Foundations and Trends® in
Machine Learning
Vol. 6, No. 1 (2013) 1–143

DOI: 10.1561/2200000045

now
the essence of knowledge

Backward Simulation Methods for Monte Carlo Statistical Inference

By Fredrik Lindsten and Thomas B. Schön

Contents

Foundations and Trends® in
Machine Learning
Vol. 6, No. 1 (2013) 1–143

DOI: 10.1561/2200000045

now
the essence of knowledge

Backward Simulation Methods for Monte Carlo Statistical Inference

Fredrik Lindsten[1] and Thomas B. Schön[2]

[1] *Division of Automatic Control, Linköping University, Linköping, 581 83, Sweden, lindsten@isy.liu.se*
[2] *Division of Automatic Control, Linköping University, Linköping, 581 83, Sweden, schon@isy.liu.se*

Abstract

Monte Carlo methods, in particular those based on Markov chains and on interacting particle systems, are by now tools that are routinely used in machine learning. These methods have had a profound impact on statistical inference in a wide range of application areas where probabilistic models are used. Moreover, there are many algorithms in machine learning which are based on the idea of processing the data sequentially, first in the forward direction and then in the backward direction. In this tutorial, we will review a branch of Monte Carlo methods based on the forward–backward idea, referred to as backward simulators. These methods are useful for learning and inference in probabilistic models containing latent stochastic processes. The theory and practice of backward simulation algorithms have undergone a significant development in recent years and the algorithms keep finding new applications. The foundation for these methods is sequential Monte Carlo (SMC). SMC-based backward simulators are capable of addressing smoothing problems in sequential latent variable models, such as

general, nonlinear/non-Gaussian state-space models (SSMs). However, we will also clearly show that the underlying backward simulation idea is by no means restricted to SSMs. Furthermore, backward simulation plays an important role in recent developments of Markov chain Monte Carlo (MCMC) methods. Particle MCMC is a systematic way of using SMC within MCMC. In this framework, backward simulation gives us a way to significantly improve the performance of the samplers. We review and discuss several related backward-simulation-based methods for state inference as well as learning of static parameters, both using a frequentistic and a Bayesian approach.

1

Introduction

A basic strategy to address many inferential problems in machine learning is to process data sequentially, first in the forward direction and then in the backward direction. Examples of this approach are the well-known forward–backward algorithm for hidden Markov models (HMMs) and the Rauch–Tung–Striebel smoother [119] for linear Gaussian state-space models. Moreover, two decades of research on sequential Monte Carlo and Markov chain Monte Carlo have enabled inference in increasingly more challenging models. Many developments have been made in order to make use of the forward–backward idea together with these Monte Carlo methods, providing inferential techniques collectively referred to as backward simulation. This tutorial provides a unifying view of these methods. In this introductory section we review some relevant background materials and also derive a first backward simulator for the special case of linear Gaussian state-space models.

1.1 Background and Motivation

For over half a century, Monte Carlo methods have been recognized as potent tools for statistical inference in complex probabilistic

models; see [103] for an early discussion. A continuous development and refinement of these methods have enabled inference in increasingly more challenging models. A key milestone in this development was the introduction of Markov chain Monte Carlo (MCMC) methods through the inventions of the Metropolis–Hastings algorithm [71, 102] and the Gibbs sampler [58]. Parallel to this, sequential importance sampling [70] and sampling/importance resampling [122] laid the foundation of sequential Monte Carlo (SMC). In its modern form, SMC was first introduced in [64, 129]. During the 1990s, several independent developments were made by, among others, [77, 83]. Recently, SMC and MCMC have been combined in a systematic manner through the developments of pseudo-marginal methods [6, 11] and particle MCMC [3].

Backward simulation is a strategy which is useful as a Monte Carlo method for learning of probabilistic models containing latent stochastic processes. In particular, we will consider inference in dynamical systems, i.e., systems that evolve over time. Dynamical systems play a central role in a wide range of scientific fields, such as signal processing, automatic control, epidemiology and econometrics, to mention a few.

One of the most widely used models of a dynamical system is the state-space model (SSM), reviewed in more detail in Sections 1.4–1.6. The structure of an SSM can be seen as influenced by the notion of a physical system. At each time t, the system is assumed to be in a certain state x_t. The state contains all relevant information about the system, i.e., if we would know the state of the system we would have full insight into its internal condition. However, the state is typically not known. Instead, we measure some quantity y_t which depend on the state in some way. Given a sequence of observations $y_{1:T} \triangleq (y_1, \ldots, y_T)$, we seek to draw inference about the latent state process $x_{1:T}$ (state inference), as well as about unknown static parameters of the model (parameter inference).

The class of SSMs will play a central role in this tutorial. Indeed, many of the inferential methods that we will review have been developed explicitly for SSMs. However, as will become apparent in Sections 4 and 5, most of the methods are more general and can be used for learning interesting models outside the class of SSMs.

Backward simulation is based on the forward–backward idea. That is, the data is processed first in the forward direction and then in the backward direction. 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 $y_{1:T}$, which can then be used for Monte Carlo integration. State smoothing is of key relevance, e.g., to obtain refined state estimates in offline settings. Furthermore, it lies at the core of many parameter inference methods (see Section 1.5) and it can be used to address problems in optimal control (see Section 4.1).

Backward simulation is also useful in MCMC, as a way of grouping variables to improve the mixing of the sampler. A common way to construct an MCMC sampler for an SSM is to sample the state variables x_t, for different t, one at a time (referred to as single-state sampling). However, since the states are often strongly dependent across time, this can lead to poor performance. Backward simulation provides a mean of grouping the state variables and sampling the entire trajectory $x_{1:T}$ as one entity. As we will illustrate in Section 1.3, this can lead to a considerable improvement upon the single-state sampler.

In Section 1.7 we will derive a first backward simulator for the class of linear Gaussian state-space (LGSS) models. Apart from LGSS models, exact backward simulation is tractable, basically only for finite state-space HMMs (see also Section 4.1.1). The main focus in this tutorial will be on models outside these restricted classes, for which exact backward simulation is not possible. Instead, we will make use of SMC (and MCMC) to enable backward simulation in challenging probabilistic models, such as nonlinear/non-Gaussian SSMs, as well as more general non-Markovian latent variable models.

1.2 Notation and Definitions

For any sequence $\{x_k\}_{k\in\mathbb{N}}$ and integers $m \leq n$ we write $x_{m:n} \triangleq (x_m, \ldots, x_n)$. We let $\wedge$ be the minimum operator, i.e., $a \wedge b \triangleq \min(a, b)$.

For a matrix A, the matrix transpose is written as A^{T}. For two probability distributions μ_1 and μ_2, the total variation distance is given by $\|\mu_1 - \mu_2\|_{\mathrm{TV}} \triangleq \sup_A |\mu_1(A) - \mu_2(A)|$. A Dirac point-mass located at some point x' is denoted as $\delta_{x'}(dx)$. We write $X \sim \mu$ to mean that the random variable X is either distributed according to μ, or sampled from μ. The uniform probability distribution on the interval $[a, b]$ is written as $\mathcal{U}([a, b])$. $\mathrm{Cat}(\{p_i\}_{i=1}^n)$, with $\sum_{i=1}^n p_i = 1$, is the categorical (i.e., discrete) probability distribution on the set $\{1, \ldots, n\}$, with probabilities $\{p_i\}_{i=1}^n$. Finally, $\mathcal{N}(m, \Sigma)$ and $\mathcal{N}(x; m, \Sigma)$ are the Gaussian (i.e., normal) probability distribution and density function, respectively, with mean vector m, covariance matrix Σ and argument x.

1.3 A Preview Example

Before we continue with this section on background theory, we consider an example to illustrate the potential benefit of using backward simulation. A simple stochastic volatility SSM is given by,

$$x_{t+1} = ax_t + v_t, \qquad v_t \sim \mathcal{N}(0, q), \tag{1.1a}$$

$$y_t = e_t \exp\left(\tfrac{1}{2} x_t\right), \qquad e_t \sim \mathcal{N}(0, 1), \tag{1.1b}$$

where the state process $\{x_t\}_{t \geq 1}$ is latent and observations are made only via the measurement process $\{y_t\}_{t \geq 1}$. Similar models have been used to generalize the Black–Scholes option pricing equation to allow for the variance to change over time [27, 101]. The same model was used by [30] to illustrate the poor mixing of a single-state Gibbs sampler; an example which is replicated here.

For simplicity, we assume that the parameters $a = 0.99$ and $q = 0.01$ are known. We seek the density $p(x_{1:T} \mid y_{1:T})$, i.e., the conditional density of the state process $x_{1:T}$ given a sequence of observations $y_{1:T}$ for some fixed final time point T. This conditional density is referred to as the joint smoothing density (JSD). For the model under study, the JSD is not available in closed form due to the nonlinear measurement Equation (1.1b). To remedy this, we construct an MCMC method to approximately sample from it. MCMC will be reviewed in more detail in Section 2.2. However, the basic idea is to simulate a Markov chain which is constructed in such a way that it admits the target

distribution as limiting distribution. The sample path from the Markov chain can then be used to draw inference about the target density $p(x_{1:T} \mid y_{1:T})$.

As an initial attempt, we try a single-state Gibbs sampler. That is, we sample each state x_t conditionally on $\{x_{1:t-1}, x_{t+1:T}\}$ (and the observations $y_{1:T}$). At each iteration of the Gibbs sampler we thus simulate according to,

$$\begin{aligned} x'_1 &\sim p(x_1 \mid x_{2:T}, y_{1:T}); \\ &\vdots \\ x'_t &\sim p(x_t \mid x'_{1:t-1}, x_{t+1:T}, y_{1:T}); \\ &\vdots \\ x'_T &\sim p(x_T \mid x'_{1:T-1}, y_{1:T}). \end{aligned}$$

This procedure will leave $p(x_{1:T} \mid y_{1:T})$ invariant (see Section 2.2 for more on Gibbs sampling) and it results in a valid MCMC sampler. The conditional densities $p(x_t \mid x_{1:t-1}, x_{t+1:T}, y_{1:T})$ are not available in closed form. However, for this model (Equation (1.1)), they are log-concave and we can employ the efficient rejection sampling strategy by [145] to sample exactly from these distributions.

The single-state Gibbs sampler will indeed converge to samples from $p(x_{1:T} \mid y_{1:T})$. However, it is well recognized that single-state samplers can suffer from poor mixing, due to the often strong dependencies between consecutive state variables. That is, the convergence can be slow in the sense that we need to iterate the above sampling scheme a large number of times to get reliable samples.

To analyze this, we generate $T = 100$ samples from the model (Equation (1.1)) and run the Gibbs sampler for 100 000 iterations (in each iteration, we loop over all the state variables for $t = 1, \ldots, T$). The first 10 000 iterations are discarded, to avoid transient effects. We then compute the empirical autocorrelation function (ACF) of the state x_{50}, which is given in Figure 1.1. As can be seen, the ACF decreases very slowly, indicating a poorly mixing Gibbs kernel. This simply reflects the fact that, when the state variables are highly correlated, the single-state sampler will be inefficient at exploring the state-space. This is a

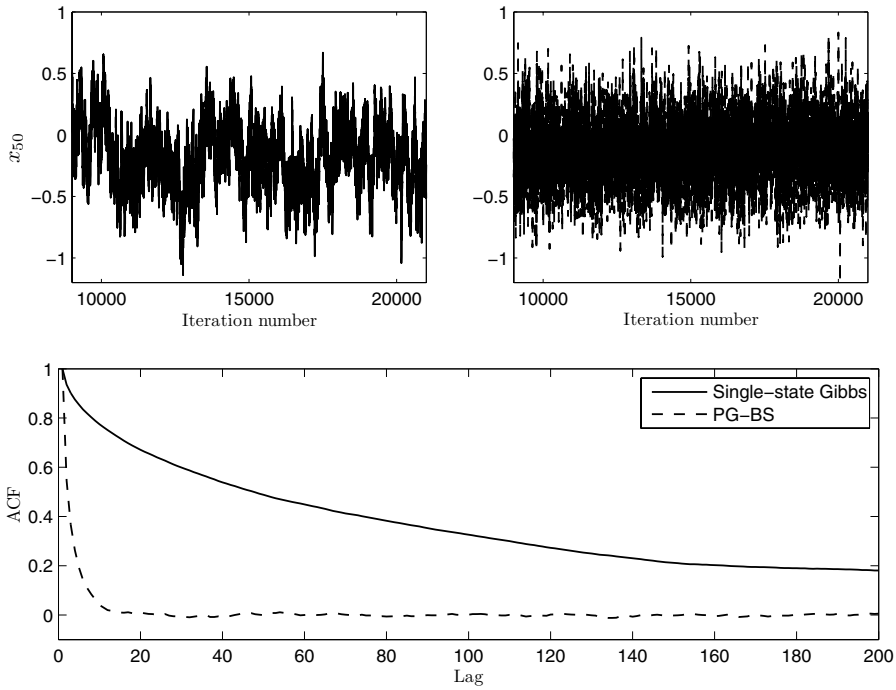

Fig. 1.1 (Top left) Part of sample path for the single-state Gibbs sampler; (Top right) Part of sample path for PGBS; (Bottom) Empirical ACF for x_{50} for the single-state Gibbs sampler and for PGBS using $N = 15$ particles.

common and well-recognized problem when addressing the state inference problem for SSMs.

One way to remedy this is to group the variables and sample a full state trajectory $x_{1:T}$ jointly. This is what a backward simulator aims to accomplish. Grouping variables in a Gibbs sampler will in general improve upon the mixing of the single-state sampler [97, Section 6.7], and in practice the improvement can be quite considerable.

To illustrate this, we have included the ACF for a backward-simulation-based method in Figure 1.1. Since the model (Equation (1.1)) is nonlinear, exact backward simulation is not possible. Instead, the results reported here are from a backward simulator based on SMC, using (only) $N = 15$ particles. The specific method that we have used is denoted as particle Gibbs with backward simulation (PGBS), and it will be discussed in detail in Section 5.4. For the PGBS,

the ACF drops off much more rapidly, indicating a more efficient sampler. Furthermore, a key property of PGBS is that, despite the fact that it relies on a crude SMC approximation, it does not alter the stationary distribution of the Gibbs sampler, nor does it introduce any additional bias. That is, PGBS will, just as the single-state Gibbs sampler, target the exact JSD $p(x_{1:T} \mid y_{1:T})$. This property is known as *exact approximation*, a concept that we will return to in Section 5.

1.4 State-Space Models

State-space models (SSMs) are commonly used to model time series and dynamical systems. Additionally, many models that are not sequential "by nature" can also be written on state-space form. It is a comprehensive and important class of models, and it serves as a good starting point for introducing the concepts that will be discussed throughout this tutorial.

We consider here discrete-time SSMs on a general state-space X. The system state is a Markov process $\{x_t\}_{t\geq 1}$ on X, evolving according to a Markov transition kernel $F(dx_{t+1} \mid x_t)$ and with initial distribution $\nu(dx_1)$. The state x_t is assumed to summarize all relevant information about the system at time t. However, the state process is latent and it is observed only implicitly through the observations $\{y_t\}_{t\geq 1}$, taking values in some set Y. Given x_t, the measurement y_t is conditionally independent of past and future states and observations, and it is distributed according to a kernel $G(dy_t \mid x_t)$. A graphical model, illustrating the conditional dependencies in an SSM, is given in Figure 1.2.

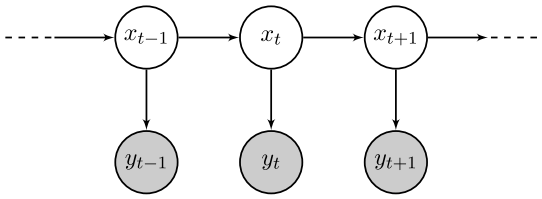

Fig. 1.2 Graphical model of an SSM. The white nodes represent latent variables and the gray nodes represent observed variables.

We shall assume that the observation kernel G admits a probability density g w.r.t. some dominating measure, which we simply denote dy. Such models are referred to as partially dominated. If, in addition, the transition kernel F admits a density f and the initial distribution ν admits a density μ, both w.r.t. some dominating measure dx, the model is called fully dominated. In summary, a fully dominated SSM can be expressed as,

$$x_{t+1} \sim f(x_{t+1} \mid x_t), \tag{1.2a}$$

$$y_t \sim g(y_t \mid x_t), \tag{1.2b}$$

and $x_1 \sim \mu(x_1)$. Two examples of SSMs follow below.

Example 1.1 (Finite state-space hidden Markov model). A finite state-space HMM, or simply HMM, is an SSM with $\mathsf{X} = \{1, \ldots, K\}$ for some finite K. The transition density (w.r.t. counting measure) can be summarized in a $K \times K$ transition matrix Π, where the (i,j)th entry is given by,

$$\Pi_{i,j} = P(x_{t+1} = j \mid x_t = i) = f(j \mid i).$$

Hence, $f(j \mid i)$ denotes the probability of moving from state i at time t, to state j at time $t+1$.

Example 1.2 (Additive noise model). In engineering applications, SSMs are often expressed on functional form with additive noise,

$$x_{t+1} = a(x_t) + v_t,$$
$$y_t = c(x_t) + e_t,$$

for some functions a and c. Here, the noises v_t and e_t are commonly referred to as process noise and measurement noise, respectively. If the noise distributions admit densities w.r.t. dominating measures, then the model is fully dominated. The transition density is then given by $f(x_{t+1} \mid x_t) = p_{v_t}(x_{t+1} - a(x_t))$ and similarly for the observation density.

Throughout this tutorial, we will mostly be concerned with fully dominated SSMs and therefore do most of our derivations in terms of probability densities. There are, however, several examples of interesting models that are *degenerate*, i.e., that are not fully dominated. We will return to this in the sequel and discuss how it affects the methods presented in here.

1.5 Parameter Learning in SSMs

The basic inference problem for SSMs is typically that of state inference, i.e., to infer the latent states given measurements from the system. In fact, even when the actual task is to learn a model of the system dynamics, state inference tends to play a crucial role as an intermediate step of the learning algorithm. To illustrate this, assume that the SSM (Equation (1.2)) is parameterized by some unknown parameter $\theta \in \Theta$,

$$x_{t+1} \sim f_\theta(x_{t+1} \mid x_t), \tag{1.3a}$$

$$y_t \sim g_\theta(y_t \mid x_t), \tag{1.3b}$$

and $x_1 \sim \mu_\theta(x_1)$. Given a batch of measurements $y_{1:T}$, we wish to draw inference about θ. In the Bayesian setting, a prior distribution $\pi(\theta)$ is assigned to the parameter and the learning problem amounts to computing the posterior distribution $p(\theta \mid y_{1:T})$.

A complicating factor is that the likelihood $p(y_{1:T} \mid \theta)$ in general cannot be computed in closed form. To address this difficulty, it is common to make use of *data augmentation* [136, 132]. That is, we target the joint state and parameter posterior $p(\theta, x_{1:T} \mid y_{1:T})$, rather than the marginal posterior $p(\theta \mid y_{1:T})$. The latent states are thus viewed as auxiliary variables. This opens up for using Gibbs sampling (see Section 2.2), for instance by initializing $\theta[0] \in \Theta$ and iterating;

(i) Draw $x_{1:T}[r] \sim p(x_{1:T} \mid \theta[r-1], y_{1:T})$;
(ii) Draw $\theta[r] \sim p(\theta \mid x_{1:T}[r], y_{1:T})$.

Under weak assumptions, this procedure will generate a Markov chain $\{\theta[r], x_{1:T}[r]\}_{r \geq 1}$ with stationary distribution $p(\theta, x_{1:T} \mid y_{1:T})$. Consequently, the stationary distribution of the subchain $\{\theta[r]\}_{r \geq 1}$ will be the

marginal parameter posterior distribution $p(\theta \mid y_{1:T})$. Note that Step (i) of the above sampling scheme requires the computation of the JSD, for a fixed value of the parameter θ. That is, we need to address an intermediate smoothing problem in order to implement this Gibbs sampler.

Data augmentation is commonly used also in the frequentistic setting. Assume that we, instead of the posterior distribution, seek the maximum likelihood estimator (MLE),

$$\widehat{\theta}_{\mathrm{ML}} = \arg\max_{\theta \in \Theta} \log p_\theta(y_{1:T}), \tag{1.4}$$

where $p_\theta(y_{1:T})$ is the likelihood of the observed data for a given value of the system parameter θ. Again, since the log-likelihood $\log p_\theta(y_{1:T})$ is not available in closed form, direct maximization in Equation (1.4) is problematic. Instead, we can make use of the expectation maximization (EM) algorithm [33] (see also [100]). The EM algorithm is an iterative method, which maximizes $p_\theta(y_{1:T})$ by iteratively maximizing an auxiliary quantity,

$$Q(\theta, \theta') = \int \log p_\theta(x_{1:T}, y_{1:T}) p_{\theta'}(x_{1:T} \mid y_{1:T})\, dx_{1:T}. \tag{1.5}$$

The EM algorithm is useful when maximization of $\theta \mapsto Q(\theta, \theta')$, for fixed θ', is simpler than direct maximization of the log-likelihood, $\theta \mapsto \log p_\theta(y_{1:T})$. The procedure is initialized at some $\theta[0] \in \Theta$ and then iterates between two steps, expectation (E) and maximization (M);

(E) Compute $Q(\theta, \theta[r-1])$;
(M) Compute $\theta[r] = \arg\max_{\theta \in \Theta} Q(\theta, \theta[r-1])$.

The resulting sequence $\{\theta[r]\}_{r \geq 0}$ will, under weak assumptions, converge to a stationary point of the likelihood $p_\theta(y_{1:T})$ [148].

Using the conditional independence properties of an SSM, we can write the complete data log-likelihood as

$$\begin{aligned} &\log p_\theta(x_{1:T}, y_{1:T}) \\ &\quad = \log \mu_\theta(x_1) + \sum_{t=1}^{T} \log g_\theta(y_t \mid x_t) + \sum_{t=1}^{T-1} \log f_\theta(x_{t+1} \mid x_t). \end{aligned} \tag{1.6}$$

From Equation (1.5), we note that the auxiliary quantity is defined as the expectation of expression (1.6) under the JSD. Hence, to carry out the E-step of the EM algorithm, we again need to address an intermediate smoothing problem for fixed values of the system parameters.

1.6 Smoothing Recursions

As noted above, the JSD is a quantity of central interest for learning and inference problems in SSMs. It summarizes all the information about the latent states which is available in the observations. Many densities that arise in various state inference problems are given as marginals of the JSD. There are a few that are of particular interest, which we summarize in Table 1.1. To avoid a cluttered notation, we now drop the (possible) dependence on an unknown parameter θ from the notation and write the JSD as $p(x_{1:T} \mid y_{1:T})$.

As in Equation (1.6), the conditional independence properties of an SSM implies that the complete data likelihood can be written as,

$$p(x_{1:T}, y_{1:T}) = \mu(x_1) \prod_{t=1}^{T} g(y_t \mid x_t) \prod_{t=1}^{T-1} f(x_{t+1} \mid x_t). \tag{1.7}$$

The JSD is related to the above expression by Bayes' rule,

$$p(x_{1:T} \mid y_{1:T}) = \frac{p(x_{1:T}, y_{1:T})}{\int p(x_{1:T}, y_{1:T})\, dx_{1:T}}. \tag{1.8}$$

Despite the simplicity of this expression, it is of limited use in practice due to the high-dimensional integration needed to compute the normalization factor in the denominator. Instead, most practical methods

Table 1.1 Filtering and smoothing densities of particular interest.

	Density
Filtering[a]	$p(x_t \mid y_{1:t})$
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]	$p(x_{t-\ell+1:t} \mid y_{1:t})$

[a] The filtering and fixed-lag smoothing densities are marginals of the JSD at time t, $p(x_{1:t} \mid y_{1:t})$.

(and in particular the ones discussed in this tutorial) are based on a recursive evaluation of the JSD.

Again by using Bayes' rule, we get the following two-step procedure,

$$p(x_{1:t} \mid y_{1:t}) = \frac{g(y_t \mid x_t)p(x_{1:t} \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}, \tag{1.9a}$$

$$p(x_{1:t+1} \mid y_{1:t}) = f(x_{t+1} \mid x_t)p(x_{1:t} \mid y_{1:t}). \tag{1.9b}$$

The above equations will be denoted as the forward recursion for the JSD, since they evolve forward in time. Step (1.9a) is often referred to as the measurement update, since the current measurement y_t is taken into account. Step (1.9b) is known as the time update, moving the density forward in time, from t to $t + 1$.

An interesting fact about SSMs is that, conditioned on $y_{1:T}$, the state process $\{x_t\}_{t=1}^T$ is an inhomogeneous Markov process. Under weak assumptions (see [23, Section 3.3.2] for details), the same holds true for the time-reversed chain, starting at time T and evolving backward in time according to the so-called backward kernel,

$$B_t(A \mid x_{t+1}) \triangleq P(x_t \in A \mid x_{t+1}, y_{1:T}). \tag{1.10}$$

Note that the backward kernel is time inhomogeneous. In the general case, it is not possible to give an explicit expression for the backward kernel. However, for a fully dominated model, this can always be done, and its density is given by

$$p(x_t \mid x_{t+1}, y_{1:T}) = \frac{f(x_{t+1} \mid x_t)p(x_t \mid y_{1:t})}{\int f(x_{t+1} \mid x_t)p(x_t \mid y_{1:t})\,dx_t}. \tag{1.11}$$

From the conditional independence properties of an SSM, it also holds that $p(x_t \mid x_{t+1}, y_{1:T}) = p(x_t \mid x_{t+1}, y_{1:t})$.

Using the backward kernel, we get an alternative recursion for the JSD, evolving backward in time,

$$p(x_{t:T} \mid y_{1:T}) = p(x_t \mid x_{t+1}, y_{1:t})p(x_{t+1:T} \mid y_{1:T}), \tag{1.12}$$

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

The backward kernel density at time t depends only on the transition density $f(x_{t+1} \mid x_t)$ 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 (Equation (1.12)) for computing the JSD, the filtering densities must first be computed for $t = 1, \ldots, T$. Consequently, this procedure is generally called forward filtering/backward smoothing.

1.7 Backward Simulation in Linear Gaussian SSMs

An important special case of Equation (1.2) is the class of linear Gaussian state-space models. A functional representation of an LGSS model is given by,

$$x_{t+1} = Ax_t + v_t, \qquad v_t \sim \mathcal{N}(0, Q), \tag{1.13a}$$

$$y_t = Cx_t + e_t, \qquad e_t \sim \mathcal{N}(0, R). \tag{1.13b}$$

Here, y_t is an n_y-dimensional vector of observations, x_t is an n_x-dimensional state vector and the system matrices A and C are of appropriate dimensions. The process and measurement noises are multivariate Gaussian with zero means and covariances Q and R, respectively.

Example 1.3 (Partially or fully dominated SSM). Assume that the measurement noise covariance R in Equation (1.13b) is full rank. Then, the observation kernel is Gaussian and dominated by Lebesgue measure. Hence, the model is partially dominated. If, in addition, the process noise covariance Q in Equation (1.13a) is full rank, then the transition kernel is also Gaussian and dominated by Lebesgue measure. In this case, the model is fully dominated.

However, for singular Q the model is degenerate (i.e., not fully dominated). Rank deficient process noise covariances arise in many applications, for instance if there is a physical connection between some of the states (such as between position and velocity).

A fully dominated LGSS model can equivalently be expressed as in Equation (1.2) with,

$$f(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1}; Ax_t, Q), \tag{1.14a}$$

$$g(y_t \mid x_t) = \mathcal{N}(y_t; Cx_t, R). \tag{1.14b}$$

LGSS models are without doubt one of the most important and well-studied classes of SSMs. There are basically two reasons for this. First, LGSS models provide sufficiently accurate descriptions of many interesting dynamical systems. Second, LGSS models are one of the few model classes, simple enough to allow for a fully analytical treatment.

