also consider function-free diagnostic tools.

One such diagnostic which is frequently used in the context of importance sampling is the notion of *effective sample size (ESS)*.

4.3.4 Sequential importance sampling

It is nontrivial to design proposal distributions for high-dimensional problems. One useful approach is to construct the proposal distribution sequentially, leading to the so-called sequential importance sampler (SIS). For concreteness, consider the nonlinear smoothing problem. This involves computing the joint filtering PDF $p(x_{0:t} \mid y_{1:t})$ when the state and measurement processes are modelled according to an SSM. Note that the dimension of the space X^t grows with t! We will try to solve this problem by deriving an importance sampler targeting $p(x_{0:t} \mid y_{1:t})$ (i.e. in the notation used above, $x = x_{0:t}$ and $\pi(x)$ corresponds to $p(x_{0:t} \mid y_{1:t})$), where the computations are performed sequentially in

time t. This allows us to exploit the structure present in the model (2.30) by exploring the state space in a systematic fashion. Furthermore, in case we are interested in computing the joint smoothing densities online, for $t = 1, 2 \ldots$, a direct application of importance sampling for each $p(x_{0:t} \mid y_{1:t})$ would result in a computational complexity growing quadratically with t. This can be avoided if some of the computations are reused over time. The *key* for accomplishing this lies in choosing a proposal density $q(x_{0:t} \mid y_{1:t})$ which factorizes according to,

$$q(x_{0:t} \mid y_{1:t}) = q(x_0)q(x_1 \mid y_1) \prod_{s=2}^{t} q(x_s \mid x_{1:s-1}, y_{1:s}) = q(x_0)q(x_1 \mid y_1) \prod_{s=2}^{t} q(x_s \mid x_{s-1}, y_s). \tag{4.38}$$

Here, the last equality is suggested by the conditional independence properties inherent in the SSM. That is, we assume that the proposal density respects the conditional independence structure of the target density, e.g. being Markovian. For instance, from the model (2.30), we may choose the proposal density according to

$$q(x_{0:t} \mid y_{1:t}) = q(x_{0:t}) = p(x_0) \prod_{s=1}^{t} p(x_s \mid x_{s-1}), \tag{4.39}$$

corresponding to the prior distribution of the latent states $X_{1:t}$. Hence, sampling from the above proposal density corresponds to a pure simulation of the system dynamics, without taking the observed measurements $y_{1:t}$ into account. In practice this means that at time $t = 0$ we sample $x_0^i \sim p(x_0)$ and then for time $s = 1, \ldots, t$ we sample $x_s^i \sim p(x_s \mid x_{s-1}^i)$. Again, note that this is just *one* of many possible proposal distributions and below we will make use of the more general expression (4.38).

With the proposal density (4.38) in place, we have completed step 1 in Algorithm 4.2. It remains to compute the importance weights. The target density can be expressed as,

$$p(x_{0:t} \mid y_{1:t}) = \frac{p(x_{0:t}, y_{1:t})}{p(y_{1:t})} \propto p(x_0) \left[\prod_{s=1}^{t} p(x_s \mid x_{s-1}) \right] \left[\prod_{s=1}^{t} p(y_s \mid x_s) \right], \tag{4.40}$$

where the right hand side can be evaluated at any value of $x_{0:t}$ (note that the target density can be evaluated only up to proportionality, since the normalizing constant $Z_\pi = p(y_{1:t})$ is typically unknown). In principle, the above expression can be used to compute the importance weights. However, a straightforward evaluation of (4.40) is a computationally expensive operation, since the computational cost grows with time t. To avoid this, we again seek to exploit the structure of the model to find a recursive expression for the importance weights. Analogously to (4.19), the weight function is here given by

$$\omega_t(x_{0:t}) = \frac{p(x_{0:t}, y_{1:t})}{q(x_{0:t} \mid y_{1:t})}. \tag{4.41}$$

For notational convenience, we have suppressed the dependence on $y_{1:t}$ in the weight function, since the measurements are considered known and fixed. By using the conditional independence properties of the SSM and the assumed structure of the proposal density (4.38), we can write (4.41) as,

$$\omega_t(x_{0:t}) = \frac{p(y_t \mid x_t)p(x_t \mid x_{t-1})}{q(x_t \mid x_{t-1}, y_t)} \underbrace{\frac{p(x_{1:t-1}, y_{1:t-1})}{q(x_{1:t-1} \mid y_{1:t-1})}}_{=\omega_{t-1}(x_{1:t-1})}. \tag{4.42}$$

It follows that the weight function can be evaluated recursively, initialized with $\omega_0(x_0) = p(x_0)/q(x_0)$ and updated according to (4.42). We have thus arrived at a sequential importance sampler (Algorithm 4.4) where the samples $\{x^i_{1:t}\}_{i=1}^N$ and the importance weights $\{w^i_t\}_{i=1}^N$ are computed sequentially.

Algorithm 4.4: Sequential importance sampler targeting $p(x_{0:t} \mid y_{1:t})$

1 Sample $x^i_0 \sim p(x_0)$ and initialize the importance weights, $\bar{w}^i_0 = 1/N$.
2 **for** $t = 1, 2 \ldots$ **do**
3 Sample $x^i_t \sim p(x_t \mid x^i_{t-1})$ and store the new samples $x^i_{0:t} = \{x^i_{0:t}, x^i_t\}$.
4 Compute the unnormalized importance weights $\bar{w}^i_t = p(y_t \mid x^i_t)\bar{w}^i_{t-1}$.
5 Normalize the importance weights $w^i_t = \bar{w}^i_t / \sum_{j=1}^N \bar{w}^j_t$.
6 **end**

As pointed out above, sequential importance sampling is a special case of importance sampling where the proposal density is chosen as (4.38) to allow for a sequential implementation. Sequential importance sampling is of interest whenever we are required to sample in high-dimensional spaces and, in particular, when there is structure present in the model. The nonlinear smoothing problem under study here is indeed an example of exactly this kind, where the dimension of the state space is high ($t \times n_x$). However, sequential sampling is not a silver bullet for tackling high-dimensional problems. Indeed, while SIS provides a practical way of addressing the joint smoothing problem, its performance in practice will degenerate quickly with increasing t. This issue is illustrated in the example below.

Example 4.5: A sequential importance sampler targeting $p(x_{0:t} \mid y_{1:t})$

In this example we will illustrate the *key problem* that arises in making use of the sequential importance sampler derived in Algorithm 4.4. To be concrete we will make use of Algorithm 4.4 to target the joint filtering distribution $p(x_{0:t} \mid y_{1:t})$ for the LGSS model defined in (4.25), with $\theta = 0.7$. For simplicity we will only study the filtering marginal $p(x_t \mid y_{1:t})$, since this is enough to show that the sequential importance sampler is in fact unable to solve the nonlinear filtering problem. While this is of course a negative result, the nature of the problem will lead us to a first working particle filter in the subsequent section.

The estimate of the filtering PDF $p(x_t \mid y_{1:t})$ produced by Algorithm 4.4 is of the

form

$$\widehat{p}^N(x_t \mid y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{X_t^i}(x_t). \tag{4.43}$$

To study the quality of this estimate we will investigate the conditional mean estimate of the state

$$\widehat{x}_{t\mid t}^{\mathrm{SIS}} = \int x_t \widehat{p}(x_t \mid y_{1:t}) \mathrm{d}x_t = \sum_{i=1}^{N} w_t^i x_t^i \tag{4.44}$$

We are studying an LG-SSM, which means that the true filtering PDF is Gaussian, with mean value $\widehat{x}_{t\mid t}^{\mathrm{KF}}$ and covariance $P_{t\mid t}^{\mathrm{KF}}$ provided by the KF. This allows us to investigate how close the SIS estimate $\widehat{x}_{t\mid t}^{\mathrm{SIS}}$ produced by Algorithm 4.4 is to the truth. This is performed by generating 1 000 realisations of data ($T = 100$) from the model (4.25) (with $\theta = 0.7$) and then making use of both Algorithm 4.4 and the KF to estimate the mean value. In Figure 4.4 (upper left) the root-mean-squared-error (RMSE) of $|\widehat{x}_{t\mid t}^{\mathrm{SIS}} - \widehat{x}_{t\mid t}^{\mathrm{KF}}|$ is plotted for three different cases, corresponding to $N = 500, N = 5\,000$ and $N = 50\,000$, respectively. From this figure it is clear that the results of Algorithm 4.4 do not appear to be very good, since it provides an error that grows over time. This can be intuitively understood by studying the importance weights w_t^i used in computing the estimate (4.43). Figure 4.4 (top right) shows all the 500 weights for each point in time for a particular realisation using $N = 500$ samples.

A quick look at this figure reveals that there must be something seriously wrong, since all but a few weights are zero. This is further illustrated in the two bottom plots where the 500 importance weights at time $t = 100$ are shown (left) and histograms of the weights are shown for times $t = 2, 5, 10, 20$ and 50 (right), respectively. Again these three plots clearly show the problem that most of the weights are zero, implying that the estimate (4.43) only consists of one or a few non-zero terms. This is not a problem that is specific to just this particular example, but rather this is a well-known problem inherent in the sequential importance sampler when applied to an SSM. It is referred to as *weight degeneracy*.

Unfortunately, the effect experienced in the example above is not a coincidence. Indeed, for SIS it is a rule, rather than an exception, that the variance of the normalized importance weights increases over time; see Section 4.6 for references and further discussion. This has the effect that one of the weights tends to one, whereas all the remaining weights tend to zero, i.e. the ESS goes to one. If we stop to think about it for a moment this is not a surprising result. As pointed out above, SIS is nothing but a standard IS with a specific choice of proposal density (4.38), targeting a density in a space of increasing dimension. As the dimension increases, SIS will be affected by the curse of dimensionality, i.e. an apparently small discrepancy between the proposal density and

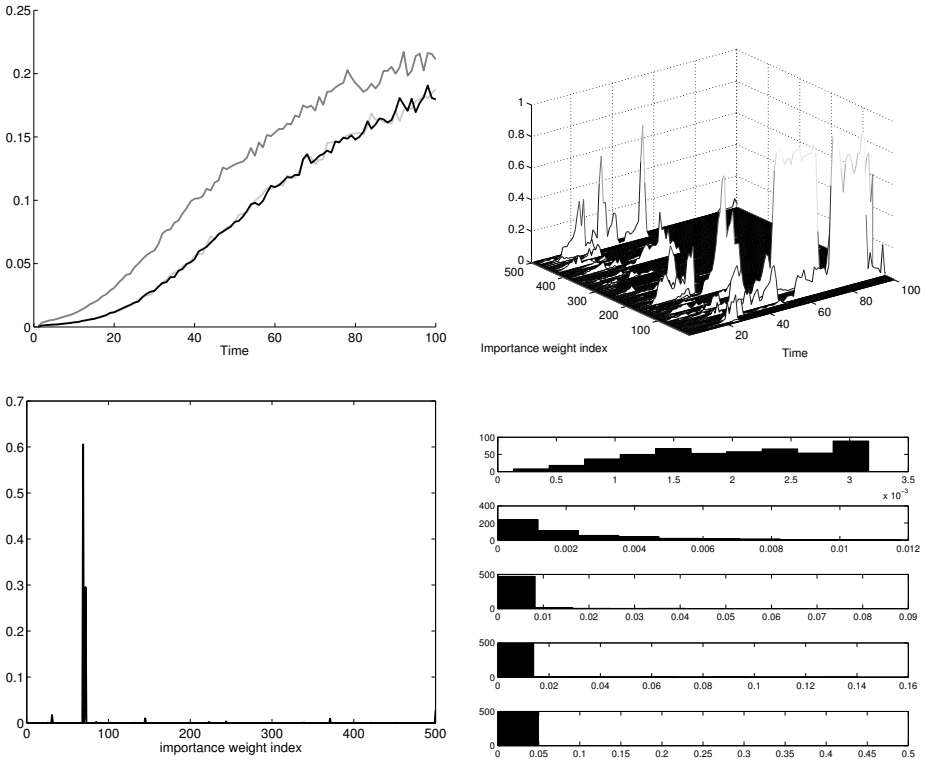

Figure 4.4: See the text for explanation.

the target density gives rise to a large weight variance. This so-called *weight degeneracy* problem means that the SIS algorithm is bound to fail in its attempt to solve the smoothing problem for large enough t (which, in practice, need not be very large).

However, this development motivates how to make progress toward a more practical algorithm. As some of the samples become negligible due to small weights they can be discarded and the focus is instead directed towards the samples with larger weights. One way to accomplish this is through a procedure referred to as *resampling*. The combination of SIS and resampling results in the family of sequential Monte Carlo (SMC) methods. These methods will be discussed in detail in Chapter 5, where we illustrate how they – among other things – provide a solution to the general nonlinear filtering problem.

4.4 Resampling

Motivated by the problems experienced above with degenerating weights, we are interested in finding a way of generating an unweighted set of samples from a weighted set. Let us return to the general problem of sampling from some target density $\pi(x)$. As in Section 4.3 we assume that it is not possible to sample directly from $\pi(x)$, but that it is possible to target it using importance sampling. The importance sampler generates a weighted sample $\{x^i, w^i\}_{i=1}^N$ defining a point-mass approximation of $\pi(x)$ according to (4.24). Given this approximation, we can proceed to generate an unweighted sample, approximately distributed according to $\pi(x)$. The idea is very simple–we draw, say M, conditionally independent samples $\{\widetilde{x}^j\}_{j=1}^M$ from the point-mass probability distribution (4.24),

$$\widetilde{x}^j \overset{\text{i.i.d.}}{\sim} \sum_{i=1}^N w^i \delta_{x^i}(z), \qquad j = 1, \ldots, M. \tag{4.45}$$

Due to the fact that this is a discrete distribution, supported at the points $\{x^i\}_{i=1}^N$, it is straightforward to sample from it. More precisely, we set $\widetilde{x}^j = x^i$ with probability w^i, i.e.

$$\mathbb{P}\big(\widetilde{x}^j = x^i \,\big|\, \{x^i, w^i\}_{i=1}^N\big) = w^i, \qquad j = 1, \ldots, M. \tag{4.46}$$

The resulting samples can be used to construct an unweighted point-mass approximation of the target density $\pi(x)$,

$$\widehat{\pi}_{\text{SIR}}(x) = \frac{1}{M} \sum_{j=1}^M \delta_{\widetilde{x}^j}(x). \tag{4.47}$$

The procedure which (randomly) turns a weighted set of samples into an unweighted one is known as *resampling*. The method outlined above, i.e. to target $\pi(x)$ with an

importance sampler followed by a resampling step, is referred to as *sampling importance resampling (SIR)*. The method is summarized in Algorithm 4.5.

Algorithm 4.5: Sampling Importance Resampler (SIR)

1 Sample $x^i \sim q(x)$.
2 Compute the importance weights $\bar{w}^i = \widetilde{\pi}(x^i)/\widetilde{q}(x^i)$.
3 Normalize the importance weights $w^i = \bar{w}^i / \sum_{j=1}^{N} \bar{w}^j$.
4 Resample $\{x^i, w^i\}_{i=1}^N$ to obtain equally weighted samples $\{\widetilde{x}^i, 1/N\}_{i=1}^N$.

It is instructive to think about SIR as a double Monte Carlo approximation. First, we construct an importance sampling approximation of the target density as in (4.24). We then make a second (vanilla) Monte Carlo approximation of (4.24), resulting in (4.47). Contrary to, for instance, rejection sampling, SIR does not produce i.i.d. samples from the exact target distribution. One aspect of this is that even when the original target density $\pi(x)$ is continuous, there is a non-zero probability that there are duplicates among the samples $\{\widetilde{x}^j\}_{j=1}^M$, since these are drawn from the discrete distribution (4.24).

As we will see in the subsequent chapter, it is often natural to set $M = N$, i.e. to generate equally many samples in the resampling step as used in the importance sampler. In this case it is indeed very likely that some of the possible values $\{x^i\}_{i=1}^N$ will be (randomly) selected more than once in the resampling procedure (4.46), whereas other values are not selected at all. Indeed, this is how resampling works: the importance weights are replaced by multiplicity of samples. Resampling is thus in effect a way of duplicating samples with large importance weights and discarding samples with small importance weights. This is the key property which makes resampling useful for tackling the weight degeneracy problem encountered in Section 4.3.4. We will see how this can be done in the coming chapter.

The resampling step (4.46) can be modeled more explicitly by introducing auxiliary index variables. From (4.46) it is clear that $\widetilde{x}^j = x^i$ for some $i \in \{1, \ldots, N\}$. Hence, we can introduce the index variables $\{a^j\}_{j=1}^M$, such that

$$\widetilde{x}^j = x^{a^j}, \qquad j = 1, \ldots, M. \tag{4.48}$$

Sampling $\widetilde{x}^j$ as in (4.46) thus corresponds to sampling the index variable a^j. It follows that $\{a^j\}_{j=1}^M$ are i.i.d. categorical random variables (i.e. discrete, see Appendix B.1.2 for details on the categorical distribution),

$$a^j \sim \mathcal{C}\left(N, \{w^i\}_{i=1}^N\right), \qquad j = 1, \ldots, M. \tag{4.49}$$

Furthermore, since the index variables are i.i.d., the entire resampling step can be interpreted as generating a sample from the distribution

$$r(\mathbf{a} \,|\, \mathbf{w}) \triangleq \prod_{j=1}^{M} \mathcal{C}\left(a^j \,\middle|\, N, \mathbf{w}\right). \tag{4.50}$$

Note that we make use of bold face notation to denote all samples of a certain variable, e.g. $\mathbf{w} = \{w^1, \ldots, w^M\}$.

4.5 Useful constructions

4.5.1 Conditional Monte Carlo

Whenever there is some structure present in a model, it can be exploited in devising algorithms working with the model. We will in this section provide a useful mechanism that is applicable whenever there is a certain analytically tractable substructure available in the model. To be specific, we are concerned with the case where the random variable X can be partitioned according to $X = (A, B)$, such that the underlying model has a structure allowing us to compute the conditional density $\pi(a \,|\, b)$ analytically. Examples of models where this is possible are provided by the conditionally linear Gaussian SSMs in Section 2.7. Should we choose not to exploit this structure in computing the estimator (4.4), the result is

$$I(\varphi) = \mathbb{E}[\varphi(A, B)] = \int\int \varphi(a, b)\pi(a, b)\mathrm{d}a\mathrm{d}b, \tag{4.51}$$

where $a \in \mathsf{A}$ and $b \in \mathsf{B}$. On the other hand, the analytically tractable structure $\pi(a \,|\, b)$ allows us to rewrite the integrals in (4.51) according to

$$I(\varphi) = \int \left[\int \varphi(a, b)\pi(a \,|\, b)\mathrm{d}a\right] \pi(b)\mathrm{d}b = \int \varphi'(b)\pi(b)\mathrm{d}b, \tag{4.52}$$

where the second equality highlights the fact that the structure allows us to analytically marginalize (integrate out) the variables A, by defining

$$\varphi'(b) \triangleq \int \varphi(a, b)\pi(a \,|\, b)\mathrm{d}a. \tag{4.53}$$

The two expressions for $I(\varphi)$ above opens up for two different ways in which we can construct Monte Carlo estimators for the function $\varphi(A, B)$. The first alternative is to target the joint density $\pi(a, b)$ rendering a set of samples $\{A^m, B^m\}_{m=1}^M$, which inserted into (4.51) results in

$$\widehat{I}^1(\varphi) = \frac{1}{M}\sum_{m=1}^{M} \varphi(A^m, B^m). \tag{4.54}$$

The second and intuitively more appealing alternative is to target the marginal density $\pi(b)$ using Monte Carlo and exploit the analytically tractable structure inherent in the model according to (4.52), resulting in

$$\widehat{I}^2(\varphi) = \frac{1}{M}\sum_{m=1}^{M} \int \varphi(a, B^m)\pi(a \,|\, B^m)\mathrm{d}a = \frac{1}{M}\sum_{m=1}^{M} \varphi'(B^m). \tag{4.55}$$

We move from a Monte Carlo integration in (4.54) to a partially analytical integration in (4.55). The *marginalized estimator* (4.55) should be better due to the fact that the

marginal space B is smaller than the original space $\mathsf{A}\times\mathsf{B}$. This intuition can be formalized in using the following relationship between conditional and unconditional variance. For clarity we explicitly state w.r.t. which variables the expected values and variances are defined.

Theorem 2 (Conditional variance)**.** *Let $\varphi(A, B)$ be a scalar function depending on the two random variables A and B. Then, the variance of $\varphi(A, B)$ can be decomposed according to*

$$\mathrm{Var}_{AB}[\varphi(A,B)] = \mathrm{Var}_B[\mathbb{E}_A[\varphi(A,B)\,|\,B]] + \mathbb{E}_B[\mathrm{Var}_A[\varphi(A,B)\,|\,B]]\,. \tag{4.56}$$

Proof. □ □

A direct consequence of Theorem 2 is that

$$\mathrm{Var}_B[\mathbb{E}_A[\varphi(A,B)\,|\,B]] \leq \mathrm{Var}_{AB}[\varphi(A,B)]\,, \tag{4.57}$$

since $\mathrm{Var}_A[\varphi(A, B)\,|\,B] \geq 0$ by definition. The relationship (4.57) is interesting in that it can be used to compare the variance of the two estimators $\widehat{I}^1(\varphi)$ and $\widehat{I}^2(\varphi)$ introduced in (4.54) and (4.55), respectively. The result is that whenever $\pi(a\,|\,b)$ is analytically available it is statistically justified to exploit it, since it improves the quality of the estimator,

$$\mathrm{Var}\Big[\widehat{I}^2(\varphi)\Big] \leq \mathrm{Var}\Big[\widehat{I}^1(\varphi)\Big]\,. \tag{4.58}$$

We have just showed that by first integrating out the random variable A (i.e. computing $\varphi'(B)$ according to (4.55)) we will obtain a resulting estimator of better quality in terms of lower variance. There are in general many different ways in which the partitioning of Z can be performed and they are not all useful.

The process of marginalizing analytically tractable variables according to (4.53) is commonly referred to as *Rao-Blackwellization*, as a direct consequence of the fact that Theorem 2 is a version of the so-called Rao-Blackwell theorem. Rao-Blackwellization constitutes an important mechanism when it comes to assembling efficient Monte Carlo algorithms, since it allows for construction of estimators that are typically better and never worse (in terms of minimum variance) compared to not using it. It is a constructive procedure for combining basic algorithms into more complex algorithms. We provide several highly useful instances of this later in the manuscript; for example in Section **??**, where it is shown how Rao-Blackwellization allows us to combine the particle filter and the Kalman filter to solve the nonlinear filtering problem when the model has a linear Gaussian substructure.

4.5.2 Monte Carlo with auxiliary variables

Importance sampling for mixture distributions

Assume that the target is given by a mixture distribution, i.e. we wish to sample from

$$\pi(x) = \sum_{k=1}^{K} v_k \pi_k(x), \tag{4.59}$$

where K is the, possibly large, number of components and $\{v_k\}_{k=1}^K$ are the mixture probabilities. Consequently, $v_k \geq 0$ and $\sum_{k=1}^K v_k = 1$.