When addressing inferential problems for SSMs, we are often asked to generate samples from the JSD, typically as part of an MCMC sampler used to learn a model of the system dynamics, as discussed above. For an LGSS model, the JSD is Gaussian and it can be computed using Kalman filtering and smoothing techniques (see e.g., [80]). Hence, we can make use of standard results for Gaussian distributions to generate a sample from $p(x_{1:T} \mid y_{1:T})$. This is possible for small T, but for increasing T it soon becomes infeasible due to the large matrix inversions involved.

To address this issue, it was recognized by [24, 56] that we can instead use the backward recursion (Equation (1.12)). It follows that the JSD can be factorized as,

$$p(x_{1:T} \mid y_{1:T}) = \left(\prod_{t=1}^{T-1} p(x_t \mid x_{t+1}, y_{1:t}) \right) p(x_T \mid y_{1:T}). \tag{1.15}$$

Initially, we generate a sample from the filtering density at time T,

$$\widetilde{x}_T \sim p(x_T \mid y_{1:T}). \tag{1.16a}$$

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

$$\widetilde{x}_t \sim p(x_t \mid \widetilde{x}_{t+1}, y_{1:t}), \tag{1.16b}$$

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 (Equation (1.15)).

To compute the backward kernel, we first run a forward filter to find the filtering densities $p(x_t \mid y_{1:t})$ for $t = 1, \ldots, T$. For an LGSS model, this is done by a standard Kalman filter [81]. It follows that the filtering densities are Gaussian according to,

$$p(x_t \mid y_{1:t}) = \mathcal{N}(x_t; \hat{x}_{t|t}, P_{t|t}), \tag{1.17}$$

for some tractable sequences of mean vectors $\{\hat{x}_{t|t}\}_{t\geq 1}$ and covariance matrices $\{P_{t|t}\}_{t\geq 1}$, respectively. From Equation (1.14a), we note that the transition density function is Gaussian and affine in x_t. Using Equations (1.11) and (1.17) and standard results on affine transformations of Gaussian variables, it then follows that

$$p(x_t \mid x_{t+1}, y_{1:t}) = \mathcal{N}(x_t; \mu_t, M_t), \tag{1.18a}$$

with

$$\mu_t = \hat{x}_{t|t} + P_{t|t}A^{\mathsf{T}}(Q + AP_{t|t}A^{\mathsf{T}})^{-1}(x_{t+1} - A\hat{x}_{t|t}), \tag{1.18b}$$

$$M_t = P_{t|t} - P_{t|t}A^{\mathsf{T}}(Q + AP_{t|t}A^{\mathsf{T}})^{-1}AP_{t|t}. \tag{1.18c}$$

Note that, if more than one sample is desired, multiple backward trajectories can be generated independently, without having to rerun the forward Kalman filter. We illustrate the backward simulator in the example below.

Example 1.4. To illustrate the possibility of generating samples from the JSD using backward simulation, we consider a first-order LGSS model,

$$\begin{aligned} x_{t+1} &= 0.9x_t + v_t, & v_t &\sim \mathcal{N}(0, 0.1), \\ y_t &= x_t + e_t, & e_t &\sim \mathcal{N}(0, 1), \end{aligned}$$

and $x_1 \sim \mathcal{N}(x_1; 0, 10)$. 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 by running a Kalman filter followed by a Rauch–Tung–Striebel smoother [119]. 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 $\{\widetilde{x}^j_{1:T}\}_{j=1}^M$, by first running a Kalman filter and then repeating the backward simulation procedure given by Equations (1.16) and (1.18) M times. Histograms over the simulated states at three specific time points, $t = 1$, $t = 25$ and $t = 50$, are given in Figure 1.3. As expected, the histograms are in close agreement with the true marginal smoothing distributions.

The strategy given by Equation (1.16), 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

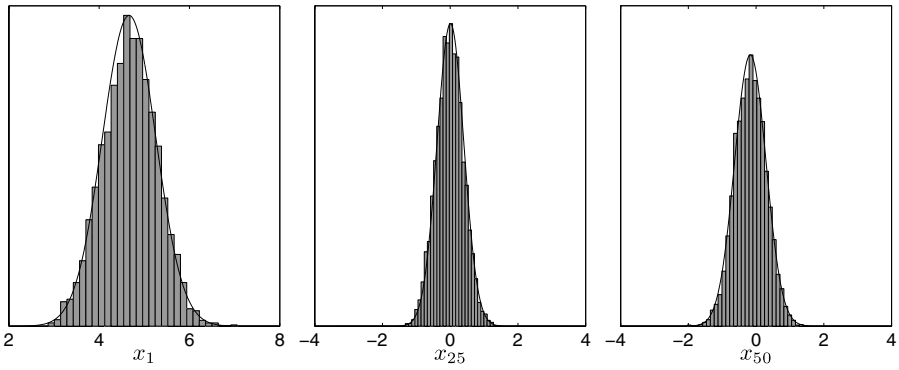

Fig. 1.3 Histograms of $\{\widetilde{x}_t^j\}_{j=1}^M$ for $t = 1$, $t = 25$ and $t = 50$ (from left to right). The true marginal smoothing densities $p(x_t \mid y_{1:T})$ are shown as black lines.

backward simulation. We will now leave the world of LGSS models. In the remainder of this tutorial we address backward simulation for general nonlinear/non-Gaussian models. In these cases, the backward kernels will in general not be available in closed form. Instead, we will rely on SMC approximations of the kernels to carry out the backward simulation.

Before we leave this section, it should be noted that the backward simulator for LGSS models derived here is provided primarily to illustrate the concept. For LGSS models, more efficient samplers exist, e.g., based on disturbance simulation. See [30, 47, 146] for further details and extensions.

1.8 Outline

The rest of this tutorial is organized as follows. Section 2 reviews the two main Monte Carlo methods that are used throughout SMC and MCMC. The section is self-contained, but for obvious reasons it does not provide an in-depth coverage of these methods. Several references which may be useful for readers with no background in this area are given in Section 2.

Section 3 addresses SMC-based backward simulation for SSMs. The focus in this section is on smoothing in general nonlinear/non-Gaussian SSMs. More precisely, we discuss algorithms for generating

state trajectories, approximately distributed according to the joint smoothing distribution. These algorithms can be categorized as *particle smoothers*. Hence, readers with particular interest in smoothing problems may want to focus their attention on this section. However, smoothing is also addressed in Section 5 (see in particular Section 5.7), and the methods presented there can be useful alternatives to the particle smoothers discussed in Section 3.

Section 4 generalizes the backward simulation idea to latent variable models outside the class of SSMs. A general backward simulator is introduced and we discuss its properties and the type of models for which it is applicable. As a special case of the general backward simulator, we derive a Rao–Blackwellized particle smoother for conditionally linear Gaussian SSMs.

In Section 5, we discuss backward simulation in the context of so-called particle MCMC (PMCMC) methods. The focus in this section is on parameter inference, primarily in the Bayesian setting, but we also discuss PMCMC for maximum-likelihood-based inference. As mentioned above, the smoothing problem is also addressed. Finally, in Section 6 we conclude with a discussion about the various methods reviewed throughout this tutorial and outline possible directions for future work.

2

Monte Carlo Preliminaries

In this section, we review the two main Monte Carlo tools on which the subsequent development is based; sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC).

2.1 Sequential Monte Carlo

For general nonlinear non-Gaussian SSMs the backward kernel (Equation (1.10)) is not available on closed form. To address this issue, the basic idea that will be employed is to use an SMC approximation of the backward kernel. In this section, we review the basics of SMC applied to SSMs, also referred to as particle filtering.

Since the focus of this tutorial is on backward simulation, and not SMC, we will only present a basic SMC sampler and discuss some of its important properties. For a more in-depth treatment, we refer to one of the many tutorials and textbooks on the topic, see e.g., [10, 22, 38, 41, 66, 126]. A comprehensive collection of convergence results for SMC can be found in [31]. It should be emphasized that the backward simulators that are derived in the subsequent sections indeed rely on SMC approximations of the backward kernels, but not directly

on how these approximations are generated. Hence, it is straightforward to replace the basic sampler presented here with a more advanced SMC method, if desired.

2.1.1 The Particle Filter

SMC methods are based on importance sampling and resampling techniques to sequentially sample from a sequence of target densities. They can be seen as combinations of the sequential importance sampling [70] and the sampling/importance resampling [122] algorithms. 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 $p(x_{1:t} \mid y_{1:t})$ for $t = 1, 2, \ldots$.

In a standard importance sampler, targeting $p(x_{1:t} \mid y_{1:t})$, we generate N independent samples $\{x^i_{1:t}\}_{i=1}^N$ from some proposal density $r_t(x_{1:t} \mid y_{1:t})$. These samples are then assigned importance weights,

$$w^i_t = \frac{p(x^i_{1:t} \mid y_{1:t})}{r_t(x^i_{1:t} \mid y_{1:t})}, \tag{2.1}$$

for $i = 1, \ldots, N$. The weights can only be computed up to proportionality, since the normalization constant for the target density is unknown. However, this is easily coped with by normalizing the weights to sum to one. That is, we first compute the unnormalized importance weights

$$\bar{w}^i_t = \frac{p(x^i_{1:t}, y_{1:t})}{r_t(x^i_{1:t} \mid y_{1:t})}, \tag{2.2}$$

and then set $w^i_t = \bar{w}^i_t / \sum_l \bar{w}^l_t$.

Now, to allow for a sequential method, we construct the proposal density so that it factorizes according to,

$$\begin{aligned} r_t(x_{1:t} \mid y_{1:t}) &= r_t(x_t \mid x_{t-1}, y_t) r_{t-1}(x_{1:t-1} \mid y_{1:t-1}) \\ &= r_1(x_1 \mid y_1) \prod_{s=2}^{t} r_s(x_s \mid x_{s-1}, y_s). \end{aligned} \tag{2.3}$$

Hence, we can draw a sample from the proposal $x^i_{1:t} \sim r_t(x_{1:t} \mid y_{1:t})$ by first generating $x^i_1 \sim r_1(x_1 \mid y_1)$, then $x^i_2 \sim r_2(x_2 \mid x^i_1, y_2)$, etc. This

implies that the importance weights are given by,

$$w_t^i = \frac{p(x_{1:t}^i \mid y_{1:t})}{r_t(x_{1:t}^i \mid y_{1:t})} \propto \frac{g(y_t \mid x_t^i) f(x_t^i \mid x_{t-1}^i)}{r_t(x_t^i \mid x_{t-1}^i, y_t)} \frac{p(x_{1:t-1}^i \mid y_{1:t-1})}{r_{t-1}(x_{1:t-1}^i \mid y_{1:t-1})}. \quad (2.4)$$

Hence, if we define the weight function

$$W_t(x_{t-1}, x_t; y_t) = \frac{g(y_t \mid x_t) f(x_t \mid x_{t-1})}{r_t(x_t \mid x_{t-1}, y_t)}, \quad (2.5)$$

the importance weights can be computed as

$$\bar{w}_t^i = W_t(x_{t-1}^i, x_t^i; y_t) w_{t-1}^i, \quad (2.6)$$

and $w_t^i = \bar{w}_t^i / \sum_l \bar{w}_t^l$. We thus obtain a sequential updating formula also for the importance weights. The initial weight function at time $t = 1$ is given by $W_1(x_1; y_1) = g(y_1 \mid x_1) p(x_1) / r_1(x_1 \mid y_1)$. As indicated by the notation, the proposal at time t is allowed to depend on both the previous state x_{t-1} and on the current observation y_t. This is important in practice, to be able to make good use of the available information when designing the proposal.

In the SMC literature, the samples $\{x_{1:t}^i\}_{i=1}^N$ are called particles and the collection $\{x_{1:t}^i, w_t^i\}_{i=1}^N$ is referred to as a *weighted particle system*. This weighted sample defines an empirical point-mass distribution on X^t,

$$\widehat{p}^N(dx_{1:t} \mid y_{1:t}) \triangleq \sum_{i=1}^N w_t^i \delta_{x_{1:t}^i}(dx_{1:t}), \quad (2.7)$$

which is an approximation of the target distribution. Equivalently, for any test function $\varphi : \mathsf{X}^T \to \mathbb{R}$ we can construct the estimator

$$\widehat{\varphi}_{t|t}^N \triangleq \sum_{i=1}^N w_t^i \varphi(x_{1:t}^i) \approx \mathbb{E}[\varphi(x_{1:t}) \mid y_{1:t}]. \quad (2.8)$$

This estimator is consistent, as $N \to \infty$, and a central limit theorem holds (see e.g., [31, 87, 36]).

The procedure resulting from sampling according to Equation (2.3) and computing the weights (Equation (2.6)) is the sequential importance sampler [70], dating back to the late 60's. Though applicable for short data lengths, a serious drawback with this method is that the

weight update (Equation (2.6)) is unstable. More precisely, the variance of the normalized importance weights increases over time [23, p. 232], typically at an exponential rate. 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). This does not come as a surprise, since we are in fact applying an importance sampler in a space with dimension increasing with t. In high dimensions, even what appears to be a small discrepancy between the proposal and the target densities will result in poor performance of the sampler (see e.g., [41] for an illustrative example).

A way to mitigate this problem was proposed by [129, 64], resulting in the first functional SMC sampler. The idea is to rejuvenate the sample by replicating particles with high weights and discarding particles with low weights. This is done by resampling the particle system, similar to the sampling/importance resampling method [122]. Since the weighted particle system $\{x_{1:t}^i, w_t^i\}_{i=1}^N$ approximates the target according to Equation (2.7), we can generate a new, unweighted set of particles by sampling independently from Equation (2.7). That is, we set $\check{x}_{1:t}^j = x_{1:t}^i$ with probability w_t^i, for $j = 1, \ldots, N$. The equally weighted particle system $\{\check{x}_{1:t}^j, \frac{1}{N}\}_{j=1}^N$ can then be used to construct an estimator similar to Equation (2.8),

$$\check{\varphi}_{t|t}^N \triangleq \frac{1}{N} \sum_{j=1}^{N} \varphi(\check{x}_{1:t}^j) \approx \mathbb{E}[\varphi(x_{1:t}) \mid y_{1:t}]. \tag{2.9}$$

It should be noted that the resampling introduces some additional variance, so the estimator in Equation (2.9) will be dominated by the estimator in Equation (2.8). However, when applied sequentially, the resampling is critical since it allows us to put focus on the promising particles and discard the improbable ones. We summarize the PF in Algorithm 1.

A common choice in practice is to run the PF with $r_t(x_t \mid x_{t-1}, y_t) = f(x_t \mid x_{t-1})$, i.e., we propose samples according to the transition density function. This results in a vanilla method referred to as the *bootstrap filter*. If possible, it is recommended to use a proposal density which takes the current observation y_t into account. The *optimal proposal* is given by the conditional density $r_t(x_t \mid x_{t-1}, y_t) = p(x_t \mid x_{t-1}, y_t)$. This

Algorithm 1 Particle filter (all operations are for $i = 1, \ldots, N$)

1: Draw $x_1^i \sim r_1(x_1 \mid y_1)$.
2: Compute $\bar{w}_1^i = W_1(x_1^i; y_1)$.
3: Normalize: set $w_1^i = \bar{w}_1^i / \sum_l \bar{w}_1^l$.
4: **for** $t \geq 2$ **do**
5: Resample with $P(\check{x}_{t-1}^i = x_{t-1}^j) = w_{t-1}^j$.
6: Draw $x_t^i \sim r_t(x_t \mid \check{x}_{t-1}^i, y_t)$ and set $x_{1:t}^i = \{\check{x}_{1:t-1}^i, x_t^i\}$.
7: Compute $\bar{w}_t^i = W_t(x_t^i, \check{x}_{t-1}^i; y_t)$.
8: Normalize: set $w_t^i = \bar{w}_t^i / \sum_l \bar{w}_t^l$.
9: **end for**

choice of proposal density will minimize the variance of the incremental importance weights at time t [41]. Unfortunately, it is often the case that this density is not available in practice, but various approximations can be used instead.

The resampling scheme outlined above is known as multinomial resampling. Alternative methods that introduce less variance exist, e.g., residual resampling [144, 98] and stratified resampling [83]. Furthermore, instead of resampling at every iteration, it is recommended to do so only when there is need for it. For this cause, it is common to introduce a measure of imbalance of the importance weights, such as the effective sample size (ESS) [97, Section 2.5]. Resampling is then only carried out when the ESS drops below some prespecified threshold.

Another modification of the resampling step is used in the auxiliary particle filter (APF) by [116]. The idea is to use an adjusted proposal distribution in the resampling step. This is accomplished by computing a set of weights $\{w_t^i \nu_t^i\}_{i=1}^N$, and using these (after normalization) as probabilities in the resampling. Here, the adjustment weights ν_t^i are user defined. They allow us to, for instance, take the observation y_{t+1} into account when resampling the particles at time t. The fact that we modify the proposal for the resampling is compensated for in the importance weight computation. If the APF is run with an optimal proposal kernel and with adjustment weights given by $\nu_t^i = p(y_{t+1} \mid x_t^i)$, the importance weights will be identically equal to one. In this case, the filter is said to be *fully adapted*. See [116, 79, 37] for more information about the APF and how the adjustment weights can be chosen.

2.1.2 Path Degeneracy

As pointed out in the previous section, the PF targets the sequence of JSDs, $p(x_{1:t} \mid y_{1:t})$. By keeping track of the *genealogy* of the particle filter, i.e., the ancestral dependence of the particles, we obtain weighted particle trajectories $\{x^i_{1:t}, w^i_t\}_{i=1}^N$. These provide an approximation, not only of the filtering distribution, but of the complete joint smoothing distribution as in Equation (2.7) [83]. It thus appears as if the PF solves the general state inference problem, since any smoothing density can be attained from $p(x_{1:t} \mid y_{1:t})$ by marginalization. However, this is not truly the case.

To see why, let s be some fixed time point and assume that we apply a PF to approximate the marginal smoothing density $p(x_s \mid y_{1:t})$ for $t \geq s$. This is given straightforwardly from Equation (2.7) by simply discarding everything except the sth time point from the particle trajectories $\{x^i_{1:t}\}_{i=1}^N$. Now, if $t = s$ (i.e., we are in fact looking at the filtering density) we have N unique particles in the point-mass approximation. However, each time we resample the particle system, the unique number of particles at time s will decrease. This is in fact the purpose of the resampling, to remove particles with small weights and duplicate particles with large weights. Consequently, for large enough $t \gg s$, all the particles $\{x^i_{1:t}\}_{i=1}^N$ will share a common ancestor at time s, due to the consecutive resampling steps. This problem, known as *path degeneracy*, is further illustrated in the example below.

Example 2.1 (Path degeneracy). A PF with $N = 30$ particles is applied to a one-dimensional Gaussian random walk process measured in Gaussian noise. At time $t = 50$ the JSD is targeted by a weighted particle system $\{x^i_{1:50}, w^i_{50}\}_{i=1}^{30}$. Figure 2.1 depicts the particle trajectories. As can be seen, all particles share a common ancestor at time $s = 32$.

Assume, for instance, that we are interested in the smoothed estimate of the initial state, $\mathbb{E}[x_1 \mid y_{1:50}]$. We thus construct an estimator,

$$\hat{x}_{1|50} = \sum_{i=1}^{30} w^i_{50} x^i_1,$$

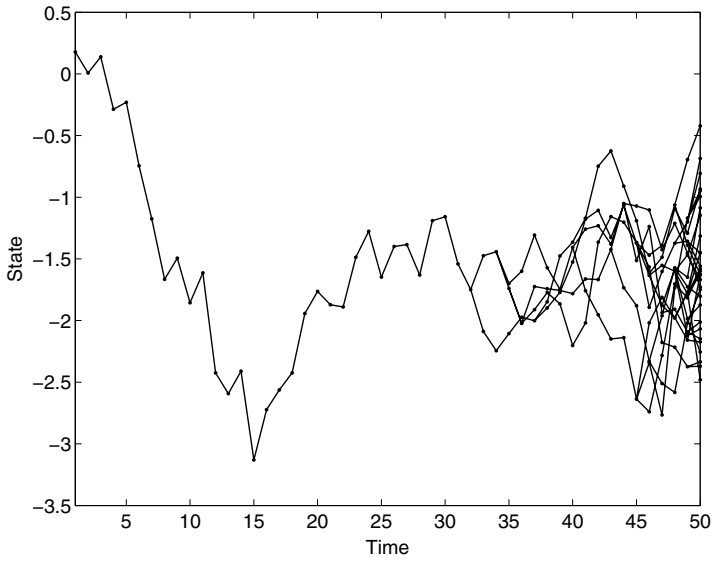

Fig. 2.1 Particle trajectories at time $t = 50$. For any time $s \leq 32$, all trajectories coincide.

but since x_1^i are identical for all $i = 1, \ldots, 30$, this is in effect a Monte Carlo integration using a single sample. Hence, due to the path degeneracy, we cannot expect to obtain an accurate estimate of $\mathbb{E}[x_1 \mid y_{1:50}]$ from the PF.

Path degeneracy arises as an effect of resampling, but it should rather be thought of as a manifestation of a deeper problem, namely the degeneracy of the importance weights. In the sequential importance sampler, all weight will be put on a single sample for large enough t, having a similar effect as path degeneracy. This issue is in fact to a large extent mitigated by introducing resampling.

Due to path degeneracy, the PF is in practice used mostly for filtering (hence the name) and fixed-lag smoothing. At time t or $s \lesssim t$, the particle diversity is in general high, leading to accurate approximations of the filtering and fixed-lag smoothing distributions. How SMC can be used to address the marginal and joint smoothing problems will be the topic of Section 3.

2.2 Markov Chain Monte Carlo

Another comprehensive class of Monte Carlo methods is MCMC. In this section, we review some of the basic concepts of MCMC that are of key relevance. For further reading, see the tutorial papers [2, 128] or one of the many textbooks, e.g., [20, 97, 120].

2.2.1 Markov Chains and Limiting Distributions

MCMC methods allow us to approximately generate samples from an arbitrary target distribution $f(z)$ on some space Z. More precisely, an MCMC sampler will simulate a Markov chain, or Markov process, $\{z[r]\}_{r\geq 0}$ which is constructed in such a way that the limiting distribution of the chain is given by $f(z)$. The sample path of the Markov chain can then be used to draw inference about the target distribution. For instance, the target can be the JSD $p(x_{1:T} \mid y_{1:T})$, which means that MCMC can be used as an alternative to SMC for addressing the state inference problem. However, we will primarily make use of MCMC in a different way, either as a component of SMC or vice versa. Hence, in this section, the methods are presented in a general setting, which is specialized to different scenarios in the consecutive sections.

A Markov chain $\{z[r]\}_{r\geq 0}$ is completely specified by its initial distribution $\mu(z)$ and its transition kernel $K(z' \mid z)$ (for simplicity, we assume that there exists a dominating measure and express all distributions in terms of their probability densities). Here, $K(z' \mid z)$ encodes the probability density for the consecutive state of the chain, denoted as z', given that the current state is z. That is, the chain can be simulated according to $z[0] \sim \mu(z)$ and $z[r] \sim K(z' \mid z[r-1])$ for $r = 1, 2, \ldots$. As an example, the state process $\{x_t\}_{t\geq 1}$ of the SSM (Equation (1.2)) is a Markov chain. However, the view on this process is quite different, as $\{x_t\}_{t\geq 1}$ is thought of as the internal state of a physical system evolving over time. On the contrary, the Markov chain $\{z[r]\}_{r\geq 0}$ generated in an MCMC sampler is simulated as part of an inferential method, and it does not have a physical interpretation in the same sense.

The Markov chains encountered in MCMC have, by construction, a special stability property, namely that of a *stationary distribution*. That is, there exist some distribution g such that, if $z[r]$ is marginally

distributed according to g, then so is $z[r+1]$ (and therefore all $z[m]$ for $m \geq r$). If g is a stationary distribution for the Markov kernel K, then we say that K leaves g *invariant*. Furthermore, if the marginals of the chain approach the stationary distribution (in total variation), it is referred to as the *limiting distribution* of the chain. An important consequence of this convergence property is that the sample path average,

$$\frac{1}{R}\sum_{r=0}^{R}\varphi(z[r]), \tag{2.10}$$

converges almost surely to the expectation $\mathbb{E}_g[\varphi(z)] = \int \varphi(z)g(z)\,dz$. This result, known as the ergodic theorem (see e.g., [120, Section 6.6]), ensures that the Markov chain can be used to compute Monte Carlo estimators of expectations under the stationary distribution.

There are two properties of the chain that are sufficient for a stationary distribution to be also a limiting distribution; irreducibility and aperiodicity. The Markov chain is said to be *g-irreducible* if, for any initial state, it has positive probability of entering any set which has a positive probability under g. The chain is periodic if, with probability 1, certain subsets only can be visited at regularly spaced intervals. If the chain is not periodic, it is said to be *aperiodic*. For an irreducible and aperiodic chain with stationary distribution g, the marginals converge to the stationary distribution.

Theorem 2.1. Let $\{z[r]\}_{r\geq 0}$ be a g-irreducible and aperiodic Markov chain with stationary distribution g. Then, for g-almost all starting points,

$$\|\mathcal{L}(z[r] \in \cdot) - g(\cdot)\|_{\mathrm{TV}} \to 0,$$

as $r \to \infty$, where $\mathcal{L}(z[r] \in \cdot)$ is the marginal distribution of $z[r]$ and $\|\cdot\|_{\mathrm{TV}}$ is the total variation norm.

Proof. See [134, Theorem 1]. □

Stronger forms of convergence are also common in the MCMC literature, such as geometric ergodicity and uniform ergodicity. It is

also possible to establish central limit theorems for the estimator in Equation (2.10). There is by now a well-developed theory on Markov chains, safeguarding the theoretical validity of MCMC methods, see e.g., [104, 120, 134].

2.2.2 Metropolis–Hastings Algorithm

A sample path from a Markov chain with a specific limiting distribution can be used to compute estimators according to Equation (2.10). However, since we are interested in computing expectations under a given target distribution $f(z)$, it remains to construct the chain so that $f(z)$ is its limiting distribution. To accomplish this, we have freedom in the choice of transition kernel $K(z' \mid z)$, as long as the kernel is chosen in such a way that it is possible to simulate the Markov chain.

A property which is sufficient for $f(z)$ to be a stationary distribution of the chain is that the kernel satisfies *detailed balance*,

$$f(z)K(z' \mid z) = f(z')K(z \mid z') \quad \text{for } (z, z') \in \mathsf{Z}^2. \tag{2.11}$$

Based on this criterion, it is possible to construct a Markov kernel with the correct stationary distribution using an accept/reject procedure. This results in the Metropolis–Hastings (MH) algorithm [102, 71], which is the main tool in the MCMC toolbox.

The idea behind the MH algorithm is to generate samples from some arbitrary proposal kernel $q(z' \mid z)$. The simulation is followed by an accept/reject decision to make sure that detailed balance holds for the given target distribution. The MH acceptance probability, for a move from z to z', is given by

$$\rho(z, z') = 1 \wedge \frac{f(z')}{f(z)} \frac{q(z \mid z')}{q(z' \mid z)}, \tag{2.12}$$

where $a \wedge b = \min(a, b)$. That is, with $z[r-1]$ being the previous state of the Markov chain at iteration r, we proceed by sampling a candidate value $z' \sim q(z' \mid z[r-1])$. With probability $\rho(z[r-1], z')$, the sample is accepted and we set $z[r] = z'$. Otherwise, the sample is rejected and we set $z[r] = z[r-1]$. It is interesting to note that the acceptance probability depends on the target density only through the ratio $f(z')/f(z)$.

Algorithm 2 Metropolis–Hastings sampler

1: Set $z[0]$ arbitrarily.
2: **for** $r \geq 1$ **do**
3: Draw $z' \sim q(z' \mid z[r-1])$.
4: With probability

$$\rho(z[r-1], z') = 1 \wedge \frac{f(z')}{f(z[r-1])} \frac{q(z[r-1] \mid z')}{q(z' \mid z[r-1])},$$

set $z[r] = z'$, otherwise set $z[r] = z[r-1]$.
5: **end for**

Hence, it is possible to run the MH algorithm even when the normalizing constant of the target density is unknown, which is key to its applicability in practice. The MH sampler is summarized in Algorithm 2.

The transition kernel given by the sampling procedure above is,

$$K(dz' \mid z) = \rho(z, z')q(dz' \mid z) + (1 - \eta(z))\delta_z(dz'), \tag{2.13}$$

with $\eta(z) = \int \rho(z, z')q(z' \mid z)\, dz'$. It can be verified that this kernel satisfies the detailed balance condition (Equation (2.11)). Consequently, $f(z)$ is a stationary distribution of the chain. Under additional, mild, assumptions (basically, positivity of the proposal kernel), it can also be verified that the chain is irreducible and aperiodic, which by Theorem 2.1 implies that $f(z)$ is the limiting distribution of the Markov chain.

In practice, the choice of the proposal kernel is critical. If the proposal is not chosen appropriately, the average acceptance probability can be very low. This results in that the chain gets "stuck" at the same value for many iterations. We say that the chain suffers from poor *mixing*. Furthermore, a high acceptance probability does not necessarily imply good mixing. Indeed, a proposal kernel which only makes small, local moves can yield a high acceptance probability, but at the same time a very poor exploration of the state-space. To construct a proposal kernel with good mixing properties is particularly challenging in high dimensions. To address this issue, sophisticated proposal mechanisms have been developed for the MH algorithm. One approach is to construct a proposal kernel based on the simulation of

a Hamiltonian dynamical system, leading to the so-called Hamiltonian Monte Carlo method [44] (see also [109]). See [61, 78, 124] for applications and extensions of this method. Another approach is to make use of SMC within MCMC, which is the idea that underlies the particle MCMC framework [3]. These methods will be discussed in more detail in Section 5.

2.2.3 Gibbs Sampling

The Gibbs sampler [58, 57] is an MCMC method of particular interest. It can be seen as a special case of the MH algorithm. However, it differs both in terms of basic methodology and historical motivation. Assume that the random variable of interest can be partitioned according to $z = \{z_1, \ldots, z_p\}$, where the z_is can be either uni- or multidimensional. A Markov kernel is then constructed by sampling each component from its full conditional under f, i.e.,

$$z_i' \mid z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_p \sim f_i(z_i \mid z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_p). \quad (2.14)$$

The sampling procedure given above leaves $f(z)$ invariant, i.e., if z is distributed according to f, then so is $\{z_1, \ldots, z_{i-1}, z_i' z_{i+1}, \ldots, z_p\}$. Hence, sampling each of the components in this manner, either in a deterministic or in a random order, defines a Markov kernel on Z, with stationary distribution $f(z)$. Since only the conditionals $f_1, \ldots, f_p$ are used in the sampling procedure, all of the simulations can be univariate (or low-dimensional), even for high-dimensional problems, which is a key property of the Gibbs sampler. We summarize the method in Algorithm 3.

The specific parameterization that is used in the construction of a Gibbs sampler can have very significant effects on its performance, see e.g., [72, 121]. If the individual components $z_1, \ldots, z_p$ are strongly dependent, sampling from the full conditionals can lead to insignificant updates of the components, resulting in a poorly mixing Gibbs sampler. An example of this phenomena, which is of particular interest to the present work, is single-state Gibbs sampling for SSMs. Assume that the target distribution is the JSD $p(x_{1:T} \mid y_{1:T})$. A natural parameterization for a Gibbs sampler is to draw the individual states at different time

Algorithm 3 Gibbs sampler (deterministic scan)

1: Set $z[0]$ arbitrarily.
2: **for** $r \geq 1$ **do**
3: Sample according to

$$
\begin{aligned}
z_1[r] &\sim f_1(z_1 \mid z_2[r-1], \ldots, z_p[r-1]). \\
&\vdots \\
z_i[r] &\sim f_i(z_i \mid z_1[r], \ldots, z_{i-1}[r], z_{i+1}[r-1], \ldots, z_p[r-1]). \\
&\vdots \\
z_p[r] &\sim f_p(z_p \mid z_1[r], \ldots, z_{p-1}[r]).
\end{aligned}
$$

4: **end for**

points from their full conditional, i.e.,

$$x_t' \sim p(x_t \mid x_{1:t-1}, x_{t+1:T}, y_{1:T}), \tag{2.15}$$

for $t = 1, \ldots, T$. This leads to the single-state sampler which was used in the illustrating example in Section 1.3. As was experienced in this example, the resulting Gibbs sampler can suffer from very poor mixing. The reason is that there is often a strong dependence between consecutive states in an SSM. Hence, when x_t' is sampled, the resulting value will be in close agreement with the conditioning on x_{t-1} and x_{t+1}, resulting in small updates at each iteration.

Two important techniques, used to improve the mixing of the Gibbs sampler, are *grouping* and *collapsing*. To illustrate these concepts, assume that the variable of interest is partitioned into three parts, $z = \{z_1, z_2, z_3\}$. The basic Gibbs sampler would then, at each iteration, simulate according to

$$z_1' \sim f_1(z_1 \mid z_2, z_3), \qquad z_2' \sim f_2(z_2 \mid z_1', z_3), \qquad z_3' \sim f_3(z_3 \mid z_1', z_2').$$

Grouping, or blocking, amounts to sample a group of variable, say $\{z_2, z_3\}$, jointly. That is, we simulate according to

$$z_1' \sim f_1(z_1 \mid z_2, z_3), \quad \{z_2', z_3'\} \sim f_{2,3}(z_2, z_3 \mid z_1').$$

This corresponds to a reparameterization of the Gibbs sampler, with only two components instead of three. If z_2 and z_3 are strongly dependent, sampling them jointly can result in much larger updates, improving the mixing of the basic Gibbs sampler.

Collapsing is another word for marginalization. In the simplest case, one component of the model, say z_3, is marginalized in all the steps of the sampler. That is, we simulate according to

$$z_1' \sim \bar{f}_1(z_1 \mid z_2), \quad z_2' \sim \bar{f}_2(z_2 \mid z_1'),$$

where $\bar{f}_i$, $i = 1, 2$, are the full conditionals under the marginal distribution $\bar{f}(z_1, z_2) = \int f(z_1, z_2, z_3)\, dz_3$. This corresponds to a standard Gibbs sampler, targeting the marginal distribution $\bar{f}$ instead of f.

Perhaps more interestingly, the Gibbs sampler does not have to be fully collapsed, as above, to attain an improvement in mixing. In a *partially collapsed* Gibbs sampler, we marginalize a certain variable only in some of the Gibbs steps. For instance, we could simulate according to

$$z_1' \sim f_1(z_1 \mid z_2, z_3), \quad z_2' \sim \bar{f}_2(z_2 \mid z_1'), \quad z_3' \sim f_3(z_3 \mid z_1', z_2').$$

This partially collapsed Gibbs sampler results in a valid MCMC method, with stationary distribution $f(z_1, z_2, z_3)$. Again, if z_2 and z_3 are strongly dependent, sampling z_2 unconditionally on z_3 enables a larger update, which in turn results in a larger update also of z_3.[1]

However, care needs to be taken when constructing a partially collapsed sampler, since the order of the Gibbs steps in fact can affect the stationary distribution. If, for instance, we would collapse all the Gibbs steps to a maximum degree and sample each of the variables z_1, z_2 and z_3 from their respective marginals under f, then we would lose all correlation between the variables. A more intricate example of an improperly collapsed sampler is,

$$z_1' \sim \bar{f}_1(z_1 \mid z_2), \quad z_2' \sim f_2(z_2 \mid z_1', z_3), \quad z_3' \sim f_3(z_3 \mid z_1', z_2').$$

This sampling scheme is seemingly similar to the one above, but this latter scheme does not leave $f(z_1, z_2, z_3)$ invariant. Again, the reason is

[1] In fact, in this specific example, the partially collapsed Gibbs sampler is equivalent to the grouped Gibbs sampler.

that the sampler does not take the dependences between the variables into account in a correct way. To see why this is the case and how to construct a *properly collapsed* Gibbs sampler, see [137].

Grouping and collapsing will in general improve the mixing properties of the Gibbs sampler (see [97, Section 6.7]). In practice, the gain can be quite considerable. In the motivating example in Section 1.3, we pointed out that a backward simulator aims at sampling the state sequence $x_{1:T}$ jointly from the JSD, instead of resorting to single-state sampling of the individual state components. In light of the above discussion, we see that the backward simulator thus is a way of grouping the state variables to alleviate the strong dependencies between consecutive states.

A different technique which can also be useful is to reparameterize the model by making a change of variables. If the new variables are chosen in a way which decreases the posterior interdependence, then the mixing of the Gibbs sampler can be improved; see [114] and Section 4.6.3.

To be able to implement the Gibbs samplers described above, we need to be able to sample from all the involved conditionals under the target distribution f. While this is indeed possible for many interesting problems, it is not always the case. To alleviate this, one possibility is to use a mixed strategy, in which Gibbs steps are used whenever possible and MH steps are used for the intractable conditionals. Since the MH sampler leaves its target distribution invariant, each step of the mixed sampler will leave f invariant, resulting in a valid MCMC kernel. This approach is sometimes referred to as *Hastings-within-Gibbs*.

3

Backward Simulation for State-Space Models

SMC provides an approximation of the joint smoothing distribution, given by Equation (2.7). However, as discussed in Section 2.1.2, path degeneracy makes this approximation unreliable for anything but filtering or fixed-lag smoothing with a short enough lag. In the present section, we will see how the joint smoothing problem can be addressed by complementing the forward filter with a second recursion, evolving in the time-reversed direction. In particular, we will make use of SMC to enable backward simulation for general SSMs, allowing us to generate samples approximately distributed according to the JSD.

A related approach is the two-filter smoother [17], which is based on one filter moving forward in time and one filter moving backward in time. When the two filters meet "in the middle", the information is merged, enabling computation of smoothed estimates. This approach will not be considered here, and for further reading on SMC implementations of the two-filter smoother we refer to [18, 51].

3.1 Forward Filter/Backward Simulator

As was recognized in Section 1.7, it is possible to make use of the backward recursion (Equation (1.12)) to generate samples from the JSD. The key ingredient is the backward kernel. Let us assume that the model under study is fully dominated. The backward kernel density is then given by Equation (1.11). As previously pointed out, this expression depends explicitly on the filtering density $p(x_t \mid y_{1:t})$. The basic idea, underlying the particle-based forward filter/backward simulator (FFBSi) [62, 40], is to make use of a PF to approximate the backward kernel.

Assume that we have recorded a sequence of observations $y_{1:T}$ up to some final time point T. Assume also that we have applied a PF to this batch of data. For each time $t = 1, \ldots, T$ we thus have a weighted particle system $\{x_t^i, w_t^i\}_{i=1}^N$ approximating the filtering distribution at time t,

$$\widehat{p}^N(dx_t \mid y_{1:t}) \triangleq \sum_{i=1}^N w_t^i \delta_{x_t^i}(dx_t). \tag{3.1}$$

These approximations are given by marginalization of Equation (2.7), or, put differently, by discarding everything from the particle trajectories except for the last time point. Using Equation (3.1) in Equation (1.11), we obtain an approximation of the backward kernel,

$$\widehat{B}_t^N(dx_t \mid x_{t+1}) \triangleq \sum_{i=1}^N \frac{w_t^i f(x_{t+1} \mid x_t^i)}{\sum_l w_t^l f(x_{t+1} \mid x_t^l)} \delta_{x_t^i}(dx_t). \tag{3.2}$$

It is important to note that the particles $\{x_t^i, w_t^i\}_{i=1}^N$ that are used to approximate the filtering distribution in Equation (3.1) are those that were generated at time t in the PF. Hence, there is in general a rich diversity of particles in this system, and the approximations in Equations (3.1) and (3.2) do not suffer (directly) from path degeneracy. This is also the reason for why FFBSi can succeed where the PF fails. Even though the PF results in a degenerate approximation of the JSD, it can generally provide accurate approximations of the filtering distributions, which is all that is needed to compute an approximation of the backward kernel.

We can now make use of Equations (3.1) and (3.2) to generate a backward trajectory by sampling (cf. Equation (1.16)),

$$\widetilde{x}_T \sim \widehat{p}^N(dx_T \mid y_{1:T}), \tag{3.3a}$$

$$\widetilde{x}_t \sim \widehat{B}_t^N(dx_t \mid \widetilde{x}_{t+1}), \tag{3.3b}$$

for $t = T-1, \ldots, 1$. The backward trajectory $\widetilde{x}_{1:T}$ is an approximate realization from $p(x_{1:T} \mid y_{1:T})$. How close the distribution of $\widetilde{x}_{1:T}$ is to the JSD, clearly depends on how accurate the PF approximations in Equations (3.1) and (3.2) are. The convergence properties of FFBSi will be the topic of Section 3.2.2.

Sampling according to Equation (3.3a) simply consists of drawing among the filter particles at time T with $P(\widetilde{x}_T = x_T^i) = w_T^i$. Similarly, the empirical backward kernel has finite support and it can be written as

$$\widehat{B}_t^N(dx_t \mid \widetilde{x}_{t+1}) = \sum_{i=1}^{N} \widetilde{w}_{t|T}^i \delta_{x_t^i}(dx_t), \tag{3.4}$$

where we have defined the smoothing weights

$$\widetilde{w}_{t|T}^i = \frac{w_t^i f(\widetilde{x}_{t+1} \mid x_t^i)}{\sum_l w_t^l f(\widetilde{x}_{t+1} \mid x_t^l)}. \tag{3.5}$$

Hence, to sample according to Equation (3.3b), we evaluate the smoothing weights $\{\widetilde{w}_{t|T}^i\}_{i=1}^N$ and draw among the forward filter particles at time t with $P(\widetilde{x}_t = x_t^i) = \widetilde{w}_{t|T}^i$. Note that, in general, $\widetilde{x}_t$ will differ from the ancestor particle of $\widetilde{x}_{t+1}$, resulting in an increased particle diversity.

We emphasize that Equation (3.2) relies on the assumption that the model under study is fully dominated. Applying backward simulation to degenerate (non fully dominated) models, turns out to be much more tricky. We will return to this issue in Section 4.

When using FFBSi to address the smoothing problem, we typically generate multiple backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$, by repeating the backward simulation (Equation (3.3)) M times. Hence, conditionally on the forward filter particles, the collection $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$ are i.i.d. samples. The conditional distribution of these samples will be given explicitly in Section 3.2.1. The backward trajectories define an unweighted

Algorithm 4 FFBSi [62]

Input: Forward filter particle systems $\{x_t^i, w_t^i\}_{i=1}^N$ for $t = 1, \ldots, T$.
Output: Backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$.

1: Sample independently $\{b_T(j)\}_{j=1}^M \sim \mathrm{Cat}\left(\{w_T^i\}_{i=1}^N\right)$.
2: Set $\widetilde{x}_T^j = x_T^{b_T(j)}$ for $j = 1, \ldots, M$.
3: **for** $t = T-1$ **to** 1 **do**
4: **for** $j = 1$ **to** M **do**
5: Compute $\widetilde{w}_{t|T}^{i,j} \propto w_t^i f(\widetilde{x}_{t+1}^j \mid x_t^i)$ for $i = 1, \ldots, N$.
6: Normalize the smoothing weights $\{\widetilde{w}_{t|T}^{i,j}\}_{i=1}^N$ to sum to one.
7: Draw $b_t(j) \sim \mathrm{Cat}\left(\{\widetilde{w}_{t|T}^{i,j}\}_{i=1}^N\right)$.
8: Set $\widetilde{x}_t^j = x_t^{b_t(j)}$ and $\widetilde{x}_{t:T}^j = \{\widetilde{x}_t^j, \widetilde{x}_{t+1:T}^j\}$.
9: **end for**
10: **end for**

point-mass approximation of the joint smoothing distribution,

$$\widetilde{p}^M(dx_{1:T} \mid y_{1:T}) \triangleq \frac{1}{M} \sum_{j=1}^M \delta_{\widetilde{x}_{1:T}^j}(dx_{1:T}). \tag{3.6}$$

Clearly, this approximation can also be used to approximate any marginal or fixed-interval smoothing distribution.

We summarize the FFBSi in Algorithm 4, and illustrate the backward simulation procedure in Example 3.1 below.

Example 3.1 (Backward simulation). We illustrate the backward simulation process on a toy example. In Figure 3.1, we show the particle trajectories generated by a forward PF in a one-dimensional problem. The dots show the particle positions for the $N = 4$ particles over $T = 5$ time steps and their sizes represent the particle weights. The dots are connected, to illustrate the ancestral dependence of the particles. All particles at time $t = 5$ share a common ancestor at time $t = 3$, i.e., the particle paths are degenerate.

In Figure 3.2 we illustrate the simulation of one backward trajectory. In the upper left plot, the backward trajectory is initialized by sampling

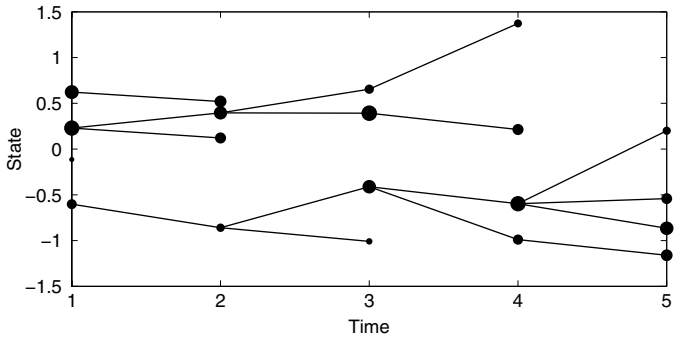

Fig. 3.1 Particle trajectories for $N = 4$ particles over $T = 5$ time steps after a completed forward filtering pass. The sizes of the dots represent the particle weights.

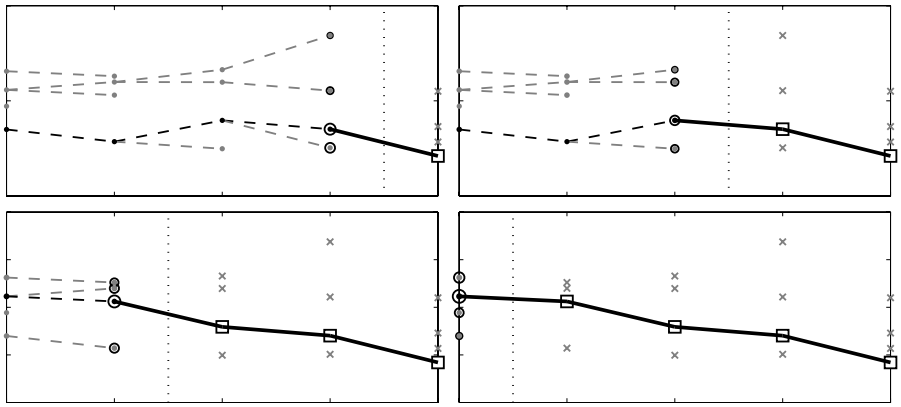

Fig. 3.2 Backward simulation of a single backward trajectory. See the text for details.

from the forward filter particles at time $t = 5$. The probability of sampling a particle x_T^i is given by its importance weight w_T^i. The initialized backward trajectory is shown as a square. The particle weights at $t = 4$ are thereafter recomputed according to Equation (3.5). The smoothing weights $\{\widetilde{w}_{t|T}^i\}_{i=1}^N$ are shown as circles, whereas the filter weights are illustrated with dots. Another particle is then drawn and appended to the backward trajectory. In the upper right and lower left plots, the trajectory is augmented with new particles at $t = 3$ and $t = 2$, respectively. Finally, in the lower right plot, a final particle is appended at $t = 1$, forming a complete backward trajectory $\widetilde{x}_{1:5}$. Observe that the generated backward trajectory differs from the ancestral line of the forward

filter particle as shown in Figure 3.1. The procedure can be repeated as many times as needed (using the same forward filter particles), to generate a collection of backward trajectories.

The backward trajectories are conditionally i.i.d. and the particle diversity among the trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$ will thus in general be larger than that among the forward filter trajectories $\{x_{1:T}^i\}_{i=1}^N$, since the latter suffer from path degeneracy. However, an interesting question to ask is the following. Assume that we generate only a single backward trajectory $\widetilde{x}_{1:T}$. Assume also that we extract a single trajectory $x_{1:T}^k$ from the PF, by sampling once from the empirical distribution (Equation (2.7)) at time T. Then, will the distribution of $\widetilde{x}_{1:T}$ be closer to the true JSD, than the distribution of $x_{1:T}^k$?

The answer to this question is not obvious in the general case. Intuitively, we might think that the answer is *yes*, since $\widetilde{x}_{1:T}$ is sampled by a forward/backward smoothing procedure, whereas $x_{1:T}^k$ is sampled by a forward-only procedure. However, at least when the underlying SMC sampler is a bootstrap PF, this turns out not to be the case, as is stated in the following proposition due to [113].

Proposition 3.1. Assume that the weighted particle system $\{x_{1:T}^i, w_T^i\}_{i=1}^N$ is generated by a bootstrap particle filter. Let k be sampled with $P(k=i) = w_T^i$ and let $\widetilde{x}_{1:T}$ be generated by a backward simulator. Then, $P(x_{1:T}^k \in A) = P(\widetilde{x}_{1:T} \in A)$ for any set A.

Proof. See [113, Proposition 5]. □

Hence, the backward simulation procedure should be thought of as a way to *reduce the correlation* between the degenerate forward filter trajectories, rather than as a way to actually improve the quality of the individual trajectories.

As can be seen in Algorithm 4 (page 38), to generate M backward trajectories, the smoothing weights need to be computed for index i ranging from 1 to N and index j ranging from 1 to M. Hence, the computational complexity of the algorithm is $O(NM)$. The number of

backward trajectories M, generated by Algorithm 4, is arbitrary. However, to obtain accurate approximations of the smoothing distributions of interest, it is clear that M needs to be fairly large. The computational complexity of Algorithm 4 may therefore be prohibitive. One option is to make use of parallel architectures, e.g., graphics processing units. Algorithm 4, and in fact most backward simulators, is particularly well suited for parallel implementation, since the backward trajectories can be generated independently. In Section 3.3, we will discuss alternative methods for reducing the computational complexity of the algorithm.

The FFBSi is illustrated in the example below, which also provides some additional insight into how to reason about the design parameters N and M.

Example 3.2 (FFBSi). We consider a standard nonlinear time-series model previously used by, among others, Andrade Netto et al. [111], Gordon et al. [64] and Godsill et al. [62]. The model is given by,

$$\begin{aligned} x_{t+1} &= 0.5x_t + 25\frac{x_t}{1+x_t^2} + 8\cos(1.2t) + v_t, \\ y_t &= 0.05x_t^2 + e_t, \end{aligned}$$

with $v_t \sim \mathcal{N}(0, \sigma_v^2)$, $e_t \sim \mathcal{N}(0, \sigma_e^2)$ and $x_1 \sim \mathcal{N}(0, 5)$. We take $\sigma_v^2 = 10$ and $\sigma_e^2 = 1$ and generate $T = 100$ observations $y_{1:T}$ from the system. The smoothing distribution for this model is distinctively bimodal when $\sigma_e < \sigma_v$, rendering state inference problematic.

To address the joint smoothing problem, we apply the FFBSi of Algorithm 4 to generate M backward trajectories. A bootstrap PF with N particles is used in the forward direction. To see how the performance is affected by different numbers of particles and backward trajectories, we run the FFBSi several times on the same data, with different values for N and M.

To evaluate the performance, we compute the posterior mean estimates $\widetilde{x}_{t|T}^{N,M} = \frac{1}{M}\sum_{j=1}^{M} \widetilde{x}_t^j$ for each smoother, for $t = 1, \ldots, T$. We then compute the root-mean-square error (RMSE) relative to the true[1]

[1] To obtain a ground truth estimate, we run an FFBSi with 100 000 forward filter particles and 50 000 backward trajectories, yielding very accurate results.

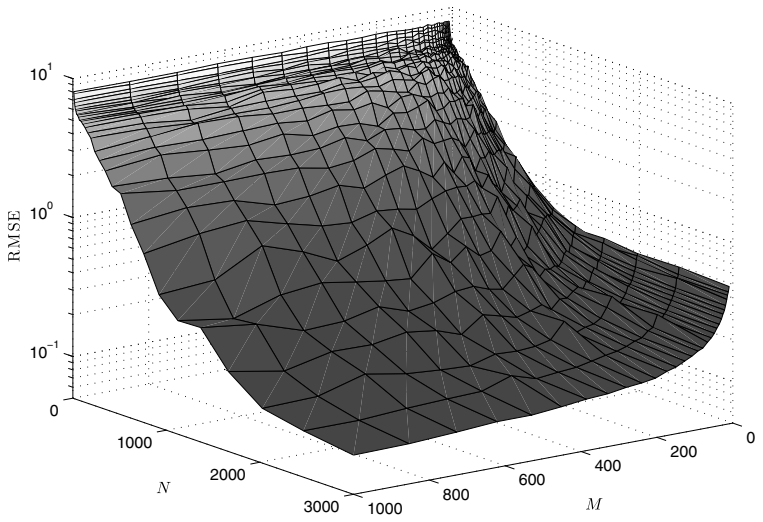

Fig. 3.3 Averaged RMSE values (log-scale), for different combinations of N and M.

posterior means,

$$\varepsilon(N, M) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\widetilde{x}_{t|T}^{N,M} - \mathbb{E}[x_t \mid y_{1:T}]\right)^2}.$$

Since the RMSE is defined relative to the true posterior mean, we expect $\varepsilon(N, M) \to 0$ as $N \to \infty$ and $M \to \infty$ (see Section 3.2.2). We run the smoothers on a grid of values of N and M, with N ranging from 10 to 3000 and with M ranging from 10 to 1000. Furthermore, we use 500 independent runs at each grid point and average the RMSE values, to reduce the effects of randomness. The results are reported in Figure 3.3.

As expected, the error is decreasing with both N and M. It is interesting to note, however, that the dependence on N is much more pronounced than the dependence on M. Even for a very modest number of backward trajectories ($M = 10$), the error can be reduced a fair bit by just increasing N. This is not the case if we instead fix $N = 10$ and only increase M.

These results can be explained by the fact that the FFBSi relies heavily on the empirical backward kernel (Equation (3.2)). Since this

only depends on the forward filter, it is clear that we need to take N large to obtain an accurate approximation of the kernel. Conditionally on the forward filter particles, Algorithm 4 generates M i.i.d. backward trajectories. If the distribution of these trajectories, which only depends on N, is "close enough" to the true JSD, a modest number of samples can be sufficient to obtain accurate posterior estimates.

The results in Figure 3.3 suggests that we can try to find an optimal trade-off between M and N for a given computational time. However, the results can vary between different models. Also, in this example, we have only considered estimating the posterior means $\mathbb{E}[x_t \mid y_{1:T}]$. For other test functions, the results may also differ. Furthermore, for the FFBSi of Algorithm 4, the computational cost scales rather predictively as MN. However, as we will see in Section 3.3, there are alternative implementations of the FFBSi with smaller computational costs, which on the other hand are harder to predict. Hence, it can be difficult to find an optimal trade-off, even if a graph such as in Figure 3.3 would be available. However, as a rule of thumb, it is better to put most effort on the forward filter to obtain accurate backward kernel approximations, i.e., to take $N > M$.