Assume that the individual components of the mixture, i.e. $\{\pi_k(x)\}_{k=1}^K$ are complicated distributions which we are unable to sample directly from. To address this difficulty, one possibility is to make use of importance sampling. Below, an efficient IS will be constructed for (4.59) by exploiting the auxiliary variable principle. First, however, consider a direct application of IS to the target distribution (4.59). We do this in order to understand why this approach can be problematic. Since (4.59) is a special case of the general target distribution considered in Section 4.3 the importance sampler from Algorithm **??** can thus be used as it stands. To construct a proposal distribution, one possibility is to mimic the structure of the target (4.59) and use a mixture distribution

$$q(x) = \sum_{k=1}^{K} v_k q_k(x), \tag{4.60}$$

where $\{q_k(x)\}_{k=1}^K$ are the mixture components for the proposal. This is not the only possible choice of proposal. Indeed, any proposal density which satisfies the conditions stated in Section 4.3.1 can be used. However, (4.60) is a useful starting point and we will use a similar construction when deriving the particle filter in Chapter 5.

Now, let $\{x^i\}_{i=1}^N$ be i.i.d. draws from (4.60). As in Section 4.3, these samples are assigned importance weights $w^i = \omega(x^i)$ for $i = 1, \ldots, N$. The weight function is given by (4.19), i.e.

$$\omega(x) = \frac{\widetilde{\pi}(x)}{q(x)} = \frac{\sum_{k=1}^K v_k \widetilde{\pi}_k(x)}{\sum_{\ell=1}^K v_\ell q_\ell(x)}, \tag{4.61}$$

where, to comply with the presentation in Section 4.3, we have assumed that $\pi(x)$ can only be evaluated up to proportionality and where $\widetilde{\pi}(x)$ is the unnormalized target density.

The computational complexity of evaluating the weight function (4.61) scales linearly with the number of components K. Since this has to be done for all the samples $\{x^i\}_{i=1}^N$, we thus obtain a method with a computational complexity of $\mathcal{O}(NK)$. If the number of components is large this may be prohibitive. It is therefore of interest to find a more practical Monte Carlo method for the target distribution (4.59). To accomplish this, we will make use of auxiliary variables.

Consider the process of generating a sample from a mixture density, such as (4.59) or (4.60). To do this, we would first randomly select one of the components, and then draw a sample from that specific component. The idea is to introduce the discrete random variable, which picks out one of the components of the mixture, as an auxiliary variable. This will not affect the sampling procedure itself. However, it will affect the way in which the importance weights are computed, which is indeed the source of the high computational complexity of the IS presented above. To make this more precise, let K be a discrete random variable, taking values on the set of integers $\{1, \ldots, K\}$ with probabilities $\{v_k\}_{k=1}^K$, i.e.

$$K \sim \mathcal{C}\left(\{v_k\}_{k=1}^K\right). \tag{4.62}$$

The auxiliary random variable K can be thought of as a variable selecting one of the terms in the sum in (4.59). Let us now define a PDF on the product space $\mathsf{Z} \times \{1, \ldots, K\}$ according to,

$$\pi(x, k) \triangleq \frac{1}{C_\pi} v_k \widetilde{\pi}_k(x), \tag{4.63}$$

where Z_π is a normalization constant, independent of (x, k). This is a slightly non-standard PDF, since it has one continuous component (assuming that X is a continuous space) and one discrete component. However, keeping in mind that the PDF of a discrete distribution is given by the discrete probabilities, this is a perfectly valid construction.

The distribution given by (4.63) is useful since it admits the target distribution (4.59) as a marginal. More precisely, by marginalizing (4.63) over K we get

$$\sum_{k=1}^K \pi(x, k) = \frac{1}{Z_\pi} \sum_{k=1}^K v_k \widetilde{\pi}_k(x) = \pi(x). \tag{4.64}$$

This means that we can use (4.63) as a surrogate for (4.59) in accordance with the auxiliary variable principle. Note that the definition of (4.63) does not contain any summation, which is promising since the sum in (4.59) is the source of the high computational complexity.

Next, we note that the proposal density on the product space $\mathsf{X} \times \{1, \ldots, K\}$, implied by (4.60), is given by

$$q(x, k) = v_k q_k(x). \tag{4.65}$$

To understand the structure of the above PDF, it can be instructive to write it as $q(x, k) = q(x \,|\, k) q(k)$, where

$$q(x \,|\, k) = q_k(x), \tag{4.66a}$$

$$q(k) = v_k. \tag{4.66b}$$

That is, the variable K follows the categorical distribution (4.62) and, given $K = k$, the continuous variable $_x$ is distributed according to the k^{th} component of the mixture

(4.60). Sampling $\{(x^i, k^i)\}_{i=1}^N$ from the joint proposal (4.65) can thus be done by first generating $\{k^i\}_{i=1}^N$ independently from the categorical distribution (4.62) and then generating $\{x^i\}_{i=1}^N$ from the corresponding components of the mixture (4.60). As noted above, this is exactly how one would go about to sample from the mixture proposal (4.60) in the first place.

As noted above, the advantage of using auxiliary variables lies in the computation of the importance weights. The weight function, i.e. the ratio between the target density (4.63) and the proposal density (4.65), is given by,

$$\omega(x, k) = \frac{v_k \widetilde{\pi}_k(x)}{v_k q_k(x)} = \frac{\widetilde{\pi}_k(x)}{q_k(x)}, \tag{4.67}$$

which can be evaluated in constant time. By computing $\bar{w}^i = \omega(x^i, k^i)$ for $i = 1, \ldots, N$ and normalizing, we obtain a weighted sample $\{(x^i, k^i), w^i\}_{i=1}^N$ targeting (4.63). However, since we are only interested in the marginal distribution of X, the indices k^i may be discarded, leaving us with the weighted sample $\{x^i, w^i\}_{i=1}^N$ targeting the mixture distribution (4.59).

4.6 History and further reading

Metropolis and Ulam (1949) Kahn and Marshall (1953) Geweke (1989) Hammersley and Morton (1954) Handschin and Mayne (1969) Handschin (1970) Liu (2001). Liu (2001) Kong et al. (1994) Cappé et al. (2005) Rubin (1987, 1988) Andrieu et al. (2003)

Chapter 5

Sequential Monte Carlo

Sequential Monte Carlo (SMC) methods are constituted by a combination of sequential importance sampling and resampling, both of which were introduced in Chapter 4. SMC is used to sequentially sample from a *sequence* of target densities. The name particle filter (PF) is often used interchangeably with SMC, though here we reserve it for the case when the sequence of target densities is given by $\{p(x_{0:t} \mid y_{1:t})\}_{t\geq 1}$ or the marginals $\{p(x_t \mid y_{1:t})\}_{t\geq 1}$. The basic idea is to leverage sequential importance sampling and resampling to propagate a collection of N weighted random samples $\{x_t^i, w_t^i\}_{i=1}^N$ forward in time. These samples (commonly referred to as particles) constitute an empirical approximation that converge asymptotically (as $N \to \infty$) to the underlying target density $\pi(x_t)$.

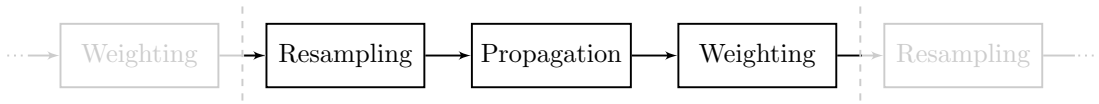

Figure 5.1: Illustrating the three parts making up sequential Monte Carlo.

5.1 Introducing the particle filter

The particle filter (PF) provides an approximate solution to the nonlinear filtering problem, where the accuracy of the approximation is only limited by our computational resources. More specifically, the PF can be viewed as a way of realizing the forward filtering strategy outlined in Section 3.2, i.e. as a way of sequentially approximating the filtering densities $\{p(x_t \mid y_{1:t})\}_{t\geq 1}$. The resulting approximation is an empirical distribution of the form

$$\widehat{p}^N(x_t \mid y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{x_t^i}(x_t). \tag{5.1}$$

The samples $\{x_t^i\}_{i=1}^N$ are often referred to as *particles*—they are point-masses "spread out" in the state space, each particle being one hypothesis about the state of the system. For intuition we can think of each particle x_t^i as a possible system state and the corresponding weight w_t^i contains information about how probable that particular state is. As a way of illustrating the versatility of using the empirical approximation employed by the PF we provide Example 5.1, where the PF is used to solve a localization problem exploiting map information.

Example 5.1: A simple localization problem

Localization is the process of determining the position of an object. Consider a simplified aircraft localization problem, where the aircraft is moving along a straight line at a varying altitude, illustrated by the blue trajectory in Figure 5.2 (upper left). From various onboard sensors the aircraft has knowledge of its velocity v_t and its altitude over

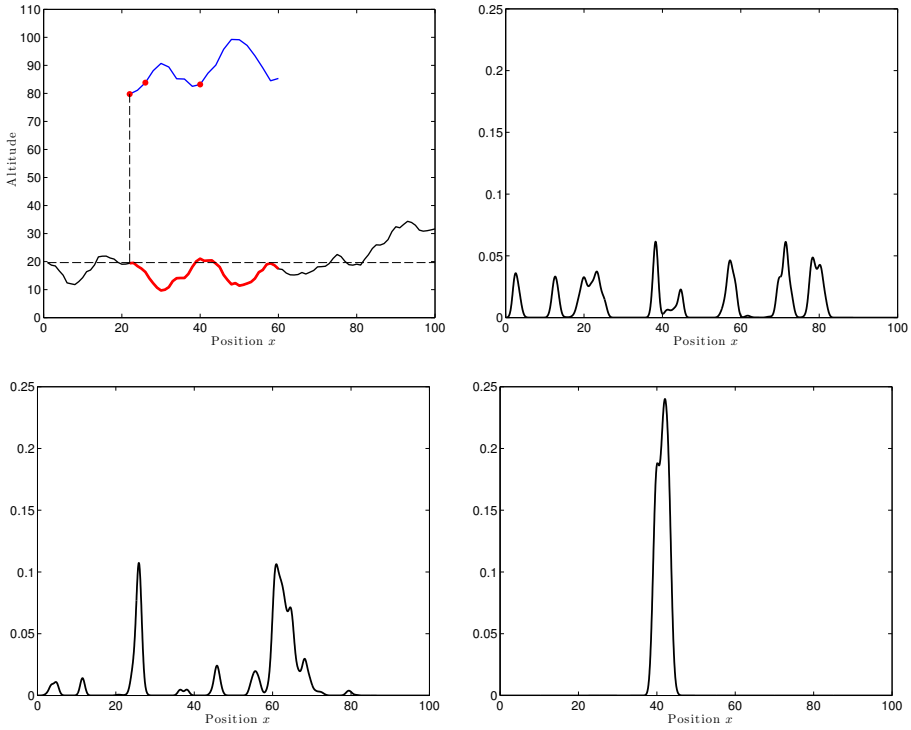

Figure 5.2: The upper left plot illustrates the scenario involving an aircraft moving along the x axis with a varying altitude. The aircraft trajectory is illustrated by the blue curve and the terrain elevation is shown by the black curve. The aircraft measures the distance to ground using a radar and the first measurement at time $t = 1$ is illustrated by the vertical dashed line. The area of the ground that is illuminated by the radar as the aircraft moves from $x = 20$ to $x = 60$ is illustrated by the red curve. The PF estimates of the filtering PDF $p(x_t \mid y_{1:t})$ at time $t = 1$, $t = 3$ and $t = 10$ are shown in the upper right, lower left and lower right plots, respectively. The corresponding true aircraft positions are indicated with the red dots along the trajectory in the upper left plot. See text for all details.

sea level (0 m). A simple model of the aircraft motion is provided by a (discrete-time)

integrator

$$x_{t+1} = x_t + v_t + w_t, \qquad w_t \sim \mathcal{N}(0, 5), \tag{5.2a}$$

where x_t denotes the position and w_t denotes process noise. The velocity can only help in establishing a relative position, which means that additional information is needed to solve the localization problem. One solution is to make use of a map of the terrain elevation and a downward-facing radar, measuring the distance between the aircraft and the ground. Under the assumption that the ground is not entirely flat these radar measurements together with the terrain elevation map can be used in determining the position of the aircraft. The corresponding measurement equation is

$$y_t = h(x_t) + e_t, \qquad e_t \sim \mathcal{N}(0, 1), \tag{5.2b}$$

where y_t denotes the distance over ground measured by the radar, $h(\,\cdot\,)$ denotes a look-up in the map encoding the terrain elevation (terrain height over sea level) and e_t denote the measurement noise. This is in fact the principle employed to compute the absolute position in several fighter aircraft. Let us assume that we do not know the initial position of the aircraft, hence the initial guess encoded via $\mu(x_1)$ is a uniform distribution $\mathcal{U}(0, 100)$. The localization problem can now be formulated as a state inference problem, where we want to find $p(x_t \,|\, y_{1:t})$ for $t = 1, 2, \ldots$, i.e. as more and more measurements are acquired.

Let us pause for a moment and think about the nature of this problem. The measurement equation (5.2b) is nonlinear and the function $h(\,\cdot\,)$ is only defined in discrete points (according to the resolution of the map). Furthermore, as already mentioned there are an unknown and dynamically changing number of modes present in the filtering PDF. It is a hard problem, but it fits the PF perfectly, since it can deal with nonlinear functions and a varying number of possible hypotheses. Furthermore, the PF provides a natural representation of the uncertainty inherent in the problem and a principled way of working with this uncertainty.

The result of using the PF with $N = 200$ particles is shown in Figure 5.2. At time 1 we have the situation depicted in Figure 5.2 (upper left), where the aircraft is flying at an altitude of roughly 80 m at position $x_1 = 22$ m, where the terrain elevation is 20 m. This means that the radar is measuring a distance over ground of roughly 60 m. The function $h(\,\cdot\,)$ will look into the map and return the positions x_t in the map that are consistent with an altitude of 80 m and a distance to ground of 60 m (indicated by the horizontal dashed line in the figure). Hence, intersecting this horizontal dashed line with the black curve (representing the map) provides the locations that are consistent with y_1. Now, compare these positions to the PDF $\widehat{p}^N(x_1 \,|\, y_1)$ provided by the PF after the first measurement has been received and processed (upper right plot). The conclusion is that the PF is able to accurately represent the different possible position hypotheses in a good way, illustrating the PFs capability of representing the uncertainty inherent in the problem. The more the aircraft moves, the more information is acquired about its position. Let us study the situation leading up to time $t = 3$, explaining why it is natural

that there are two dominating modes present in $\widehat{p}^N(x_3 \,|\, y_{1:3})$ as shown in the lower left plot in Figure 5.2.

In studying the terrain elevation for three consecutive samples (the first one being at 20 m), the map reveals that the most likely positions at time $t = 3$ are $x_3 = 25$ and $x_3 = 60$. The position $x_3 = 46$ is also relevant, but the slope of the terrain is significantly steeper there, which the PF has accurately encoded in the PDF by only assigning a small probability mass around this position. Finally, once 10 measurements have been obtained the aircraft is localized quite well, illustrated in the lower right plot, showing $\widehat{p}^N(x_{10} \,|\, y_{1:10})$.

The principle illustrated in this example of using a PF to solve the localization problem by combining the information from sensors and maps has been successfully used to solve many different localization problems, including for example underwater vessels, ships, cars and people moving around in both indoor and outdoor environments.

An alternative view of the PF is to derive it as a realization of the forward smoothing strategy, i.e. as a way of sequentially approximating the joint smoothing densities $\{p(x_{1:t} \,|\, y_{1:t})\}_{t\geq 1}$, again using an empirical distribution, this time in the form

$$\widehat{p}^N(x_{1:t} \,|\, y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{x_{1:t}^i}(x_{1:t}). \tag{5.3}$$

In comparing the two objects (5.1) and (5.3), the most striking difference is that the dimension of the variable we are estimating is fixed in the former, whereas it increases without an upper bound for the latter. More specifically, $\widehat{p}^N(x_t \,|\, y_{1:t})$ is defined on the space X of fixed dimension (independent of t), whereas $\widehat{p}^N(x_{1:t} \,|\, y_{1:t})$ is defined on a space X^t of increasing dimension. This will have consequences, which will be precisely quantified later in this chapter. However, pragmatically we can already now draw the conclusion that the PF is bound to fail in its attempt of realizing the forward smoothing strategy, since it is inherently impossible to make use of a finite number of N particles to represent a distribution of arbitrarily high dimension.

It is important to note that the joint smoothing density $p(x_{1:T} \,|\, y_{1:T})$ (or some of its marginals) can indeed be approximated using particle methods as well. The resulting method relies on particle implementations of not only the forward computational strategies, but also of the backward computational strategies, which leads to particle smoothers. However, in the present chapter we offer two different views of the PF, both interesting in their own way. Firstly, the PF is in Section 5.2 derived as a way of approximately realizing the forward filtering strategy, i.e. as a way of sequentially approximating the filtering densities $\{p(x_t \,|\, y_{1:t})\}_{t\geq 0}$. Secondly, it is in Section 5.3 derived as an approximate realization of the forward smoothing strategy, i.e. as a way of sequentially approximating the joint smoothing densities $\{p(x_{0:t} \,|\, y_{1:t})\}_{t\geq 0}$. The filtering density is of course a marginal of the joint smoothing density obtained by integrating out $x_{0:t-1}$. Hence, after this marginalization we would expect the resulting algorithms to be identical—as we will see, this is indeed the case.

5.2 The particle filter – targeting the filtering PDF

In Section 4.3.4, we attempted to solve the nonlinear smoothing problem by using the SIS algorithm. The attempt failed due to the increasing dimension of the state trajectories $x_{0:t}$, leading to the so-called weight degeneracy problem. To make progress, we will start our exploration of SMC methods by considering a different, but still challenging problem, namely that of nonlinear filtering (which is of significant importance on its own). Filtering amounts to computing the filtering PDFs $\{p(x_t \mid y_{1:t})\}$ when the state and measurement processes are modelled according to an SSM (Definition **??**). Since the dimension of the state space X, on which this density is defined, does not depend on t, we are now in a much better situation.

5.2.1 The marginal and the bootstrap particle filters

The nonlinear filtering problem amounts to computing the filtering PDFs $\{p(x_t \mid y_{1:t})\}$ sequentially in time. A principled solution is provided by the following two recursive equations (see Section 3.2.1 for details):

$$p(x_t \mid y_{1:t}) = \frac{p(y_t \mid x_t)p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}, \tag{5.4a}$$

$$p(x_t \mid y_{1:t-1}) = \int p(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1})\mathrm{d}x_{t-1}. \tag{5.4b}$$

As we saw in the previous chapter, importance sampling offers a way to approximate a probability distribution of interest by an empirical weighted point-mass distribution. The problem in designing an importance sampler for $p(x_t \mid y_{1:t})$, however, is that to evaluate (5.4a) even up to proportionality, we need to solve the integral in (5.4b). This is in general not analytically tractable. However, the integral can in principle be approximated using an importance sampler targeting the filtering distribution at time $t-1$. This motivates us to proceed in an inductive fashion. Hence, assume that we have an empirical approximation of the filtering distribution at time $t-1$, constituted by N weighted samples, $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N$, i.e.

$$\widehat{p}^N(x_{t-1} \mid y_{1:t-1}) = \sum_{i=1}^N w_{t-1}^i \delta_{x_{t-1}^i}(x_{t-1}). \tag{5.5}$$

At time $t = 1$, we can obtain a point-mass approximation according to (5.5), by targeting $p(x_1 \mid y_1) \propto p(y_1 \mid x_1)\mu(x_1)$ with an importance sampler. Inserting the approximation $\widehat{p}^N(x_{t-1} \mid y_{1:t-1})$ into (5.4b), results in

$$\widehat{p}^N(x_t \mid y_{1:t-1}) = \int p(x_t \mid x_{t-1}) \sum_{i=1}^N w_{t-1}^i \delta_{x_{t-1}^i}(x_{t-1})\mathrm{d}x_{t-1} = \sum_{i=1}^N w_{t-1}^i p(x_t \mid x_{t-1}^i). \tag{5.6}$$

Using (5.6) in (5.4a), we can thus evaluate *an approximation of* the filtering PDF $p(x_t \mid y_{1:t})$ up to proportionality

$$p(x_t \mid y_{1:t}) \approx \frac{1}{p(y_t \mid y_{1:t-1})} \sum_{i=1}^{N} w_{t-1}^i p(y_t \mid x_t) p(x_t \mid x_{t-1}^i). \tag{5.7}$$

This opens up for targeting $p(x_t \mid y_{1:t})$ with an importance sampler. Guided by the structure of (5.7), we choose a similar type of mixture as proposal density, namely

$$q(x_t \mid y_{1:t}) = \sum_{i=1}^{N} w_{t-1}^i q(x_t \mid x_{t-1}^i, y_t). \tag{5.8}$$

There are many different options when it comes to choosing the components $q(x_t \mid x_{t-1}^i, y_t)$ in this mixture. We will investigate some of the most common choices and also explain their advantages and disadvantages in the sequel. For now, however, we simply assume that each component in the mixture is chosen in such a way that it is possible to sample from it. It should be noted that the mixture (5.8) depends on the particle system $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N$, but this dependence is not explicitly accounted for on the left hand side of the expression for notational simplicity. However, since (5.8) is to be used at time t of the algorithm, the particle system $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N$ (which is generated at time $t-1$) can be viewed as fixed. Note also that, in general, the proposal density at time t is allowed to depend on the current observation y_t, as indicated by the notation used in (5.8).

Sampling from the proposal density $q(x_t \mid y_{1:t})$, which is a weighted mixture comprising N components, can be done by the following two-step procedure; first we randomly select one of the components, and then we generate a sample from that specific component. For the first part, the probability of selecting a particular component $q(x_t \mid x_{t-1}^i, y_t)$ is encoded via the corresponding weight w_{t-1}^i. Slightly reformulated this amounts to selecting one of the N particles $\{x_{t-1}^i\}_{i=1}^N$ according to

$$\mathbb{P}\Big(\bar{x}_{t-1} = x_{t-1}^i \,\Big|\, \{x_{t-1}^j, w_{t-1}^j\}_{j=1}^N\Big) = w_{t-1}^i, \qquad \text{for } i = 1, \ldots, N, \tag{5.9}$$

where the selected particle is denoted by $\bar{x}_{t-1}$. We can then draw $x_t \sim q(x_t \mid \bar{x}_{t-1}, y_t)$ to generate a realization from the proposal distribution (5.8). Since x_t is generated conditionally on $\bar{x}_{t-1}$, we will refer to $\bar{x}_{t-1}$ as the *ancestor* particle of x_t.

It is interesting to note that, by repeating (5.9) N times we obtain an *unweighted* particle system $\{\bar{x}_{t-1}^i, 1/N\}_{i=1}^N$. This procedure which (randomly) turns a weighted set of samples into an unweighted set is commonly referred to as *resampling*. Analogously to (5.5), these unweighted particles define an empirical point-mass approximation of $p(x_{t-1} \mid y_{1:t-1})$,

$$\bar{p}^N(x_{t-1} \mid y_{1:t-1}) := \frac{1}{N} \sum_{i=1}^{N} \delta_{\bar{x}_{t-1}^i}(x_{t-1}). \tag{5.10}$$

In (5.5), the weights contain information about how useful each particle is for the approximation of the target distribution. For the approximation (5.10), this information is instead encoded via the multiplicity of the particles. Indeed, if some particle in the approximation (5.5) is associated with a large weight, then this particle is likely to be selected multiple times when sampling $\{\bar{x}_{t-1}^i\}_{i=1}^N$ according to (5.9).