3.2 Analysis and Convergence

In this section we discuss the relationship between the FFBSi and a method referred to as forward filter/backward smoother (FFBSm). As we will see, analyzing the FFBSm is useful in order to assess convergence properties for FFBSi and to better understand the properties of the method.

3.2.1 Forward Filter/Backward Smoother

So far, we have seen how the empirical backward kernel (Equation (3.2)) can be used to simulate backward trajectories, approximately distributed according to the JSD. From these trajectories, we obtained an approximation of the joint smoothing distribution given by Equation (3.6). However, an alternative way to address the smoothing problem is to simply plug the approximation in Equation (3.2) into the

backward recursion (Equation (1.12)). Contrary to the backward simulator, this will result in a deterministic approximation of the smoothing distribution, conditionally on the forward filter particles. This method is referred to as the forward filter/backward smoother (FFBSm).

FFBSm was introduced for marginal smoothing in [39] and thus predates the FFBSi. The computational complexity for FFBSm is at least quadratic in the number of particles. For FFBSi, it is possible to obtain a lower computational cost (see Section 3.3), making it the preferred method of choice (among these two) for most models. However, the two methods are closely related and, as noted above, by studying the FFBSm we can gain additional insight into the properties of FFBSi.

We start by reviewing the marginal FFBSm by [39]. This method aims at computing the marginal smoothing densities $p(x_t \mid y_{1:T})$ for $t = 1, \ldots, T$. These densities are given recursively, backward in time, by

$$p(x_t \mid y_{1:T}) = \int p(x_t \mid x_{t+1}, y_{1:t}) p(x_{t+1} \mid y_{1:T}) \, dx_{t+1}. \tag{3.7}$$

Assume that we have obtained a weighted particle system $\{x_{t+1}^i, \omega_{t+1|T}^i\}_{i=1}^N$, targeting $p(x_{t+1} \mid y_{1:T})$. At the final time point, this is given by the PF since the filtering distribution and the marginal smoothing distribution coincide at time T, i.e., $\omega_{t|T}^i = w_T^i$ for $i = 1, \ldots, N$. Plugging the empirical distribution defined by this particle system, and the approximation of the backward kernel (Equation (3.2)), into Equation (3.7) results in

$$\begin{aligned}
\widehat{p}_{\mathrm{FFBSm}}^N(dx_t \mid y_{1:T}) &\triangleq \sum_{j=1}^N \omega_{t+1|T}^j \sum_{i=1}^N \frac{w_t^i f(x_{t+1}^j \mid x_t^i)}{\sum_l w_t^l f(x_{t+1}^j \mid x_t^l)} \delta_{x_t^i}(dx_t) \\
&= \sum_{i=1}^N \omega_{t|T}^i \delta_{x_t^i}(dx_t),
\end{aligned} \tag{3.8}$$

where we have defined the smoothing weights,

$$\omega_{t|T}^i \triangleq w_t^i \sum_{j=1}^N \omega_{t+1|T}^j \frac{f(x_{t+1}^j \mid x_t^i)}{\sum_l w_t^l f(x_{t+1}^j \mid x_t^l)}, \tag{3.9}$$

for $i = 1, \ldots, N$. The smoothing weights defined above are self-normalized, in the sense that they sum to one by construction.

We have added "FFBSm" explicitly in the notation to emphasize that the approximation of the marginal smoothing distribution given by Equation (3.8) is not the same as we would get by marginalization of the PF approximation (Equation (2.7)). Note, however, that the FFBSm reuses the forward filter particles $\{x_t^i\}_{i=1}^N$, but updates the importance weights of these particles to target the marginal smoothing distribution, rather than the filtering distribution.

The computational complexity of the marginal FFBSm is $O(N^2)$, which can be seen from the weight expression (Equation (3.9)). It is possible to apply the same approach to approximate any fixed interval distribution or the joint smoothing distribution, albeit at an increased computational cost. For the joint smoothing distribution, we make use of the following factorization of the JSD,

$$p(x_{1:T} \mid y_{1:T}) = \left(\prod_{t=1}^{T-1} p(x_t \mid x_{t+1}, y_{1:t}) \right) p(x_T \mid y_{1:T}), \tag{3.10}$$

following from Equation (1.12). By plugging the empirical filter and backward kernel approximations into this expression, we get

$$\begin{aligned} &\widehat{p}^N_{\mathrm{FFBSm}}(dx_{1:T} \mid y_{1:T}) \\ &\qquad \triangleq \sum_{i_1=1}^{N} \cdots \sum_{i_T=1}^{N} \underbrace{\left(\prod_{t=1}^{T-1} \frac{w_t^{i_t} f(x_{t+1}^{i_{t+1}} \mid x_t^{i_t})}{\sum_l w_t^l f(x_{t+1}^{i_{t+1}} \mid x_t^l)} \right) w_T^{i_T}}_{\triangleq \omega_{1:T|T}(i_1,\ldots,i_T)} \delta_{x_1^{i_1} \cdots x_T^{i_T}}(dx_{1:T}). \end{aligned} \tag{3.11}$$

The expression above defines a point-mass distribution on X^T, and the cardinality of its support is N^T. The meaning of the distribution can be understood in the following way. For each time $t = 1, \ldots, T$, the particles $\{x_t^i\}_{i=1}^N$ generated by the PF is a set in X of cardinality N. By picking one particle from each time point, we obtain a particle trajectory, i.e., a point in X^T,

$$(x_1^{i_1}, \ldots, x_T^{i_T}) \in \mathsf{X}^T. \tag{3.12}$$

By letting all of the indices i_1, i_2, …, i_T range from 1 to N, we get in total N^T such trajectories. The empirical distribution (Equation (3.11)) assigns, to each such trajectory, an importance weight $\omega_{1:T|T}(i_1, \ldots, i_T)$.

Clearly, Equation (3.11) is impractical for any real problem of interest, since the computation of the weights is an $O(N^T)$ operation, both in terms of computational complexity and storage. However, Expression (3.11) provides an interesting connection between FFBSm and FFBSi.

From the construction of the backward simulator (Equation (3.3)), it follows that FFBSi generates i.i.d. draws from the distribution in Equation (3.11), conditionally on the forward filter particles. We thus make the following observation. Let $\varphi : \mathsf{X}^T \to \mathbb{R}$ be some test function, of which we seek to compute the expectation under the JSD, $\mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}]$. By Equation (3.11), the FFBSm estimator is given by,

$$\widehat{\varphi}^N_{\mathrm{FFBSm}} = \sum_{i_1=1}^{N} \cdots \sum_{i_T=1}^{N} \omega_{1:T|T}(i_1, \ldots, i_T)\varphi(x_1^{i_1}, \ldots, x_T^{i_T}). \tag{3.13}$$

This estimator is, unfortunately, also intractable. However, the FFBSi approximation (Equation (3.6)) provides an unbiased estimator of Equation (3.13), i.e.,

$$\widehat{\varphi}^N_{\mathrm{FFBSm}} = \mathbb{E}\left[\widetilde{\varphi}^M_{\mathrm{FFBSi}} \,\middle|\, \{x_t^i, w_t^i\}_{i=1}^N, t = 1, \ldots, T\right], \tag{3.14}$$

where $\widetilde{\varphi}^M_{\mathrm{FFBSi}} = \frac{1}{M}\sum_{j=1}^{M} \varphi(\widetilde{x}^j_{1:T})$. Here, the expectation is taken w.r.t. the random components of the backward simulation, conditionally on all the random variables generated by the forward filter. That is, the expectation in Equation (3.14) is given by a summation over the backward trajectory indices, corresponding exactly to Expression (3.13).

From Equation (3.14), we see that the FFBSm estimator can be seen as a Rao–Blackwellization of the FFBSi estimator (see e.g., [90]). Put the other way around, FFBSi is an "anti-Rao–Blackwellization" of FFBSm. Rao–Blackwellization usually aims at reducing the variance of an estimator, but generally at the cost of increased computational complexity. Here, we go the other way, and (significantly) reduce the complexity of the FFBSm estimator by instead employing FFBSi, albeit at the cost of a slight increase in variance.

3.2.2 Convergence of FFBSm and FFBSi

The close connection between FFBSm and FFBSi is useful in order to transfer convergence results from the former method to the latter. In

this section, a few key results from [35] are reviewed. For simplicity, we assume $M = N$ throughout this section. Let $\|\varphi\|_\infty = \sup_x |\varphi(x)|$ and $\mathrm{osc}(\varphi) = \sup_{(x,x')} |\varphi(x) - \varphi(x')|$ be the supremum and oscillator norms, respectively. Let $\xrightarrow{\mathrm{D}}$ denote convergence in distribution. We make the following assumption on bounds on the likelihood and the weight function.

(A1) For any $1 \leq t \leq T$,

- $g(y_t \mid \cdot) > 0$ and $\|g(y_t \mid \cdot)\|_\infty < \infty$.
- $\|W_t(\cdot, \cdot; y_t)\|_\infty < \infty$ where $W_t(x_t, x_{t-1}; y_t)$ is defined in Equation (2.5).

We start with a deviation inequality for FFBSm.

Theorem 3.2 (Exponential deviation inequality for FFBSm). Assume (A1). Then there exist constants $c_{1,T}$ and $c_{2,T} > 0$ such that for all N, $\varepsilon > 0$ and all bounded measurable functions $\varphi : \mathsf{X}^T \to \mathbb{R}$,

$$P(|\widehat{\varphi}^N_{\mathrm{FFBSm}} - \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}]| \geq \varepsilon) \leq c_{1,T} \exp(-c_{2,T} N \varepsilon^2 / \mathrm{osc}^2(\varphi)).$$

Proof. See [35, Theorem 5]. □

The exponential deviation inequality of Theorem 3.2 can be more or less directly extended to FFBSi by using the identity in Equation (3.14).

Corollary 3.3 (Exponential deviation inequality for FFBSi). Assume (A1). Then there exist constants $c_{1,T}$ and $c_{2,T} > 0$ such that for all N, $\varepsilon > 0$ and all bounded measurable functions $\varphi : \mathsf{X}^T \to \mathbb{R}$,

$$P(|\widetilde{\varphi}^N_{\mathrm{FFBSi}} - \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}]| \geq \varepsilon) \leq c_{1,T} \exp(-c_{2,T} N \varepsilon^2 / \mathrm{osc}^2(\varphi)).$$

Proof. See [35, Corollary 6]. □

Hence, when using FFBSi to compute an expectation, the probability that the Monte Carlo error exceeds some value ε is bounded by a function decreasing exponentially fast with the number of particles. Among other things, this non-asymptotic result implies almost sure convergence of the estimator $\widetilde{\varphi}^N_{\mathrm{FFBSi}}$.

In the asymptotic regime, it is also possible to establish a central limit theorem (CLT) with rate $\sqrt{N}$. To do so, we make an additional assumption on the transition density function and the proposal density.

(A2) $\|f(\cdot \mid \cdot)\|_\infty < \infty$ and, for any $1 \leq t \leq T$, $\|r_t(\cdot \mid \cdot, y_t)\|_\infty < \infty$.

A CLT can now be stated for FFBSm.

Theorem 3.4 (CLT for FFBSm). Assume (A1) and (A2). Then, for any bounded measurable function $\varphi : \mathsf{X}^T \to \mathbb{R}$,

$$\sqrt{N}(\widehat{\varphi}^N_{\mathrm{FFBSm}} - \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}]) \xrightarrow{\mathrm{D}} \mathcal{N}(0, \Gamma_{1:T|T}[\varphi]),$$

where $\Gamma_{1:T|T}[\varphi]$ is defined in [35, Equation (48)].

Proof. See [35, Theorem 8]. □

Similar to above, we can extend the CLT to the FFBSi by using the relation in Equation (3.14).

Corollary 3.5 (CLT for FFBSi). Assume (A1) and (A2). Then, for any bounded measurable function $\varphi : \mathsf{X}^T \to \mathbb{R}$,

$$\begin{aligned}\sqrt{N}(\widetilde{\varphi}^N_{\mathrm{FFBSi}} - \mathbb{E}[\varphi(x_{1:T}) \mid y_{1:T}]) \\ \xrightarrow{\mathrm{D}} \mathcal{N}(0, \mathrm{Var}(\varphi(x_{1:T}) \mid y_{1:T}) + \Gamma_{1:T|T}[\varphi]),\end{aligned}$$

where $\Gamma_{1:T|T}[\varphi]$ is defined in [35, Equation (48)].

Proof. See [35, Corollary 9]. □

The asymptotic variance for the FFBSi estimator contains an additional term, compared to the FFBSm estimator. This is in agreement with what we can expect from Equation (3.14). However, as pointed

out above, the increase in variance is compensated for by a significant reduction in computational complexity.

Additional convergence results are derived in [35], e.g., time uniform bounds for marginal smoothing estimators. Non-asymptotic deviation inequalities for FFBSi estimators of smoothed additive functionals are given in [45].

3.3 Backward Simulation with Rejection Sampling

The particle smoothers considered above have quadratic computational complexity, which can be prohibitive for many practical applications. FFBSi requires $O(MN)$ operations and the computational cost of FFBSm is (at least) $O(N^2)$.

There exist several different approaches to reduce the computational complexity of various particle smoothers, based on both numerical approximations and algorithmic modifications. In [84], so-called N-body methods are used to derive a marginal FFBSm with $O(N \log N)$ complexity. The same approach can be used also for FFBSi. These methods impose additional approximations, though the tolerance can usually be specified beforehand. Similar N-body methods have also been used in [85, 89] to compute *exact* joint maximum *a posteriori* estimates of the state trajectory in $O(N \log N)$ computational complexity.

There have also been developments based on the two-filter algorithm [18]. In its original formulation, this method is quadratic in the number of particles. However, Fearnhead et al. [51] have proposed a modified two-filter algorithm with linear complexity. In [49], quasirandom numbers are used for particle smoothing resulting in a method, albeit with quadratic complexity, but at the same time with a quadratic decrease in variance (at least for one-dimensional systems).

In this section, we will see how the computational complexity of FFBSi can be reduced, without introducing any further approximations, by using rejection sampling (RS).

3.3.1 Rejection Sampling

The basic idea that we will explore is to make use of a rejection sampler within the FFBSi algorithm, as suggested by [35]. The key insight is

that we do not need to evaluate all the smoothing weights $\{\widetilde{w}_{t|T}^i\}_{i=1}^N$ to be able to sample from the empirical backward kernel (Equation (3.4)). To convince ourselves that there indeed is room for improvement, note that in the FFBSi in Algorithm 4 on page 38 we evaluate N smoothing weights on line 5, draw a single sample from the categorical distribution on line 7 and then discard all the weights. Instead of making an exhaustive evaluation of the categorical distribution, we can sample from it using RS. For this to be applicable, we assume that the transition density function is bounded from above,

$$f(x_{t+1} \mid x_t) \leq \rho, \quad (x_t, x_{t+1}) \in \mathsf{X}^2, \tag{3.15}$$

which is true for many models arising in practical applications.

At time t, we wish to sample an index $b_t(j)$, corresponding to the forward filter particle which is to be appended to the jth backward trajectory. The target distribution is categorical over the index space $\{1, \ldots, N\}$, with probabilities $\{\widetilde{w}_{t|T}^{i,j}\}_{i=1}^N$ (which we have not computed yet). As proposal, we take another categorical distribution over the same index space, with (known) probabilities $\{w_t^i\}_{i=1}^N$. That is, we propose samples based on the filter weights, rather than on the smoothing weights.

Assume that a sample index $I(j)$ is proposed for the jth backward trajectory. To compute the acceptance probability, we consider the ratio between the target and the proposal distributions. Using the definition of the smoothing weights (Equation (3.5)) we get,

$$\frac{\widetilde{w}_{t|T}^{I(j),j}}{w_t^{I(j)}} = \frac{f(\widetilde{x}_{t+1}^j \mid x_t^{I(j)})}{\sum_l w_t^l f(\widetilde{x}_{t+1}^j \mid x_t^k)} \propto f(\widetilde{x}_{t+1}^j \mid x_t^{I(j)}) \leq \rho. \tag{3.16}$$

This implies that the sample should be accepted with probability $f(\widetilde{x}_{t+1}^j \mid x_t^{I(j)})/\rho$ (see e.g., [13, Section 11.1.2]). The rejection sampling-based FFBSi (RS-FFBSi) is given in Algorithm 5. We also provide MATLAB code in Listing 3.1.

The rationale for using RS is that in the limit $N \to \infty$, the computational cost will be linear in the number of particles. This is formalized in the following proposition, due to [35].

Algorithm 5 Rejection sampling FFBSi [35]

Input: Forward filter particle systems $\{x_t^i, w_t^i\}_{i=1}^N$ for $t = 1, \ldots, T$.
Output: Backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$.

Sample independently $\{b_T(j)\}_{j=1}^M \sim \mathrm{Cat}\left(\{w_T^i\}_{i=1}^N\right)$.
Set $\widetilde{x}_T^j = x_T^{b_T(j)}$ for $j = 1, \ldots, M$.
for $t = T-1$ **to** 1 **do**
  $L \leftarrow \{1, \ldots, M\}$.
  while L is not empty **do**
    $n \leftarrow \mathrm{card}(L)$.
    $\delta \leftarrow \emptyset$.
    Sample independently $\{I(k)\}_{k=1}^n \sim \mathrm{Cat}(\{w_t^i\}_{i=1}^N)$.
    Sample independently $\{U(k)\}_{k=1}^n \sim \mathcal{U}([0,1])$.
    for $k = 1$ **to** n **do**
      if $U(k) \leq f(\widetilde{x}_{t+1}^{L(k)} \mid x_t^{I(k)})/\rho$ **then**
        $b_t(L(k)) \leftarrow I(k)$.
        $\delta \leftarrow \delta \cup \{L(k)\}$.
      end if
    end for
    $L \leftarrow L \setminus \delta$.
  end while
  Set $\widetilde{x}_t^j = x_t^{b_t(j)}$ and $\widetilde{x}_{t:T}^j = \{\widetilde{x}_t^j, \widetilde{x}_{t+1:T}^j\}$ for $j = 1, \ldots, M$.
end for

Proposition 3.6. Assume Equation (3.15). Let $M = N$ and let C_t^N be the total number of simulations required in the accept–reject procedure at time t in Algorithm 5. Assume that $\int g(y_t \mid x_t)\, dx_t < \infty$ for all $t = 1, \ldots, T$. Then, for a bootstrap PF or a fully adapted PF, C_t^N/N converges in probability to a finite constant.

Proof. See [35, Proposition 1]. □

Proposition 3.6 implies that for large N, RS-FFBSi will have close to linear computational complexity. However, it is worth to note that there is no upper bound on the number of times that the while-loop at

```
1  % INPUT:
2  %    x_pf         - nx * N * T array with forward filter particles.
3  %    w_pf         - 1 * N * T array with forward filter weights.
4  %    T, M, N,     - Constants, see text for explanation.
5  %    nx, rho
6  % OUTPUT:
7  %    x_ffbsi      - nx * M * T array with backward trajectories.
8
9  x_ffbsi = zeros(nx, M, T);
10 bins = [0 cumsum(w_pf(:,:,T))];
11 [~, b] = histc(rand(M,1), bins);          % Sample I ~ Cat(w_T)
12 x_ffbsi(:,:,T) = x_pf(:,b,T);
13 for(t = (T-1) : (-1) : 1)
14     bins = [0 cumsum(w_pf(:,:,t))];       % Precomputation
15     L = 1:M;
16     while(~isempty(L))
17         n = length(L);
18         [~, I] = histc(rand(n,1), bins); % Sample I ~ Cat(w_t)
19         U = rand(1, n);                  % Sample U ~ U([0,1])
20
21         x_t1 = x_ffbsi(:, L, t+1);       % x_{t+1}^k for k in L
22         x_t =  x_pf(:, I, t);            % x_t^i      for i in I
23
24         p = transition_density_function(x_t1, x_t);
25         accept = (U <= p/rho);           % Accepted draws
26
27         % L is the index list of backward trajectories at time t
28         % that still need assignment. I is the index list of
29         % candidate particles at time t. The forward filter
30         % particle with (random) index I(k) is either accepted as
31         % the smoothing particle with index L(k), or not.
32
33         x_ffbsi(:, L(accept), t) = x_t(:,accept);
34         L = L(~accept);                  % Remove accepted
35     end
36 end
```

Listing 3.1 MATLAB code for RS-FFBSi. We have assumed that a function `transition_density_function(x_t1, x_t)` is available, where `x_t1` and `x_t` are $n_x \times N$ matrices (n_x being the state dimension). The function computes the transition density function value $f(x_{t+1} \mid x_t)$ for each pair of columns in the two matrices, and returns the result as a $1 \times N$ row vector.

line 5 may be executed. It has been observed in practice that different backward trajectories can get very different acceptance probabilities. This has the effect that most of the time required by Algorithm 5 is spent on just a few trajectories. Furthermore, RS-FFBSi has been found to be sensitive to the dimension of the state-space X [21, 131].

Technically, the rejection sampling in Algorithm 5 is done over the finite space $\{1, \ldots, N\}$. However, the underlying idea is to sample from the backward kernel using rejection sampling, with the filtering distribution as a proposal. That is, the rejection sampling is in effect done in the space X. Hence, it is natural to expect that the acceptance probability is diminished as the dimension of X increases, due to the curse of dimensionality.

Due to the effects mentioned above, RS-FFBSi can in fact require more computational time than the standard FFBSi (Algorithm 4) for many models. To alleviate this, Taghavi et al. [131] have proposed a hybrid strategy which switches between RS-FFBSi and FFBSi. This approach will be reviewed in the next section.

Finally, as a practical detail, we note that the sampling at line 8 should be conducted prior to the for-loop at line 10, for Algorithm 5 to reach linear complexity. That is, when proposing indices $\{I(k)\}_{k=1}^{n}$ from the categorical distribution with probabilities $\{w_t^i\}_{i=1}^{N}$, we draw the samples all at once. This allows us to use the efficient multinomial sampler by Douc et al. [35, Algorithm 2], which generates N i.i.d. samples from a categorical distribution with support at N points in $O(N)$ time.

3.3.2 Early Stopping

Since Expression (3.15) depends on x_{t+1}, different backward trajectories will get different acceptance probabilities in RS-FFBSi. The trajectories with high probabilities are typically accepted early in the process, whereas the trajectories with low probabilities can remain for many iterations. This has the effect that the cardinality of the set L (see Algorithm 5, page 51) decreases fast in the beginning, but it can linger for a long time close to zero. Consequently, most of the time required by RS-FFBSi is spent on just a few trajectories.

To speed up the algorithm, a hybrid strategy has been proposed by Taghavi et al. [131]. The idea is to run the rejection sampling loop for a certain number of iterations, and then switch to standard FFBSi for the remaining backward trajectories (i.e., by making an exhaustive evaluation of the remaining weights). The general method based on this idea, RS-FFBSi with early stopping, is given in Algorithm 6.

Algorithm 6 RS-FFBSi with early stopping [131]

Input: Forward filter particle systems $\{x_t^i, w_t^i\}_{i=1}^N$ for $t = 1, \ldots, T$.
Output: Backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$.

1: Sample independently $\{b_T(j)\}_{j=1}^M \sim \mathrm{Cat}\left(\{w_T^i\}_{i=1}^N\right)$.
2: Set $\widetilde{x}_T^j = x_T^{b_T(j)}$ for $j = 1, \ldots, M$.
3: **for** $t = T - 1$ **to** 1 **do**
4: $L \leftarrow \{1, \ldots, M\}$.
5: **while** Stopping criterion is not met **and** L is not empty **do**
6: Run one RS iteration (lines 6 to 16 in Algorithm 5, page 51).
7: **end while**
8: **if** L is not empty **then**
9: Sample $b_t(j)$ for $j \in L$ using FFBSi (Algorithm 4, page 38).
10: **end if**
11: Set $\widetilde{x}_t^j = x_t^{b_t(j)}$ and $\widetilde{x}_{t:T}^j = \{\widetilde{x}_t^j, \widetilde{x}_{t+1:T}^j\}$ for $j = 1, \ldots, M$.
12: **end for**

Two stopping criteria are proposed in [131], one deterministic and one adaptive. The first is a simple stopping rule, in which a maximum number of K rejection sampling iterations are allowed. If L is not empty after K iterations, an exhaustive evaluation of the backward sampling weights is made for the remaining trajectories. The second stopping criterion is an adaptive rule, which monitors the number of acceptances at each rejection sampling iteration. Based on this information, the average acceptance probability is estimated and used to make a decision about when to stop.

The benefit of using the adaptive rule is that it avoids a hard (and possibly difficult) choice for the design parameter K. Instead, it has the ability to automatically adapt the stopping time to the properties of the model, depending on the acceptance probabilities. As we will see in Example 3.3, early stopping can be quite useful in reducing the computational cost compared to both FFBSi and RS-FFBSi.

Example 3.3 (RS-FFBSi with early stopping). This example is taken from [131]. We consider a second-order LGSS model, previously

used to evaluate the two-filter smoother presented in [51],

$$\begin{aligned} x_{t+1} &= Ax_t + v_t, & v_t &\sim N(0, Q), \\ y_t &= Cx_t + e_t, & e_t &\sim \mathcal{N}(0, \sigma^2), \end{aligned}$$

with

$$C = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad Q = \begin{pmatrix} \frac{1}{3} & \frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix}.$$

We let $\sigma \in \{0.1, 1, 10\}$. The reason to choose different values for σ is to obtain different acceptance probabilities in the rejection sampler, where $\sigma = 0.1$, correspond to a fairly high acceptance probability and $\sigma = 10$ to a very low acceptance probability. We choose $N = 5\,000$ forward filter particles and $M = 1\,000$ backward trajectories. For the deterministic stopping rule, we consider three different thresholds: $K \in \{\frac{M}{5}, \frac{M}{10}, \frac{M}{20}\}$. We also use the adaptive stopping rule from [131].

We emphasize that all the backward simulators considered here are equivalent in terms of accuracy. Hence, they are evaluated only in terms of computational time. We clock the CPU times using the `tic` and `toc` commands in Matlab. All the simulations are done on a standard laptop Intel(R) Core(TM) i7-3720Qm 2.60GHz platform with 8GB of RAM. For each value of σ, we generate five unique datasets, each consisting of $T = 100$ samples. For each dataset, we run the algorithms ten times and average the results, to reduce the effects of randomness.

The results for different choices of σ are shown in Table 3.1. For high acceptance probabilities ($\sigma = 0.1$), there is a gain in using RS-FFBSi instead of FFBSi. However, as σ is increased, some backward trajectories get very small acceptance probabilities with many rejections as a result. By using early stopping, a large improvement in computation time is attained for both small and large acceptance probabilities, compared to both FFBSi and RS-FFBSi.

Table 3.1 Average CPU times in seconds for RS-FFBSi with early stopping.

σ	FFBSi	RS-FFBSi	$K = \frac{M}{5}$	$K = \frac{M}{10}$	$K = \frac{M}{20}$	Adaptive
0.1	44.65	19.50	2.03	1.94	2.36	1.92
1	45.28	77.71	3.49	3.65	5.33	3.79
10	48.63	355.70	14.83	20.21	25.86	15.25

3.4 Backward Simulation with MCMC Moves

In this section, we review a few related methods in which MCMC moves are used in the context of backward simulation.

3.4.1 Metropolis–Hastings FFBSi

The bottleneck of the FFBSi algorithm is the computation of the backward sampling weights (Equation (3.5)). In RS-FFBSi, rejection sampling is used to avoid an exhaustive evaluation of the weights, which is possible since the weights can be evaluated up to proportionality in constant time. However, under the same conditions, it is also possible to apply MCMC. An alternative is thus to run a Metropolis–Hastings sampler for a fixed number of iterations, say R, to approximately sample from the empirical backward kernel. This method, proposed by [21], is referred to as Metropolis–Hastings FFBSi (MH-FFBSi).