Based on the resampled particles, we can generate N values from the proposal density (5.8) by sampling $x_t^i \sim q(x_t \,|\, \bar{x}_{t-1}^i, y_t)$ for $i = 1, \ldots, N$. As usual in importance sampling, the samples are then assigned importance weights to account for the discrepancy between the target distribution and the proposal distribution, recall (4.19). From (5.7) and (5.8) it follows that the weight function is given by

$$\omega(x_t, y_t) = \frac{p(y_t \,|\, x_t) \sum_{j=1}^N w_{t-1}^j p(x_t \,|\, x_{t-1}^j)}{\sum_{j=1}^N w_{t-1}^j q(x_t \,|\, x_{t-1}^j, y_t)}. \tag{5.11}$$

Again, for notational simplicity, we have not made the dependence on the previous particle system $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N$ explicit on the left-hand-side. By evaluating $\bar{w}_t^i = \omega(x_t^i, y_t)$ for $i = 1, \ldots, N$ and normalizing the weights, we obtain a new particle system $\{x_t^i, w_t^i\}_{i=1}^N$, constituting an empirical approximation of $p(x_t \,|\, y_{1:t})$. This completes the algorithm, since these weighted particles in turn can be used to approximate the filtering PDF at time $t+1$, then at time $t+2$ and so on.

The resulting algorithm is referred to as the *marginal particle filter* since its starting point is the filtering density $p(x_t \,|\, y_{1:t})$, rather than the smoothing density $p(x_{1:t} \,|\, y_{1:t})$ (the former density is a marginal of the latter). In Section 5.3 we will discuss particle filtering from the perspective of targeting the smoothing density. While the above development indeed results in a working method, we notice a problem with the resulting algorithm at this point. Due to the sums appearing in (5.11), the computational complexity of evaluating the weight function is $\mathcal{O}(N)$. This has to be done for each of the particles $\{x_t^i\}_{i=1}^N$, meaning that the overall computational complexity of the algorithm will be $\mathcal{O}(N^2)$. Since N is typically a large number, this quadratic complexity can be impractical in many applications.

However, from the expression (5.11), we see that there is an easy remedy to the problem of quadratic computational complexity. A pragmatic solution is to choose the proposal density according to

$$q(x_t \,|\, y_{1:t}) := \widehat{p}^N(x_t \,|\, y_{1:t-1}) = \sum_{j=1}^N w_{t-1}^j p(x_t \,|\, x_{t-1}^j). \tag{5.12}$$

With this choice, the sums cancel in (5.11) and the expression for the weight function reduces to $\omega(x_t, y_t) = p(y_t \,|\, x_t)$. This function can be evaluated in constant time, bringing the total computational complexity of the algorithm down to $\mathcal{O}(N)$.

The PF which uses the predictive distribution (5.12) as proposal distribution is commonly referred to as the *bootstrap particle filter*. It is without doubt the most commonly

used implementation in practice, owing to its simplicity and intuitive appeal. We summarize the method in Algorithm 5.1.

Algorithm 5.1: Bootstrap particle filter

Initialization ($t = 1$)**:**
  Sample $x_1^i \sim \mu(x_1)$.
  Compute $\bar{w}_1^i = p(y_1 \,|\, x_1^i)$ and normalize, $w_1^i = \bar{w}_1^i / \sum_{j=1}^N \bar{w}_1^j$.
for $t = 2$ **to** T **do**
  Resampling: Generate the equally weighted particle system $\{\bar{x}_{t-1}^i, 1/N\}_{i=1}^N$ by resampling of $\{x_{t-1}^i, w_{t-1}^i\}_{i=1}^N$ according to (5.9).
  Propagation: Sample $x_t^i \sim p(x_t \,|\, \bar{x}_{t-1}^i)$.
  Weighting: Compute $\bar{w}_t^i = p(y_t \,|\, x_t^i)$ and normalize, $w_t^i = \bar{w}_t^i / \sum_{j=1}^N \bar{w}_t^j$.
end

Using the predictive density (5.12) to propose new samples can seem natural, since it leverages the predictive capabilities of the model. The weight w_t^i carries information about how well the corresponding sample x_t^i describes the true state of the system. Hence, computing the weights according to the measurement model $p(y_t \,|\, x_t^i)$ is natural, since $p(y_t \,|\, x_t^i)$ describes how well the state x_t^i explains the current measurement y_t. Nevertheless, as we will see in the subsequent section, it is possible to design more efficient proposal distributions than (5.12) while still, importantly, retaining the $\mathcal{O}(N)$ computational complexity.

Example 5.2

Consider the following nonlinear time-varying SSM

$$x_{t+1} = \frac{1}{2}x_t + \frac{25x_t}{1+x_t^2} + 8\cos(1.2t) + v_t, \qquad v_t \sim \mathcal{N}(0, 0.5), \tag{5.13a}$$

$$y_t = \frac{1}{20}x_t^2 + e_t, \qquad e_t \sim \mathcal{N}(0, 0.5), \tag{5.13b}$$

where the initial state is $x_1 \sim \mathcal{N}(0, 1)$. Note that one complication with the model is that the measurement does not contain any information about the sign of the state, since the measurement is a function of x_t^2. Based on $T = 100$ measurements $y_{1:100}$ from (5.13) the task is now to find an approximation of the filtering PDFs $p(x_t \,|\, y_{1:t})$. To solve this task we make use of the bootstrap PF, which amounts to using Algorithm 5.1 with the following design choices

$$q_1(x_1 \,|\, y_1) = \mathcal{N}\left(x_1 \,|\, 0,\, 1\right), \tag{5.14a}$$

$$q_t\left(x_t \,|\, x_{t-1}, y_{1:t}\right) = \mathcal{N}\left(x_t \,\middle|\, \frac{1}{2}x_{t-1} + \frac{25x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2(t-1)),\, 0.5\right). \tag{5.14b}$$

This results in the weight function,

$$\omega\left(x_t, y_t\right) = g(y_t \,|\, x_t) = \mathcal{N}\left(y_t \,\middle|\, 1/20x_t^2,\, 0.5\right). \tag{5.15}$$

In the simulations $N = 10\,000$ particles are used, which is quite a lot for a one-dimensional state space. The left plot in Figure 5.3 shows the conditional mean estimate provided by the PF and the true state. Comparing the particle filter mean estimate to

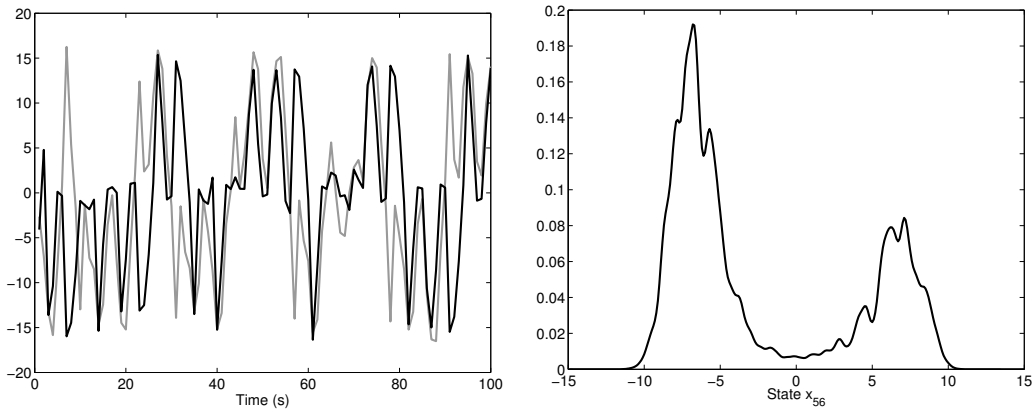

Figure 5.3: Left plot: the conditional mean estimate (gray curve) and the true state (black curve). Right plot: the estimate of the filter PDF at time $t = 56$, i.e. $\widehat{p}^N(x_{56} \mid y_{1:56})$.

the true state reveals at least two things. First, this is an inherently hardproblem, especially due to the squared state in the measurement equation. Second, just looking at a point estimate can be dangerous and misleading in a complex problem like this. To understand this we have provided a plot of $\widehat{p}^N(x_{56} \mid y_{1:56})$ in the right plot in Figure 5.3. From this plot we can see that the PF is capable of representing the uncertainty of the sign of x_t due to the square in the measurement equation. It seems natural that this square can result in a bimodal filter PDF.

The bootstrap PF is the simplest possible implementation of SMC, but nevertheless, it incorporates the essential methodological ideas that underpin the general SMC framework. Importance sampling and resampling is used sequentially to approximate a sequence of probability distributions of interest. A highly valid concern at this point might therefore be the following:

> *The approximation of the filtering distribution at time t is used to construct an approximation of the filtering distribution at time $t+1$. This approximation is subsequently used at time $t+2$, then at $t+3$, and so on. Should we then not expect that the approximation errors accumulate over time, rendering the solution useless as $t \to \infty$?*

The question raised above concerns the stability of the PF approximation. We will discuss this in more detail in Section 5.8, but somewhat simplified the answer is that the PF is indeed stable under certain *forgetting* conditions on the model under study. That is, if the model is such that the there is a (sufficiently large) decay in the influence of the past on the present, then there will be no indefinite accumulation of errors in the PF.

5.2.2 Using auxiliary variables

Consider again the *bootstrap proposal* (5.12). While intuitively and computationally appealing, this choice of proposal distribution is unfortunately also suboptimal. The reason is as follows. In the propagation step, when simulating the particles $\{x_t^i\}_{i=1}^N$ from the mixture distribution (5.12), the current measurement y_t is not taken into account. However, intuitively, y_t contains information about the current state of the system, i.e. about the shape of the filtering density $p(x_t \mid y_{1:t})$. Since y_t is indeed available at time t, it would be preferable to make use of this information already when simulating the particles $\{x_t^i\}_{i=1}^N$, to increase the probability of producing samples in the most relevant parts of the state space. Indeed, this is why we indicate a possible dependence on y_t in the mixture components of the proposal density in (5.8).

In Section 5.2.4 we will be more specific about how it is possible to adapt the proposal distribution to the available information. To prepare for such a development we will now—with the above discussion in mind—focus on enabling the use of the proposal density (5.8), while still enjoying a linear computational cost. This can be accomplished by making use of auxiliary variables, as discussed in Section 4.5.2. In fact, in Section 4.5.2 the problem of designing an efficient importance sampler for a mixture distribution was considered in a general setting. We now make use of the same idea specifically in the context of particle filtering.

As pointed out above, to sample from the mixture proposal (5.8), we would first randomly select one of the components, and then draw a sample from that specific component. In (5.9), we noted that selecting one of the mixture components is equivalent to selecting an ancestor $\bar{x}_{t-1}$ among the particles $\{x_{t-1}^i\}_{i=1}^N$ at time $t-1$. We can then generate a realization from (5.8) by simulating $x_t \sim q(x_t \mid \bar{x}_{t-1}, y_t)$. To make the selection procedure in (5.9) more explicit, let us introduce an auxiliary variable in the form of a discrete random variable a_t which takes values on the set of integers $\{1, \ldots, N\}$. We can then express (5.9) as $\bar{x}_{t-1} := x_{t-1}^{a_t}$ where

$$\mathbb{P}\Big(A_t = i \,\Big|\, \{x_{t-1}^j, w_{t-1}^j\}_{j=1}^N\Big) = w_{t-1}^i, \qquad \text{for } i = 1, \ldots, N. \tag{5.16}$$

Hence, A_t is the index of the ancestor particle at time $t-1$. Consequently, we will refer to a_t as an *ancestor index*. Introducing such ancestor indices as auxiliary variables is useful, since it makes the stochasticity of the resampling step explicit. Furthermore, analogously to the development in Section 4.5.2, it allows us to reduce the computational complexity of the weight computation from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. The key in this development is to target the joint distribution of (x_t, a_t) with an importance sampler, instead of directly targeting the marginal distribution of x_t as was done in the previous section.

Analogously to above, the mixture proposal (5.8) can be interpreted as a joint proposal distribution for the pair (x_t, a_t), given by

$$q(x_t, a_t \mid y_{1:t}) = w_{t-1}^{a_t} q(x_t \mid x_{t-1}^{a_t}, y_t). \tag{5.17}$$

Here, the ancestor index a_t should be thought of as an index selecting one of the components in the sum (5.8). Generating, independently, N realizations from this joint proposal distribution can be done as follows.

1. Sample the ancestor indices $\{a_t^i\}_{i=1}^N$ according to (5.16). This corresponds to the resampling step of the algorithm. The resampled particles are given as $\bar{x}_{t-1}^i = x_{t-1}^{a_t^i}$ for $i = 1, \ldots, N$.

2. Propagate the particles to time t by simulating $x_t^i \sim q(x_t \,|\, \bar{x}_{t-1}^i, y_t)$ for $i = 1, \ldots, N$.

The advantage of explicitly introducing and using the ancestor indices as auxiliary variables lies in the computation of the importance weights (cf. Section 4.5.2). From (5.7), we have that the *joint* target distribution for (x_t, a_t) is proportional to

$$w_{t-1}^{a_t} p(y_t \,|\, x_t) p(x_t \,|\, x_{t-1}^{a_t}). \tag{5.18}$$

Note that there is no summation over previous particles involved in this expression—the summation in (5.7) is recovered by marginalizing the aforementioned joint target distribution over the auxiliary variable a_t. In computing the ratio between the unnormalized joint target density (5.18) and the joint proposal density (5.17), the factors $w_{t-1}^{a_t}$ will cancel:

$$\frac{w_{t-1}^{a_t^i} p(y_t \,|\, x_t^i) p(x_t^i \,|\, x_{t-1}^{a_t^i})}{w_{t-1}^{a_t^i} q(x_t^i \,|\, x_{t-1}^{a_t^i}, y_t)} = \frac{p(y_t \,|\, x_t^i) p(x_t^i \,|\, x_{t-1}^{a_t^i})}{q(x_t^i \,|\, x_{t-1}^{a_t^i}, y_t)} = \frac{p(y_t \,|\, x_t^i) p(x_t^i \,|\, \bar{x}_{t-1}^i)}{q(x_t^i \,|\, \bar{x}_{t-1}^i, y_t)}.$$

Hence, we define the weight function $\omega : \mathsf{X}^2 \times \mathsf{Y} \mapsto \mathbb{R}$ as,

$$\omega(x_{t-1}, x_t, y_t) = \frac{p(y_t \,|\, x_t) p(x_t \,|\, x_{t-1})}{q(x_t \,|\, x_{t-1}, y_t)}. \tag{5.19}$$

By computing $\bar{w}_t^i = \omega(\bar{x}_{t-1}^i, x_t^i, y_t)$ for $i = 1, \ldots, N$ and normalizing the weights to sum to one, we obtain the joint weighted particle system $\{(x_t^i, a_t^i), w_t^i\}_{i=1}^N$. However, assuming that we are only interested in addressing the filtering problem, the auxiliary variables $\{a_t^i\}_{i=1}^N$ may be discarded. This leaves us with the weighted particle system $\{x_t^i, w_t^i\}_{i=1}^N$ approximating the (marginal) target density (5.7). Importantly, evaluating (5.19) can be done in constant time, leading to a total computational complexity of the particle filter which is linear in N. Note that, contrary to (5.11), the weight function (5.19) depends not only on the current state x_t, but also on the previous state x_{t-1}. This implicitly encodes the dependence on the ancestor index a_t through the corresponding ancestor particle $\bar{x}_{t-1}$.

Interestingly, if we use the bootstrap proposal $q(x_t \,|\, x_{t-1}, y_t) = f(x_t \,|\, x_{t-1})$, the weight function (5.19) reduces to $\omega(x_{t-1}, x_t, y_t) = g(y_t \,|\, x_t)$ and we recover exactly the same *bootstrap PF* algorithm that was derived in the previous section.

5.2.3 The auxiliary particle filter

The introduction of ancestor indices as auxiliary variables opens up for further adaption of the proposal distribution used in the particle filter. In particular, it is possible to use the information available in the current observation y_t, not only when proposing the new

state x_t, but also when proposing its ancestor index a_t. Simulating the ancestor indices $\{a_t^i\}_{i=1}^N$, or equivalently resampling the particles $\{x_{t-1}^i\}_{i=1}^N$, is done to facilitate sampling of the particles $\{x_t^i\}_{i=1}^N$ in the propagation step of the filter. It is natural to attempt to take as much information as possible into account when sampling the ancestor indices, and this includes the current observation y_t. The idea is that we can thereby increase the probability of resampling particles at time $t-1$ that are in agreement with the observation y_t. This results in what is often referred to as the *auxiliary particle filter* (APF).

Recall that we use (5.17) as a joint proposal distribution for the pair (x_t, a_t). In this expression, it is assumed that a_t is distributed according to the categorical distribution induced by the importance weights $\{w_{t-1}^i\}_{i=1}^N$. However, we are free to use any proposal distribution that we find appropriate, as long as this is compensated for when computing the importance weights. Specifically, let $\nu : \mathsf{X} \times \mathsf{Y} \mapsto \mathbb{R}$ be a user-specified nonnegative function that will be used to adapt the proposal distribution for the ancestor indices. See Example 5.3 below for an illustration of how this function can be chosen. For each particle, i.e. for $i = 1, \ldots, N$, we then compute the quantities

$$\boldsymbol{\nu}_{t-1}^i := \nu(x_{t-1}^i, y_t), \tag{5.20}$$

referred to as *adjustment multipliers*. Importantly, these quantities depend only on the previous particles and on the current observation, both of which are available when simulating the ancestor indices (i.e., in the resampling step) in the algorithm at time t. The adjustment multipliers are used to construct a proposal distribution for the ancestor index variable a_t according to (cf. (5.16))

$$\mathbb{P}\Big(A_t = i \,\Big|\, \{x_{t-1}^j, w_{t-1}^j\}_{j=1}^N, y_t\Big) = \frac{w_{t-1}^i \boldsymbol{\nu}_{t-1}^i}{\sum_{l=1}^N w_{t-1}^l \boldsymbol{\nu}_{t-1}^l}, \qquad \text{for } i = 1, \ldots, N. \tag{5.21}$$

From the above expression we see that we can think of ν as adjusting the original weights w via multiplication, explaining the name adjustment multipliers. As before, once the ancestor indices are generated, we propagate the particles to time t by simulating $x_t^i \sim q(x_t \,|\, \bar{x}_{t-1}^i, y_t)$ for $i = 1, \ldots, N$, where $\bar{x}_{t-1} = x_{t-1}^{a_t}$. As mentioned above, the idea underlying the use of adjustment multipliers is that by carefully choosing the function ν in (5.20), we can adapt the sampling of the ancestor indices in (5.21) to make use of the information available in the current measurement y_t. Intuitively, $\nu(x_{t-1}, y_t)$ should take a large value if it is likely to observe the measurement y_t at time t, given that the system state is x_{t-1} at time t. To make this more concrete, we provide an example of how the adjustment multipliers can be chosen below.

Example 5.3: Computing adjustment multipliers

To compensate for the fact that we use an adjusted proposal distribution (5.21) for the ancestor index, we have to modify the weight function (5.19) accordingly. The

normalization constant in (5.21) can be neglected, since the weights are normalized to sum to one. We get the following expression for the ith importance weight:

$$\frac{w_{t-1}^{a_t^i} p(y_t \,|\, x_t^i) p(x_t^i \,|\, x_{t-1}^{a_t^i})}{w_{t-1}^{a_t^i} \boldsymbol{\nu}_{t-1}^{a_t^i} q(x_t^i \,|\, x_{t-1}^{a_t^i}, y_t)} = \frac{p(y_t \,|\, x_t^i) p(x_t^i \,|\, \bar{x}_{t-1}^i)}{\nu(\bar{x}_{t-1}^i, y_t) q(x_t^i \,|\, \bar{x}_{t-1}^i, y_t)}.$$

Hence, the weight function is now given by

$$\omega(x_{t-1}, x_t, y_t) = \frac{p(y_t \,|\, x_t) p(x_t \,|\, x_{t-1})}{\nu(x_{t-1}, y_t) q(x_t \,|\, x_{t-1}, y_t)}. \tag{5.22}$$

The particle filter discussed in the previous section is a special case of the APF with the choice $\nu(x_{t-1}, y_t) \equiv 1$. We summarize the APF in Algorithm 5.2.

Algorithm 5.2: Auxiliary particle filter (APF)

Initialization ($t = 1$):
  Sample $x_1^i \sim q(x_1 \,|\, y_1)$.
  Compute the importance weights $\bar{w}_1^i = p(y_1 \,|\, x_1^i)\mu(x_1^i)/q(x_1^i \,|\, y_1)$ and normalize, $w_1^i = \bar{w}_1^i / \sum_{j=1}^N \bar{w}_1^j$.
for $t = 2$ **to** T **do**
  Compute the adjustment multipliers $\boldsymbol{\nu}_{t-1}^i = \nu(x_{t-1}^i, y_t)$
  Resampling: Resample $\{x_{t-1}^i\}_{i=1}^N$ with probabilities proportional to $\{w_{t-1}^i \boldsymbol{\nu}_{t-1}^i\}_{i=1}^N$ to generate the equally weighted particle system $\{\bar{x}_{t-1}^i, 1/N\}_{i=1}^N$.
  Propagation: Sample $x_t^i \sim q(x_t \,|\, \bar{x}_{t-1}^i, y_t)$.
  Weighting: Compute $\bar{w}_t^i = \omega(\bar{x}_{t-1}^i, x_t^i, y_t)$ and normalize, $w_t^i = \bar{w}_t^i / \sum_{j=1}^N \bar{w}_t^j$.
end

The APF is sometimes viewed as a look-ahead procedure, since we make use of the information available in y_t when resampling the particles $\{x_{t-1}^i\}_{i=1}^N$. In fact, we can take this idea even further and adapt the proposal distributions of the particle filter to, say, the next ℓ measurements $y_{t:t+\ell}$. Of course, the algorithm then needs to run with a time lag of ℓ time steps, so that $y_{t:t+\ell}$ are indeed available to us when proposing the particles at time t. In practice, however, it can be hard to design a good adaption scheme, especially when adapting to future observations.

5.2.4 Adapting the proposal distribution

To make the above development more practical, we will in this section discuss specific choices for the proposal distributions used in the APF. We have argued that it is useful to adapt the proposals to the information that is available in the current observation y_t. Consider first the proposal for the state x_t (we will return to the choice of adjustment multipliers below). A natural choice is to use

$$q(x_t \,|\, x_{t-1}, y_t) = p(x_t \,|\, x_{t-1}, y_t), \tag{5.23}$$

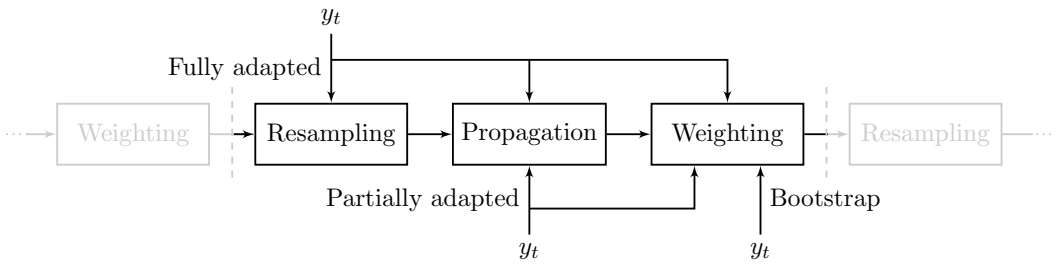

Figure 5.4:

i.e. we propose samples according to the posterior distribution of x_t, conditionally on the previous state x_{t-1} and the current observation y_t. This choice is referred to as the *optimal* proposal, since it minimizes the variance of the importance weights (see below). For many models, the optimal proposal distribution is not available in closed form, though there are some special cases of interest for which it can be computed, as we illustrate in Example 5.4 below.

Example 5.4: Exploiting the current measurement in the proposal

Let us now consider the effect on the weight function (5.22) by using the optimal proposal (5.23). From Bayes' rule, we can write

$$p(x_t \,|\, x_{t-1}, y_t) = \frac{p(y_t \,|\, x_t)p(x_t \,|\, x_{t-1})}{p(y_t \,|\, x_{t-1})}. \tag{5.24}$$

Plugging this expression into the denominator of (5.22) we obtain

$$\omega(x_{t-1}, x_t, y_t) = \omega(x_{t-1}, y_t) = \frac{p(y_t \,|\, x_{t-1})}{\nu(x_{t-1}, y_t)}, \tag{5.25}$$

which depends only on the previous state and the current observation (which we have indicated by overloading the function notation ω). Put differently, the proposal distribution for x_t is such that the actual values of the particles $\{x_t^i\}_{i=1}^N$ do not affect the corresponding importance weights. We say that the proposal is adapted to the observation y_t. This also motivates the word *optimal*—conditionally on the ancestor particle $\bar{x}_{t-1}$ the choice (5.23) for proposing x_t trivially minimizes the variance of $w_t = \omega(\bar{x}_{t-1}, x_t, y_t)$, since it results in zero variance.

The expression (5.25) also suggests how to choose the adjustment multipliers. In particular, wee see the choice $\nu(x_{t-1}, y_t) = p(y_t \,|\, x_{t-1})$ will result in a weight function that is identically equal to 1. That is, we obtain particles that are generated in such a way that, by construction, they are equally informative about the target distribution. We say that the APF is *fully adapted*. This choice of adjustment multipliers is in agreement with the discussion in Section 5.2.3. The function $\nu(x_{t-1}, y_t)$ should intuitively be large

when the state x_{t-1} at time $t-1$ is in agreement with the observation y_t at time t; the one-step predictive likelihood $p(y_t \mid x_{t-1})$ accomplishes exactly this. A schematic illustration of the differences between no, partial, or full adaption of the APF is given in Figure 5.4. We continue to study the ARCH model from Example 5.4 and illustrate how to construct a fully adapted APF in the example below.

Example 5.5: A fully adapted APF

In the example above we exploited the fact that, conditionally on x_{t-1}, the current state and observation (x_t, y_t) are jointly Gaussian. As a result, both the conditional density $p(x_t \mid x_{t-1}, y_t)$ (i.e., the optimal proposal) and the marginal density $p(y_t \mid x_{t-1})$ (i.e., the optimal adjustment multiplier function) turned out to be Gaussian and we could express them on closed form. Using the same technique, we can conclude that full adaptation is possible whenever (x_t, y_t) are jointly Gaussian, conditionally on x_{t-1}. This is the case whenever the transition density $f(x_t \mid x_{t-1})$ is Gaussian and the measurement equation is linear and Gaussian: $g(y_t \mid x_t) = \mathcal{N}(y_t \mid Cx_t, R)$ for some parameters C and R.

For models where the optimal proposals cannot be computed, it is still possible to make use of the APF to construct a filter which, in many cases, is more efficient than a simple bootstrap PF. One common approach is to use some approximate method, for instance an extended Kalman filter , to approximate the optimal proposal (5.23). Note that the approximation error associated with such an approach will only affect the (sub-)optimality of the proposal distribution, not the asymptotic consistency of the PF. Typically, whenever it is possible to adapt $q(x_t \mid x_{t-1}, y_t)$, then it is also possible to adapt $\nu(x_{t-1}, y_t)$. For instance, if (5.23) is approximated using an EKF, then this will result in an approximation of $p(y_t \mid x_{t-1})$ as a byproduct. Hence, if one puts effort into adapting one part of the proposal then it is usually worthwhile to adapt the other part as well.

Compared to a bootstrap PF, the type of adaption that we have discussed above amounts to making more efficient use of the information that is available in the observation y_t. This implies that adaption is particularly useful when the observation is informative. Indeed, if the measurement model $p(y_t \mid x_t)$ is "peaky", then the bootstrap PF tends to perform poorly. Intuitively, this can be understood by noting that the bootstrap proposal distribution (5.12) results in particles $\{x_t^i\}_{i=1}^N$ that are simulated blindly, i.e. without taking y_t into account. However, if y_t is highly informative about the underlying state of the system, then this can result in particles with very small posterior probability given y_t. The result is that the variance of the (normalized) importance weights will be large, effectively resulting in a small number of useful particles. The effect of adapting the APF is illustrated in a numerical example below.

Example 5.6

5.3 The particle filter – targeting the smoothing PDF

In this section we provide an alternative derivation of the PF. Specifically, we combine SIS and resampling to approximate the sequence of joint smoothing densities $\{p(x_{0:t} \mid y_{1:t})\}$ sequentially in time. It should be noted that we will *not* derive a new set of algorithms. Indeed, from an algorithmic point of view, the methods discussed in this section are equivalent to the PFs that we have already seen. However, by targeting the sequence of smoothing densities, rather than the filtering densities, we obtain an alternative view of the PF. This will help to illuminate certain properties of the method and, indeed, embracing both viewpoints is necessary in order to truly understand all the nuances of SMC.

5.3.1 Approximating the forward smoothing strategy

As mentioned above, we will derive the PF as a numerical approximation of the forward smoothing strategy (see Section 3.2.2 for details), given by the recursion

$$p(x_{0:t} \mid y_{1:t}) = p(x_{0:t-1} \mid y_{1:t-1}) \frac{p(x_t \mid x_{t-1}) p(y_t \mid x_t)}{p(y_t \mid y_{1:t-1})}. \tag{5.26}$$

Rather than sampling individual states x_t^i we are now sampling entire trajectories $x_{0:t}^i$, resulting in a representation of the form

$$\widehat{p}^N(x_{0:t} \mid y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{x_{0:t}^i}(x_{1:t}). \tag{5.27}$$

The starting point for the derivation is the SIS algorithm, considered already in Section 4.3.4. We briefly review the algorithm here for convenience. The SIS algorithm is a standard importance sampler, albeit with the constraint that the weighted particles are updated sequentially as new measurements are obtained. This is achieved by using a proposal density that factorizes as,

$$q(x_{0:t} \mid y_{1:t}) = q(x_1 \mid y_1) \prod_{s=2}^{t} q(x_s \mid x_{s-1}, y_s). \tag{5.28}$$

Hence, we can draw a sample from the proposal $x_{0:t}^i \sim q(x_{0:t} \mid y_{1:t})$ by first generating $x_0^i \sim q(x_0)$, then $x_1^i \sim q(x_1 \mid y_1)$, etc. It follows that the corresponding importance weights can be updated sequentially as well. Recall that the weights are given by the

ratio between the (unnormalized) target density and the proposal density (see (4.19)), resulting in

$$\begin{aligned}\omega_t(x_{0:t}, y_t) &= \frac{p(y_t \mid x_t)p(x_t \mid x_{t-1})p(x_{0:t-1} \mid y_{0:t-1})}{q(x_{0:t} \mid y_{1:t})} \\ &= \frac{p(y_t \mid x_t)p(x_t \mid x_{t-1})}{q(x_t \mid x_{t-1}, y_t)}\omega_{t-1}(x_{0:t-1}, y_{t-1}),\end{aligned} \tag{5.29}$$

where (5.28) was used to establish the last equality. The initial weight function at time $t = 1$ is given by $\omega_1(x_1, y_1) = p(y_1 \mid x_1)\mu(x_1)/q(x_1 \mid y_1)$.

In SIS, the N particles (and corresponding importance weights) evolve independently, i.e. there is no interaction between the particles. The problem with this approach, as discussed and illustrated empirically in Section 4.3.4, is that the weight update (5.29) is not stable, in the sense that the variance of the normalized importance weights increases over time. This has the effect that all but one of the weights decreases to zero, and all emphasis is thus put on one of the particles (recall that the weights are normalized to sum to one).

The PF algorithm arises as a way to mitigate this limitation of SIS. The idea is to rejuvenate the sample by replicating particles with high weights and discarding particles with low weights. This is done by *resampling*. To comply with the previous notation, we study the process of resampling the particle system $\{x^i_{0:t-1}, w^i_{t-1}\}_{i=1}^N$ at time $t-1$. Hence, we have at hand a weighted particle approximation of the joint smoothing distribution at time $t-1$ (generated by SIS, say),

$$\widehat{p}^N(x_{0:t-1} \mid y_{1:t-1}) = \sum_{i=1}^N w^i_{t-1}\delta_{x^i_{0:t-1}}(x_{0:t-1}). \tag{5.30}$$

It is possible to construct a new, unweighted set of particles by sampling independently from (5.30). That is, we set $\bar{x}^j_{0:t-1} = x^i_{0:t-1}$ with probability w^i_{t-1}, for $j = 1, \ldots, N$. The equally weighted particle system $\{\bar{x}^j_{0:t-1}, \frac{1}{N}\}_{j=1}^N$ can be used to construct a point-mass approximation of $p(x_{0:t-1} \mid y_{1:t-1})$ analogously to (5.30),

$$\bar{p}^N(x_{0:t-1} \mid y_{1:t-1}) = \frac{1}{N}\sum_{i=1}^N \delta_{\bar{x}^i_{0:t-1}}(x_{1:t-1}). \tag{5.31}$$

In fact, this is nothing but a *vanilla Monte Carlo* approximation of the distribution (5.30). Based on this insight, we can conclude that resampling will introduce additional Monte Carlo variance, which implies that the approximation (5.31) will be dominated by (5.30). However, when applied sequentially, resampling is critical since it ensures that emphasis is put on the most likely hypotheses about the system state (i.e., particles). It introduces interactions between the particles by eliminating (with high probability) those particles that have low weights and duplicating particles with high weights.

Note that the sampling procedure outlined above is exactly the same as the one in (5.9). There, the interpretation was to select components in the mixture proposal (5.8);

here, the motivation was to rejuvenate an unequally weighted particle system. Both interpretations are correct and algorithmically they are equivalent.

Other concepts introduced in Section 5.2, such as ancestor indices and adjustment multipliers, can also be used in the present context. In particular, in accordance with the previous presentation, we can make use of the ancestor indices $\{a_t^i\}_{i=1}^N$ to make the sampling from (5.30) explicit. Indeed, if the discrete random variable a_t is drawn from a categorical distribution on $\{1, \ldots, N\}$ with probabilities $\{w_{t-1}^i\}_{i=1}^N$, then $\bar{x}_{1:t-1} := x_{0:t-1}^{a_t}$ is a draw from (5.30). The APF, derived in Section 5.2.3, employs the resampling step (5.21), in which the weights w_t are adjusted with the multipliers $\boldsymbol{\nu}_t$ to obtain the resampling probabilities. This corresponds to replacing the vanilla Monte Carlo approximation of (5.30) with an importance sampling approximation. Consequently, the APF can also equivalently be viewed as targeting the sequence of smoothing distributions.

To propagate the *resampled* particles $\{\bar{x}_{t-1}^i\}_{i=1}^N$ to time t we sample $x_t^i \sim q(x_t \mid \bar{x}_{t-1}^i, y_t)$ for $i = 1, \ldots, N$. Consequently, particle x_t^i originates from $\bar{x}_{t-1}^i = x_{t-1}^{a_t^i}$. Hence, when considering the particle *trajectory* $x_{0:t}^i$, this should be thought of as the "ancestral path" of the particle x_t^i. That is, $x_{0:t}^i$ is defined recursively through the ancestor indices by $x_{0:t}^i := (x_{0:t-1}^{a_t^i}, x_t^i)$. Since the concepts of ancestor indices and ancestral paths will be of importance later on, we illustrate the idea with an example.

Example 5.7: Ancestral paths

Figure 5.5 shows the evolution of three particles for $t = 1, 2, 3$. At time $t = 1$, particle x_1^2 is resampled twice and particle x_1^3 is resampled once. At time $t = 2$, the ancestors are thus given by $a_2^1 = 2$, $a_2^2 = 2$ and $a_2^3 = 3$. Similarly, at time $t = 3$, the ancestors are given by $a_3^1 = 2$, $a_3^2 = 3$ and $a_3^3 = 3$. The ancestral path of particle x_3^1, which we denote as $x_{1:3}^1$, is shown as a thick line in the figure. This path is given recursively from the ancestor indices,

$$x_{1:3}^1 = (x_1^{a_2^{a_3^1}}, x_2^{a_3^1}, x_3^1) = (x_1^{a_2^2}, x_2^2, x_3^1) = (x_1^2, x_2^2, x_3^1).$$

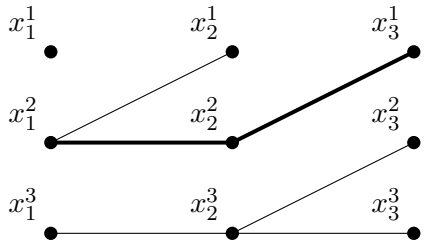

Figure 5.5: Evolution of a particle system. The ancestral path of x_3^1, i.e. $x_{1:3}^1$, is shown as a thick line.

The smoothing view of the PF offers one immediate advantage over the filtering view: since each particle consists of a complete state trajectory it is possible to let the proposal distribution depend on the complete history of the state process. That is, when simulating x_t^i at time t, this can be done from some proposal $q(x_t \mid \bar{x}_{0:t-1}^i, y_t)$ which may depend on the complete trajectory $\bar{x}_{0:t-1}^i$. For SSMs, this feature is usually not particularly useful due to the Markovian dependencies of the model. For other types of models, however, it can be key to designing an efficient sampler. Relying on the complete state trajectory for the proposal construction must be done with care, however, due to the path degeneracy issues that we will discuss in the subsequent section.

5.3.2 Path degeneracy

The SMC algorithm provided in Algorithm **??** is very ambitious in the sense that it is trying to make use of a representation consisting of a fixed sample size N in approximating a sequence of densities $\{p(x_{1:t} \mid y_{1:t})\}_{t\geq 1}$ of growing dimension. While we can show that this can be done *asymptotically* as $N \to \infty$, it is inherently impossible to solve this task using a fixed number of N particles. For any implementation of the PF we are of course limited to making use of a finite number of particles. Let us now investigate how this fundamental problem manifests itself, first by a thought experiment and then via a simulation in Example 5.8.

Assume that we employ a PF to target the joint smoothing density. At time s we generate N unique particles $\{x_s^i\}_{i=1}^N$ from the proposal density and augment these to the existing particle trajectories, according to step 6 in Algorithm **??**. We thus have a weighted particle system $\{x_{1:s}^i, w_s^i\}_{i=1}^N$ approximating the joint smoothing density at time s. Assume that the particle trajectories are then resampled, resulting in an unweighted particle system $\{\bar{x}_{1:s}^i, 1/N\}_{i=1}^N$. Recall that the purpose of the resampling step is to remove particles with small weights and duplicate particles with large weights. This has the effect of decreasing the number of unique particles. Hence, over time each consecutive resampling of the particle trajectories will reduce the number of unique particles at time s. Eventually the particle system at time s will consist of one unique particle copied to all particle trajectories. This problem is referred to as *path degeneracy*. In other words, the resampling step inevitable results in that for any given time s there exists $t > s$ such that the PF approximation $p(x_{1:t} \mid y_{1:t})$ consists of a single particle at time s. This is illustrated in Example 5.8, where $s = 6$ and $t = 25$, see the right plot in Figure 5.6.

Example 5.8: Path degeneracy

Consider a one-dimensional Gaussian random walk process measured in Gaussian noise. The joint smoothing density $p(x_{1:t} \mid y_{1:t})$ is targeted using a bootstrap PF with $N = 30$ particles. Let us study the path degeneracy problem by plotting the particle trajectories in Figure 5.6. Note that all particle trajectories $\{x_{1:25}^i\}_{i=1}^N$ share a common ancestor at time $s = 6$ (and consequently for all times prior to $s = 6$).

Despite the particle degeneracy problem, it is important to note that the PF does indeed generate weighted samples from the joint smoothing density. The problem that we face when using these samples for MC integration is that the resampling step introduces

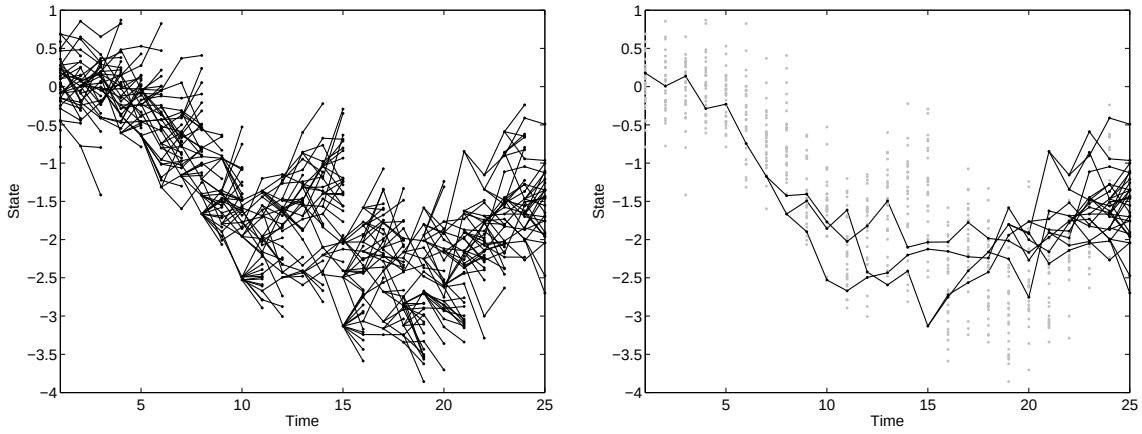

Figure 5.6: Left plot: At each point in time all $N = 30$ particles are plotted using a black dot and each particle is connected with its ancestor using a black line. Right plot: The grey dots represents the filtering density at each point in time. The black lines shows the particle trajectories $\{x^i_{1:25}\}_{i=1}^{30}$ at time $t = 25$. Hence, the right plot corresponds to the left plot with all trajectories that are not resampled removed. Note that all particles are still visualized using gray dots.

a dependency among the samples. When the trajectories have degenerated to the extent that there is only a single sample left, the particle trajectories are perfectly correlated. Consider for example the result of using the particle system $\{x^i_{1:25}\}_{i=1}^{N}$ to compute an MC estimate of $\mathbb{E}[x_3 \,|\, y_{1:25}]$, resulting in

$$\widehat{x}_{3\,|\,25} = \sum_{i=1}^{30} w^i_{25} x^i_3. \tag{5.32}$$

This boils down to an MC integration using a single sample, since x^i_3 are identical for all $i = 1, \ldots, N$.

The above development brings an interesting perspective on resampling. The resampling step successfully resolves the weight degeneracy problem. However, at the same time the resampling step results in path degeneracy, which makes it impossible to make use of SMC in order to obtain a good approximation of the joint smoothing density $p(x_{1:t} \,|\, y_{1:t})$ for large t using a finite number of particles. However, for many dynamical systems the future is more or less independent of the past. This is quantified using mixing conditions and ergodic theory. We will return to this in our discussion on convergence results, for now it will suffice with intuition. The ergodicity that is present in most dynamical systems effectively prevents the accumulation of error over time. The result is that SMC samplers are often good at approximating the fixed-lag smoothing densities $p(x_{t-L+1:t} \,|\, y_{1:t})$ for small lags $L \geq 1$. A direct consequence of this is that the particle degeneracy problem will *not* limit the applicability of the SMC sampler in targeting the filtering density $p(x_t \,|\, y_{1:t})$ (corresponding to $L = 1$). Looking at the right plot in

Figure 5.6 this does indeed seem plausible, since the particle system $\{x^i_{1:25}, w^i_{25}\}_{i=1}^{30}$ has not degenerated close to $t = 25$.

A relevant question at this stage is to consider if it is somehow possible (at least partly) to "undo" the path degeneracy. Above we just concluded that the PF is able to maintain a good representation of the filtering density. One idea is to investigate if it is somehow possible to initiate a backwards sweep starting from the PF representation of $p(x_t \mid y_{1:t})$ and somehow reintroduce diversity among the particles by some form of backwards computation. Recall the general backward computations derived in Section 3.3. This will be explored in quite some detail in Chapter **??** resulting in a family of algorithms referred to as *particle smoothers*.

5.4 Resampling algorithms

The resampling step is performed by a randomized algorithm generating an unweighted particle system from a weighted particle system, in such a way that the resulting unweighted representation is as similar to the original weighted representation as possible. The problem that is solved by resampling is formulated in Section 5.4.1, where we also derive a first algorithm that can be used to implement resampling. In Section 5.4.2 we will then derive two improvements to the first algorithms, where the improvements lies both in reduced computational complexity and a reduced variance.

5.4.1 Problem formulation

Pragmatically speaking resampling amounts to (randomly) eliminating particles with low probability and duplicating particles with high weights, such that for each $l = 1, \ldots, N$

$$\mathbb{P}\Big(\bar{x}^l = x^i \;\Big|\; \{x^j, w^j\}_{j=1}^N\Big) = w^i. \tag{5.33}$$

The resampling step is the same for all times, hence we have without loss of generality removed the time index t throughout this section. According to (5.33) we are generating samples approximately distributed according to the target distribution by selecting particle x^i with probability w^i. In repeating this N times we associate a *number of offspring* O^i with each particle x^i, i.e. O^i denotes the number of times particle x^i was selected in the resampling step. Hence, $O \triangleq (O^1, \ldots, O^N)$ is distributed according to a categorical distribution with parameters $\{w^i\}_{i=1}^N$. Resampling allows us to approximate the weighted distribution $\widehat{\pi}(x) = \sum_{i=1}^N w^i \delta_{x^i}(x)$ by an unweighted distribution

$$\bar{\pi}(x) = \sum_{i=1}^N \frac{1}{N} \delta_{x^{a^i}}(x) = \sum_{i=1}^N \frac{1}{N} \delta_{\bar{x}^i}(x) = \sum_{i=1}^N \frac{O^i}{N} \delta_{x^i}(x), \tag{5.34}$$

where we have made use of three equivalent formulations, based on the ancestor indices $\{a^i\}_{i=1}^N$, the resampled particles $\{\bar{x}^i\}_{i=1}^N$ and the offspring $\{O^i\}_{i=1}^N$, respectively. The resampling algorithms takes the weights $\{w^i\}_{i=1}^N$ as input and generates ancestor indices $\{a^i\}_{i=1}^N$ or $\{O^i\}_{i=1}^N$ as outputs.