Given a partial backward trajectory $\widetilde{x}_{t+1:T}$, the backward weights are given by,

$$\widetilde{w}^i_{t|T} \propto w^i_t f(\widetilde{x}_{t+1} \mid x^i_t), \tag{3.17}$$

for $i = 1, \ldots, N$. Let $\{\nu^i_t\}_{i=1}^N$ be probabilities on the index set $\{1, \ldots, N\}$, defining a proposal distribution for an independent Metropolis–Hastings sampler. To propose a move from I to C, we thus draw $C \sim \mathrm{Cat}(\{\nu^i_t\}_{i=1}^N)$ and accept this sample with probability

$$1 \wedge \frac{w^C_t f(\widetilde{x}_{t+1} \mid x^C_t)}{w^I_t f(\widetilde{x}_{t+1} \mid x^I_t)} \frac{\nu^I_t}{\nu^C_t}. \tag{3.18}$$

Note that the acceptance probability can be evaluated in constant time (independent of N). After R iterations, the state of the Markov chain is used as an approximate sample from the empirical backward kernel.

To generate M backward trajectories, we have to run M independent chains. Hence, the computational complexity of MH-FFBSi is $O(RM)$. However, in [21] it is argued that a small number of iterations, say $R = 10$–30, are enough to get good performance in many cases. To avoid burn-in, they propose to initialize the Markov chains according to the genealogy of the forward filter. This means that, for $R = 0$, the MH-FFBSi reduces to a degenerate smoother relying on the

Algorithm 7 Metropolis–Hastings FFBSi [21]

Input: Forward filter particle systems $\{x_t^i, w_t^i\}_{i=1}^N$ for $t = 1, \ldots, T$.
Output: Backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$.

Sample independently $\{I(j)\}_{j=1}^M \sim \mathrm{Cat}\left(\{w_T^i\}_{i=1}^N\right)$.
Set $\widetilde{x}_T^j = x_T^{I(j)}$ for $j = 1, \ldots, M$.
for $t = T - 1$ **to** 1 **do**
  For $j = 1, \ldots, M$, set $I(j)[0]$ to the index of the ancestor of $\widetilde{x}_{t+1}^j$.
  for $r = 1$ **to** R **do**
    Sample independently $\{C(j)\}_{j=1}^M \sim \mathrm{Cat}(\{\nu_t^i\}_{i=1}^N)$.
    for $j = 1$ **to** M **do**
      With probability

$$1 \wedge \frac{w_t^{C(j)} \nu_t^{I(j)[r-1]}}{w_t^{I(j)[r-1]} \nu_t^{C(j)}} \frac{f(\widetilde{x}_{t+1}^j \mid x_t^{C(j)})}{f(\widetilde{x}_{t+1}^j \mid x_t^{I(j)[r-1]})},$$

      set $I(j)[r] = C(j)$, otherwise set $I(j)[r] = I(j)[r-1]$.
    end for
  end for
  Set $\widetilde{x}_t^j = x_t^{I(j)[R]}$ and $\widetilde{x}_{t:T}^j = \{\widetilde{x}_t^j, \widetilde{x}_{t+1:T}^j\}$ for $j = 1, \ldots, M$.
end for

forward filter particle trajectories. At the other extreme, as R tends to infinity, it approaches the FFBSi. The parameter R thus gives a trade-off between performance and computational cost. We summarize the MH-FFBSi sampler in Algorithm 7.

The use of MCMC within FFBSi opens up for additional modifications of the basic FFBSi, which is also recognized in [21]. The backward simulators considered so far are limited by the fact that the states are only selected from those which appear in the collection of forward filter particles. If there is a significant discrepancy between the filtering and the smoothing distributions, then the forward filter particles are unlikely to be in the right locations of the state-space to represent the smoothing distribution well.

For this reason, [21] propose a second method geared toward this issue. The method, referred to as Metropolis–Hastings forward

filtering/backward proposing (MH-FFBP), proceeds in a similar manner as MH-FFBSi. However, at time t, rather than running a Markov chain on the index set $\{1, \ldots, N\}$ to select one of the forward filter particles $\{x_t^i\}_{i=1}^N$, the chain runs on the product space $\{1, \ldots, N\} \times \mathsf{X}$ to sample jointly $\{x_{t-1}^I, x_t'\}$. That is, we draw one of the forward filter particles at time $t-1$ and, given this particle, we propose a new value for x_t from some continuous proposal distribution on X.

This approach is enabled by the fact that the backward kernel is given by a marginal of the joint density,

$$p(x_{t-1:t} \mid x_{t+1}, y_{1:t}) \propto f(x_{t+1} \mid x_t) g(y_t \mid x_t) f(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}). \tag{3.19}$$

Hence, for a fixed backward trajectory $\widetilde{x}_{t+1:T}$, if we obtain a sample $x_{t-1:t}'$ from the joint kernel (Equation (3.19)), we can simply discard x_{t-1}' and augment the backward trajectory according to $\widetilde{x}_{t:T} = \{x_t', \widetilde{x}_{t+1:T}\}$. By using the forward filter particles at time $t-1$, Equation (3.19) can be approximated by

$$\begin{aligned} &p(dx_{t-1:t} \mid \widetilde{x}_{t+1}, y_{1:t}) \\ &\quad \approx \sum_{i=1}^N \frac{w_{t-1}^i f(\widetilde{x}_{t+1} \mid x_t) g(y_t \mid x_t) f(x_t \mid x_{t-1}^i) dx_t}{\sum_l w_{t-1}^l p(\widetilde{x}_{t+1}, y_t \mid x_{t-1}^l)} \delta_{x_{t-1}^i}(dx_{t-1}). \end{aligned} \tag{3.20}$$

To sample from Equation (3.20) using MCMC, we choose a proposal distribution on $\{1, \ldots, N\} \times \mathsf{X}$,

$$\nu_{t-1}^i q(x_t \mid \widetilde{x}_{t+1}, y_t, x_{t-1}^i), \tag{3.21}$$

for $i = 1, \ldots, N$. To propose a move from $\{I, x_t\}$ to $\{C, x_t'\}$, we draw $C \sim \mathrm{Cat}(\{\nu_{t-1}^i\}_{i=1}^N)$ and $x_t' \sim q(x_t \mid \widetilde{x}_{t+1}, y_t, x_{t-1}^C)$. The sample is accepted with probability

$$1 \wedge \frac{w_{t-1}^C f(\widetilde{x}_{t+1} \mid x_t') g(y_t \mid x_t') f(x_t' \mid x_{t-1}^C)}{w_{t-1}^I f(\widetilde{x}_{t+1} \mid x_t) g(y_t \mid x_t) f(x_t \mid x_{t-1}^I)} \frac{\nu_{t-1}^I q(x_t \mid \widetilde{x}_{t+1}, y_t, x_{t-1}^I)}{\nu_{t-1}^C q(x_t' \mid \widetilde{x}_{t+1}, y_t, x_{t-1}^C)}. \tag{3.22}$$

As for MH-FFBSi, the acceptance probability can be evaluated in constant time. Hence, the computational complexity of MH-FFBP is of the same order, $O(RM)$, though, the overhead is clearly larger. The

Algorithm 8 Metropolis–Hastings FFBP [21]

Input: Forward filter particle systems $\{x_t^i, w_t^i\}_{i=1}^N$ for $t = 1, \ldots, T$.
Output: Backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$.

Sample independently $\{I(j)\}_{j=1}^M \sim \mathrm{Cat}\left(\{w_T^i\}_{i=1}^N\right)$.
Set $\widetilde{x}_T^j = x_T^{I(j)}$ for $j = 1, \ldots, M$.
for $t = T - 1$ **to** 1 **do**
  For $j = 1, \ldots, M$, set $\widetilde{x}_t^j[0]$ to the ancestor of $\widetilde{x}_{t+1}^j$ and set $I(j)[0]$ to the index of the ancestor of $\widetilde{x}_t^j[0]$.
  for $r = 1$ **to** R **do**
    Sample independently $\{C(j)\}_{j=1}^M \sim \mathrm{Cat}(\{\nu_{t-1}^i\}_{i=1}^N)$.
    for $j = 1$ **to** M **do**
      Sample $x_t'^{,j} \sim q(x_t \mid \widetilde{x}_{t+1}^j, y_t, x_{t-1}^{C(j)})$.
      With probability (3.22), set $\{I(j)[r], \widetilde{x}_t^j[r]\} = \{C(j), x_t'^{,j}\}$, otherwise set $\{I(j)[r], \widetilde{x}_t^j[r]\} = \{I(j)[r-1], \widetilde{x}_t^j[r-1]\}$.
    end for
  end for
  Set $\widetilde{x}_{t:T}^j = \{\widetilde{x}_t^j[R], \widetilde{x}_{t+1:T}^j\}$ for $j = 1, \ldots, M$.
end for

benefit of MH-FFBP, as pointed out above, is that it is able to sample new positions for the particles when generating the backward trajectories, instead of just recycling the particles from the forward filter. The MH-FFBP is given in Algorithm 8. Note that a straightforward modification is needed for the proposal mechanism at time $t = 1$, but, for brevity, we have not made this explicit in the algorithm.

3.4.2 Metropolis–Hastings Improved Particle Smoother

A method which is related to MH-FFBP is the Metropolis–Hastings improved particle smoother (MH-IPS), suggested by Dubarry and Douc [46]. The method is also reminiscent of the resample-move algorithm [60]. Instead of making use of the intermediate filtering distributions, as has been the common theme for the backward simulators considered so far, they start from the (degenerate) paths $\{\breve{x}_{1:T}^i\}_{i=1}^N$, obtained by resampling the forward filter trajectories at time T. To

increase the diversity among these paths, the particle positions are updated by running N independent, single-state MCMC samplers, one for each particle trajectory.

If possible, the variables x_t are sampled from their full conditionals,

$$p(x_t \mid x_{1:t-1}, x_{t+1:T}, y_{1:T}) \propto f(x_{t+1} \mid x_t) g(y_t \mid x_t) f(x_t \mid x_{t-1}). \quad (3.23)$$

In the general case, however, these conditionals are not available. Instead, Hastings-within-Gibbs moves are used to update the state trajectories. Let $\widetilde{x}_{1:T}$ be the current state of one of the Markov chains (i.e., one of the current particle trajectories). To update the tth component, we simulate from some proposal density $x'_t \sim q(x_t \mid \widetilde{x}_{t+1}, y_t, \widetilde{x}_{t-1})$, targeting Equation (3.23). With probability

$$1 \wedge \frac{f(\widetilde{x}_{t+1} \mid x'_t) g(y_t \mid x'_t) f(x'_t \mid \widetilde{x}_{t-1})}{f(\widetilde{x}_{t+1} \mid \widetilde{x}_t) g(y_t \mid \widetilde{x}_t) f(\widetilde{x}_t \mid \widetilde{x}_{t-1})} \frac{q(\widetilde{x}_t \mid \widetilde{x}_{t+1}, y_t, \widetilde{x}_{t-1})}{q(x'_t \mid \widetilde{x}_{t+1}, y_t, \widetilde{x}_{t-1})}, \quad (3.24)$$

the proposed sample is accepted and a new state trajectory is constructed as $\{\widetilde{x}_{1:t-1}, x'_t, \widetilde{x}_{t+1:T}\}$. If not, the sample is rejected and the current trajectory $\widetilde{x}_{1:T}$ is retained. Obvious modifications to the proposal density and the acceptance probability are needed at times $t = T$ and $t = 1$.

In [46], it is suggested to sample the state variables in a deterministic order, backward in time, for $t = T, \ldots, 1$, which makes the MH-IPS reminiscent of a backward simulator. The incentive for this is to propagate the, in general, rich particle diversity at time points close to T, down to time points far from T. The MH-IPS is given in Algorithm 9, where we use a fixed number of R MCMC iterations. That is, we sweep through the data R times, from time $t = T$ to time $t = 1$. The algorithm produces a (unweighted) collection of backward trajectories $\{\widetilde{x}^i_{1:T}[R]\}_{i=1}^N$ which can be seen as approximate draws from the JSD.

In the simulation studies conducted in [46], only a few MCMC iterations, say $R = 5$–10, improve the particle diversity considerably. The computational complexity of MH-IPS is $O(RN)$. Hence, MH-IPS can be an interesting, and computationally cheaper, alternative to FFBSi, much like the MH-FFBP. It should be noted, however, that the method

Algorithm 9 Metropolis–Hastings improved particle smoother [46]

Input: Forward filter particle trajectories $\{x_{1:T}^i, w_T^i\}_{i=1}^N$.
Output: Improved particle trajectories $\{\widetilde{x}_{1:T}^i\}_{i=1}^N$.

1: Resample the forward filter particle system at time T to obtain an equally weighted particle system $\{\check{x}_{1:T}^i, \frac{1}{N}\}_{i=1}^N$.
2: Initialize: set $\widetilde{x}_{1:T}^i[0] = \check{x}_{1:T}^i$ for $i = 1, \ldots, N$.
3: **for** $r = 1$ **to** R **do**
4: **for** $t = T$ **to** 1 **do**
5: **for** $i = 1$ **to** N **do**
6: Sample $x_t'^i \sim q_t(x_t \mid \widetilde{x}_{t+1}^i[r], y_t, \widetilde{x}_{t-1}^i[r-1])$.
7: With probability given by (3.24), set $\widetilde{x}_t^i[r] = x_t'^i$, otherwise set $\widetilde{x}_t^i[r] = \widetilde{x}_t^i[r-1]$.
8: **end for**
9: **end for**
10: **end for**
11: Set $\widetilde{x}_{1:T}^i = \widetilde{x}_{1:T}^i[R]$ for $i = 1, \ldots, N$.

relies on single-state updates, and it might thus to a larger extent than a backward simulator be subject to strong dependencies between the state variables.

The main differences between MH-FFBP and MH-IPS are: (i) MH-FFBP uses all the intermediate filtering approximations from the forward filter, whereas MH-IPS only makes use of the degenerate particle trajectories at time T; (ii) MH-FFBP loops over the time indices $t = T, \ldots, 1$ and runs R MCMC iterations for each time t, whereas MH-IPS runs R MCMC iterations, looping over the time indices $t = T, \ldots, 1$ in each iteration.

Finally, as pointed out in [46], the MH-IPS procedure does not have to be initialized by the forward filter trajectories. In fact, it can be used as an add-on to any particle smoother, to increase the diversity among the particle trajectories. Hence, it is possible to think of various combinations, such as FFBSi combined with MH-IPS or even MH-FFBP combined with MH-IPS.

3.5 Backward Simulation for Maximum Likelihood Inference

As noted in Section 1.5, state smoothing lies at the core of many common learning algorithms for SSMs. The particle smoothers discussed throughout this section can thus be useful for parameter inference in nonlinear and/or non-Gaussian models. To illustrate this, we consider the problem of maximum likelihood parameter inference in the parameterized SSM (Equation (1.3)).

To compute the MLE (Equation (1.4)), we make use of the EM algorithm as discussed in Section 1.5. However, computing the auxiliary quantity in the E-step amounts to solving a smoothing problem, which cannot be done in closed form for a general nonlinear/non-Gaussian SSM. Instead, we run one of the FFBSi (e.g., Algorithm 6 on page 54, RS-FFBSi with early stopping) in the E-step. This results in an SMC-analogue of the well known Monte Carlo EM algorithm [140]. Similar particle smoother EM (PSEM) algorithms have previously been successfully applied to maximum likelihood inference in challenging scenarios [23, 112, 127, 147].

More precisely, given a collection of backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^{M_r}$ targeting $p_{\theta[r-1]}(x_{1:T} \mid y_{1:T})$, we compute an estimate of the auxiliary quantity (Equation (1.5)) according to

$$\widehat{Q}_r(\theta) = \frac{1}{M_r} \sum_{j=1}^{M_r} \log p_\theta(\widetilde{x}_{1:T}^j, y_{1:T}), \tag{3.25}$$

where the summand is given by Equation (1.6). This Monte Carlo approximation is then maximized with respect to θ in the M-step, to obtain the next parameter iterate. To obtain convergence of PSEM, we require the accuracy of the approximation (Equation (3.25)) to increase with the iteration number [23, 54]. Consequently, we need to let the number of particles N_r and the number of backward trajectories M_r increase with the iteration number r. One way to circumvent this issue will be presented in Section 5.6.

The backward-simulation-based PSEM algorithm is summarized in Algorithm 10. We illustrate the method in Example 3.4 below.

Algorithm 10 Backward-simulation-based PSEM

Set $\theta[0]$ arbitrarily.
for $k \geq 1$ **do**
  Run Algorithm 1 with N_r particles, targeting $p_{\theta[r-1]}(x_{1:T} \mid y_{1:T})$.
  Run a backward simulator to generate M_r trajectories $\{\widetilde{x}^j_{1:T}\}_{j=1}^N$.
  Compute $\widehat{Q}_r(\theta)$ according to Equation (3.25).
  Compute $\theta[r] = \arg\max_{\theta \in \Theta} \widehat{Q}_r(\theta)$.
  if convergence criterion is met **then**
    break
  end if
end for
return $\hat{\theta}_{\mathrm{PSEM}} = \theta[r]$.

Example 3.4 (PSEM). Consider again the nonlinear time-series model studied in Example 3.2. We let the process noise variance be given by $\sigma_v^2 = 1$ and the measurement noise variance be given by $\sigma_e^2 = 0.1$, and assume that these parameters are unknown. Given a batch of $T = 1\,500$ observations $y_{1:T}$, we wish to infer the unknown noise variances and thus set $\theta = (\sigma_v^2, \sigma_e^2)$. We apply Algorithm 10 using an RS-FFBSi sampler with early stopping in the E-step. The parameter estimates are initialized at $\theta[0] = (2, 2)$. We run the algorithm for 2 000 iterations and let the number of particles N_r and the number of backward trajectories M_r increase cubically with the iteration number r, from 500 to 5 000 and from 50 to 500, respectively. The resulting parameter estimates $\theta[r]$ are shown in Figure 3.4. As can be seen, the estimates converge to values close to the true parameters. However, as pointed out above, it is necessary to increase the number of particles and backward trajectories with the iteration number to kill the Monte Carlo variance and obtain a convergent sequence of estimates.

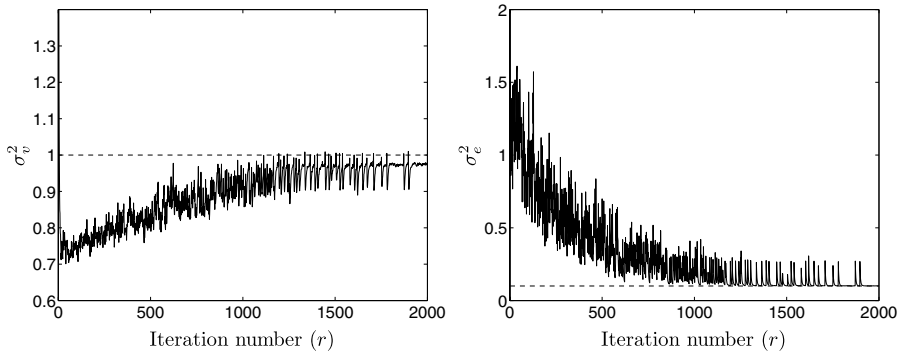

Fig. 3.4 Estimates of σ_v^2 (left) and σ_e^2 (right) vs. the iteration number k. The true parameter values are shown as dashed lines.

4

Backward Simulation for General Sequential Models

So far, we have been looking closely at SSMs. In the previous sections, we saw how SMC and backward simulation can be used to sample approximately from the JSD $p(x_{1:T} \mid y_{1:T})$. One of the strengths of SMC, however, is that it is applicable to a much wider range of models than SSMs. The same is true for backward simulation. However, contrary to SMC, the application of backward simulation will in general be less straightforward when we leave the class of SSMs. In this section, we will derive a general backward simulator and discuss how this can be applied for inference in various models of interest. We will also highlight some pitfalls that might limit the applicability of backward simulation for certain types of models.

4.1 Motivating Examples

Let us start by considering a few examples of sequential inference problems which fall outside the class of SSMs. These examples are included as motivation for the development of the present section, but also to outline possible directions for future work. We will return to these examples in Section 6, when discussing possible extensions of the

methods presented throughout this tutorial. Readers not interested in these examples can safely skip directly to Section 4.2.

4.1.1 Blocked Gibbs Sampling in Markov Random Fields

Markov random fields (MRFs) are a class of undirected graphical models. They play an important role in spatial statistics and computer vision, see e.g., [15, 138]. Let $\{x_t\}_{t=1}^T$ be a collection of latent variables and let $\{y_t\}_{t=1}^T$ be a collection of observations. In this setting, t is just an index variable for the data set and it has no temporal meaning. The conditional independence properties among these variables are specified in terms of an undirected graph with edges $\mathcal{E}$ and vertices $\mathcal{V}$ as illustrated in Figure 4.1 (left). The complete data likelihood can be factorized according to

$$p(x_{1:T}, y_{1:T}) = \frac{1}{Z} \prod_{i \in \mathcal{V}} \phi(x_i, y_i) \prod_{(i,j) \in \mathcal{E}} \psi(x_i, x_j), \tag{4.1}$$

where Z is a normalization constant (referred to as the partition function) and the functions ϕ and ψ are referred to as the observation potential and the interaction potential, respectively.

As in the case of SSMs, Gibbs samplers that sample the latent variables one at a time tend to be slow to converge, due to strong

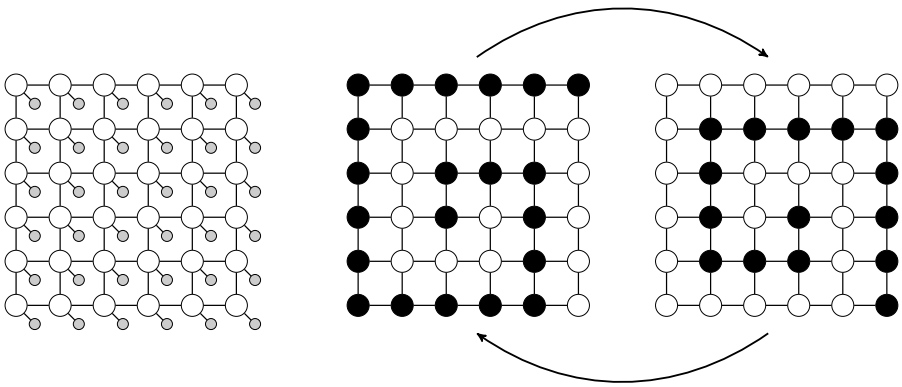

Fig. 4.1 (Left) MRF with white nodes representing latent variables and small gray nodes representing observations; (Right) partitioning of the latent variables into groups $x_{\mathcal{I}_1}$ and $x_{\mathcal{I}_2}$ used in the tree sampling algorithm (for clarity, the nodes corresponding to the observations are not shown).

dependencies among the sampled variables. To alleviate this, a Gibbs sampler which exploits the structure of the MRF has been proposed in [68]. The method, referred to as a tree sampling algorithm, is based on the fact that the MRF graph can be partitioned into two disjoint chains, or more generally, trees; see Figure 4.1 (right). With $\mathcal{I}_1$ and $\mathcal{I}_2$ representing the vertex indices for the two trees, respectively, a two-stage Gibbs sampler is constructed according to;

(i) Draw $x'_{\mathcal{I}_1} \sim p(x_{\mathcal{I}_1} \mid x_{\mathcal{I}_2}, y_{1:T})$;
(ii) Draw $x'_{\mathcal{I}_2} \sim p(x_{\mathcal{I}_2} \mid x'_{\mathcal{I}_1}, y_{1:T})$.

By block sampling the latent variables in each tree, a considerable improvement in mixing can be obtained compared to a one-at-a-time sampler [69, 68]. This is in agreement with our previous findings for SSMs.

Conditionally on $x_{\mathcal{I}_2}$, the variables $x_{\mathcal{I}_1}$ form an acyclic graph. In [68], it is assumed that the latent variables are discrete valued (cf. a finite state-space HMM). Hence, it is possible to compute their exact marginal distributions by using belief propagation [115]. Based on these marginals, exact backward simulation can be employed to sample from the conditional densities in the above Gibbs scheme.

By using SMC it is possible to generalize this idea to models for which the exact marginals are not available. In particular, in combination with PMCMC (see Section 5), SMC-based backward simulation can be used within Gibbs sampling in a systematic manner. This approach would thus generalize the tree sampling algorithm to, for instance, models with continuous latent variables. This requires a backward simulator for the chain structures in the MRF in Figure 4.1. The same approach can also be applied to other types of graphical models, such as factor graphs [69].

4.1.2 Inference Strategies for Optimal Control Problems

An interesting aspect of parameter inference is that it is useful, not only for learning predictive models, but also for policy optimization in control problems. Consider a dynamical system with state x_t, affected

by a controlled input signal u_t,

$$x_{t+1} \sim f(x_{t+1} \mid x_t, u_t) \tag{4.2}$$

and $x_1 \sim \mu(x_1)$. For simplicity, we assume that the state is completely observed. We consider a state-feedback policy (i.e., a control law) $u_t = \pi_\theta(x_t)$, parameterized by some parameter θ. Furthermore, let $r_t = r(x_t, u_t)$ be the reward, or return, obtained at time t. In optimal control problems we wish to find the policy which maximizes the expected future return,

$$V_\mu^\pi(\theta) = \mathbb{E}\left[\sum_{t=1}^{\infty} \alpha^t r_t\right], \tag{4.3}$$

where $\alpha < 1$ is a discount factor. It has been recognized [42, 75, 135] that it is possible to view Equation (4.3) as the normalization constant for an artificial transdimensional probability distribution, defined on $\bigcup_{T\in\mathbb{N}}\{T\} \times \mathsf{X}^T$,

$$\widetilde{p}_\theta(T, x_{1:T}) \triangleq \frac{r(x_T, u_T)\alpha^T}{V_\mu^\pi(\theta)}\mu(x_1)\prod_{t=1}^{T-1} f(x_{t+1} \mid x_t, u_t). \tag{4.4}$$

What is interesting with this reformulation is that maximization of the normalization constant is completely analogous to maximization of the marginal likelihood in the parameterized SSM (Equation (1.3)). It follows that the inference strategies discussed in Section 1.5 can be used also for policy optimization. Indeed, EM algorithms have been used by Toussaint and Storkey [135] and Hoffman et al. [74] for discrete and linear Gaussian models, respectively. In [42, 75] the problem is addressed in a general setting by using transdimensional MCMC, such as reversible jump samplers [65]. By factorizing $\widetilde{p}_\theta(T, x_{1:T}) = \widetilde{p}_\theta(x_{1:T} \mid T)\widetilde{p}_\theta(T)$, we note that $\widetilde{p}_\theta(x_{1:T} \mid T)$ takes the role of a joint smoothing density. Hence, the intermediate state inference step of these algorithms requires us to address a nonstandard "smoothing problem" for the artificial distribution $\widetilde{p}_\theta(x_{1:T} \mid T)$.

4.1.3 Gaussian Process Regression

The Markovian assumption in SSMs can be limiting for many applications of interest. Consider a regression problem with regressors $\xi_t \in \mathbb{R}^{n_\xi}$

and observations $y_t \in \mathbb{R}^{n_y}$. We observe a batch of data $\{\xi_t, y_t\}_{t=1}^T$ and wish to find a predictive model for y_t given ξ_t. As in Section 4.1.1, t is simply an index variable with no temporal meaning.