In order to derive the *multinomial* resampling algorithm, let us start by considering an example consisting of 7 particles with weights $\{w^i\}_{i=1}^7$ illustrated in Figure 5.7 (left plot), where its cumulative distribution function $F(w)$ is shown.

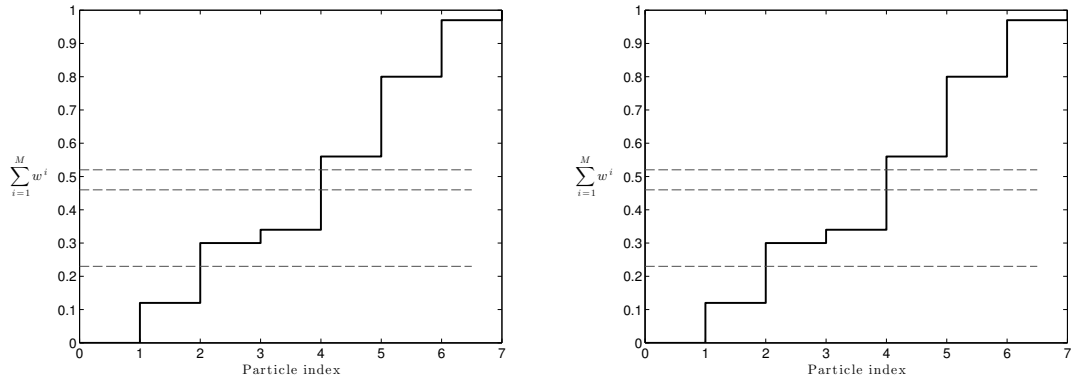

Figure 5.7: Left plot: an example of the cumulative distribution function $F(w)$ for an example consisting of 7 particles with weights $\{w^i\}_{i=1}^7$. Right plot: illustration of multinomial resampling, where five particles are resampled. In line 1 of Algorithm 5.3, five realizations $\{u^i\}_{i=1}^5$ of $\mathcal{U}(0,1]$ are drawn and in step 2 the corresponding ancestor indices are computed according to $a^i = F_{\mathcal{C}}^{-1}(u^i) = j$ for $u^i \in \left(\sum_{l=1}^{j-1} w^l, \sum_{l=1}^{j} w^l\right]$. The five realizations of the uniform random variables are shown by the thin dashed lines. Hence, in this particular example we have, $a^1 = 3, a^2 = 5, a^3 = 5, a^4 = 5, a^5 = 2,$, which using the offspring variables corresponds to $o^1 = 1, o^2 = 0, o^3 = 1, o^4 = 0, o^5 = 3$.

We can now implement a first realization from (5.33) by first generating a uniform random variable according to $U^1 \sim \mathcal{U}(0,1]$. We then make use of the inverse of the cumulative distribution function of the weights $F^{-1}(u^1)$ to figure out which particle this corresponds to. Alternatively, this corresponds to selecting the ancestor index $a^1 = j$ for which

$$u^1 \in \left[\sum_{l=1}^{j-1} w^l, \sum_{l=1}^{j} w^l\right). \tag{5.35}$$

This index is then returned and the procedure is repeated. The procedure just described is referred to as *multinomial* resampling and it is provided in Algorithm 5.3 and it is further illustrated and explained for an example in Figure 5.7 (right plot). More formally, the multinomial resampling algorithm samples ancestor indices a^i independently from the categorical distribution $\mathcal{C}\left(\{w^i\}_{i=1}^N\right)$.

Algorithm 5.3: Multinomial resampling (all operations are for $i = 1, \ldots, N$)

1 Sample $U^i \sim \mathcal{U}(0,1]$.
2 Set $a^i = F_{\mathcal{C}}^{-1}(U^i)$.

Algorithm 5.3 produces a set of ancestor indices $\{a^i\}_{i=1}^N$ which we can straightforwardly convert into a set of offspring $\{O^i\}_{i=1}^N$, where $O^i \triangleq \sum_{j=1}^N \mathbb{I}\left(a^j = i\right)$. The

multinomial resampling algorithm is unbiased in the sense that

$$\mathbb{E}\big[O^i \,|\, \{x^j, w^j\}_{j=1}^N\big] = w^i N, \qquad \text{for } i = 1, \ldots, N. \tag{5.36}$$

Furthermore, the corresponding variance is given by $\mathrm{Var}\Big[O^i \,|\, \{x^j, w^j\}_{j=1}^N\Big] = Nw^i(1 - w^i)$. There are ways of producing better resampling algorithms where the variance of the produced offspring is smaller than $Nw^i(1-w^i)$, while they still remain ubiased. Two of these algorithms are introduced in the subsequent section.

5.4.2 Reducing the variance

The key idea in reducing the variance incurred by the resampling algorithm is to enforce a more uniform selection of the particles to be resampled. Rather than generating $U^i \in \mathcal{U}(0,1]$, let us first partition the interval $(0,1]$ into N equidistant disjoint sets, $(0,1] = (0,1/N] \cup \cdots \cup ((N-1)/N, 1]$. We then draw one uniform random variable from each of these sets, i.e.

$$U^i \sim \mathcal{U}\left((i-1)/N, i/N\right], \qquad i = 1, \ldots, N. \tag{5.37}$$

These random variables are then used to select which particles to resample via the same inversion method employed by the multinomial resampling algorithm, i.e. $a^i = F^{-1}(U^i)$. The resulting algorithm is referred to as *stratified resampling* due to the stratification of the interval and it is summarized in Algorithm 5.4.

Algorithm 5.4: Stratified resampling (all operations are for $i = 1, \ldots, N$)

1 Sample $U^i \sim \mathcal{U}\left((i-1)/N, i/N\right]$.
2 Set $a^i = F^{-1}(U^i)$.

The stratification idea (5.37) can be taken one step further by only generating a single random variabel $U^1 \sim \mathcal{U}(0, 1/N]$ and then deterministically assigning the remaining variables according to

$$U^i = \frac{i-1}{N} + U = U^{i-1} + \frac{1}{N}, \qquad i = 2, \ldots, N. \tag{5.38}$$

Due to the systematic nature of this assignment, the resulting algorithm is referred to as *systematic resampling* and it is summarized in Algorithm 5.5. The resulting ancestor indices are again provided by the inversion method described above, i.e. $a^i = F^{-1}(U^i)$.

Algorithm 5.5: Systematic resampling

1 Sample $U^1 \sim \mathcal{U}(0, 1/N]$.
2 Set $U^i = (i-1)/N + U$ for $i = 2, \ldots, N$.
3 Set $a^i = F^{-1}(U^i)$, for $i = 1, \ldots, N$.

Figure 5.8 provides a graphical illustration of how the three resampling algorithms operate in generating an equally weighted particle system.

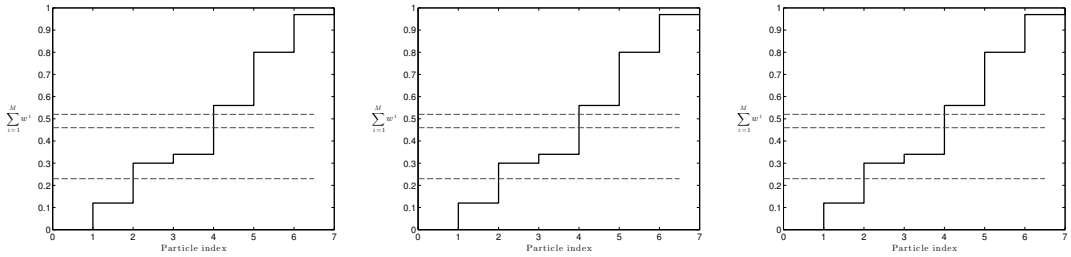

Figure 5.8: Graphical illustration of the three resampling algorithms (from left to right; multinomial, stratified and systematic) introduced. We use the same five weights as in Figure 5.7 and the values of the weights are visualized by the lengths of the bars at the bottom of the figures. The short vertical lines indicate the resampled particles. In multinomial resampling (left) we make use of a uniformly randomly sampled particle positions, whereas in stratified resampling (middle) we first generate a uniform deterministic grid (visualized using vertical dashed lines) of the entire interval and we then generate on random position within each grid cell. In systematic resampling (right) we make use of the same deterministically assigned grid as was used in the stratified resampler, but rather than using a random position for each grid cell we use the same offset in each grid cell.

5.5 Rao-Blackwellized particle filters

Let us assume that there is an analytically tractable sub-structure available in the model, implying that we can solve the filtering problem in closed form for the involved variables. The Rao-Blackwellized particle filter (RBPF) allows us to make use of this by analytically marginalizing over the part of the state vector that is conditionally tractable. As a result the particle filter only needs to be applied in a lower dimensional marginal space, improving the efficiency. The RBPF is one specific application of the general conditional Monte Carlo idea (Rao-Blackwellization) introduced in Section 4.5.1.

The strategy underlying the Rao-Blackwellized particle filter is introduced in Section 5.5.1. A common and useful example of a model with an analytically tractable sub-structure is provided by the CLG-SSM (recall Definition 2), where part of the state vector is conditionally linear and Gaussian. In order to make the development more concrete we will then derive an RBPF for a specific CLG-SSM in Section 5.5.2.

5.5.1 Strategy and key idea

The problem under consideration is to compute the filtering PDF when the state vector $X_t = (S_t, Z_t)$ is partitioned into one "linear" state z_t and one "nonlinear" state s_t, such that the conditional process $\{(Z_t, Y_t) \,|\, s_{1:t} = s_{1:t}\}_{t \geq 1}$ is analytically tractable and hence encoded in a finite dimensional parameterization. The RBPF exploits this tractable substructure of the model through the following key factorization of the target density (with the filtering PDF $p(z_t, s_t \,|\, y_{1:t})$ as one of its marginals),

$$p(z_t, s_{1:t} \,|\, y_{1:t}) = p(z_t \,|\, s_{1:t}, y_{1:t}) p(s_{1:t} \,|\, y_{1:t}). \tag{5.39}$$

The first density in the above factorization is (by assumption) available in closed form. The second factor $p(s_{1:t} \,|\, y_{1:t})$ is targeted by a PF and it is approximated by a weighted

particle system according to

$$\widehat{p}^N(s_{1:t} \mid y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{s_{1:t}^i}(s_{1:t}). \tag{5.40}$$

Hence, the RBPF is a PF in which each particle is equipped with a finite dimensional filter tracking the sufficient statistics for the conditional densities $p(z_t \mid s_{1:t}, y_{1:t})$.

Equivalently, we can view the marginalization of the z-process as a way to reduce the original model to

$$s_{t+1} \mid (s_t = s_t) \sim p(s_{t+1} \mid s_t), \tag{5.41a}$$

$$Y_t \mid (s_t = s_t) \sim p(y_t \mid s_{1:t}, y_{1:t-1}). \tag{5.41b}$$

The RBPF is then simply a PF targeting the sequence of conditional densities $p(s_{1:t} \mid y_{1:t})$ for the reduced SSM in (5.41) coupled with a recursion for the associated sufficient statistics. The model (5.41) shares many of the properties of the SSM (**??**). However, as an effect of the marginalization of the z-process, the measurement model (5.41b) depends on the complete history $s_{1:t}$, rendering the model non-Markovian. This is not a problem when it comes to deriving filtering solutions, but as we shall see in Chapter **??** it makes the corresponding smoothing problem more challenging.

An appealing property inherent in the RBPF is that it allows us to improve the quality of the filter, in a similar way as we can expect from Rao-Blackwellization. Any estimator derived from the RBPF will have a lower (or the same) variance when compared to the corresponding estimator derived from the standard PF. Informally, the reason is that in the RBPF, the particles are spread in a lower dimensional space, yielding a denser point-mass approximation of the target density. The standard PF amounts to a Monte Carlo solution in the $\mathsf{Z}^t \times \mathsf{S}^t$ space, whereas the RBPF only has to make use of a Monte Carlo solution in the smaller S^t space according to (5.40). The remaining analytically tractable states are computed explicitly.

5.5.2 Rao-Blackwellization in CLGSS models

The construction of the CLGSS model implies that the density $p(z_t \mid s_{1:t}, y_{1:t})$ is Gaussian and hence it is completely characterized by its mean and covariance according to

$$p(z_t \mid s_{1:t}, y_{1:t}) = \mathcal{N}\left(z_t \,\middle|\, \bar{z}_{t\mid t}^i,\, P_{t\mid t}^i\right), \tag{5.42}$$

where $\bar{z}_{t\mid t}^i = \bar{z}_{t\mid t}(s_{1:t}^i)$ and $P_{t\mid t}^i = P_{t\mid t}(s_{1:t}^i)$. Deriving the RBPF now amounts to finding a recursion for $\bar{z}_{t\mid t}^i$ and $P_{t\mid t}^i$ and then to properly combine this with the PF estimate (5.40) of the nonlinear state. The final RBPF approximation is obtained by inserting (5.40) and (5.42) into (5.39) resulting in the following Gaussian mixture approximation,

$$\widehat{p}^N(z_t, s_{1:t} \mid y_{1:t}) = \sum_{i=1}^{N} w_t^i \mathcal{N}\left(z_t \,\middle|\, \bar{z}_{t\mid t}(s_{1:t}^i),\, P_{t\mid t}(s_{1:t}^i)\right) \delta_{s_{1:t}^i}(s_{1:t}). \tag{5.43}$$

This means that in the RBPF we run one conditional Kalman filtering type of recursion per particle trajectory and keep track of the sufficient statistics for the corresponding conditionally Gaussian density (5.42). The RBPF generates an augmented weighted particle system defined below.

Definition 6 (Augmented weighted particle system)**.** *An augmented weighted particle system targeting a factorized density $p(z \mid s)p(s)$ is given by a collection of quadruples $\{s^i, w^i, \bar{z}^i, P^i\}_{i=1}^N$ such that*

i) $\{s^i, w^i\}_{i=1}^N$ is a weighted particle system targeting $p(s)$.

ii) The conditional density $p(z \mid s)$ is Gaussian with $p(z \mid s^i) = \mathcal{N}\left(z \mid \bar{z}^i, P^i\right)$ for $i = 1, \ldots, N$.

Let us now specialize one step further and consider a specific instance of the CLGSS model class, namely the MGSS model (recall Definition 4),

$$X_{t+1} = f_t(s_t) + A_t(s_t)z_t + z_t(s_t), \tag{5.44a}$$
$$Y_t = h_t(s_t) + C_t(s_t)z_t + E_t(s_t), \tag{5.44b}$$

where

$$X_t = \begin{pmatrix} s_t \\ z_t \end{pmatrix}, \qquad f_t(s_t) = \begin{pmatrix} f_t^s(s_t) \\ f_t^z(s_t) \end{pmatrix}, \qquad A_t(s_t) = \begin{pmatrix} A_t^s(s_t) \\ A_t^z(s_t) \end{pmatrix}. \tag{5.44c}$$

Deriving the RBPF for (5.44) can be divided into two steps. Firstly, we have to sequentially compute the sufficient statistics ($\bar{z}_{t\,|\,t}^i$ and $P_{t\,|\,t}^i$) for the conditionally linear Gaussian state (5.42). Secondly, in Section 5.5.2 we target $p(s_{1:t} \mid y_{1:t})$ using a PF, drawing the nonlinear state trajectories, which generates a sequence of weighted particle systems (5.40).

Updating the linear states

Let us start the derivation of the RBPF by showing how to obtain the conditional filtering density $p(z_t \mid s_{1:t}, y_{1:t})$ sequentially. This density is Gaussian, which implies that it is described entirely by its first and second moments. The updating formulas will show great resemblance with the Kalman filter, which is not surprising since the conditional process $\{z_t, Y_{1:t} \mid s_{1:t}\}_{t\geq 1}$ obeys an LGSS model. The only difference from the standard Kalman filter is that, in the eyes of the linear z_t state, the dynamics of the nonlinear state will behave like an "extra measurement" that we have to acknowledge in the derivation.

The derivation will be given as a proof by induction. By the end of this section we shall see that $p(z_1 \mid s_1, y_1)$ is Gaussian and can thus be written according to $p(z_1 \mid s_1, y_1) = \mathcal{N}\left(z_1 \mid \bar{z}_{1\,|\,1}(s_1),\, P_{1\,|\,1}(s_1)\right)$, where we have defined $\bar{z}_{1\,|\,1}(s_1)$ and $P_{1\,|\,1}(s_1)$ as the mean and covariance of the distribution, respectively. Hence, assume that, for $t \geq 2$,

$$p(z_{t-1} \mid s_{1:t-1}, y_{1:t-1}) = \mathcal{N}\left(z_{t-1} \mid \bar{z}_{t-1\,|\,t-1}(s_{1:t-1}),\, P_{t-1\,|\,t-1}(s_{1:t-1})\right), \tag{5.45}$$

where the mean and covariance are functions of the nonlinear state trajectory $s_{1:t-1}$ (naturally, they do also depend on the measurements $y_{1:t-1}$, but we shall not make that dependence explicit). We shall now see that this implies

$$p(z_t \mid s_{1:t}, y_{1:t}) = \mathcal{N}\left(z_t \mid \bar{z}_{t\mid t}(s_{1:t}),\, P_{t\mid t}(s_{1:t})\right), \tag{5.46}$$

and show how we can obtain the sufficient statistics for this density. Using the Markov property and the state transition density given by the model (5.44a), we have,

$$\begin{aligned} p(z_t, s_t \mid z_{t-1}, s_{1:t-1}, y_{1:t-1}) &= p(z_t, s_t \mid z_{t-1}, s_{t-1}) \\ &= \mathcal{N}\left(x_t \mid f(s_{t-1}) + A(s_{t-1})z_{t-1},\, Q(s_{t-1})\right), \end{aligned} \tag{5.47}$$

which is affine in z_{t-1}. We know that affine transformations preserves Gaussianity. More specifically, (5.45) and (5.47) together with Theorem 9 allow us to conclude

$$\begin{aligned} &p(z_t, s_t \mid s_{1:t-1}, y_{1:t-1}) \\ &\quad = \mathcal{N}\Bigg(\begin{pmatrix} s_t \\ z_t \end{pmatrix} \mid \underbrace{\begin{pmatrix} \alpha_{t\mid t-1}(s_{1:t-1}) \\ \beta_{t\mid t-1}(s_{1:t-1}) \end{pmatrix}}_{\chi_{t\mid t-1}(s_{1:t-1})}, \underbrace{\begin{pmatrix} \Sigma^s_{t\mid t-1}(s_{1:t-1}) & \Sigma^{sz}_{t\mid t-1}(s_{1:t-1}) \\ \left(\Sigma^{sz}_{t\mid t-1}(s_{1:t-1})\right)^{\mathsf{T}} & \Sigma^z_{t\mid t-1}(s_{1:t-1}) \end{pmatrix}}_{\Sigma_{t\mid t-1}(s_{1:t-1})} \Bigg), \end{aligned} \tag{5.48a}$$

where

$$\chi_{t\mid t-1}(s_{1:t-1}) = f + A\bar{z}_{t-1\mid t-1}, \tag{5.48b}$$

$$\Sigma_{t\mid t-1}(s_{1:t-1}) = Q + AP_{t-1\mid t-1}A^{\mathsf{T}}. \tag{5.48c}$$

To keep the notation simple, the dependencies on s_{t-1} and $s_{1:t-1}$ have been dropped from the right hand side. Note that (5.48a) is simply a prediction of the state at time t conditioned on $s_{1:t-1}$ and $y_{1:t-1}$. In (5.48b), the dynamics (5.44a) is simulated and (5.48c) shows how the uncertainty depends on the contributions from the process noise and the prior uncertainty in the linear state.

Marginalizing (5.48a) w.r.t. the linear state z_t results in (Theorem 7)

$$p(s_t \mid s_{1:t-1}, y_{1:t-1}) = \mathcal{N}\left(s_t \,\middle|\, \alpha_{t\mid t-1}(s_{1:t-1}),\, \Sigma^s_{t\mid t-1}(s_{1:t-1})\right), \tag{5.49}$$

and conditioning (5.48a) on the nonlinear state s_t (Theorem 8) we get,

$$p(z_t \mid s_{1:t}, y_{1:t-1}) = \mathcal{N}\left(z_t \mid \bar{z}_{t\mid t-1}(s_{1:t}),\, P_{t\mid t-1}(s_{1:t})\right), \tag{5.50a}$$

where

$$\bar{z}_{t\mid t-1}(s_{1:t}) = \beta_{t\mid t-1} + L_t\left(s_t - \alpha_{t\mid t-1}\right), \tag{5.50b}$$

$$P_{t\mid t-1}(s_{1:t}) = \Sigma^z_{t\mid t-1} - L_t\Sigma^{sz}_{t\mid t-1}, \tag{5.50c}$$

$$L_t(s_{1:t}) = \left(\Sigma^{sz}_{t\mid t-1}\right)^{\mathsf{T}}\left(\Sigma^s_{t\mid t-1}\right)^{-1}. \tag{5.50d}$$

The prediction of the nonlinear state (5.49) is used for sampling new particles in the bootstrap version of the particle filter. Once the nonlinear state trajectory is augmented with a new sample we can condition the prediction of the linear state on this sample according to (5.50). This provides some information about the linear state via the interconnections between the linear and the nonlinear parts of the state vector. From (5.50b) we see that the state is updated accordingly and the covariance is reduced (5.50c). Note that this update is very similar to a Kalman filter measurement update (3.11), which explains why it is sometimes referred to as the "extra measurement" update of the RBPF. We have still not made use of the information that is present in the current measurement y_t. This is what we will do next.

From the measurement model (5.44b) and the conditional independence properties inherent in the model we have

$$p(y_t \mid s_{1:t}, z_t, y_{1:t-1}) = p(y_t \mid s_t, z_t) = \mathcal{N}\left(y_t \mid h(s_t) + C(s_t)z_t,\ R(s_t)\right), \tag{5.51}$$

which is affine in the linear state z_t. Again appealing to Theorem 9 and making use of the result (5.50) we obtain the measurement prediction density according to

$$p(y_t \mid s_{1:t}, y_{1:t-1}) = \mathcal{N}\left(y_t \mid \widehat{y}_t(s_{1:t}),\ S_t(s_{1:t})\right), \tag{5.52a}$$

where

$$\widehat{y}_t(s_{1:t}) = h + C\bar{z}_{t\mid t-1}, \tag{5.52b}$$
$$S_t(s_{1:t}) = R + CP_{t\mid t-1}C^{\mathsf{T}}. \tag{5.52c}$$

The posterior of z_t conditioned on y_t is given by

$$p(z_t \mid s_{1:t}, y_{1:t}) = \mathcal{N}\left(z_t \,\middle|\, \bar{z}_{t\mid t}(s_{1:t}),\ P_{t\mid t}(s_{1:t})\right), \tag{5.53a}$$

where

$$\bar{z}_{t\mid t}(s_{1:t}) = \bar{z}_{t\mid t-1} + K_t\left(y_t - \widehat{y}_t\right), \tag{5.53b}$$
$$P_{t\mid t}(s_{1:t}) = P_{t\mid t-1} - K_tCP_{t\mid t-1}, \tag{5.53c}$$
$$K_t(s_{1:t}) = P_{t\mid t-1}C^{\mathsf{T}}S_t^{-1}. \tag{5.53d}$$

This concludes the induction proof for $t > 1$ and it remains to take care of the start where $t = 1$. In order to do so, let us first define $y_{1:0} \triangleq \emptyset$, so that $p(z_1 \mid s_{1:1}, y_{1\mid 0}) = p(z_1 \mid s_1)$ and analogously for the other densities. From this we see that the expression (5.50a) coincides with the prior for $z_1 \mid (s_1 = s_1)$ at time $t = 1$. The computations in (5.50) to (5.53) will thus hold at $t = 1$, which in turn confirms the validity of the induction assumption, $p(z_1 \mid s_1, y_1) = \mathcal{N}\left(z_1 \,\middle|\, \bar{z}_{1\mid 1}(s_1),\ P_{1\mid 1}(s_1)\right)$.

Sampling the nonlinear state trajectories

In the previous section we obtained closed form expressions for the conditional filtering density for the linear state $p(z_t \mid s_{1:t}, y_{1:t})$. However, according to (5.39) it remains to

find the smoothing density for the nonlinear state $p(s_{1:t} \mid y_{1:t})$. The model involves a nonlinear dependence on s_t, implying that there is no closed form expression available for this density. Instead we target it using a PF.

Let us assume that $t \geq 2$. Sampling at time $t = 1$ is covered at the end of this section, since it follows from straightforward modifications of the results for $t \geq 2$. First, using Bayes' theorem we note the following concerning the target density

$$\begin{aligned} p(s_{1:t} \mid y_{1:t}) &\propto p(y_t \mid s_{1:t}, y_{1:t-1}) p(s_{1:t} \mid y_{1:t-1}) \\ &= p(y_t \mid s_{1:t}, y_{1:t-1}) p(s_t \mid s_{1:t-1}, y_{1:t-1}) p(s_{1:t-1} \mid y_{1:t-1}). \end{aligned} \tag{5.54}$$

Second, similarly to (5.28) we choose a proposal density which factorizes according to,

$$q_t(s_{1:t} \mid y_{1:t}) = q_t(s_t \mid s_{1:t-1}, y_{1:t}) \underbrace{q_{t-1}(s_{1:t-1} \mid y_{1:t-1})}_{\text{previous proposal}}. \tag{5.55}$$

Given a weighted particle system $\{s_{1:t-1}^{A_t^i}, w_{t-1}^i\}_{i=1}^N$ at time $t-1$ targeting $p(s_{1:t-1} \mid y_{1:t-1})$, sample trajectories are constructed as in step 6 of Algorithm **??**,

$$s_t^i \sim q_t\left(s_t \mid s_{1:t-1}^{a_t^i}, y_{1:t}\right), \tag{5.56a}$$

$$s_{1:t}^i = \left\{s_{1:t-1}^{a_t^i}, s_t^i\right\}, \tag{5.56b}$$

for $i = 1, \ldots, N$. The importance weights are computed according to

$$\bar{w}_t^i = \frac{p(s_{1:t}^i \mid y_{1:t})}{q_t(s_{1:t}^i \mid y_{1:t})} \propto \underbrace{\frac{p(y_t \mid s_{1:t}^i, y_{1:t-1}) p(s_t^i \mid s_{1:t-1}^i, y_{1:t-1})}{q_t(s_t^i \mid s_{1:t-1}^i, y_{1:t})}}_{\omega(s_t^i)} \underbrace{\frac{p(s_{1:t-1}^i \mid y_{1:t-1})}{q_{t-1}(s_{1:t-1}^i \mid y_{1:t-1})}}_{=w_{t-1}^i}. \tag{5.57}$$

Inserting (5.52a) and (5.49) into (5.57) results in the following weight function

$$\omega(s_t^i) = \frac{\mathcal{N}\left(y_t \mid \widehat{y}_t(s_{1:t}^i),\, S_t(s_{1:t}^i)\right) \mathcal{N}\left(s_t \,\middle|\, \alpha_{t \mid t-1}(s_{1:t-1}^i),\, \Sigma_{t \mid t-1}^s(s_{1:t-1}^i)\right)}{q(s_t^i \mid s_{1:t-1}^i, y_{1:t})}. \tag{5.58}$$

Once the unnormalized weights are computed according to the weight function (5.58) they are normalized to sum to one. Finally, the particles are resampled similarly to the standard PF.

An RBPF algorithm

The (bootstrap) RBPF for the MGSS model (5.44) derived above is summarized in Algorithm 5.6. To simplify the notation; for functions in s_t or $s_{1:t}$, e.g. $R(s_t)$ and $\bar{z}_{t \mid t}(s_{1:t})$ we write $R_t^i = R(s_t^i)$ and $\bar{z}_{t \mid t}^i = \bar{z}_{t \mid t}(s_{1:t}^i)$, etc.

The RBPF provided in Algorithm 5.6 is just one example of a working algorithm. We are free to make use of all the standard embellishments (such as for example adaptive resampling and auxiliary variables) that are available for the PF to obtain a more efficient algorithm. The RBPF algorithm is illustrated in Example 5.9.

Algorithm 5.6: Rao-Blackwellized particle filter (RBPF) for MGSS models

Initialization ($t = 1$)**:**

Sample $s_1^i \sim q_1(s_1 \,|\, y_1)$ and set $\left\{\bar{z}_1^i, P_1^i\right\} = \{\bar{z}_1, P_1\}$.

Compute $\bar{w}_1^i = \omega\left(s_1^i\right)$ and normalize $w_1^i = \bar{w}_1^i / \sum_{j=1}^N \bar{w}_1^j$.

for $t = 2$ **to** T **do**

Resampling: Generate the equally weighted augmented particle system $\{\bar{s}_{1:t-1}^i, 1/N, z_{t-1\,|\,t-1}^i, P_{t-1\,|\,t-1}^i\}$ by resampling.

Propagation: Sample $s_t^i \sim q\left(s_t \,|\, s_{1:t-1}^i, y_{1:t}\right)$ and store the samples $s_{1:t}^i = \left\{s_{1:t-1}^i, s_t^i\right\}$.

Predict the state and condition the linear state on the newly drawn particles s_t^i.

$$\begin{aligned}
\alpha_{t\,|\,t-1}^i &= f_{t-1}^{s,i} + A_{t-1}^{s,i} \bar{z}_{t-1\,|\,t-1}^i, \\
\bar{z}_{t\,|\,t-1}^i &= f_{t-1}^{z,i} + A_{t-1}^{z,i} \bar{z}_{t-1\,|\,t-1}^i + \left(\Sigma_{t\,|\,t-1}^{sz,i}\right)^{\mathsf{T}} \left(\Sigma_{t\,|\,t-1}^{s,i}\right)^{-1} \left(s_t^i - \alpha_{t\,|\,t-1}^i\right), \\
P_{t\,|\,t-1}^i &= \Sigma_{t\,|\,t-1}^{z,i} - \left(\Sigma_{t\,|\,t-1}^{sz,i}\right)^{\mathsf{T}} \left(\Sigma_{t\,|\,t-1}^{s,i}\right)^{-1} \Sigma_{t\,|\,t-1}^{sz,i},
\end{aligned}$$

where $\Sigma_{t\,|\,t-1}^i = A_{t-1}^i P_{t-1\,|\,t-1}^i \left(A_{t-1}^i\right)^{\mathsf{T}} + Q_{t-1}^i$.

Weighting: Compute

$$\bar{w}_t^i = \frac{\mathcal{N}\left(y_t \,|\, \widehat{y}_t(s_{1:t}^i),\, S_t(s_{1:t}^i)\right) \mathcal{N}\left(s_t \,\middle|\, \alpha_{t\,|\,t-1}(s_{1:t-1}^i),\, \Sigma_{t\,|\,t-1}^s(s_{1:t-1}^i)\right)}{q(s_t^i \,|\, s_{1:t-1}^i, y_{1:t})},$$

where $\widehat{y}_t^i = h_t^i + C_t^i \bar{z}_{t\,|\,t-1}^i$, $S_t^i = R_t^i + C_t^i P_{t\,|\,t-1}^i (C_t^i)^{\mathsf{T}}$ and normalize $w_t^i = \bar{w}_t^i / \sum_{j=1}^N \bar{w}_t^j$.

Compute the sufficient statistics for the linear states, given the current measurement.

$$\begin{aligned}
\bar{z}_{t\,|\,t}^i &= \bar{z}_{t\,|\,t-1}^i + K_t^i \left(y_t - \widehat{y}_t^i\right), \\
P_{t\,|\,t}^i &= P_{t\,|\,t-1}^i - K_t^i C_t^i P_{t\,|\,t-1}^i, \\
K_t^i &= P_{t\,|\,t-1}^i (C_t^i)^{\mathsf{T}} (S_t^i)^{-1}.
\end{aligned}$$

end

Example 5.9: Better estimates using the RBPF

Consider the following MGSS model

$$s_{t+1} = \arctan s_t + \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} z_t + v_t^s, \tag{5.59a}$$

$$z_{t+1} = \begin{pmatrix} 1 & 0.3 & 0 \\ 0 & 0.92 & -0.3 \\ 0 & 0.3 & 0.92 \end{pmatrix} z_t + v_t^z, \tag{5.59b}$$

$$y_t = \begin{pmatrix} 0.1 \operatorname{sign}(s_t)\, s_t^2 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & -1 & 1 \end{pmatrix} s_t + e_t, \tag{5.59c}$$

where $v_t = \begin{pmatrix} v_t^s & v_t^z \end{pmatrix}^\mathsf{T} \sim \mathcal{N}(0,\, 0.01 I_4)$, $e_t \sim \mathcal{N}(0,\, 0.1 I_2)$ and $\operatorname{sign}(s_t)$ denotes the sign function extracting the sign of s_t. The initial states are given by $s_1 \sim \mathcal{N}(0,\, 1)$ and $z_1 \sim \mathcal{N}(0_{3\times 1},\, 0_{3\times 3})$.

5.6 Computing estimates

5.6.1 Likelihood estimates

The likelihood $\mathcal{L}(\theta)$ is an important quantity when it comes to learning parameters, both for maximum likelihood and Bayesian approaches. Using the weighted particle system generated by an SMC sampler we can compute consistent and unbiased estimate of the likelihood. Let us show how to do this using the APF introduced in Algorithm 5.2, similar derivations for other PFs can be carried out analogously.

Recall that the likelihood can be computed via the one-step predicted likelihoods (cf. (**??**)), which using marginalization over $x_{t-1:t}$ can be computed according to

$$\begin{aligned} p_\theta(y_t \mid y_{1:t-1}) &= \int p_\theta(y_t, x_{t-1:t} \mid y_{1:t-1}) \mathrm{d}x_{t-1:t} \\ &= \int p_\theta(y_t \mid x_t) p_\theta(x_t \mid x_{t-1}) p_\theta(x_{t-1} \mid y_{1:t-1}) \mathrm{d}x_{t-1:t} \\ &= \int \omega(x_{t-1}, x_t, y_t) \nu(x_{t-1}, y_t) q(x_t \mid x_{t-1}, y_t) p_\theta(x_{t-1} \mid y_{1:t-1}) \mathrm{d}x_{t-1:t} \end{aligned} \tag{5.60}$$

where the final equality follows from the definition of the weight function provided in (5.22). The particles $\{\bar{x}_{t-1}^i, x_t^i\}_{i=1}^N$ are by the construction Algorithm 5.2 approximately distributed according to $q(x_t \mid x_{t-1}, y_t) p_\theta(x_{t-1} \mid y_{1:t-1})$, implying that they can be used in approximating the integral in (5.60) according to

$$\begin{aligned} \widehat{p}_\theta(y_t \mid y_{1:t-1}) &= \int \omega(x_{t-1}, x_t, y_t) \nu(x_{t-1}, y_t) \frac{1}{N} \sum_{i=1}^N \delta_{\bar{x}_{t-1}^i, x_t^i}(x_{t-1:t}) \mathrm{d}x_{t-1:t} \\ &= \frac{1}{N} \sum_{i=1}^N \omega_\theta(\bar{x}_{t-1}^i, x_t^i, y_t) \nu_\theta(\bar{x}_{t-1}^i, y_t) = \frac{1}{N} \sum_{i=1}^N \bar{w}_t^i \nu_{t-1}^i. \end{aligned} \tag{5.61}$$

Using these estimates of the one-step predicted likelihoods we obtain the following estimate of the likelihood

$$\widehat{\mathcal{L}}(\theta) = \widehat{p}(y_1) \prod_{t=2}^{T} \widehat{p}_\theta(y_t \,|\, y_{1:t-1}) = \left(\frac{1}{N} \sum_{i=1}^{N} \bar{w}_1^i \right) \left(\prod_{t=2}^{T} \frac{1}{N} \sum_{i=1}^{N} \bar{w}_t^i \nu_{t-1}^i \right). \tag{5.62}$$

5.7 Generic sequential Monte Carlo

In this chapter we have so far been focusing on SSMs and derived the PF as a way of targeting the filtering density $p(x_t \,|\, y_{1:t})$ and the joint smoothing density $p(x_{0:t} \,|\, y_{1:t})$. One of the strengths of SMC, however, is that it is applicable to a much wider range of models than what is offered by the SSMs. We will in this section make this more explicit by deriving a general SMC formulation that is not specialized to the SSM.

Example 5.10: General latent variable model

Let $\pi_t(x_{0:t})$ for $t = 0, \ldots, T$ be a sequence of target densities on some increasing space X^t. We assume that these densities can be written as

$$\pi_t(x_{0:t}) = \frac{\tilde{\pi}_t(x_{0:t})}{Z_t}, \tag{5.63}$$

where the unnormalized density $\tilde{\pi}_t(x_{0:t})$ can be evaluated point-wise, whereas the normalization constant $Z_t = \int \tilde{\pi}_t(x_{0:t}) \mathrm{d}x_{1:t}$ is possibly unknown. SMC can be used to target any such sequence, in similarly to what was done for the sequence of JSDs earlier in this chapter. For that (important) special case, we had $\pi_t(x_{0:t}) = p(x_{0:t} \,|\, y_{1:t})$, $\tilde{\pi}_t(x_{0:t}) = p(x_{0:t}, y_{1:t})$ and $Z_t = p(y_{1:t})$. We will continue to refer to the index t as time, even though it might not at all have any temporal meaning.

To make inference about the latent variables $x_{0:T}$, as well as to enable learning model parameter, a useful approach is to construct a Monte Carlo algorithm to draw samples from $\pi_T(x_{0:T})$. The SMC method exploits the sequential nature of the problem.

Let $\{x_{0:t-1}^i, w_{t-1}^i\}_{i=1}^N$ be a weighted particle system, targeting $\pi_{t-1}(x_{0:t-1})$, meaning that the weighted particles define an empirical point-mass approximation of the target distribution at $t-1$ according to

$$\widehat{\pi}_{t-1}(x_{0:t-1}) = \sum_{i=1}^{N} \frac{\bar{w}_{t-1}^i}{\sum_l \bar{w}_{t-1}^l} \delta_{x_{0:t-1}^i}(x_{0:t-1}) \tag{5.64}$$

This particle system is propagated to time t by resampling and sequential importance sampling, but we will now take a slightly different view and perform these two sampling operations at once. The propagation to time t is thus performed by sampling $\{a_t^i, x_t^i\}_{i=1}^N$

independently, conditionally on the particles generated up to time $t-1$, from a proposal kernel on the product space $\{1, \ldots, N\} \times \mathsf{X}$,

$$M_t(a_t, x_t) = \frac{\bar{w}_{t-1}^{a_t}}{\sum_l \bar{w}_{t-1}^l} q_t(x_t \mid x_{0:t-1}^{a_t}). \tag{5.65}$$

It should be noted that the kernel M_t depends on all the random variables generated by the SMC sampler up to time $t-1$ (i.e. all the particles and all the ancestor indices), but to avoid a very cumbersome notation we have not made this dependence explicit.

Once we have generated N ancestor indices and particles from the proposal kernel (5.65), the particles are weighted according to $\bar{w}_t^i = \omega_t(x_{0:t}^i)$, where the weight function is given by

$$\omega_t(x_{0:t}) = \frac{\tilde{\pi}_t(x_{0:t})}{\tilde{\pi}_{t-1}(x_{0:t-1}) q_t(x_t \mid x_{0:t-1})}, \tag{5.66}$$

for $t \geq 2$.

In this formulation, the resampling step is implicit and corresponds to sampling the ancestor indices $\{A_t^i\}_{i=1}^N$ in (**??**).

The procedure is initialized by targeting $\pi_1(x_1)$ using importance sampling. We thus sample from some proposal density $x_1^i \sim q_1(x_1)$ and compute importance weights $\bar{w}_t^i = \omega_1(x_1)$, where the weight function is given by $\omega_1(x_1) = \tilde{\pi}_1(x_1)/q_1(x_1)$. The resulting SMC sampler is summarized in Algorithm **??**.

Algorithm 5.7: Generic SMC (all operations are for $i = 1, \ldots, N$)

Initialization ($t = 0$)**:**
  Sample $x_0^i \sim q_0(x_0)$.
  Compute $\bar{w}_0^i = \omega_0(x_0^i)$.
for $t = 1$ **to** T **do**
  Sample $\{a_t^i, x_t^i\} \sim M_t(a_t, x_t)$.
  Set $x_{0:t}^i = \{x_{0:t-1}^{a_t^i}, x_t^i\}$.
  Compute $\bar{w}_t^i = \omega_t(x_{0:t}^i)$.
end

Just like any algorithm that is used to generate a realization of random variables, there is an underlying density encodes the probabilistic properties of the involved random variables also for the SMC sampler in Algorithm 5.7. This is interesting in its own right, since it can help in providing a deeper understanding of SMC. Furthermore, it is also instrumental when it comes to the analysis of SMC and it is highly essential when deriving the particle MCMC samplers in Chapter **??**, where SMC samplers are used as proposal distributions.

Let

$$X_t \triangleq \{x_t^1, \ldots, x_t^N\}, \qquad A_t \triangleq \{a_t^1, \ldots, a_t^N\}, \tag{5.67}$$

refer to all the particles and ancestor indices, respectively, generated by the SMC sampler at time t. It follows that the SMC sampler in Algorithm 5.7 generates a *single*

realization of a collection of random variables $\{X_{1:T}, A_{2:T}\} \in \mathsf{X}^{NT} \times \{1, \ldots, N\}^{N(T-1)}$. Furthermore, $\{a_t^i, x_t^i\}_{i=1}^N$ are drawn independently (conditionally on the weighted particle system generated up to time $t-1$) from the proposal kernel M_t, and similarly at time $t = 1$. The joint probability density function of these variables is by construction given by,

$$\psi(X_{1:T}, A_{2:T}) \triangleq \prod_{i=1}^{N} q_1(x_1^i) \prod_{t=2}^{T} \prod_{i=1}^{N} M_t(a_t^i, x_t^i). \tag{5.68a}$$

That is, at time $t = 1$, we sample $\{x_1^i\}_{i=1}^N$ independently from the proposal density $q_1(x_1)$. Then, for each time point $t = 2, \ldots, T$, we sample $\{a_t^i, x_t^i\}_{i=1}^N$ independently from the proposal kernel $M_t(a_t, x_t)$.

The fact that SMC is a combination of two sampling operations—resampling and propagation—is implicit in (5.68). However, we can of course make this more explicit by breaking the joint proposal apart, resulting in,

$$\psi(X_{1:T}, A_{2:T}) = \underbrace{\left\{ \prod_{i=1}^{N} q_1(x_1^i \mid y_1) \right\}}_{\text{Initialization}} \left\{ \prod_{t=2}^{T} \prod_{i=1}^{N} \underbrace{r(a_t^i \mid w_{t-1})}_{\text{Resampling}} \underbrace{q_t\left(x_t^i \mid x_{1:t-1}^{a_t^i}, y_{1:t}\right)}_{\text{Propagation}} \right\} \tag{5.68b}$$

The SMC sampler (and implicitly also all PF) can thus be interpreted as a mechanism for generating a realization from the $2NT - N$ random variables $\mathbf{X}_{1:T}, \mathbf{A}_{2:T}$, described by the PDF (5.68).

5.8 Convergence results

5.9 History and further reading

History: The first paper introducing what is now known as the bootstrap particle filter was published by Steward and McCarty (1992). This paper has for some reason been more or less unnoticed. At the same time there were several independent derivations of the PF. The derivation of the bootstrap PF as a solution to the nonlinear filtering problem by Gordon et al. (1993) has gained a lot of influence. Kitagawa (1993, 1996) also introduced the PF at the same time. Other early contributions include Hürzeler and Künsch (1998), Liu and Chen (1998), where the latter shows that SMC can be used to any sequence of distributions of increasing dimension. The APF was originally derived by Pitt and Shephard (1999) using auxiliary variables, which explains the name. Pitt and Shephard (1999) made use of two resampling steps in deriving the APF, whereas we only made use of one resampling step according to the derivation of Carpenter et al. (1999).

Further reading: Textbooks: Douc et al. (2014), Särkkä (2013), Cappé et al. (2005), Ristic et al. (2004), Del Moral (2013), Liu (2001) Edited book: Doucet et al. (2001a)

Tutorials: Doucet and Johansen (2011), Cappé et al. (2007), Gustafsson (2010), Arulampalam et al. (2002), Künsch (2013) The first unifying paper Doucet et al. (2000b) still provides a very nice introduction to the PF.

We have in this chapter derived the PF leveraging sequential importance sampling and resampling. It is also possible to derive PFs based on the rejection sampler (recall Algorithm 4.1), which was first done by Hürzeler and Künsch (1998), Tanizaki and Mariano (1998). Later on Künsch (2005) studied the theoretical properties of this idea and provided some extensions.

There are many examples of highly relevant latent variable models which are not given by SSMs for which SMC has been successfully applied, see Del Moral et al. (2006) for a general introduction. There are also many concrete examples of this, e.g. Dirichlet process mixture models MacEachern et al. (1999), Fearnhead (2004), phylogenetic trees Bouchard-Côté et al. (2012), agglomerative clustering models Teh et al. (2008) and probabilistic graphical models Naesseth et al. (2014), **?**, just to mention a few.

Multinomial resampling was used by Gordon et al. (1993) in the bootstrap PF. Kitagawa (1996) introduced stratified resampling for the first time and residual resampling was used by Liu and Chen (1998). The systematic resampling algorithm was introduced into the particle filter literature by Carpenter et al. (1999). The residual and systematic resampling algorithms had previously been introduced by Whitley (1994) under different names and in a different context. The properties of the resampling algorithms introduced in this chapter has been thoroughly analysed, see e.g. Douc et al. (2005), Hol et al. (2006), Murray et al. (2013) As pointed out by Fearnhead and Clifford (2003) and Chen et al. (2005) resampling algorithm can be taylored for specific model classes. A specific example of this was provided by the DPF (introduced by Fearnhead (1998)) in Section 10, which should be used for any model involving discrete variables, for example the SLG-SSM in Definition 3 (Fearnhead and Clifford, 2003, Fearnhead, 2004, Whiteley et al., 2010).