To model the dependency between the regressors and the observations, we use a nonparametric Gaussian process (GP) model. GPs are widely used for both classification and regression problems; see [118] for a general introduction. In the regression setting, we have

$$f(\cdot) \sim \mathcal{GP}(m(\xi), \kappa(\xi, \xi')), \tag{4.5a}$$

$$x_t = f(\xi_t), \tag{4.5b}$$

$$y_t \mid x_t \sim g(y_t \mid x_t), \tag{4.5c}$$

where $m(\cdot)$ is a mean function and $\kappa(\cdot, \cdot)$ is a covariance kernel. Hence, the function f is modeled as a sample path from a GP, i.e., such that any finite collection of variables $\{f(\xi_t)\}_{t\in\mathcal{I}}$ (where $\mathcal{I}$ is an index set) has a joint Gaussian distribution,

$$\begin{aligned} \{f(\xi_t)\}_{t\in\mathcal{I}} &\sim \mathcal{N}(\mathbf{m}, K), \\ \mathbf{m}_t &= m(\xi_t), & t &\in \mathcal{I}, \\ K_{s,t} &= \kappa(\xi_s, \xi_t) & s, t &\in \mathcal{I}. \end{aligned}$$

The x_ts can be seen as noise free sample points from the GP. These variables are latent, but observed indirectly through the measurement likelihood $g(y_t \mid x_t)$. The model (Equation (4.5)) is reminiscent of Equation (1.2). However, for the GP model, the x_ts are not given by a Markovian process and the model thus falls outside the class of SSMs. We return to this problem in Examples 4.2 and 4.3 below.

4.2 SMC Revisited

The examples provided in the previous section are examples of latent variable models which are not given by SSMs. There are numerous other models of this kind for which SMC has been successfully applied, e.g., Dirichlet process mixture models [99, 48], phylogenetic trees [16] and agglomerative clustering models [133], to mention a few.

We thus seek a framework for backward simulation which is more generally applicable than the one presented in Section 3. For this cause, let $\gamma_t(x_{1:t})$ for $t = 1, \ldots, T$ be a sequence of target densities on some

increasing space X^t. We assume that these densities can be written as

$$\gamma_t(x_{1:t}) = \frac{\bar{\gamma}_t(x_{1:t})}{Z_t}, \tag{4.6}$$

where the unnormalized density $\bar{\gamma}_t(x_{1:t})$ can be evaluated point-wise, whereas the normalization constant Z_t is possibly unknown. SMC can be used to target any such sequence, in the same way as was done for the sequence of JSDs in Section 2. In that case, we had $\gamma_t(x_{1:t}) = p(x_{1:t} \mid y_{1:t})$ and $\bar{\gamma}_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$. We will continue to refer to the index t as time, even though it might not at all be a temporal index. In [107], SMC samplers are designed for a sequence of target distributions on a common, fixed dimensional space. However, by making use of auxiliary variables, they transform these problems into a form which coincides with the SMC framework discussed here.

Assume that $\{x^i_{1:t-1}, w^i_{t-1}\}_{i=1}^N$ is a weighted particle system, targeting $\gamma_{t-1}(x_{1:t-1})$. This particle system is propagated to time t by resampling and sequential importance sampling, in the same way as was done in Section 2. However, we now take a slightly different view on this procedure, which will prove to be convenient in the sequel.

Let a^i_t be the index of the ancestor at time $t-1$, of particle x^i_t. That is, if $\{\check{x}^i_{1:t-1}\}_{i=1}^N$ is the collection of resampled particle trajectories at time $t-1$, we have $\check{x}^i_{1:t-1} = x^{a^i_t}_{1:t-1}$. The ancestor indices are auxiliary variables in the SMC sampler, and they were (among other things) used to derive the auxiliary particle filter (APF) in [116]. When we write $x^k_{1:t}$, this should be interpreted as the ancestral path of the particle x^k_t. Since the concepts of ancestor indices and ancestral paths will be of importance later on, we illustrate them with an example.

Example 4.1 (Ancestral paths). Figure 4.2 shows the evolution of three particles for $t = 1, 2, 3$. At time $t = 1$, particle x^2_1 is resampled twice and particle x^3_1 is resampled once. At time $t = 2$, the ancestors are thus given by $a^1_2 = 2$, $a^2_2 = 2$ and $a^3_2 = 3$. Similarly, at time $t = 3$, the ancestors are given by $a^1_3 = 2$, $a^2_3 = 3$ and $a^3_3 = 3$. The ancestral path of particle x^1_3, which we denote $x^1_{1:3}$, is shown as a thick line in the figure. This path is given recursively from the ancestor indices,

$$x^1_{1:3} = (x_1^{a_2^{a_3^1}}, x_2^{a_3^1}, x_3^1) = (x^2_1, x^2_2, x^1_3).$$

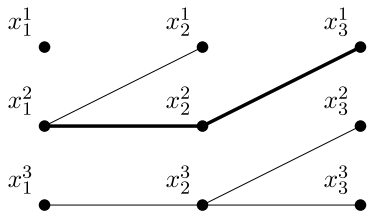

Fig. 4.2 Evolution of a particle system. The ancestral path of x_3^1, i.e., $x_{1:3}^1$, is shown as a thick line.

Let us introduce a sequence of proposal kernels on the product space $\{1, \ldots, N\} \times \mathsf{X}$,

$$M_t(a_t, x_t) = w_{t-1}^{a_t} r_t(x_t \mid x_{1:t-1}^{a_t}), \tag{4.7}$$

for $t = 2, \ldots, T$. Here, r_t is a kernel from which we propose a new sample at time t, given its ancestor $x_{1:t-1}^{a_t}$. 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.

The particle system $\{x_{1:t-1}^i, w_{t-1}^i\}_{i=1}^N$ can now be propagated to time t by sampling independently from Equation (4.7),

$$\{a_t^i, x_t^i\} \sim M_t(a_t, x_t), \tag{4.8}$$

for $i = 1, \ldots, N$. The particles $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$ are thereafter assigned (unnormalized) importance weights $\bar{w}_t^i = W_t(x_{1:t}^i)$, where the weight function is given by,

$$W_t(x_{1:t}) = \frac{\bar{\gamma}_t(x_{1:t})}{\bar{\gamma}_{t-1}(x_{1:t-1}) r_t(x_t \mid x_{1:t-1})}. \tag{4.9}$$

As before, since the weights are only known up to proportionality, they are normalized to sum to one, $w_t^i = \bar{w}_t^i / \sum_l \bar{w}_t^l$. This results in a new weighted particle system $\{x_{1:t}^i, w_t^i\}_{i=1}^N$ targeting $\gamma_t(x_{1:t})$. In this formulation, the resampling step is implicit and corresponds to sampling the ancestor indices a_t in Equation (4.8).

The procedure is initialized by targeting $\gamma_1(x_1)$ using importance sampling. We thus sample from some proposal density $x_1^i \sim r_1(x_1)$ and compute importance weights $\bar{w}_t^i = W_1(x_1)$, where the weight function

Algorithm 11 SMC (all operations are for $i = 1, \ldots, N$)

1: Draw $x_1^i \sim r_1(x_1)$.
2: Compute $\bar{w}_1^i = W_1(x_1^i)$.
3: Normalize: set $w_1^i = \bar{w}_1^i / \sum_l \bar{w}_1^l$.
4: **for** $t = 2$ **to** T **do**
5: Draw $\{a_t^i, x_t^i\} \sim M_t(a_t, x_t)$.
6: Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$.
7: Compute $\bar{w}_t^i = W_t(x_{1:t}^i)$.
8: Normalize: set $w_t^i = \bar{w}_t^i / \sum_l \bar{w}_t^l$.
9: **end for**

is given by $W_1(x_1) = \bar{\gamma}_1(x_1)/r_1(x_1)$. The SMC sampler is summarized in Algorithm 11.

4.3 A General Backward Simulator

In many inferential problems for which SMC is applied, the actual quantity of interest is the final target density $\gamma_T(x_{1:T})$, or some marginal thereof. The densities $\gamma_t(x_{1:t})$ are then used as intermediate quantities, as a way to sequentially construct this target. The SMC procedure outlined above generates a sequence of weighted particle systems $\{x_{1:t}^i, w_t^i\}_{i=1}^N$ targeting the densities $\gamma_t(x_{1:t})$ for $t = 1, \ldots, T$. These systems define empirical point-mass distributions according to

$$\widehat{\gamma}_t^N(dx_{1:t}) \triangleq \sum_{i=1}^N w_t^i \delta_{x_{1:t}^i}(dx_{1:t}). \tag{4.10}$$

Hence, an approximation of $\gamma_T(x_{1:T})$ is given for $t = T$. However, this approximation will be inaccurate due to path degeneracy (see Section 2.1.2). To mitigate this issue, we seek a way to increase the diversity of the particle trajectories used to approximate $\gamma_T(x_{1:T})$. We will strive to do this in a similar way as for the FFBSi in the SSM setting, i.e., by reusing information from the intermediate approximations (Equation(4.10)).

Consider the conditional density

$$\gamma_T(x_{1:t} \mid x_{t+1:T}) = \frac{\gamma_T(x_{1:T})}{\int \gamma_T(x_{1:T})\, dx_{1:t}}. \tag{4.11}$$

This density defines a transition kernel (the backward kernel) of invariant density γ_T. Hence, if $x_{1:T}$ is distributed according to γ_T, we can generate a new sample with the same distribution by sampling from this kernel, much in the same way as in a Gibbs sampler. That is, we draw $x'_{1:t} \sim \gamma_T(x_{1:t} \mid x_{t+1:T})$ and construct the trajectory $\{x'_{1:t}, x_{t+1:T}\}$. This suggests that we can use the following strategy:

- Draw $x'_{1:T} \sim \gamma_T(x_{1:T})$, set $\widetilde{x}_T = x'_T$ and discard $x'_{1:T-1}$;
- For $t = T-1, \ldots, 1$:
  - Draw $x'_{1:t} \sim \gamma_T(x_{1:t} \mid \widetilde{x}_{t+1:T})$;
  - Set $\widetilde{x}_{t:T} = \{x'_t, \widetilde{x}_{t+1:T}\}$ and discard $x'_{1:t-1}$.

The resulting trajectory $\widetilde{x}_{1:T}$ is distributed according to γ_T. At first, this backward simulator might appear superfluous, since it is initialized by sampling directly from the target density of interest. However, in the case of SMC, it allows us to make use of the intermediate quantities (Equation (4.10)) to improve upon the degenerate particle trajectories resulting from an initial SMC sweep.

The backward kernel for this simulator is given by

$$\gamma_T(x_{1:t} \mid x_{t+1:T}) \propto \gamma_T(x_{1:T}) \propto \frac{\bar{\gamma}_T(x_{1:T})}{\bar{\gamma}_t(x_{1:t})}\gamma_t(x_{1:t}). \tag{4.12}$$

By plugging Equation (4.10) into this expression, we obtain an approximation of the backward kernel according to

$$\widehat{\gamma}_T^N(dx_{1:t} \mid \widetilde{x}_{t+1:T}) = \sum_{i=1}^{N} \widetilde{w}_{t|T}^i \delta_{x_{1:t}^i}(dx_{1:t}), \tag{4.13}$$

with

$$\widetilde{w}_{t|T}^i \propto w_t^i \frac{\bar{\gamma}_T(\{x_{1:t}^i, \widetilde{x}_{t+1:T}\})}{\bar{\gamma}_t(x_{1:t}^i)}, \tag{4.14}$$

and where the weights are normalized to sum to one. Here, $\{x_{1:t}^i, \widetilde{x}_{t+1:T}\}$ should be understood as a point in X^T formed by concatenating the two partial trajectories. As expected, for an SSM, with $\bar{\gamma}_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$, we obtain the weight expression from Equation (3.5). Using the empirical backward kernel, we construct an SMC-based backward

Algorithm 12 Backward simulator

Input: Forward filter particle systems $\{x^i_{1:t}, w^i_t\}_{i=1}^N$ for $t = 1, \ldots, T$.
Output: Backward trajectories $\{\widetilde{x}^j_{1:T}\}_{j=1}^M$.

1: Sample independently $\{b_T(j)\}_{j=1}^M \sim \mathrm{Cat}\left(\{w^i_T\}_{i=1}^N\right)$.
2: Set $\widetilde{x}^j_T = x_T^{b_T(j)}$ for $j = 1, \ldots, M$.
3: **for** $t = T-1$ **to** 1 **do**
4: **for** $j = 1$ **to** M **do**
5: Compute

$$\widetilde{w}^{i,j}_{t|T} \propto w^i_t \frac{\bar{\gamma}_T(\{x^i_{1:t}, \widetilde{x}^j_{t+1:T}\})}{\bar{\gamma}_t(x^i_{1:t})},$$

 for $i = 1, \ldots, N$.
6: Normalize the smoothing weights $\{\widetilde{w}^{i,j}_{t|T}\}_{i=1}^N$ to sum to one.
7: Draw $b_t(j) \sim \mathrm{Cat}\left(\{\widetilde{w}^{i,j}_{t|T}\}_{i=1}^N\right)$.
8: Set $\widetilde{x}^j_t = x_t^{b_t(j)}$ and $\widetilde{x}^j_{t:T} = \{\widetilde{x}^j_t, \widetilde{x}^j_{t+1:T}\}$.
9: **end for**
10: **end for**

simulator as in Algorithm 12. The algorithm generates M independent trajectories, similar to the FFBSi in Algorithm 4 (page 38). We illustrate the backward simulation on the nonparametric regression problem from Section 4.1.3.

Example 4.2 (Gaussian process regression). Let us return to the GP regression problem considered in Section 4.1.3. We observe a heteroscedastic data set $\{y_{1:T}, \xi_{1:T}\}$ for $T = 1000$ and wish to infer the momentaneous volatility. The data is illustrated in Figure 4.3. We put a zero-mean Gaussian process prior on the log-volatility, i.e., the model is given by

$$f(\cdot) \sim \mathcal{GP}(0, \kappa(\xi, \xi')), \tag{4.15}$$

$$x_t = f(\xi_t), \tag{4.16}$$

$$y_t = e_t \exp\left(\tfrac{1}{2} x_t\right), \quad e_t \sim \mathcal{N}(0,1). \tag{4.17}$$

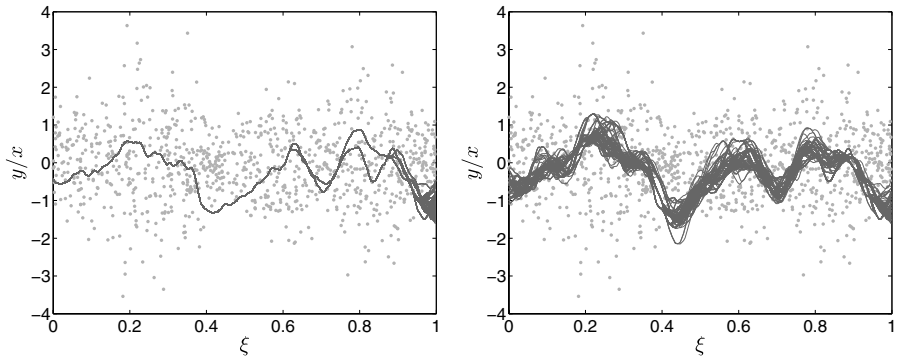

Fig. 4.3 Regression results using SMC; (left) applied only in the forward direction; (right) complemented with a backward simulation pass. The dots show the observations and the gray lines the generated trajectories, representing the log-volatility.

To complete the model, we use a covariance function in the Matérn class of kernels, namely,

$$\kappa(\xi,\xi') = \left(1 + \frac{\sqrt{3}|\xi - \xi'|}{\ell}\right) \exp\left(-\frac{\sqrt{3}|\xi - \xi'|}{\ell}\right).$$

For simplicity, we fix the length scale to $\ell = 0.1$. To infer the latent process $x_{1:T}$, we employ an SMC sampler with $N = 500$ particles to the data set.[1] The results are given in the left panel of Figure 4.3. The SMC sampler clearly suffers from path degeneracy, resulting in poor regression results and overconfidence in the estimated volatility. To mitigate these issues, we complement the SMC sampler with a run of the backward simulator given in Algorithm 12. We generate $M = 100$ backward trajectories. The results are illustrated in the right panel of Figure 4.3. The backward simulator mitigates the degeneracy problem to a large extent and results in more reliable credibility intervals.

The generality of Algorithm 12 suggests that the method can be used for a very wide range of problems. While this is indeed true, it should be noted that its usefulness will depend heavily on the properties of the problem at hand.

[1] For clarity of illustration, we sort the data points in increasing order of the regressors. However, this is not necessary and the algorithms can be applied using any ordering of the data points.

To start with, the computational complexity of Algorithm 12 is $O(MN)$, just as for the original formulation of the FFBSi. Unfortunately, for many models outside the class of SSMs, the rejection sampling technique applied in Section 3.3 is of limited use. The reason is that Algorithm 12 samples complete trajectories at each iteration. Hence, the sampling is done in a high-dimensional space and a rejection sampler will suffer from the curse of dimensionality, with very low acceptance probabilities as a result. Furthermore, the computation of the weights (Equation (4.14)) will for many models scale with T (see Section 4.5). This differs from the SSM setting, where the weights (Equation (3.5)) are independent of T. Whether or not this is prohibitive clearly depends on the application.

Another pitfall to look out for is that the backward sampling weights can get badly skewed when there are strong and long-ranging dependencies among the variables $x_{1:T}$. If this is the case, there is a high probability that the backward simulator does not alter the trajectories generated in the forward pass to any considerable degree. The problem is illustrated in the example below.

Example 4.3 (Gaussian process regression, cont'd). Consider again the GP regression problem of Example 4.2. Assume that we want to impose further smoothness constraints on the latent process x_t. This can be done by modifying the covariance kernel of the GP prior. For instance, we can use a squared exponential kernel,

$$\kappa(\xi, \xi') = \exp\left(-\frac{(\xi - \xi)^2}{2\ell^2}\right),$$

which results in a very smooth process [118]. The same fixed length scale as before is used, $\ell = 0.1$. We apply an SMC sampler and a backward simulator to the data with $N = 500$ particles and $M = 100$ backward trajectories. The results are given in Figure 4.4. For this choice of covariance kernel, the results are quite different from what we experienced in Example 4.2. Most notably, the backward simulator does not provide any significant improvement over just running SMC in the forward direction, and the path degeneracy problem is still present. To generate a backward trajectory, at each time t we consider the union of

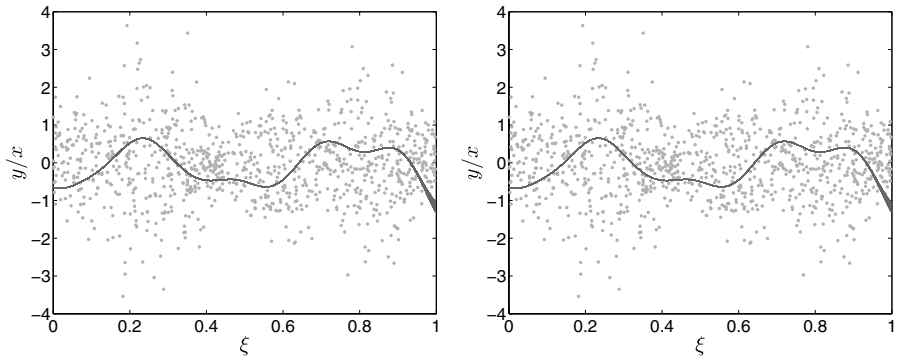

Fig. 4.4 Regression results using SMC; (left) applied only in the forward direction; (right) complemented with a backward simulation pass. The dots show the observations and the gray lines the generated trajectories, representing the log-volatility.

the partial backward trajectory $\widetilde{x}_{t+1:T}$ with partial forward trajectories $x^i_{1:t}$. These unions, $\{x^i_{1:t}, \widetilde{x}_{t+1:T}\}$, are used to compute the weights $\widetilde{w}^i_{t|T}$; see Equation (4.14). In order to improve the particle diversity in the backward pass, we rely on having several possible candidates among the partial forward trajectories, i.e., on having several weights $\widetilde{w}^i_{t|T}$ which are significantly larger than zero. However, the long-ranging dependencies imposed by the squared exponential kernel tend to result in weights close to zero, except for the one particle $x^i_{1:t}$ which was the ancestor of $\widetilde{x}_{t+1}$ in the forward pass. Hence, the backward simulator will with high probability just recover the ancestral paths from the SMC sampler, and consequently suffer from the same degeneracy problems.

The problem illustrated in the example above is caused by the strong dependence between the partial backward and partial forward trajectories. It was illustrated on the GP regression problem with a squared exponential kernel, because the long-ranging dependencies imposed by this kernel (with this specific length scale) made the problem very distinct. However, it should be noted that the same effect can be experienced when there is a strong dependence between consecutive states, even when the dependence is not long ranging. In fact, even in a Markovian SSM, we might get skewed weights when the transition density f is "peaky". The extreme case is of course when

the transition kernel is degenerate, a case which is discussed in more detail in Section 4.6.1.

4.4 Rao–Blackwellized FFBSi

To illustrate the applicability of the general backward simulator, we consider a special case which is of particular interest. A popular approach to increase the efficiency of SMC samplers for SSMs is to marginalize over one component of the state, and apply an SMC sampler in the lower dimensional marginal space. This leads to what is known as the Rao–Blackwellized particle filter (RBPF) [25, 39, 125]. It has been shown that the RBPF always will have lower asymptotic variance than the corresponding non-marginalized filter [28, 96]. Intuitively, this can be understood by the fact that in an RBPF, the particles are spread in a lower dimensional space and will thus provide a denser point-mass approximation of the target distribution. The RBPF has been successfully applied to a range of applications, e.g., simultaneous localization and mapping [105, 106], aircraft positioning [125], underwater navigation [82], communications [26, 139] and audio source separation [4].

The same approach has also been applied to state smoothing [53, 92, 123, 142], but it turns out that Rao–Blackwellization is less straightforward in this case. The reason is that the marginal state-process will be non-Markov. As an example, consider the following conditionally linear Gaussian state-space (CLGSS) model,

$$x_{t+1} \sim f(x_{t+1} \mid x_t), \tag{4.18a}$$

$$z_{t+1} = A(x_t)z_t + F(x_t)v_t, \qquad v_t \sim \mathcal{N}(0, I), \tag{4.18b}$$

$$y_t = C(x_t)z_t + e_t, \qquad e_t \sim \mathcal{N}(0, R(x_t)). \tag{4.18c}$$

If the state-variable x_t can take only a finite number of values, the above model reduces to a jump Markov linear system where x_t is a mode variable determining which linear mode that is active at time t. The CLGSS model is thus a generalization of a switching system, with a (possibly) continuous mode variable.

The RBPF exploits the tractable substructure of the model through the factorization $p(z_t, x_{1:t} \mid y_{1:t}) = p(z_t \mid x_{1:t}, y_{1:t})p(x_{1:t} \mid y_{1:t})$. Since the

model is CLGSS, the first factor of this expression is Gaussian and it can be evaluated by running a conditional Kalman filter (see e.g., [125]). That is,

$$p(z_t \mid x_{1:t}, y_{1:t}) = \mathcal{N}(z_t; \bar{z}_{t|t}(x_{1:t}), P_{t|t}(x_{1:t})), \tag{4.19}$$

for some tractable sequences of mean and covariance functions. The factor $p(x_{1:t} \mid y_{1:t})$ is targeted by an SMC sampler and it is approximated by a weighted particle system $\{x_{1:t}^i, w_t^i\}_{i=1}^N$. Consequently, we obtain the Gaussian mixture approximation,

$$p(dz_t, dx_{1:t} \mid y_{1:t}) \approx \sum_{i=1}^{N} w_t^i \mathcal{N}(dz_t; \bar{z}_{t|t}^i, P_{t|t}^i) \delta_{x_{1:t}^i}(dx_{1:t}), \tag{4.20}$$

where $\bar{z}_{t|t}^i = \bar{z}_{t|t}(x_{1:t}^i)$ and $P_{t|t}^i = P_{t|t}(x_{1:t}^i)$. Hence, the RBPF is an SMC sampler in which each particle is equipped with a Kalman filter, tracking the sufficient statistics for the conditional Gaussian densities (Equation (4.19)).

Equivalently, we can view the marginalization of the z-process as a way to reduce the model (Equation (4.18)) to,

$$x_{t+1} \sim f(x_{t+1} \mid x_t), \tag{4.21a}$$

$$y_t \sim p(y_t \mid x_{1:t}, y_{1:t-1}). \tag{4.21b}$$

Similar to Equation (4.19), the conditional density in Equation (4.21b) is Gaussian and it can be evaluated for any fixed marginal state trajectory $x_{1:t}$ by running a Kalman filter. The RBPF is then simply an SMC sampler targeting the sequence of conditional densities $p(x_{1:t} \mid y_{1:t})$ for the reduced latent variable model (Equation (4.21)). This model shares many of the properties with the SSM (Equation (1.2)). However, as an effect of the marginalization of the z-process, the measurement model (Equation (4.21b)) depends on the complete history $x_{1:t}$ and the backward simulators derived in Section 3 are not applicable.

To construct a Rao–Blackwellized FFBSi particle smoother, we instead make use of the general backward simulator in Algorithm 12 (page 74). With T being some final time point, the target distribution is given by $p(x_{1:T} \mid y_{1:T})$. To compute the weights (Equation (4.14)),

we need to evaluate the ratio

$$\frac{\bar{\gamma}_T(x_{1:T})}{\bar{\gamma}_t(x_{1:t})} = \frac{p(x_{1:T}, y_{1:T})}{p(x_{1:t}, y_{1:t})} = p(x_{t+1:T}, y_{t+1:T} \mid x_{1:t}, y_{1:t}) \propto p(y_{t+1:T} \mid x_{1:T}, y_{1:t}) f(x_{t+1} \mid x_t). \tag{4.22}$$

It remains to find an expression for the first factor of this expression (up to proportionality). In fact, this predictive density can be computed straightforwardly by running a conditional Kalman filter from time t up to T. However, using this approach to calculate the weights at time t would require N separate Kalman filters to run over $T - t$ time steps, resulting in a total computational complexity scaling quadratically with T. For this specific model, it is possible to do better.

The idea is to propagate a set of statistics backward in time, as the backward trajectory $\widetilde{x}_{1:T}$ is generated. This approach has been used for MCMC sampling in [59] and it has been adapted to Rao–Blackwellized backward simulation in [142, 123]. In [92], a similar method is derived for a different type of CLGSS model, in which there is a dependence on z_t in the updating Equation (4.18a).

For the derivation below, we use the notation $\|\mu\|_\Omega^2 \triangleq \mu^\mathsf{T}\Omega\mu$, where μ is a vector and $\Omega \succeq 0$ is a positive semidefinite matrix. To compute Equation (4.22) we consider the expression,

$$p(y_{t+1:T} \mid x_{1:T}, y_{1:t}) = \int p(y_{t+1:T} \mid z_{t+1}, x_{t+1:T}) p(z_{t+1} \mid x_{1:t}, y_{1:t})\, dz_{t+1}. \tag{4.23}$$

The second factor of the integrand is given by a one-step prediction of the RBPF, similar to Equation (4.19),

$$p(z_{t+1} \mid x_{1:t}, y_{1:t}) = \mathcal{N}(z_{t+1}; \bar{z}_{t+1|t}(x_{1:t}), P_{t+1|t}(x_{1:t})). \tag{4.24}$$

For later reference, we introduce a Cholesky factorization of the predictive covariance, $P_{t+1|t}(x_{1:t}) = \Gamma_{t+1|t}(x_{1:t})\Gamma_{t+1|t}(x_{1:t})^\mathsf{T}$.

The key quantity is the first factor of the integrand in Equation (4.23). We will show, by induction, that for any $t = 1, \ldots, T-1$,

$$p(y_{t+1:T} \mid z_{t+1}, x_{t+1:T}) \propto \exp\left(-\tfrac{1}{2}\left(\|z_{t+1}\|_{\Omega_{t+1}}^2 - 2\lambda_{t+1}^\mathsf{T} z_{t+1}\right)\right) \tag{4.25}$$

for some $\Omega_{t+1} \succeq 0$ and λ_{t+1} that only depend on $y_{t+1:T}$ and $x_{t+1:T}$.