Convergence: Del Moral (2004), Hu et al. (2008), Crisan and Doucet (2002)

RBPF: Doucet et al. (2000a,b), Schön et al. (2005), Chen and Liu (2000), Doucet et al. (2001b). Study of the variance improvements (Lindsten et al., 2011, Chopin, 2004).

MPF: Klaas et al. (2005)

APF: Developments by Johansen and Doucet (2008). (Pitt et al., 2011)

Appendices

Appendix A

Probability/statistics

Independent stochastic variables

$$p(x_a, x_b) = p(x_a)p(x_b). \tag{A.1}$$

Conditional PDF is defined as

$$p(x_a \,|\, x_b) = \frac{p(x_a, x_b)}{p(x_b)}. \tag{A.2}$$

Marginalization

$$p(x_a) = \int p(x_a, x_b) \mathrm{d}x_b \tag{A.3}$$

Bayes' theorem

$$p(x_a \,|\, x_b) = \frac{p(x_b \,|\, x_a)p(x_a)}{p(x_b)} \tag{A.4}$$

$$p(x_a \,|\, x_b, x_c) = \frac{p(x_b \,|\, x_a, x_c)p(x_a \,|\, x_c)}{p(x_b \,|\, x_c)} \tag{A.5}$$

Theorem 3 (The strong law of large numbers)**.** *Assume that* $\mathbf{z}_1, \mathbf{z}_2, \ldots$ *are i.i.d. random variables with finite expectation* μ *and let* $S_M = \sum_{i=1}^{M} \mathbf{z}_i, M \geq 1$*. Then,*

$$\lim_{M\to\infty} \frac{S_M}{M} \xrightarrow{\text{a.s.}} \mu, \tag{A.6}$$

Theorem 4 (The central limit theorem)**.** *Assume that* $\mathbf{z}_1, \mathbf{z}_2, \ldots$ *are i.i.d. random variables with finite expectation* μ *and finite variance* σ^2*. Let* $S_M = \sum_{i=1}^{M} \mathbf{z}_i, M \geq 1$*. Then,*

$$\lim_{M\to\infty} \frac{S_M - M\mu}{\sigma\sqrt{M}} \xrightarrow{\text{d}} \mathcal{N}(0, 1) \tag{A.7}$$

Appendix B

Probability Distributions

The aim in this appendix is to introduce the distributions that are most commonly used when it comes to probabilistic modelling of dynamical systems. Besides the mere definition of the distribution we will provide some useful results related to each. Material of this sort is available in many books and on the Internet. The reason for still including it here is as a service to the reader, since we are using the same parameterizations throughout the manuscript. The notation and parameterizations used are notoriously different depending on which source is used.

For continuous and discrete random variables we talk about the probability density function (PDF) and the probability mass function (PMF), respectively.

B.1 Discrete distributions

B.1.1 Dirac distribution and empirical distribution

B.1.2 Categorical distribution

The categorical distribution describes a random event that can take one of K possible outcomes and it is thus a generalization of the Bernoulli distribution. The support of a categorically distributed variable x is $\{1, \ldots, K\}$ and it is parameterized by the event probabilities $\alpha_1, \ldots, \alpha_K$, where $\sum_{k=1}^{K} \alpha_k = 1$ and $\alpha_k \geq 0, \forall k$. We use the following notation to explain that a random variable x is categorically distributed

$$x \sim \mathcal{C}\left(\alpha_{1:K}\right). \tag{B.1}$$

The PMF describing x is

$$p(x = k \,|\, \alpha_{1:K}) = \alpha_k, \tag{B.2}$$

stating that the probability of x corresponding to outcome k is given by α_k. The notation (B.2) is good for understanding, but it is not suitable for mathematical manipulations, which motivates the slightly more complicated, but much more practical

formulation,

$$p(x \mid \alpha_{1:K}) = \prod_{k=1}^{K} \alpha_k^{I_k(x)}, \tag{B.3}$$

where $I_A(x)$ denotes the *indicator function*, defined as

$$I_A(x) \triangleq \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases} \tag{B.4}$$

This can equivalently be denoted using the so called *Iverson bracket*, $[x \in A]$, i.e. $[x \in A] = I_A(x)$.

The categorical distribution describes the result of a random event where the result can be one of N different outcomes. Put in slightly different words, the result of a random event where one of N different categories is somehow selected, which explains the name categorical distribution. The probability of each outcome is known and denoted by w_i, where $\sum_{i=1}^{N} w_i = 1$ and $w_i \geq 0, \forall i = 1, \ldots, N$. The parameters of the categorically distributed random variable x are the probabilities for the categories, $w_i = \mathbb{P}(x = i), i = 1, \ldots, N$, denoted

$$x \sim \mathcal{C}(N, w_{1:N}). \tag{B.5}$$

The probability mass function of a categorically distributed random variable x is given by

$$p(x = i \mid w) = w_i. \tag{B.6}$$

In the special case of two categories, the categorical distribution is also referred to as the Bernoulli distribution. Hence, the categorical distribution is a generalization of the Bernoulli distribution to more than two categories.

B.2 Univariate continuous distributions

B.2.1 Gaussian

$$x \sim \mathcal{N}(m, p), \qquad p > 0, \tag{B.7}$$

$$\mathcal{N}(x \mid m, p) = \frac{1}{\sqrt{2\pi p}} \exp\left(-\frac{1}{2p}(x - m)^2\right) \tag{B.8}$$

B.2.2 Gamma

$$x \sim \mathcal{G}(a, b), \qquad a > 0, b > 0. \tag{B.9}$$

PDF

$$\mathcal{G}(x \mid a, b) = \frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx) \tag{B.10}$$

B.2.3 Inverse Gamma

The inverse gamma distribution is defined on the positive real line and it is characterized by the so called shape parameter a and the scale parameter b. We write

$$x \sim \mathcal{IG}(a, b), \qquad a > 0, b > 0, \tag{B.11}$$

and the PDF is given by

$$\mathcal{IG}\left(x \mid a, b\right) = \frac{b^a}{\Gamma(a)} x^{-(a+1)} \exp\left(-\frac{b}{x}\right), \qquad x > 0, \tag{B.12}$$

where $\Gamma(a)$ is the gamma function, i.e. $\Gamma(a) = \int_0^\infty t^{a-1} e^{-t} \mathrm{d}t$.

B.2.4 Normal inverse Gamma

$$x, q \sim \mathcal{NIG}(m, c, a, b), \qquad q > 0, a > 0, b > 0, \tag{B.13}$$

The support is given by $x \in (-\infty, \infty), q \in (0, \infty)$ and the PDF is given by

$$\mathcal{NIG}\left(x, q \mid m, c, a, b\right) = \mathcal{N}\left(x \mid m,\, cq\right) \mathcal{IG}\left(q \mid a, b\right) \tag{B.14a}$$

$$= \frac{1}{\sqrt{2\pi c}} \frac{b^a}{\Gamma(a)} q^{-(a+1/2+1)} \exp\left(-\frac{1}{2q}\left(\frac{1}{c}(x-m)^2 + 2b\right)\right) \tag{B.14b}$$

Interpretation in terms of a Gaussian with its variance scaled by an inverse-Gamma distributed variable q. This also provides a way of generating variables from the $\mathcal{NIG}$ distribution.

B.2.5 Student's t

B.3 Multivariate continuous distributions

B.3.1 Multivariate Gaussian distribution

In this section we provide some important properties of the multivariate Gaussian (or Normal) distribution, which is an important distribution in itself. However, it is also an important building block in more sophisticated probabilistic models.

The most common way of parameterizing the multivariate Gaussian distribution is according to

$$\mathcal{N}\left(x \mid \mu,\, \Sigma\right) \triangleq \frac{1}{(2\pi)^{n/2}\sqrt{\det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^{\mathsf{T}} \Sigma^{-1} (x-\mu)\right) \tag{B.15}$$

where $x \in \mathbb{R}^n$ denotes a random vector that is Gaussian distributed, with mean value $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in S^n_{++}$ (i.e. the n-dimensional positive definite cone). Furthermore, $x \sim \mathcal{N}(\mu, \Sigma)$ states that the random vector x is Gaussian with mean μ and covariance Σ.

The parameterizing (B.15) is sometimes referred to as the *moment* form. An alternative parameterization of the Gaussian density is provided by the *information* form, which is also referred to as the *canonical* form or the *natural* form. This parameterization

$$\mathcal{N}^{-1}(x;\nu,\Lambda) \triangleq \frac{\exp\left(-\frac{1}{2}\nu^{\mathsf{T}}\Lambda^{-1}\nu\right)}{(2\pi)^{n/2}\sqrt{\det\Lambda^{-1}}}\exp\left(-\frac{1}{2}x^{\mathsf{T}}\Lambda x + x^{\mathsf{T}}\nu\right), \tag{B.16}$$

is parameterized by the information vector ν and (Fisher) information matrix Λ. It is straightforward to see the relationship between the two parameterizations (B.15) and (B.16),

$$\begin{aligned}\mathcal{N}(x \mid \mu, \Sigma) &= \frac{1}{(2\pi)^{n/2}\sqrt{\det\Sigma}}\exp\left(-\frac{1}{2}x^{\mathsf{T}}\Sigma^{-1}x + x^{\mathsf{T}}\Sigma^{-1}\mu - \frac{1}{2}\mu^{\mathsf{T}}\Sigma^{-1}\mu\right)\\ &= \frac{\exp\left(-\frac{1}{2}\mu^{\mathsf{T}}\Sigma^{-1}\mu\right)}{(2\pi)^{n/2}\sqrt{\det\Sigma}}\exp\left(-\frac{1}{2}x^{\mathsf{T}}\Sigma^{-1}x + x^{\mathsf{T}}\Sigma^{-1}\mu\right),\end{aligned} \tag{B.17}$$

which also reveals the that the parameters used in the information form are given by a scaled version of the mean,

$$\nu = \Sigma^{-1}\mu \tag{B.18a}$$

and the inverse of the covariance matrix (sometimes also called the precision matrix)

$$\Lambda = \Sigma^{-1}. \tag{B.18b}$$

The moment form (B.15) and the information form (B.16) results in different computations. It is useful to understand these differences in order to derive efficient algorithms.

Basic Properties

Theorem 5. *(Sum of independent Gaussians)*

Theorem 6. ***(Product of Gaussians)*** *The product of two Gaussians is an un-normalized Gaussian,*

$$\mathcal{N}(x \mid a, A)\,\mathcal{N}(x \mid b, B) = C\mathcal{N}(x \mid d, D), \tag{B.19}$$

where

$$d = D(A^{-1}a + B^{-1}b), \tag{B.20a}$$

$$D = (A^{-1} + B^{-1})^{-1}, \tag{B.20b}$$

$$C = \frac{1}{(2\pi)^{n_x/2}\sqrt{\det(A+B)}}\exp\left(-\frac{1}{2}(a-b)^{\mathsf{T}}(A+B)^{-1}(a-b)\right). \tag{B.20c}$$

Partitioned Gaussian densities

Let us, without loss of generality, assume that random vector x, its mean μ and its covariance Σ can be partitioned according to

$$x = \begin{pmatrix} x_a \\ x_b \end{pmatrix}, \qquad \mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \tag{B.21}$$

where for reasons of symmetry $\Sigma_{ba} = \Sigma_{ab}^{\mathsf{T}}$. It is also useful to write down the partitioned information matrix

$$\Lambda = \Sigma^{-1} = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}, \tag{B.22}$$

since this form will provide simpler calculations below. Note that, since the inverse of a symmetric matrix is also symmetric, we have $\Lambda_{ab} = \Lambda_{ba}^{\mathsf{T}}$.

We will now derive two important and very useful theorems for partitioned Gaussian densities. These theorems will explain the two operations marginalization and conditioning.

Theorem 7. ***(Marginalization)*** *Let the random vector x be Gaussian distributed according to* (B.15) *and let it be partitioned according to* (B.21)*, then the marginal density $p(x_a)$ is given by*

$$p(x_a) = \mathcal{N}\left(x_a \,|\, \mu_a,\, \Sigma_{aa}\right). \tag{B.23}$$

Proof: The marginal density $p(x_a)$ is by definition obtained by integrating out the x_b variables from the joint density $p(x_a, x_b)$ according to

$$p(x_a) = \int p(x_a, x_b) \mathrm{d}x_b, \tag{B.24}$$

where

$$p(x_a, x_b) = \frac{1}{(2\pi)^{n/2}\sqrt{\det \Sigma}} \exp(E) \tag{B.25}$$

and Σ was defined in (B.21) and the exponent E is given by

$$\begin{aligned} E = &-\frac{1}{2}(x_a - \mu_a)^{\mathsf{T}} \Lambda_{aa} (x_a - \mu_a) - \frac{1}{2}(x_a - \mu_a)^{\mathsf{T}} \Lambda_{ab} (x_b - \mu_b) \\ &- \frac{1}{2}(x_b - \mu_b)^{\mathsf{T}} \Lambda_{ba} (x_a - \mu_a) - \frac{1}{2}(x_b - \mu_b)^{\mathsf{T}} \Lambda_{bb} (x_b - \mu_b) \\ = &-\frac{1}{2}\big(x_b^{\mathsf{T}} \Lambda_{bb} x_b - 2x_b^{\mathsf{T}} \Lambda_{bb} (\mu_b - \Lambda_{bb}^{-1} \Lambda_{ba} (x_a - \mu_a)) - 2x_a^{\mathsf{T}} \Lambda_{ab} \mu_b \\ &+ 2\mu_a^{\mathsf{T}} \Lambda_{ab} \mu_b + \mu_b^{\mathsf{T}} \Lambda_{bb} \mu_b + x_a^{\mathsf{T}} \Lambda_{aa} x_a - 2x_a^{\mathsf{T}} \Lambda_{aa} \mu_a + \mu_a^{\mathsf{T}} \Lambda_{aa} \mu_a \big) \end{aligned} \tag{B.26}$$

Completing the squares in the above expression results in

$$\begin{aligned}
E &= -\frac{1}{2}(x_b - (\mu_b - \Lambda_{bb}^{-1}\Lambda_{ba}(x_a - \mu_a)))^\mathsf{T}\Lambda_{bb}(x_b - (\mu_b - \Lambda_{bb}^{-1}\Lambda_{ba}(x_a - \mu_a))) \\
&\quad + \frac{1}{2}\big(x_a^\mathsf{T}\Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}x_a - 2x_a^\mathsf{T}\Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}\mu_a + \mu_a\Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}\mu_a\big) \\
&\quad - \frac{1}{2}\big(x_a^\mathsf{T}\Lambda_{aa}x_a - 2x_a^\mathsf{T}\Lambda_{aa}\mu_a + \mu_a^\mathsf{T}\Lambda_{aa}\mu_a\big) \\
&= -\frac{1}{2}(x_b - (\mu_b - \Lambda_{bb}^{-1}\Lambda_{ba}(x_a - \mu_a)))^\mathsf{T}\Lambda_{bb}(x_b - (\mu_b - \Lambda_{bb}^{-1}\Lambda_{ba}(x_a - \mu_a))) \\
&\quad - \frac{1}{2}(x_a - \mu_a)^\mathsf{T}(\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba})(x_a - \mu_a) \qquad \text{(B.27)}
\end{aligned}$$

Using the block matrix inversion provided in (C.4) we have $\Sigma_{aa}^{-1} = \Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba}$, which together with (B.27) results in

$$p(x_a, x_b) = \frac{1}{(2\pi)^{n/2}\sqrt{\det\Sigma}}\exp(E_1)\exp(E_2) \qquad \text{(B.28)}$$

where

$$E_1 = -\frac{1}{2}(x_b - (\mu_b - \Lambda_{bb}^{-1}\Lambda_{ba}(x_a - \mu_a)))^\mathsf{T}\Lambda_{bb}(x_b - (\mu_b - \Lambda_{bb}^{-1}\Lambda_{ba}(x_a - \mu_a))), \qquad \text{(B.29a)}$$

$$E_2 = -\frac{1}{2}(x_a - \mu_a)^\mathsf{T}(\Lambda_{aa} - \Lambda_{ab}\Lambda_{bb}^{-1}\Lambda_{ba})(x_a - \mu_a). \qquad \text{(B.29b)}$$

Now, since E_2 is independent of x_b we have

$$p(x_a) = \frac{1}{(2\pi)^{n/2}\sqrt{\det\Sigma}}\int \exp(E_1)\mathrm{d}x_b\exp(E_2). \qquad \text{(B.30)}$$

Since the integral of a density function is equal to one, we have $\int\exp(E_1)\mathrm{d}x_b = (2\pi)^{n_b/2}\sqrt{\det\Lambda_{bb}^{-1}}$, which inserted into (B.30) results in

$$p(x_a) = \frac{\sqrt{\det\Lambda_{bb}^{-1}}}{(2\pi)^{n_a/2}\sqrt{\det\Sigma}}\exp((x_a - \mu_a)^\mathsf{T}\Sigma_{aa}^{-1}(x_a - \mu_a)). \qquad \text{(B.31)}$$

Finally, the determinant property (C.2b) gives

$$\det\Sigma = \det\Sigma_{aa}\det(\Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab}) \qquad \text{(B.32)}$$

and the block matrix inversion property (C.4) provides

$$\Lambda_{bb}^{-1} = \Sigma_{bb} - \Sigma_{ba}\Sigma_{aa}^{-1}\Sigma_{ab}. \qquad \text{(B.33)}$$

Now, inserting (B.32) and (B.33) into (B.31) concludes the proof. □

Theorem 8. ***(Conditioning)*** *Let the random vector x be Gaussian distributed according to* (B.15) *and let it be partitioned according to* (B.21)*, then the conditional density $p(x_a \,|\, x_b)$ is given by*

$$p(x_a \,|\, x_b) = \mathcal{N}\left(x_a \,\middle|\, \mu_{a\,|\,b},\, \Sigma_{a\,|\,b}\right), \tag{B.34a}$$

where

$$\mu_{a\,|\,b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b), \tag{B.34b}$$
$$\Sigma_{a\,|\,b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}, \tag{B.34c}$$

which using the information matrix can be written,

$$\mu_{a\,|\,b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b), \tag{B.34d}$$
$$\Sigma_{a\,|\,b} = \Lambda_{aa}^{-1}. \tag{B.34e}$$

Proof: We will make use of the fact that $p(x_a \,|\, x_b) = p(x_a, x_b)/p(x_b)$, which according to the definition (B.15) is

$$p(x_a \,|\, x_b) = \sqrt{\frac{\det \Sigma_{bb}}{(2\pi)^{n_a/2}\det \Sigma}}\exp(E), \tag{B.35}$$

where $E = -1/2(x-\mu)^\mathsf{T}\Sigma^{-1}(x-\mu) + 1/2(x_b-\mu_b)\Sigma_{bb}^{-1}(x_b-\mu_b)$. Let us first consider the constant in front of the exponential in (B.35),

$$\sqrt{\frac{\det \Sigma_{bb}}{(2\pi)^{n_a/2}\det \Sigma}}. \tag{B.36}$$

Using the result on determinants given in (C.2b) we have

$$\det \Sigma = \det \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} = \det \Sigma_{bb} \det(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}) \tag{B.37}$$

which inserted into (B.36) results in the following expression for the constant in front of the exponent

$$\frac{1}{(2\pi)^{n_a/2}\det(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})} \tag{B.38}$$

Let us now continue by studying the exponent of (B.35), which using the precision matrix is given by

$$\begin{aligned} E &= -\frac{1}{2}(x-\mu)^\mathsf{T}\Lambda(x-\mu) + \frac{1}{2}(x_b-\mu_b)\Sigma_{bb}^{-1}(x_b-\mu_b) \\ &= -\frac{1}{2}(x_a-\mu_a)^\mathsf{T}\Lambda_{aa}(x_a-\mu_a) - \frac{1}{2}(x_a-\mu_a)^\mathsf{T}\Lambda_{ab}(x_b-\mu_b) \\ &\quad - \frac{1}{2}(x_b-\mu_b)^\mathsf{T}\Lambda_{ba}(x_a-\mu_a) - \frac{1}{2}(x_b-\mu_b)^\mathsf{T}(\Lambda_{bb} - \Sigma_{bb}^{-1})(x_b-\mu_b). \end{aligned} \tag{B.39}$$

Reordering the terms in (B.39) results in

$$E = -\frac{1}{2}x_a^\mathsf{T}\Lambda_{aa}x_a + x_a^\mathsf{T}\left(\Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b)\right) - \frac{1}{2}\mu_a^\mathsf{T}\Lambda_{aa}\mu_a + \mu_a^\mathsf{T}\Lambda_{ab}(x_b - \mu_b) - \frac{1}{2}(x_b - \mu_b)^\mathsf{T}(\Lambda_{bb} - \Sigma_{bb}^{-1})(x_b - \mu_b). \tag{B.40}$$

Using the block matrix inversion result (C.4), we have $\Sigma_{bb}^{-1} = \Lambda_{bb} - \Lambda_{ba}\Lambda_{aa}^{-1}\Lambda_{ab}$, which inserted into (B.40) results in

$$E = -\frac{1}{2}x_a^\mathsf{T}\Lambda_{aa}x_a + x_a^\mathsf{T}\left(\Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b)\right) - \frac{1}{2}\mu_a^\mathsf{T}\Lambda_{aa}\mu_a + \mu_a^\mathsf{T}\Lambda_{ab}(x_b - \mu_b) - \frac{1}{2}(x_b - \mu_b)^\mathsf{T}\Lambda_{ba}\Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b). \tag{B.41}$$

We can now complete the squares, which gives us

$$E = -\frac{1}{2}\left(x_a - (\mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b))\right)^\mathsf{T}\Lambda_{aa}\left(x_a - (\mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b))\right). \tag{B.42}$$

Finally, combining (B.38) and (B.42) results in

$$p(x_a \,|\, x_b) = \frac{1}{(2\pi)^{nx/2}\det\Lambda_{aa}^{-1}}\exp(E) \tag{B.43}$$

which concludes the proof. □

Affine transformations

In the previous section we dealt with partitioned Gaussian densities, and derived the expressions for the marginal and conditional densities expressed in terms of the parameters of the joint density. We shall now take a different starting point, namely that we are given the marginal density $p(x_a)$ and the conditional density $p(x_b \,|\, x_a)$ (affine in x_a) and derive expressions for the joint density $p(x_a, x_b)$, the marginal density $p(x_b)$ and the conditional density $p(x_a \,|\, x_b)$.

Theorem 9. ***(Affine transformation)*** *Assume that x_a, as well as x_b conditioned on x_a, are Gaussian distributed according to*

$$p(x_a) = \mathcal{N}\left(x_a \,|\, \mu_a,\, \Sigma_a\right), \tag{B.44a}$$

$$p(x_b \,|\, x_a) = \mathcal{N}\left(x_b \,\middle|\, Mx_a + b,\, \Sigma_{b\,|\,a}\right), \tag{B.44b}$$

where M is a matrix (of appropriate dimension) and b is a constant vector. The joint distribution of x_a and x_b is then given by

$$p(x_a, x_b) = \mathcal{N}\left(\begin{pmatrix} x_a \\ x_b \end{pmatrix} \,\middle|\, \begin{pmatrix} \mu_a \\ M\mu_a + b \end{pmatrix},\, R\right), \tag{B.44c}$$

with

$$R = \begin{pmatrix} M^\mathsf{T}\Sigma_{b\,|\,a}^{-1}M + \Sigma_a^{-1} & -M^\mathsf{T}\Sigma_{b\,|\,a}^{-1} \\ -\Sigma_{b\,|\,a}^{-1}M & \Sigma_{b\,|\,a}^{-1} \end{pmatrix}^{-1} = \begin{pmatrix} \Sigma_a & \Sigma_a M^\mathsf{T} \\ M\Sigma_a & \Sigma_{b\,|\,a} + M\Sigma_a M^\mathsf{T} \end{pmatrix}. \tag{B.44d}$$

Proof: We start by introducing the vector

$$x = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \tag{B.45}$$

for which the joint distribution is given by

$$p(x) = p(x_b \,|\, x_a)p(x_a) = \frac{(2\pi)^{-(n_a+n_b)/2}}{\sqrt{\det \Sigma_{b\,|\,a} \det \Sigma_a}} e^{-\frac{1}{2}E}, \tag{B.46}$$

where

$$E = (x_b - Mx_a - b)^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} (x_b - Mx_a - b) + (x_a - \mu_a)^{\mathsf{T}} \Sigma_a^{-1} (x_a - \mu_a). \tag{B.47}$$

Introduce the following variables

$$e = x_a - \mu_a, \tag{B.48a}$$
$$f = x_b - M\mu_a - b, \tag{B.48b}$$

which allows us to write the exponent (B.47) as

$$\begin{aligned} E &= (f - Me)^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} (f - Me) + e^{\mathsf{T}} \Sigma_a^{-1} e \\ &= e^{\mathsf{T}} (M^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} M + \Sigma_a^{-1}) e - e^{\mathsf{T}} M^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} f - f \Sigma_{b\,|\,a}^{-1} M e + f^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} f \\ &= \begin{pmatrix} e \\ f \end{pmatrix}^{\mathsf{T}} \underbrace{\begin{pmatrix} M^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} M + \Sigma_a^{-1} & -M^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} \\ -\Sigma_{b\,|\,a}^{-1} M & \Sigma_{b\,|\,a}^{-1} \end{pmatrix}}_{\triangleq R^{-1}} \begin{pmatrix} e \\ f \end{pmatrix} \\ &= \begin{pmatrix} x_a - \mu_a \\ x_b - M\mu_a - b \end{pmatrix}^{\mathsf{T}} R^{-1} \begin{pmatrix} x_a - \mu_a \\ x_b - M\mu_a - b \end{pmatrix} \end{aligned} \tag{B.49}$$

Furthermore, from (C.2b) we have that

$$\begin{aligned} \frac{1}{\det R} &= \det R^{-1} = \det \left(\Sigma_{b\,|\,a}^{-1} \right) \det \left(M^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} M + \Sigma_a^{-1} - M^{\mathsf{T}} \Sigma_{b\,|\,a}^{-1} \Sigma_{b\,|\,a} \Sigma_{b\,|\,a}^{-1} M \right) \\ &= \det \left(\Sigma_{b\,|\,a}^{-1} \right) \det \left(\Sigma_a^{-1} \right) = \frac{1}{\det \left(\Sigma_{b\,|\,a} \right) \det \left(\Sigma_a \right)}. \end{aligned} \tag{B.50}$$

Hence, from (B.46), (B.49) and (B.50) we can write the joint PDF for x as

$$\begin{aligned} p(x) &= \frac{(2\pi)^{-(n_a+n_b)/2}}{\sqrt{\det R}} \exp \left(-\frac{1}{2} \left(\begin{pmatrix} x_a - \mu_a \\ x_b - M\mu_a - b \end{pmatrix}^{\mathsf{T}} R^{-1} \begin{pmatrix} x_a - \mu_a \\ x_b - M\mu_a - b \end{pmatrix} \right) \right) \\ &= \mathcal{N} \left(x \,\middle|\, \begin{pmatrix} \mu_a \\ M\mu_a + b \end{pmatrix}, R \right) \end{aligned} \tag{B.51}$$

which concludes the proof. □

Combining the results in Theorems 7, 8 and 9 we also get the following corollary.

Corollary 1. ***(Affine transformation – marginal and conditional)*** *Assume that x_a, as well as x_b conditioned on x_a, are Gaussian distributed according to*

$$p(x_a) = \mathcal{N}\left(x_a \,|\, \mu_a, \Sigma_a\right), \tag{B.52a}$$
$$p(x_b \,|\, x_a) = \mathcal{N}\left(x_b \,\middle|\, Mx_a + b, \Sigma_{b\,|\,a}\right), \tag{B.52b}$$

where M is a matrix (of appropriate dimension) and b is a constant vector. The marginal density of x_b is then given by

$$p(x_b) = \mathcal{N}\left(x_b \,|\, \mu_b, \Sigma_b\right), \tag{B.52c}$$

with

$$\mu_b = M\mu_a + b, \tag{B.52d}$$
$$\Sigma_b = \Sigma_{b\,|\,a} + M\Sigma_a M^{\mathsf{T}}. \tag{B.52e}$$

The conditional density of x_a given x_b is

$$p(x_a \,|\, x_b) = \mathcal{N}\left(x_a \,\middle|\, \mu_{a\,|\,b}, \Sigma_{a\,|\,b}\right), \tag{B.52f}$$

with

$$\mu_{a\,|\,b} = \Sigma_{a\,|\,b}\left(M^{\mathsf{T}}\Sigma_{b\,|\,a}^{-1}(x_b - b) + \Sigma_a^{-1}\mu_a\right) = \mu_a + \Sigma_a M^{\mathsf{T}}\Sigma_b^{-1}(x_b - b - M\mu_a), \tag{B.52g}$$
$$\Sigma_{a\,|\,b} = \left(\Sigma_a^{-1} + M^{\mathsf{T}}\Sigma_{b\,|\,a}^{-1}M\right)^{-1} = \Sigma_a - \Sigma_a M^{\mathsf{T}}\Sigma_b^{-1}M\Sigma_a. \tag{B.52h}$$

B.4 Matrix valued continuous distributions

B.4.1 Matrix Normal

The matrix valued normal distribution is a generalization of the vector valued normal distribution.

Definition 7 (Matrix normal distribution)**.** *The stochastic matrix $X \in \mathbb{R}^{d\times m}$ has a matrix normal distribution with mean matrix $M \in \mathbb{R}^{d\times m}$ and covariance matrix $\Lambda^{-1}\otimes\Sigma$, where $\Lambda^{-1} \succ 0 \in \mathbb{R}^{m\times m}$ and $\Sigma \succ 0 \in \mathbb{R}^{d\times d}$ if*

$$\mathrm{Vec}\left(X\right) \sim \mathcal{N}\left(X \,\middle|\, \mathrm{Vec}\left(M\right), \Lambda^{-1}\otimes\Sigma\right). \tag{B.53}$$

In the above definition the matrix Λ^{-1} is referred to as the left covariance and the matrix Σ is referred to as the right covariance. From this definition we can now show that the PDF of X is given by

$$\mathcal{MN}\left(X \,|\, M, \Lambda, \Sigma\right) = \frac{|\Lambda|^{d/2}}{(2\pi)^{dm/2}\,|\Sigma|^{m/2}} \exp\left(-\frac{1}{2}\,\mathrm{Tr}\left((X-M)^{\mathsf{T}}\Sigma^{-1}(X-M)\Lambda\right)\right). \tag{B.54}$$

B.4.2 Wishart

The Wishart distribution is a probability distribution defined over symmetric and nonnegative-definite random matrices. It is commonly used in modeling random precision (inverse covariance) matrices and it is the conjugate prior of the precision (inverse covariance) matrix of a vector valued Normal random variable.

The Wishart distribution also has an interpretation in terms of a generalization of the gamma distribution or the χ^2 distribution for integer valued degrees of freedom.

B.4.3 Inverse Wishart

The inverse Wishart distribution is a matrix valued generalization of the inverse Gamma distribution. It is commonly used in modeling random covariance matrices and it is the conjugate prior of the covariance matrix of a vector valued Normal random variable.

Definition 8 (Inverse Wishart distribution)**.** *The random matrix $\Sigma \in \mathbb{R}^{d\times d}$ has an inverse Wishart distribution with ν degrees of freedom and parameter matrix $S \in \mathbb{R}^{d\times d}$ if its PDF is given by*

$$p(\Sigma) = \mathcal{IW}\left(\Sigma \,|\, \nu, S\right) = \frac{|S|^{\nu/2}}{2^{d\nu/2}\Gamma_d(\nu/2)} \left|\Sigma\right|^{-(d+\nu+1)/2} \exp\left(-\frac{1}{2}\operatorname{Tr}\left(S\Sigma^{-1}\right)\right) \tag{B.55}$$

where $\Gamma_d(\nu/2)$ is the multivariate gamma function,

$$\Gamma_d(\nu/2) = \pi^{d(d-1)/4} \prod_{i=1}^{d} \Gamma\left(\frac{\nu+1-i}{2}\right) \tag{B.56}$$

B.4.4 Matrix Normal Inverse Wishart

The $\mathcal{MNIW}$ distribution for the matrices A and Σ ($p(A,\Sigma) = p(A \,|\, \Sigma)p(\Sigma)$) consists of a conditional matrix normal distribution for A, given Σ

$$p(A \,|\, \Sigma) = \mathcal{MN}\left(A \,|\, M, \Lambda, \Sigma\right) \tag{B.57}$$

and an inverse Wishart distribution for Σ

$$p(\Sigma) = \mathcal{IW}\left(\Sigma \,|\, \nu, S\right). \tag{B.58}$$

The $\mathcal{MNIW}$ distribution is thus a *hierarchical distribution.*

B.5 Further reading

Muller and Stewart (2006) Gupta and Nagar (2000), Muller and Stewart (2006) Agresti (2007) Press et al. (2007)

Appendix C

Matrix Theory

This appendix provides some useful results from matrix theory.

Consider the following block matrix

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{C.1}$$

The following results hold for the determinant

$$\det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det A \det(\underbrace{D - CA^{-1}B}_{\Delta_A}), \tag{C.2a}$$

$$= \det D \det(\underbrace{A - BD^{-1}C}_{\Delta_D}), \tag{C.2b}$$

where $\Delta_D = A - BD^{-1}C$ is referred to as the *Schur complement* of D in M and $\Delta_A = D - CA^{-1}B$ is referred to as the Schur complement of A in M.

When the block matrix (C.1) is invertible its inverse can be written according to

$$\begin{aligned} \begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} &= \begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & 0 \\ 0 & \Delta_A^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix} \\ &= \begin{pmatrix} A^{-1} + A^{-1}B\Delta_A^{-1}CA^{-1} & -A^{-1}B\Delta_A^{-1} \\ -\Delta_A^{-1}CA^{-1} & \Delta_A^{-1} \end{pmatrix}, \end{aligned} \tag{C.3}$$

where we have made use of the Schur complement of A in M. We can also use the Schur complement of D in M, resulting in

$$\begin{aligned} \begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} &= \begin{pmatrix} I & 0 \\ -D^{-1}C & I \end{pmatrix} \begin{pmatrix} \Delta_D^{-1} & 0 \\ 0 & D^{-1} \end{pmatrix} \begin{pmatrix} I & -BD^{-1} \\ 0 & I \end{pmatrix} \\ &= \begin{pmatrix} \Delta_D^{-1} & -\Delta_D^{-1}BD^{-1} \\ -D^{-1}C\Delta_D^{-1} & D^{-1} + D^{-1}C\Delta_D^{-1}BD^{-1} \end{pmatrix}. \end{aligned} \tag{C.4}$$

The *matrix inversion lemma*

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}, \tag{C.5}$$

under the assumption that the involved inverses exist. It is worth noting that the matrix inversion lemma is sometimes also referred to as the Woodbury matrix identity, the Sherman-Morrison-Woodbury formula or the Woodbury formula.

C.1 Matrix Derivatives

C.2 Trace

$A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n}$

$$\mathrm{Tr}\,(AB) = \mathrm{Tr}\,(BA)\,, \tag{C.6}$$

$$\mathrm{Tr}\,(A) = \mathrm{Tr}\left(A^{\mathsf{T}}\right). \tag{C.7}$$

C.3 Further Reading

Matrix differential calculus Magnus and Neudecker (1999).

Bibliography

Agresti, A. (2007). *An Introduction to Categorical Data Analysis*. Wiley, second edition.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, 50:5–43.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Bernado, J. M. and Smith, A. F. M. (2000). *Bayesian theory*. John Wiley & Sons.

Bouchard-Côté, A., Sankararaman, S., and Jordan, M. I. (2012). Phylogenetic inference via sequential Monte Carlo. *Systematic Biology*, 61(4):579–593.

Box, G. E. P. and Tiao, G. C. (1992). *Bayesian inference in statistical analysis*. John Wiley & Sons, wiley classics library edition edition.

Bresler, Y. (1986). Two-filter formula for discrete-time non-linear bayesian smoothing. *International Journal of Control*, 43(2):626–641.

Cappé, O., Godsill, S., and Moulines, E. (2007). An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924.

Cappé, O., Moulines, E., and Rydén, T. (2005). *Inference in Hidden Markov Models*. Springer Series in Statistics. Springer, New York, USA.

Carpenter, J., Clifford, P., and Fearnhead, P. (1999). Improved particle filter for nonlinear problems. *IEE Proceedings – Radar, Sonar and Navigation*, 146(1):2–7.

Carter, C. K. and Kohn, R. (1994). On Gibbs sampling for state space models. *Biometrika*, 81:541–553.

Chen, R. and Liu, J. S. (2000). Mixture Kalman filters. *Journal of the Royal Statistical Society*, 62(3):493–508.

Chen, Y., Xie, J., and Liu, J. S. (2005). Stopping-time resampling for sequential Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):199–217.

Chesney, M. and Scott, L. (1989). Pricing European currency options: A comparison of the modifies Black-Scholes model and a random variance model. *Journal of Financial and Quantitative Analysis*, 24(3):267–284.

Chopin, N. (2004). Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *The Annals of Statistics*, 32(6):2385–2411.

Crisan, D. and Doucet, A. (2002). A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746.

De Jong, P. and Shephard, N. (1995). The simulation smoother for time series models. *Biometrika*, 82(2):339–350.

Del Moral, P. (2004). *Feynman-Kac formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Applications. Springer, New York, USA.

Del Moral, P. (2013). *Mean field simulation for Monte Carlo integration*. Chapman and Hall/CRC.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society. Series B (Methodological)*, 63(3):411–436.

Denison, D. G. T., Holmes, C. C., Mallick, B. K., and Smith, A. F. M. (2002). *Bayesian methods for nonlinear classifications and regression*. John Wiley & Sons.

Douc, R., Cappé, O., and Moulines, E. (2005). Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, Zagreb, Croatia.

Douc, R., Moulines, E., and Stoffer, D. (2014). *Nonlinear time series: Theory, methods and applications with R examples*. Chapman & Hall/CRC.

Doucet, A., de Freitas, N., and Gordon, N., editors (2001a). *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, USA.

Doucet, A., de Freitas, N., Murphy, K., and Russell, S. (2000a). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, Stanford, CA, USA.

Doucet, A., Godsill, S. J., and Andrieu, C. (2000b). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208.

Doucet, A., Gordon, N., and Krishnamurthy, V. (2001b). Particle filters for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624.

Doucet, A. and Johansen, A. M. (2011). A tutorial on particle filtering and smoothing: Fifteen years later. In Crisan, D. and Rozovsky, B., editors, *Nonlinear Filtering Handbook*. Oxford University Press.

Durbin, J. and Koopman, S. J. (2002). A simple and efficient simulation smoother for state space time series analysis. *Biometrika*, 89(3):603–615.

Fearnhead, P. (1998). *Sequential Monte Carlo methods in filter theory.* PhD thesis, University of Oxford, Oxford, UK.

Fearnhead, P. (2004). Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21.

Fearnhead, P. and Clifford, P. (2003). On-line inference for hidden markov models via particle filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(4):887–899.

Fisher, R. A. (1912). On an absolute criterion for fitting frequency curves. *Messenger of Mathematics*, 41:155–160.

Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society Series A*, 222:309–368.

Fox, E., Sudderth, E., Jordan, M., and Willsky, A. (2011). Bayesian nonparametric inference of switching dynamic linear models. *IEEE Transactions on Signal Processing*, 59(4):1569–1585.

Frühwirth-Schnatter, S. (1994). Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, 15(2):183–202.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2003). *Bayesian Data Analysis.* Chapman & Hall/CRC, second edition.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6):1317–1339.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113.

Gupta, A. K. and Nagar, D. K. (2000). *Matrix variate distributions.* Monographs and surveys in pure and applied mathematics. Chapman & Hall/CRC.

Gustafsson, F. (2010). Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82.

Hammersley, J. M. and Morton, K. W. (1954). Poor man's Monte Carlo. *Journal of the Royal Statistical Society. Series B (Methodological)*, 16(1):23–38.

Handschin, J. E. (1970). Monte Carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica*, 6:555–563.

Handschin, J. E. and Mayne, D. Q. (1969). Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control*, 9:547–559.

Hol, J. D., Schön, T. B., and Gustafsson, F. (2006). On resampling algorithms for particle filters. In *Nonlinear Statistical Signal Processing Workshop*, Cambridge, United Kingdom.

Hu, X.-L., Schön, T. B., and Ljung, L. (2008). A basic convergence result for particle filtering. *IEEE Transactions on Signal Processing*, 56(4):1337–1348.

Hürzeler, M. and Künsch, H. R. (1998). Monte Carlo approximations for general state-space models. *Journal of Computational and Graphical Statistics*, 7(2):175–193.

Jazwinski, A. H. (1970). *Stochastic processes and filtering theory*. Mathematics in science and engineering. Academic Press, New York, USA.

Johansen, A. M. and Doucet, A. (2008). A note on auxiliary particle filters. *Statistics and Probability Letters*, 78(12):1498–1504.

Jordan, M. I. (2004). Graphical models. *Statistical Science*, 19(1):140–155.

Kahn, H. and Marshall, A. W. (1953). Methods of reducing sample size in Monte Carlo computations. *Journal of the Operations Research Society of America*, 1(5):263–278.

Kailath, T., Sayed, A. H., and Hassibi, B. (2000). *Linear Estimation*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, NJ, USA.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45.

Kitagawa, G. (1993). A Monte Carlo filtering and smoothing method for non-Gaussian nonlinear state space models. In *Proceedings of the 2nd U.S.-Japan joint seminar on statistical time series analysis*, pages 110–131, Honolulu, Hawaii.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25.

Klaas, M., de Freitas, N., and Doucet, A. (2005). Toward practical N^2 Monte Carlo: the marginal particle filter. In *Proceedings of the 21st conference on Uncertainty in Artificial Intelligence (UAI)*, Edinburgh, Scotland.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models – Principles and Techniques*. MIT Press, Cambridge, MA, USA.

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of American Statistical Association*, 89(425):278–288.

Künsch, H. R. (2005). Recursive Monte Carlo filters: algorithms and theoretical analysis. *The Annals of Statistics*, 33(5):1983–2021.

Künsch, H. R. (2013). Particle filters. *Bernoulli*, 19(4):1391–1403.

Lindsten, F., Schön, T. B., and Olsson, J. (2011). An explicit variance reduction expression for the Rao-Blackwellised particle filter. In *Proceedings of the 18th World Congress of the International Federation of Automatic Control (IFAC)*, Milan, Italy.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics. Springer, New York, USA.

Liu, J. S. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044.

Ljung, L. (1999). *System identification, Theory for the user*. System sciences series. Prentice Hall, Upper Saddle River, NJ, USA, second edition.

MacEachern, S. N., Clyde, M., and Liu, J. S. (1999). Sequential importance sampling for nonparametric Bayes models: The next generation. *The Canadian Journal of Statistics*, 27(2):251–267.

Magnus, J. R. and Neudecker, H. (1999). *Matrix differential calculus with applications in statistics and econometrics*. Wiley series in probability and statistics. John Wiley & Sons, second edition.

Melino, A. and Turnbull, S. (1990). Pricing foreign currency options with stochastic volatility. *Journal of Econometrics*, 45(1–2):239–265.

Metropolis, N. and Ulam, S. (1949). The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341.

Muller, K. E. and Stewart, P. W. (2006). *Linear model theory univariate, multivariate and mixed models*. jw.

Murray, L. M., Lee, A., and Jacob, P. E. (2013). Rethinking resampling in the particle filter on graphics processing units. Technical report, arXiv:1301.4019 [stat.CO].

Naesseth, C. A., Lindsten, F., and Schön, T. B. (2014). Sequential Monte Carlo methods for graphical models. In *Advances in Neural Information Processing Systems (NIPS) 27*.

Ninness, B. and Henriksen, S. J. (2010). Bayesian system identification via Markov chain Monte Carlo techniques. *Automatica*, 46(1):40–51.

Peterka, V. (1981). Bayesian system identification. *Automatica*, 17(1):41–53.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599.

Pitt, M. K., Silva, R. S., Giordani, P., and Kohn, R. (2011). Auxiliary particle filtering within adaptive Metropolis-Hastings sampling. Technical report, arXiv:1006.1914.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 3rd edition.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rauch, H. E., Tung, F., and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyond the Kalman Filter: particle filters for tracking applications*. Artech House, London, UK.

Robert, C. P. (2001). *The Bayesian choice*. Springer texts in statistics. Springer, New York, USA, second edition.

Rubin, D. B. (1987). The calculation of posterior distributions by data augmentation: comment: A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithms. *Journal of the American Statistical Association*, 82(398):543–546.

Rubin, D. B. (1988). Using the SIR algorithm to simulate posterior distributions. In Bernardo, J., DeGroot, M., Lindley, D., and Smith, A., editors, *Bayesian Statistics 3*, pages 395–402. Oxford University Press.

Särkkä, S. (2013). *Bayesian filtering and smoothing*. Cambridge University Press.

Schön, T., Gustafsson, F., and Nordlund, P.-J. (2005). Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7):2279–2289.

Söderström, T. and Stoica, P. (1989). *System identification*. Systems and Control Engineering. Prentice Hall.

Steward, L. and McCarty, P. (1992). The use of Bayesian belief networks to fuse continuous and discrete information for target recognition and discrete information for target recognition, tracking, and situation assessment. In *Proceedings of SPIE Signal Processing, Sensor Fusion and Target Recognition*, volume 1699, pages 177–185.

Tanizaki, H. and Mariano, R. S. (1998). Nonlinear and non-Gaussian state-space modeling with Monte Carlo simulations. *Journal of Econometrics*, 83(1–2):263–290.

Teh, Y. W., Daumé III, H., and Roy, D. (2008). Bayesian agglomerative clustering with coalescents. *Advances in Neural Information Processing*, pages 1473–1480.

Verhaegen, M. and Verdult, V. (2007). *Filtering and System Identification - A Least Squares Approach*. Cambridge University Press.

Whiteley, N., Andrieu, C., and Doucet, A. (2010). Efficient bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods. Technical report, arXiv:1011.2437.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85.

Wilkinson, D. J. and Yeung, S. K. H. (2002). Conditional simulation from highly structured Gaussian systems, with application to blocking-MCMC for the Bayesian analysis of very large linear models. *Statistics and Computing*, 12(3):287–300.