First, we give a technical lemma. The proof is omitted for brevity, but follows straightforwardly by carrying out the integration.

Lemma 4.1. Let $z = m + \Gamma\xi$ with $\xi \sim \mathcal{N}(0, I)$ and let $\Omega \succeq 0$. Then

$$\begin{aligned}\mathbb{E}\left[\exp\left(-\tfrac{1}{2}\left(\|z\|_\Omega^2 - 2\lambda^\mathsf{T} z\right)\right)\right] \\ = |M|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}\left(\|m\|_\Omega^2 - 2\lambda^\mathsf{T} m - \|\Gamma^\mathsf{T}(\lambda - \Omega m)\|_{M^{-1}}^2\right)\right),\end{aligned}$$

with $M = \Gamma^\mathsf{T}\Omega\Gamma + I$.

To simplify the notation, we will write A_t for $A(x_t)$ and similarly for other functions. First, we note that, by Equation (4.18c),

$$p(y_t \mid z_t, x_t) \propto \exp\left(-\tfrac{1}{2}\left(z_t^\mathsf{T} C_t^\mathsf{T} R_t^{-1} C_t z_t - 2(C_t^\mathsf{T} R_t^{-1} y_t)^\mathsf{T} z_t\right)\right). \tag{4.26}$$

Hence, at time T, Equation (4.25) holds with

$$\Omega_T = C_T^\mathsf{T} R_T^{-1} C_T, \tag{4.27a}$$

$$\lambda_T = C_T^\mathsf{T} R_T^{-1} y_T. \tag{4.27b}$$

Assume that Equation (4.25) holds at time $t+1$ and consider the factorization,

$$p(y_{t:T} \mid z_t, x_{t:T}) = p(y_{t+1:T} \mid z_t, x_{t:T}) p(y_t \mid z_t, x_t). \tag{4.28}$$

The first factor in Equation (4.28) is given by

$$\begin{aligned}p(y_{t+1:T} \mid z_t, x_{t:T}) &= \int p(y_{t+1:T} \mid z_{t+1}, z_t, x_{t:T}) p(z_{t+1} \mid z_t, x_{t:T})\, dz_{t+1} \\ &= \int p(y_{t+1:T} \mid z_{t+1}, x_{t+1:T}) p(z_{t+1} \mid z_t, x_t)\, dz_{t+1}.\end{aligned} \tag{4.29}$$

Under the induction hypothesis and using Lemma 4.1 and Equation (4.18b),

$$p(y_{t+1:T} \mid z_t, x_{t:T}) \propto \exp\left(-\tfrac{1}{2}\xi_t\right), \tag{4.30a}$$

with

$$\xi_{t+1} = \|A_t z_t\|_{\Omega_{t+1}}^2 - 2\lambda_{t+1}^\mathsf{T} A_t z_t - \|F_t^\mathsf{T}(\lambda_{t+1} - \Omega_{t+1} A_t z_t)\|_{\Lambda_t^{-1}}^2 \tag{4.30b}$$

and $\Lambda_t = F_t^\mathsf{T}\Omega_{t+1}F_t + I$. Combining Equations (4.26) and (4.30) in Equation (4.28) we get $p(y_{t:T} \mid z_t, x_{t:T}) \propto \exp\left(-\frac{1}{2}\left(\|z_t\|^2_{\Omega_t} - 2\lambda_t^\mathsf{T} z_t\right)\right)$ with,

$$\Omega_t = A_t^\mathsf{T}(I - \Omega_{t+1}F_t\Lambda_t^{-1}F_t^\mathsf{T})\Omega_{t+1}A_t + C_t^\mathsf{T}R_t^{-1}C_t, \tag{4.31a}$$

$$\lambda_t = A_t^\mathsf{T}(I - \Omega_{t+1}F_t\Lambda_t^{-1}F_t^\mathsf{T})\lambda_{t+1} + C_t^\mathsf{T}R_t^{-1}y_t, \tag{4.31b}$$

which completes the induction.

Finally, applying Lemma 4.1 to Equation (4.23), with the integrands given by Equations (4.24) and (4.25) results in, $p(y_{t+1:T} \mid x_{1:T}, y_{1:t}) \propto |M_t|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\eta_t\right)$, with

$$M_t = \Gamma_{t+1|t}^\mathsf{T}\Omega_{t+1}\Gamma_{t+1|t} + I, \tag{4.32a}$$

$$\eta_t = \|\bar{z}_{t+1|t}\|^2_{\Omega_{t+1}} - 2\lambda_{t+1}^\mathsf{T}\bar{z}_{t+1|t} - \|\Gamma_{t+1|t}^\mathsf{T}(\lambda_{t+1} - \Omega_{t+1}\bar{z}_{t+1|t})\|^2_{M_t^{-1}}. \tag{4.32b}$$

Using the above expression we can compute the backward sampling weights through Equation (4.22). We present the adaption of the general backward simulator (Algorithm 12) to Rao–Blackwellized particle smoothing in Algorithm 13.

4.5 Non-Markovian Latent Variable Models

By marginalization of the CLGSS model (Equation (4.18)) we obtained the formulation in Equation (4.21). This model is in fact a special case of an important generalization of SSMs, the class of non-Markovian latent variable models is given by

$$x_{t+1} \sim f(x_{t+1} \mid x_{1:t}), \tag{4.33a}$$

$$y_t \sim g(y_t \mid x_{1:t}). \tag{4.33b}$$

Similar to the SSM (Equation (1.2)), this model is characterized by a latent process $x_t \in \mathsf{X}$ and an observed process $y_t \in \mathsf{Y}$. However, it does not share the conditional independence properties that are central to SSMs. Instead, both the transition density f and the measurement density g may depend on the entire past history of the state trajectory.

In Section 4.6 we will see additional examples in which non-Markovian models arise by some manipulation of an SSM, similar to

Algorithm 13 Rao–Blackwellized FFBSi

Input: Forward filter particle systems $\{x_t^i, w_t^i\}_{i=1}^N$ for $t = 1, \ldots, T$ and linear state statistics $\{\bar{z}_{t+1|t}^i, \Gamma_{t+1|t}^i\}_{i=1}^N$ for $t = 1, \ldots, T-1$.

Output: Backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^M$.

Sample independently $\{b_T(j)\}_{j=1}^M \sim \mathrm{Cat}\left(\{w_T^i\}_{i=1}^N\right)$.
Set $\widetilde{x}_T^j = x_T^{b_T(j)}$ for $j = 1, \ldots, M$.
Compute $\{\Omega_T^j, \lambda_T^j\}$ according to (4.27) for $j = 1, \ldots, M$.
for $t = T-1$ **to** 1 **do**
  for $j = 1$ **to** M **do**
    for $i = 1$ **to** N **do**
      Compute $\{M_t^{i,j}, \eta_t^{i,j}\}$ according to Equation (4.32).
      Compute $\widetilde{w}_{t|T}^{i,j} \propto w_t^i |M_t^{i,j}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\eta_t^{i,j}\right) f(\widetilde{x}_{t+1}^j \mid x_t^i)$.
    end for
    Normalize the smoothing weights $\{\widetilde{w}_{t|T}^{i,j}\}_{i=1}^N$ to sum to one.
    Draw $b_t(j) \sim \mathrm{Cat}\left(\{\widetilde{w}_{t|T}^{i,j}\}_{i=1}^N\right)$.
    Set $\widetilde{x}_t^j = x_t^{b_t(j)}$ and $\widetilde{x}_{t:T}^j = \{\widetilde{x}_t^j, \widetilde{x}_{t+1:T}^j\}$.
    Compute $\{\Omega_t^j, \lambda_t^j\}$ according to (4.31).
  end for
end for

the marginalization discussed in the previous section. Another example is the GP regression model considered in Section 4.1.3, which also can be written on the form of Equation (4.33).

With the target density being $\gamma_t(x_{1:t}) = p(x_{1:t} \mid y_{1:t})$, this model fits into the framework presented in Section 4.3. As we saw in the previous section, to compute the backward simulation weights (Equation (4.14)) we need to evaluate the ratio

$$\frac{\bar{\gamma}_T(x_{1:T})}{\bar{\gamma}_t(x_{1:t})} = \frac{p(x_{1:T}, y_{1:T})}{p(x_{1:t}, y_{1:t})} = \prod_{s=t+1}^{T} g(y_s \mid x_{1:s}) f(x_s \mid x_{1:s-1}). \tag{4.34}$$

For the CLGSS model (Equation (4.18)), we found a backward recursion for a set of statistics, which enabled the evaluation of this expression in constant time. However, in the general case, the computational cost of evaluating Equation (4.34) will increase with T. For instance,

if the functions f and g can be evaluated in constant time, then the computational cost of evaluating Equation (4.34) scales linearly with T. This implies that the cost of generating a full backward trajectory is $O(T^2)$. Since T is typically large, an $O(T^2)$ computational complexity can be prohibitive for many applications.

One way to mitigate this issue has been proposed in [93]. They consider non-Markovian models in which there is a decay in the influence of the past on the present, akin to that in SSMs but without the strong Markovian assumption. It is thus possible to obtain a useful approximation of Equation (4.34) by truncating the product to a smaller number of factors. In [93], an adaptive method is proposed, in which the factors of Equation (4.34) are computed sequentially until some criterion is met, after which the product is truncated. They also propose a specific backward-simulation-based inference method, particle Gibbs with ancestor sampling, which is suitable for inference in this type of models. This method will be reviewed in detail in Section 5.5.

4.6 From State-Space Models to Non-Markovian Models

Rao–Blackwellization, as considered in Section 4.4, resulted in a non-Markovian latent variable model. To provide further insight into this model class, we present a few additional examples of SSMs for which backward simulation is problematic in the original formulation. A transformation to a non-Markovian model is then useful in order to enable backward simulation, by using the general sampler provided in Algorithm 12 on page 74.

4.6.1 Degenerate State-Space Models

As was pointed out in Section 3, the approximation of the backward kernel (Equation (3.2)) relies on the assumption that the model under study is fully dominated. Hence, the backward simulators for SSMs derived in Section 3 are not applicable for degenerate models. This can be understood by noting that, if the model is degenerate, then so is the backward kernel. Consequently, the support of the backward kernel is limited to some low-dimensional subspace or manifold, embedded in the state-space. Due to this, it cannot be approximated in a natural

way by the forward filter particles. We illustrate the problem in the following example.

Example 4.4 (Degenerate backward kernel). To see why backward simulation is problematic for degenerate models, we consider a simple toy model. Let the state be given by $x_t = (x_{1,t}, x_{2,t})^\mathsf{T} \in \mathbb{R}^2$. The initial state is distributed according to $x_1 \sim \mathcal{N}(0, I)$ and the evolution of the state process is given by $x_{t+1} = x_t + v_t$ with,

$$v_t \sim \mathcal{N}\left(\begin{pmatrix}10\\0\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}\right).$$

Since the process noise covariance is singular, the model is degenerate (see Section 1.4). Still, it is straightforward to apply SMC to this model. Assume that we apply a bootstrap PF with $N = 10$ particles. The particles generated at times $t = 1$ and $t = 2$ are shown in Figure 4.5.

From the definition (Equation (1.10)), the backward kernel at time $t = 1$ is given by $B_1(A \mid \widetilde{x}_2, y_1) = P(x_1 \in A \mid \widetilde{x}_2, y_1)$. Since the transition kernel is degenerate, it holds that $B_1(\{x_1 \in \mathbb{R}^2 : x_{1,2} \neq \widetilde{x}_{2,2}\} \mid \widetilde{x}_2, y_1) = 0$. That is, for a given particle $\widetilde{x}_2$ there is, with probability 1, only one particle at time $t = 1$ which is contained in the support of the backward kernel $B_1(\cdot \mid \widetilde{x}_2, y_1)$, namely the ancestor particle of $\widetilde{x}_2$.

The same argument can be applied for any time t. If we apply a backward simulator to this model, we will only recover the genealogy

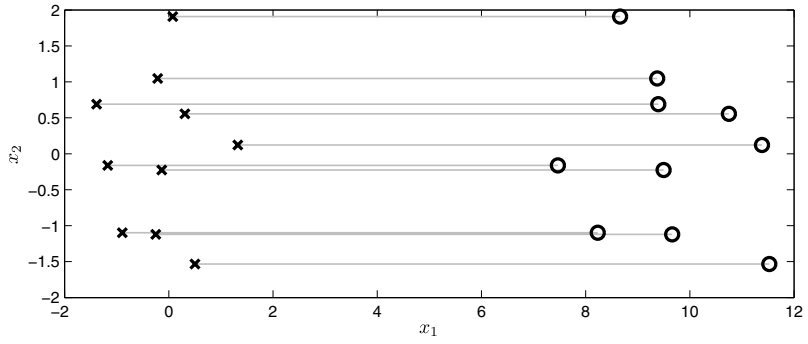

Fig. 4.5 Particles generated at time $t = 1$ (crosses) and at time $t = 2$ (circles) together with the ancestral dependence (gray lines).

of the forward PF and the backward simulator is not able to improve upon the degenerate paths of the forward filter. For this specific model, the $x_{2,t}$-state is completely deterministic given the initial state. However, the same problem will arise whenever the transition kernel is degenerate, but in general with more complicated constraints on the individual state components as a result.

This restriction of backward simulators for SSMs is unfortunate, since many interesting dynamical systems are most naturally modeled as degenerate. For instance, consider a nonlinear system with additive noise on the form,

$$\xi_{t+1} = f_\xi(\xi_t) + G\nu_t, \tag{4.35a}$$
$$y_t = h_\xi(\xi_t) + e_t, \tag{4.35b}$$

where G is a tall matrix and, consequently, $\operatorname{rank}(G) < \dim(\xi_t)$. Hence, the process noise covariance matrix GG^T is singular. SMC samplers can straightforwardly be applied to this type of model, but as pointed out above, smoothing via backward simulation is more problematic.

To alleviate the problem, we can recast the degenerate SSM as a non-Markovian model in a lower dimensional space. For the model considered in Example 4.4, the state $x_{2,t}$ is superfluous. Hence, the model can be reformulated by removing $x_{2,t}$ and running the backward simulator only for the $x_{1,t}$-state. A similar approach can be applied to the additive noise model (Equation (4.35)). Let $G = U \begin{bmatrix}\Sigma & 0\end{bmatrix}^\mathsf{T} V^\mathsf{T}$ with unitary U and V be a singular value decomposition of G. Furthermore, let,

$$\begin{bmatrix} x_{t+1} \\ z_{t+1} \end{bmatrix} \triangleq U^\mathsf{T}\xi_{t+1} = U^\mathsf{T} f(UU^\mathsf{T}\xi_t) + \begin{bmatrix} \Sigma V^\mathsf{T}\nu_t \\ 0 \end{bmatrix}. \tag{4.36}$$

Hence, with $v_t \triangleq \Sigma V^\mathsf{T}\nu_t$ and by appropriate definitions of the functions f_x, f_z and h, the model (Equation (4.35)) can be rewritten as

$$x_{t+1} = f_x(x_t, z_t) + v_t, \tag{4.37a}$$
$$z_{t+1} = f_z(x_t, z_t), \tag{4.37b}$$
$$y_t = h(x_t, z_t) + e_t. \tag{4.37c}$$

For simplicity, assume that z_1 is known. If this is not the case, z_1 can be included in the system state or treated as an unknown static parameter of the model. Hence, the sequence $z_{1:t}$ is $\sigma(x_{1:t-1})$-measurable and we can write $z_t = z_t(x_{1:t-1})$. We can then further rewrite the model as,

$$x_{t+1} = f_x(x_{1:t}) + v_t, \tag{4.38a}$$

$$y_t = h(x_{1:t}) + e_t, \tag{4.38b}$$

which is a non-degenerate, non-Markovian model. This reformulation illustrates the intimate relationship between degenerate models and non-Markovian models. In fact, this is nothing but another application of marginalization as discussed in Section 4.4, where the z-state is conditionally deterministic and thus trivially marginalizable.

The model (Equation (4.38)) will in general not allow for a simple evaluation of Equation (4.34) in constant time. Hence, the computational complexity for applying backward simulation in this model is $O(T^2)$. Whether or not this is prohibitive clearly depends upon the application. One possibility to reduce the complexity is to make use of a truncation as discussed in Section 4.5.

4.6.2 Collapsing of State-Space Models

Another subclass of SSMs, for which backward simulation in problematic, is that in which the transition density function f is not available on closed form. It is quite common that the transition density f can be simulated from, but not evaluated point-wise, see e.g., [50, 63, 67, 108]. If this is the case, it is not possible to evaluate the backward sampling weights (Equation (3.5)), since these depend explicitly on f.

To find a way around this, we note that if $f(x_t \mid x_{t-1})$ can be simulated from, then this is typically done by generating a random variable $v_t \sim p_v(v_t)$ and computing $x_t = a(x_{t-1}, v_t)$ for some function a. By a similar argument as in the previous section, it follows that x_t is $\sigma(v_{1:t})$-measurable. Hence, the model can be written as

$$v_t \sim p_v(v_t), \tag{4.39a}$$

$$y_t \sim g(y_t \mid v_{1:t-1}), \tag{4.39b}$$

which is a non-Markovian model with latent variables $v_t \in \mathsf{V}$. This formulation has been exploited by [67, 108] to construct auxiliary particle

filters for SSMs in which the transition density function is not available on closed form. By using Algorithm 12 we can employ backward simulation for the model (Equation (4.39)), thus simulating the innovation process $v_{1:T}$. Again, the computational complexity for evaluating the backward sampling weights will be $O(T^2)$ in the general case, but truncation can be used to reduce the complexity at the cost of an approximation error.

4.6.3 Non-centered Parameterizations

A different reason for reformulating an SSM into the form of Equation (4.39), can arise in the context of parameter inference using data augmentation, e.g., Gibbs sampling. Assume that we have a parametric model of an SSM as in Equation (1.3). For simplicity, we assume that only the transition density function (Equation (1.3a)) is parameterized by θ. In the Bayesian setting, i.e., with θ modeled as a random variable, a graphical model for this SSM is given in Figure 4.6 (left).

In [114], this is referred to as a *centered* parameterization. This parameterization suggest a Gibbs sampler according to;

(i) Draw $\theta' \sim p(\theta \mid x_{1:T}, y_{1:T})$;
(ii) Draw $x'_{1:T} \sim p(x_{1:T} \mid \theta', y_{1:T})$.

The conditioning on $x_{1:T}$ in Step (i) typically allows for a fairly straightforward updating of θ. Step (ii) can be addressed by using a backward simulator. In particular, the PMCMC framework (see Section 5 and, in particular, Sections 5.4–5.5) provides the means of using SMC-based backward simulators within Gibbs sampling.

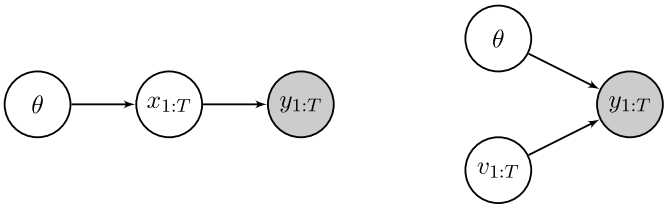

Fig. 4.6 Graphical model of a centered parameterization (left) and a non-centered parameterization (right) of an SSM.

However, as can be seen in Figure 4.6, $x_{1:T}$ and θ exhibit *a priori* dependence. In many cases, this dependence can be very strong, resulting in slow convergence of the Gibbs sampler outlined above. By instead making use of the formulation in Equation (4.39), we can choose the variables $v_{1:T}$ to be *a priori* independent of θ. The graphical model is given in Figure 4.6 (right). This is referred to as a *non-centered* parameterization, and it results in the Gibbs sampler:

(i)′ Draw $\theta' \sim p(\theta \mid v_{1:T}, y_{1:T})$;
(ii)′ Draw $v'_{1:T} \sim p(v_{1:T} \mid \theta', y_{1:T})$.

Often, the non-centered parameterization can result in a Gibbs sampler with better convergence properties than the centered parameterization, though the opposite is also possible (see [114] for a discussion). However, Steps (i)′–(ii)′ are typically more computationally involved than Steps (i)–(ii). In particular, Step (ii)′ requires the simulation of a posterior sample from the innovation process $v_{1:T}$. As pointed out above, backward simulation in the model (Equation (4.39)) can lead to a high computational complexity. Consequently, the potential benefit of using a non-centered parameterization has to be evaluated on a case-by-case basis.

5

Backward Simulation in Particle MCMC

Particle Markov chain Monte Carlo (PMCMC) is a systematic way of combining the two main tools used in Monte Carlo statistical inference: SMC and MCMC. This is a class of inferential methods, introduced in [3], in which SMC is used to construct proposal mechanisms for MCMC samplers. Within this framework, backward simulation has proved to be an important component. This section is devoted to PMCMC, and in particular to how we can benefit from using backward simulation within this framework.

5.1 Introduction to PMCMC

In the previous sections, we have mostly been focusing on state-inference under the assumption that the parameters of the model are known (or at least fixed). However, what we are often interested in is to make posterior inference about some static parameter $\theta \in \Theta$ of the model; see Section 1.5.

As an example, consider the parameterized SSM (Equation (1.3)), with prior density $\pi(\theta)$ for the parameter. Given a batch of observations $y_{1:T}$, we seek the posterior density $p(\theta \mid y_{1:T}) \propto p(y_{1:T} \mid \theta)\pi(\theta)$.

This posterior is typically not available in closed form. Furthermore, for a general nonlinear/non-Gaussian SSM, it is problematic to design an MCMC sampler targeting $p(\theta \mid y_{1:T})$ directly, since this would require us to analytically marginalize the states $x_{1:T}$. As noted in Section 1.5, a possible remedy is to include the latent states $x_{1:T}$ as auxiliary variables and instead target the joint posterior $p(\theta, x_{1:T} \mid y_{1:T})$ with an MCMC sampler. For example, this opens up for Gibbs sampling using the following scheme:

(i) Draw $\theta' \sim p(\theta \mid x_{1:T}, y_{1:T})$;
(ii) Draw $x'_{1:T} \sim p(x_{1:T} \mid \theta', y_{1:T})$.

Sampling θ in Step (i), i.e., conditionally on both the observed data and the latent states, is in general much easier than sampling θ conditionally only on the observed data. In fact, if conjugate priors are used, this step can be carried out exactly. For non-conjugate models, one option is to replace Step (i) with a Metropolis–Hastings step, which is possible since the unnormalized density $p(\theta, x_{1:T}, y_{1:T})$ can be evaluated point-wise.

Step (ii) of the above Gibbs sampler, however, is less straightforward since it requires us to generate a sample from the JSD for a fixed value of the system parameter. In some special cases this can be done exactly. For instance, if the model is linear Gaussian, we can make use of the backward simulator for LGSS models derived in Section 1.7 to sample $x'_{1:T}$.

For a general nonlinear/non-Gaussian model, on the other hand, the JSD is not available and the second step of the Gibbs sampler cannot be carried out. However, with the material of Sections 3 and 4 in mind, a natural idea is to employ SMC to approximately sample from the JSD, using the backward simulator of Algorithm 4 on page 38.

However, to simply replace Step (ii) of the Gibbs sampler with a backward simulator will result in a method suffering from some serious drawbacks. Since it is based on SMC, we only obtain an approximate sample from the JSD by running Algorithm 4. This will introduce additional bias and it is not clear how the error will propagate through the Gibbs sampler. As a consequence, the success of this approach relies heavily on using a large number of particles N in the underlying SMC sampler, to obtain accurate backward kernel approximations.

This might seem like a natural requirement, but it also means that we waste a lot of computational resources. The reason is that, at each iteration of the Gibbs sampler, we run an SMC sampler with a large number of particles, extract a single-state trajectory and then discard all the remaining particles. As we will see in Section 5.4, it is possible to do much better if we, instead of viewing the Gibbs sampler and the SMC sampler as two separate components, analyze them together.

We will start our exploration of the PMCMC framework by reviewing the particle marginal Metropolis–Hastings (PMMH) sampler [3] in Section 5.2. PMMH is an example of a pseudo-marginal method, meaning that it uses unbiased likelihood estimates in place of exact likelihoods in an MCMC sampler [6, 11]. Somewhat surprisingly, this will not alter the stationary distribution of the chain. PMMH is a good starting point for introducing PMCMC, and we will fall back on this material in the later sections. In Section 5.3 we will see how backward simulation can be used to improve the quality of the PMMH estimates, using a method proposed in [113].

We then turn to a different member of the PMCMC family, particle Gibbs (PG), for which backward simulation has turned out to play a quite different role. In Sections 5.4 and 5.5, we will discuss two approaches to backward simulation in the context of PG, which enables us to employ this method even for large data records and using very few particles. These two sections cover material previously presented in [93, 94, 142].

5.2 Particle Marginal Metropolis–Hastings

We will work within the general framework introduced in Section 4, but now with the addition of an unknown static parameter θ. Hence, the target density on $\Theta \times \mathsf{X}^T$ is given by

$$\gamma_T(\theta, x_{1:T}) = \frac{\bar{\gamma}_T(\theta, x_{1:T})}{\mathcal{Z}_T} = \frac{\bar{\gamma}_T^\theta(x_{1:T})\pi(\theta)}{\mathcal{Z}_T}, \tag{5.1}$$

where $\pi(\theta)$ is the prior density for the parameter and $\mathcal{Z}_T$ is a normalization constant. When using θ as a superscript, it should be thought of as conditionally on θ, but for notational simplicity we prefer to write $\bar{\gamma}_T^\theta(x_{1:T})$ rather than $\bar{\gamma}_T(x_{1:T} \mid \theta)$, etc.

For fixed θ, a sequence of intermediate densities

$$\gamma_t^\theta(x_{1:t}) = \frac{\bar{\gamma}_t^\theta(x_{1:t})}{Z_t(\theta)} \tag{5.2}$$

for $t = 1, \ldots, T$ is available, analogously to Equation (4.6). Note that the normalization factor $Z_t(\theta) = \int \bar{\gamma}_t^\theta(x_{1:t})\, dx_{1:t}$ depends on θ. These densities can sequentially be approximated by the application of an SMC sampler, analogously to Section 4.2 (simply add θ to the notation for all quantities introduced in Section 4.2).

Example 5.1 (The special case of SSMs). In the special case of an SSM, we typically have

$$\gamma_T(\theta, x_{1:T}) = p(\theta, x_{1:T} \mid y_{1:T}) \quad \text{and} \quad \bar{\gamma}_T(\theta, x_{1:T}) = p(\theta, x_{1:T}, y_{1:T}).$$

Hence, Equation (5.1) corresponds to

$$p(\theta, x_{1:T} \mid y_{1:T}) = \frac{p(\theta, x_{1:T}, y_{1:T})}{p(y_{1:T})} = \frac{p(x_{1:T}, y_{1:T} \mid \theta)\pi(\theta)}{p(y_{1:T})},$$

with the normalization constant $\mathcal{Z}_T = p(y_{1:T})$. Similarly, for fixed θ, the intermediate quantities of the SMC sampler are the JSDs, so Equation (5.2) corresponds to

$$p(x_{1:t} \mid \theta, y_{1:t}) = \frac{p(x_{1:t}, y_{1:t} \mid \theta)}{p(y_{1:t} \mid \theta)},$$

with the conditional normalization constant $Z_t(\theta) = p(y_{1:t} \mid \theta)$.

For large T, the high dimension of the space $\Theta \times \mathsf{X}^T$ is prohibitive when designing an MCMC sampler for the target density in Equation (5.1). As previously pointed out, we will try to alleviate this by constructing a proposal based on SMC, and use this within a Metropolis–Hastings sampler. To start with, consider a proposal kernel on $\Theta \times \mathsf{X}^T$ given by,

$$q(x'_{1:T}, \theta' \mid \theta, x_{1:T}) = q(\theta' \mid \theta)\gamma_T^{\theta'}(x'_{1:T}). \tag{5.3}$$

This proposal makes use of a marginal Markov kernel $q(\theta' \mid \theta)$ on Θ and is perfectly adapted to the target on X^T. This proposal cannot be used in practice, since the second factor on the right-hand side, $\gamma_T^\theta(x_{1:T})$, is assumed to be unavailable. However, this factor can be approximated

using SMC, resulting in an empirical point-mass distribution, similar to Equation (4.10),

$$\widehat{\gamma}_T^{\theta,N}(dx_{1:T}) \triangleq \sum_{i=1}^{N} w_T^i \delta_{x_{1:T}^i}(dx_{1:T}). \tag{5.4}$$

In order to mimic the proposal kernel in Equation (5.3), we can thus make use of the following sampling strategy:

(i) Draw $\theta' \sim q(\theta' \mid \theta)$;
(ii) Parameterize the model with θ' and run an SMC sampler (Algorithm 11, page 72) targeting $\gamma_T^{\theta'}(x_{1:T})$;
(iii) Draw k with $P(k = i) = w_T^i$ and set $x'_{1:T} = x_{1:T}^k$.

Note that the particle trajectory generated in Step (iii) is the ancestral path of particle x_T^k, as discussed in Example 4.1. That is, we can write

$$x_{1:T}^k = x_{1:T}^{b_{1:T}} \triangleq (x_1^{b_1}, \ldots, x_T^{b_T}), \tag{5.5}$$

where the indices $b_{1:T}$ are given recursively by the ancestor indices: $b_T = k$ and $b_t = a_{t+1}^{b_{t+1}}$.

The above sampling procedure can straightforwardly be implemented and it defines a proposal kernel on $\Theta \times \mathsf{X}^T$, akin to the "ideal" kernel (Equation (5.3)). The problem, however, in using this proposal mechanism in a Metropolis–Hastings sampler lies in the computation of the acceptance probability. To compute the acceptance probability, we need to be able to evaluate the proposal kernel density point-wise. For the sampling scheme defined by Steps (i)–(iii) above, the proposal kernel is given by

$$q(d\theta' \mid \theta)\mathbb{E}\left[\sum_{i=1}^{N} w_T^i \delta_{x_{1:T}^i}(dx_{1:T})\right], \tag{5.6}$$

where the expectation is taken w.r.t. the randomness of the SMC sampler (the weights $\{w_T^i\}_{i=1}^N$ and the particles $\{x_{1:T}^i\}_{i=1}^N$ are random variables). In other words, to be able to evaluate the proposal kernel, we have to marginalize over all the random variables generated by the SMC sampler, which is an utterly hopeless task.

The solution to this problem, as often, lies in including the random variables generated by the SMC sampler as auxiliary variables in the

sampling scheme. By doing so, we avoid the need to marginalizing over them. For this cause, we introduce the boldface notation

$$\mathbf{x}_t = \{x_t^1, \ldots, x_t^N\}, \quad \mathbf{a}_t = \{a_t^1, \ldots, a_t^N\}, \tag{5.7}$$

to refer to all the particles and ancestor indices, respectively, generated by the SMC sampler at time t. It follows that, for a fixed θ, the SMC sampler in Algorithm 11 (page 72) generates a *single sample* on the extended space $\mathsf{X}^{NT} \times \{1, \ldots, N\}^{N(T-1)}$. The probability density of this sample is given by,

$$\psi^\theta(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}) \triangleq \prod_{i=1}^{N} r_1^\theta(x_1^i) \prod_{t=2}^{T} \prod_{i=1}^{N} M_t^\theta(a_t^i, x_t^i). \tag{5.8}$$

That is, at time $t = 1$, we sample $\{x_1^i\}_{i=1}^N$ independently from the proposal density $r_1^\theta(x_1)$. 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 $M_t^\theta(a_t, x_t)$. Recall that $M_t^\theta(a_t, x_t)$ depends on all the random variables generated up to time $t-1$, i.e., $\{\mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}\}$, but we will not make this dependence explicit for notational convenience.

Using the notion of sampling on an extended space, the procedure given by Steps (i)–(iii) above can be seen as generating a sample $\{\theta', \mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k\}$ in the space $\Omega \triangleq \Theta \times \mathsf{X}^{NT} \times \{1, \ldots, N\}^{N(T-1)+1}$. Furthermore, the density of these random variables on Ω is given by,

$$q(\theta' \mid \theta)\psi^{\theta'}(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}) w_T^k, \tag{5.9}$$

where the three factors, from left to right, correspond to the three steps of the sampling procedure.

We are now in an unusual situation — we wish to use the proposal kernel in Equation (5.9), which is defined on Ω, in a Metropolis–Hastings sampler. However, the target density $\gamma_T(\theta, x_{1:T})$ is defined on a lower dimensional space $\Theta \times \mathsf{X}^T \subset \Omega$. In order to do so, we define a new, extended target density on Ω. Basically, what we look for is an artificial target density, which we can call ϕ, which:

(1) Admits $\gamma_T(\theta, x_{1:T})$ as a marginal;
(2) Is, in some sense, as close as possible to the proposal.

The first requirement is necessary since we wish to use ϕ as a surrogate for the original target $\gamma_T(\theta, x_{1:T})$. If ϕ admits γ_T as a marginal and if we are able to design an MCMC sampler with stationary distribution ϕ, then we implicitly target also γ_T. More precisely, the subchain constructed by only considering the variables corresponding to this marginal will have γ_T as stationary distribution.

The second requirement is more vague, but it is crucial in order to obtain an efficient (and simple) algorithm. Guided by these requirements, we define the extended target density on Ω according to

$$\begin{aligned}\phi(\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k) &= \phi(\theta, x_{1:T}^{b_{1:T}}, b_{1:T})\phi(\mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}} \mid \theta, x_{1:T}^{b_{1:T}}, b_{1:T}) \\ &\triangleq \underbrace{\frac{\gamma_T(\theta, x_{1:T}^{b_{1:T}})}{N^T}}_{\text{marginal}} \underbrace{\prod_{\substack{i=1 \\ i \neq b_1}}^{N} r_1^\theta(x_1^i) \prod_{t=2}^{T} \prod_{\substack{i=1 \\ i \neq b_t}}^{N} M_t^\theta(a_t^i, x_t^i)}_{\text{conditional}}. \end{aligned} \tag{5.10}$$

In the above expression, we have introduced the notation $\mathbf{x}_t^{-i} = \{x_t^1, \ldots, x_t^{i-1}, x_t^{i+1}, \ldots, x_t^N\}$, $\mathbf{x}_{1:T}^{-b_{1:T}} = \{\mathbf{x}_1^{-b_1}, \ldots, \mathbf{x}_T^{-b_T}\}$ and similarly for the ancestor indices. The factorization into a marginal and a conditional density, which can be done since ϕ is a probability density function, is intended to reveal some of the structure of the extended target. First, the marginal density of the variables $\{\theta, x_{1:T}^{b_{1:T}}, b_{1:T}\}$ is defined to be equal to the original target density $\gamma_T(\theta, x_{1:T}^{b_{1:T}})$, up to a factor N^{-T} (related to the index variables $b_{1:T}$). By defining the marginal in this way, we fulfill the first of the two requirements. More precisely, if $\{\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k\}$ are distributed according to ϕ, then, by construction, the variables $\{\theta, x_{1:T}^{b_{1:T}}\}$ are distributed according to γ_T. Note that the particle trajectory $x_{1:T}^{b_{1:T}}$ is the ancestral path of x_T^k, as defined in Equation (5.5).

Second, to define the conditional density of the remaining variables, we look at the second requirement. That is, we strive to make this conditional density as close as possible to the proposal density (Equation (5.9)), or in effect as close as possible to ψ^θ defined in Equation (5.8). This is done by defining the conditional, i.e., the second factor in Equation (5.10), similar to Equation (5.8). The difference is that, since we now condition on the path $x_{1:T}^{b_{1:T}}$ and the indices $b_{1:T}$, we remove the corresponding factors from the products.

With the target density (Equation (5.10)) and the proposal (Equation (5.9)) in place, we are close to having a complete Metropolis–Hastings sampler. What remains is to compute the acceptance probability. For a proposed move from $\{\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k\}$ to $\{\theta', \mathbf{x}'_{1:T}, \mathbf{a}'_{2:T}, k'\}$, this is given by a standard Metropolis–Hastings ratio,

$$1 \wedge \frac{\phi(\theta', \mathbf{x}'_{1:T}, \mathbf{a}'_{2:T}, k')}{\phi(\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k)} \frac{q(\theta \mid \theta')\psi^{\theta}(\mathbf{x}_{1:T}, \mathbf{a}_{2:T})w_T^k}{q(\theta' \mid \theta)\psi^{\theta'}(\mathbf{x}'_{1:T}, \mathbf{a}'_{2:T})w_T'^{,k'}}. \tag{5.11}$$

To simplify this expression we will rewrite Equation (5.10) on an alternative form. Note first that we can write

$$\bar{\gamma}_t^{\theta}(x_{1:t}) = \bar{\gamma}_1^{\theta}(x_1) \prod_{s=2}^{t} \frac{\bar{\gamma}_s^{\theta}(x_{1:s})}{\bar{\gamma}_{s-1}^{\theta}(x_{1:s-1})}. \tag{5.12}$$

By using the definition of the weight function (Equation (4.9)), this expression can be expanded according to

$$\bar{\gamma}_t^{\theta}(x_{1:t}) = W_1^{\theta}(x_1) r_1^{\theta}(x_1) \prod_{s=2}^{t} W_s^{\theta}(x_{1:s}) r_s^{\theta}(x_s \mid x_{1:s-1}). \tag{5.13}$$

By plugging the trajectory $x_{1:t}^{b_{1:t}}$ into the above expression, we get

$$\begin{aligned}
\bar{\gamma}_t^{\theta}(x_{1:t}^{b_{1:t}}) &= \bar{w}_1^{b_1} r_1^{\theta}(x_1^{b_1}) \prod_{s=2}^{t} \bar{w}_s^{b_s} r_s^{\theta}(x_s^{b_s} \mid x_{1:s-1}^{b_{1:s-1}}) \\
&= \left(\prod_{s=1}^{t} \sum_{l=1}^{N} \bar{w}_s^l \right) \frac{\bar{w}_1^{b_1}}{\sum_l \bar{w}_1^l} r_1^{\theta}(x_1^{b_1}) \prod_{s=2}^{t} \frac{\bar{w}_s^{b_s}}{\sum_l \bar{w}_s^l} r_s^{\theta}(x_s^{b_s} \mid x_{1:s-1}^{b_{1:s-1}}) \\
&= w_t^{b_t} \left(\prod_{s=1}^{t} \sum_{l=1}^{N} \bar{w}_s^l \right) r_1^{\theta}(x_1^{b_1}) \prod_{s=2}^{t} M_t^{\theta}(a_s^{b_s}, x_s^{b_s}),
\end{aligned} \tag{5.14}$$

where $\{\bar{w}_t^i\}_{i=1}^N$ are the unnormalized importance weights at time t. Let

$$\widehat{Z}_t^N(\theta) \triangleq \prod_{s=1}^{t} \left(\frac{1}{N} \sum_{l=1}^{N} \bar{w}_s^l \right). \tag{5.15}$$

Note that this quantity depends on all the random variables generated by the SMC sampler, $\{\mathbf{x}_{1:t}, \mathbf{a}_{2:t}\}$, though this dependence is not explicit in the notation. Using Equation (5.15) we can now rewrite the extended

target density (Equation (5.10)) as

$$\phi(\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k) \overset{(5.1)}{=} \frac{\bar{\gamma}_T^\theta(x_{1:T}^{b_{1:T}})\pi(\theta)}{\mathcal{Z}_T N^T} \frac{\psi^\theta(\mathbf{x}_{1:T}, \mathbf{a}_{2:T})}{r_1^\theta(x_1^{b_1}) \prod_{t=2}^T M_t^\theta(a_t^{b_t}, x_t^{b_t})}$$
$$\overset{(5.14)}{=} \frac{\widehat{Z}_T^N(\theta)\pi(\theta)}{\mathcal{Z}_T} w_T^k \psi^\theta(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}), \tag{5.16}$$

where we have used the identity $b_T = k$. The random variable $\widehat{Z}_t^N(\theta)$ has a natural interpretation as an estimator of the normalization constant $Z_t(\theta)$ in Equation (5.2). It is well known that this estimator is unbiased [31, 117], which in fact follows directly from Equation (5.16) since ϕ is a probability density function and thus integrates to one. By plugging Equation (5.16) into Equation (5.11), we obtain the, surprisingly simple, expression for the acceptance probability,

$$1 \wedge \frac{\widehat{Z}_T'^{,N}(\theta')}{\widehat{Z}_T^N(\theta)} \frac{\pi(\theta')}{\pi(\theta)} \frac{q(\theta \mid \theta')}{q(\theta' \mid \theta)}. \tag{5.17}$$

The resulting Metropolis–Hastings sampler, using the SMC-based proposal kernel (Equation (5.9)) for the extended target density (Equation (5.10)), is referred to as particle marginal Metropolis–Hastings (PMMH). It should be noted that, despite the rather cumbersome derivation, the method is very simple to implement. We summarize the PMMH sampler in Algorithm 14.

From the derivation above, it follows that the PMMH sampler generates a sequence $\{\theta[r], x_{1:T}[r]\}_{r \geq 0}$ with stationary distribution $\gamma_T(\theta, x_{1:T})$. Under additional weak assumptions, it also follows that the underlying Markov chain is ergodic, and PMMH is thus a valid MCMC sampler.

(A3) Let $\mathcal{S} = \{\theta \in \Theta : \pi(\theta) > 0\}$. Then, for any $\theta \in \mathcal{S}$ and any $t \in \{1, \ldots, T\}$, $\mathcal{S}_t^\theta \subseteq \mathcal{Q}_t^\theta$, where

$$\mathcal{S}_t^\theta = \{x_{1:t} \in \mathsf{X}^t : \gamma_t^\theta(x_{1:t}) > 0\},$$
$$\mathcal{Q}_t^\theta = \{x_{1:t} \in \mathsf{X}^t : r_t^\theta(x_t \mid x_{1:t-1}) \gamma_{t-1}^\theta(x_{1:t-1}) > 0\}.$$

(A4) The ideal Metropolis–Hastings sampler with target density $\gamma_T(\theta, x_{1:T})$ and proposal density given by Equation (5.3) is irreducible and aperiodic.

Algorithm 14 Particle marginal Metropolis–Hastings [3]

1: Set $\theta[0]$ arbitrarily.
2: Run an SMC sampler, targeting $\gamma_T^{\theta[0]}(x_{1:T})$, and compute an estimate of the normalization constant, $\widehat{Z}_T^N(\theta[0])$.
3: Sample k with $P(k=i) = w_T^i$ and set $x_{1:T}[0] = x_{1:T}^k$.
4: **for** $r \geq 1$ **do**
5: Draw $\theta' \sim q(\theta' \mid \theta[r-1])$.
6: Run an SMC sampler, targeting $\gamma_T^{\theta'}(x_{1:T})$, and compute an estimate of the normalization constant, $\widehat{Z}_T'^N(\theta')$.
7: Sample k with $P(k=i) = w_T^i$ and set $x'_{1:T} = x_{1:T}^k$.
8: With probability

$$1 \wedge \frac{\widehat{Z}_T'^N(\theta')}{\widehat{Z}_T^N(\theta[r-1])} \frac{\pi(\theta')}{\pi(\theta[r-1])} \frac{q(\theta[r-1] \mid \theta')}{q(\theta' \mid \theta[r-1])},$$

set $\{\theta[r], x_{1:T}[r], \widehat{Z}_T^N(\theta[r])\} = \{\theta', x'_{1:T}, \widehat{Z}_T'^N(\theta')\}$, otherwise set $\{\theta[r], x_{1:T}[r], \widehat{Z}_T^N(\theta[r])\} = \{\theta[r-1], x_{1:T}[r-1], \widehat{Z}_T^N(\theta[r-1])\}$.
9: **end for**

Theorem 5.1. Assume (A3) and (A4). Then, for any $N \geq 1$, the PMMH sampler generates a sequence $\{\theta[r], x_{1:T}[r]\}_{r \geq 0}$ whose marginal distributions $\mathcal{L}^N(\{\theta[r], x_{1:T}[r]\} \in \cdot)$ satisfy,

$$\|\mathcal{L}^N(\{\theta[r], x_{1:T}[r]\} \in \cdot) - \gamma_T(\cdot)\|_{\mathrm{TV}} \to 0 \qquad \text{as } r \to \infty,$$

where $\|\cdot\|_{\mathrm{TV}}$ is the total variation norm.

Proof. See [3, Theorem 4]. □

See [8, 6] for additional and more precise convergence results related to the PMMH method.

An interesting, and aesthetically appealing, property of PMMH is that it is reminiscent of an ideal Metropolis–Hastings sampler, targeting the marginal density

$$\gamma_T(\theta) \triangleq \int \gamma_T(\theta, x_{1:T})\, dx_{1:T}, \tag{5.18}$$

using the proposal kernel $q(\theta' \mid \theta)$ (or, equivalently, using the ideal proposal (Equation (5.3)) to target the joint density $\gamma_T(\theta, x_{1:T})$). For this marginal Metropolis–Hastings sampler, the acceptance probability is given by

$$1 \wedge \frac{Z_T(\theta')}{Z_T(\theta)} \frac{\pi(\theta')}{\pi(\theta)} \frac{q(\theta \mid \theta')}{q(\theta' \mid \theta)}. \tag{5.19}$$

The difference between this expression and the PMMH acceptance probability (Equation (5.17)) is that, in the latter, the unknown normalization constants are replaced by their SMC-based estimators (Equation (5.15)).

Hence, PMMH can be thought of as an SMC approximation of an ideal marginal Metropolis–Hastings sampler. Since the estimator in Equation (5.15) is consistent, the PMMH acceptance probability converges to Equation (5.19) as the number of particles N increases. That is, the convergence speed of PMMH converges to that of the ideal sampler. We emphasize that PMMH is *exact for any number of particles*, in the sense that the stationary distribution of the sampler is $\gamma_T(\theta, x_{1:T})$ for any $N \geq 1$ (see Theorem 5.1). However, for small N, the variance of the estimator (5.15) will be large and PMMH tends to get stuck, leading to slow convergence.

For a fixed computational time, there is a trade-off between taking N large to get a high acceptance probability, and to run many iterations of the MCMC sampler. This trade-off has been analyzed in [43, 117] who, under certain assumption, conclude that it is optimal to choose N so that the variance of the logarithm of $\widehat{Z}_T^N$ is around 1. As a rule of thumb, N should thus scale at least linearly with T to keep the variance of the normalization constant estimate in check [3].

Another thing that is interesting to note is that the variable k, which is sampled at line 7 of Algorithm 14, does not affect the accept/reject decision on line 8. That is, the acceptance probability depends only on the estimate of the normalization constant (Equation (5.15)), and not on the specific particle trajectory $x'_{1:T}$ that is extracted at line 7. Hence, if the object of interest is the marginal density (Equation (5.18)), there is no need to sample k at all.

5.3 PMMH with Backward Simulation

The PMMH targets the joint density $\gamma_T(\theta, x_{1:T})$. However, as pointed out in the previous section, if the focus is only on the marginal density (Equation (5.18)), then the state trajectory $x'_{1:T}$ does not have to be sampled.

On the contrary, if the focus is on the latent variables $x_{1:T}$, then it seems like a waste to generate N particle trajectories at each iteration of the PMMH sampler, but keep only a single one, as is done in Algorithm 14. In fact, since the acceptance probability does not depend on the specific trajectory that is extracted, it is possible to average over all the trajectories instead of randomly picking one of them.

Assume that we wish to compute

$$\mathbb{E}_{\gamma_T}[\varphi(\theta, x_{1:T})] = \int \varphi(\theta, x_{1:T})\, \gamma_T(\theta, x_{1:T})\, d\theta dx_{1:T}, \tag{5.20}$$

for some test function $\varphi : \Theta \times \mathsf{X}^T \to \mathbb{R}$. We run R iterations of Algorithm 14 (possibly with some burn-in), resulting in a realization of the process, $\{\theta[r], x_{1:T}[r]\}_{r=0}^{R}$. The most straightforward estimator of Equation (5.20) is then

$$\hat{\varphi}_{\text{PMMH}} = \frac{1}{R}\sum_{r=0}^{R} \varphi(\theta[r], x_{1:T}[r]), \tag{5.21}$$

which, by the ergodic theorem, converges almost surely to Equation (5.20). However, it was recognized by Andrieu et al. [3] that an alternative is to use the Rao–Blackwellized estimator

$$\begin{aligned}\hat{\varphi}_{\text{PMMH-RB}} &= \frac{1}{R}\sum_{r=0}^{R} \mathbb{E}\left[\varphi\left(\theta[r], x^k_{1:T}\right) \,\middle|\, \theta[r], \mathbf{x}_{1:T}[r], \mathbf{a}_{2:T}[r]\right] \\ &= \frac{1}{R}\sum_{r=0}^{R}\sum_{i=1}^{N} w_T^i[r]\varphi\left(\theta[r], x^i_{1:T}[r]\right). \end{aligned} \tag{5.22}$$

The possibility to make use of all the generated particles to reduce the variance of the estimator seems promising. However, a problem with the above estimator is that the particle systems $\{x^i_{1:T}[r], w^i_T[r]\}_{i=1}^{N}$ are generated by SMC samplers, and will thus suffer from path degeneracy.

Hence, the possible benefit of Rao–Blackwellization is limited, due to the low particle diversity for time points t far away from the final time T.

To achieve a better variance reduction effect, [113] have proposed to complement PMMH with a run of a backward simulator. That is, line 7 of Algorithm 14 on page 99 is replaced by an execution of Algorithm 12 (page 74). In practice, it is preferable to run the backward simulator only if the proposed sample is accepted. This is possible since, as pointed out above, the acceptance probability is independent of the extracted trajectories. The PMMH sampler with backward simulation is given in Algorithm 15.

Algorithm 15 PMMH with backward simulation [113]

1: Set $\theta[0]$ arbitrarily.
2: Run an SMC sampler, targeting $\gamma_T^{\theta[0]}(x_{1:T})$, and compute an estimate of the normalization constant, $\widehat{Z}_T^N(\theta[0])$.
3: Generate $\{\widetilde{x}_{1:T}^j[0]\}_{j=1}^M$ by backward simulation (Algorithm 12).
4: **for** $r \geq 1$ **do**
5: Draw $\theta' \sim q(\theta' \mid \theta[r-1])$.
6: Run an SMC sampler, targeting $\gamma_T^{\theta'}(x_{1:T})$, and compute an estimate of the normalization constant, $\widehat{Z}_T^{\prime,N}(\theta')$.
7: With probability

$$1 \wedge \frac{\widehat{Z}_T^{\prime,N}(\theta')}{\widehat{Z}_T^N(\theta[r-1])} \frac{\pi(\theta')}{\pi(\theta[r-1])} \frac{q(\theta[r-1] \mid \theta')}{q(\theta' \mid \theta[r-1])},$$

set $I_{\text{accept}} = 1$, otherwise set $I_{\text{accept}} = 0$.
8: **if** $I_{\text{accept}} = 1$ **then**
9: Set $\{\theta[r], \widehat{Z}_T^N(\theta[r])\} = \{\theta', \widehat{Z}_T^{\prime,N}(\theta')\}$.
10: Generate $\{\widetilde{x}_{1:T}^j[r]\}_{j=1}^M$ by backward simulation (Algorithm 12).
11: **else**
12: Set $\{\theta[r], \widehat{Z}_T^N(\theta[r])\} = \{\theta[r-1], \widehat{Z}_T^N(\theta[r-1])\}$.
13: Set $\{\widetilde{x}_{1:T}^j[r]\}_{j=1}^M = \{\widetilde{x}_{1:T}^j[r-1]\}_{j=1}^M$.
14: **end if**
15: **end for**

Let the M backward trajectories generated at iteration r of the PMMH sampler be denoted $\{\widetilde{x}^j_{1:T}[r]\}_{j=1}^M$. An estimator of Equation (5.20) is then given by

$$\hat{\varphi}_{\text{PMMH-BS}} = \frac{1}{RM}\sum_{r=0}^{R}\sum_{j=1}^{M}\varphi\left(\theta[r], \widetilde{x}^j_{1:T}[r]\right). \tag{5.23}$$

It is shown in [113] that the backward simulator leaves the target distribution invariant. Again, by Rao–Blackwellization type of arguments, it then follows that Equation (5.23) converges almost surely as $R \to \infty$, for any $M \geq 1$.

In [113], an expression for the variance of the estimator in Equation (5.23) is provided,

$$\begin{aligned}\operatorname{Var}\left(\hat{\varphi}_{\text{PMMH-BS}}\right) &= \mathbb{E}\left[\operatorname{Var}\left(\hat{\varphi}_{\text{PMMH-BS}} \,\middle|\, \{\theta[r], \mathbf{x}_{1:T}[r], \mathbf{a}_{2:T}[r]\}_{r=0}^R\right)\right] \\ &\quad + \operatorname{Var}\left(\mathbb{E}\left[\hat{\varphi}_{\text{PMMH-BS}} \,\middle|\, \{\theta[r], \mathbf{x}_{1:T}[r], \mathbf{a}_{2:T}[r]\}_{r=0}^R\right]\right) \\ &= \frac{1}{R}\left(\frac{\sigma^2}{M} + \sigma_R^2\right) \approx \frac{1}{R}\left(\frac{\sigma^2}{M} + \sigma_\infty^2\right),\end{aligned} \tag{5.24}$$

where

$$\begin{aligned}\sigma^2 &= \mathbb{E}\left[\operatorname{Var}\left(\varphi\left(\theta, \widetilde{x}_{1:T}\right) \mid \{\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}\}\right)\right], \\ \sigma_R^2 &= \frac{1}{R}\operatorname{Var}\left(\sum_{r=0}^{R}\mathbb{E}\left[\varphi\left(\theta[r], \widetilde{x}_{1:T}[r]\right) \mid \{\theta[r], \mathbf{x}_{1:T}[r], \mathbf{a}_{2:T}[r]\}\right]\right),\end{aligned}$$

and $\sigma_\infty^2 = \lim_{R\to\infty}\sigma_R^2$ is the time-average variance constant. This expression can be used to find an optimal trade-off between R and M, depending on the run times of the algorithm for different settings. In practice, the parameters σ^2 and σ_∞^2 are not known, and to make use of Equation (5.24) to tune the sampler it is thus necessary to estimate these parameters from data.

We illustrate the effects of Rao–Blackwellization and backward simulation in the example below.

Example 5.2 (PIMH with backward simulation). We present a simulation study similar to one of the examples considered in [113].

Consider again the nonlinear time-series model,

$$x_{t+1} = 0.5x_t + 25\frac{x_t}{1+x_t^2} + 8\cos(1.2t) + v_t,$$
$$y_t = 0.05x_t^2 + e_t,$$

with $v_t \sim \mathcal{N}(0,10)$, $e_t \sim \mathcal{N}(0,1)$ and $x_1 \sim \mathcal{N}(0,5)$. We seek the joint smoothing distribution $p(x_{1:T} \mid y_{1:T})$. For simplicity, we assume that there are no unknown parameters in the model. Since the PMCMC samplers in Algorithms 14 and 15 address the joint parameter and state inference problem, smoothing is covered as a special case (see Section 5.7). In this case, i.e., in the absence of an unknown static parameter, the PMMH sampler is referred to as particle independent Metropolis–Hastings (PIMH).

We wish to compare the estimator variances for the three alternative estimators $\hat{\varphi}_{\text{PIMH}}$, $\hat{\varphi}_{\text{PIMH-RB}}$ and $\hat{\varphi}_{\text{PIMH-BS}}$. We generate one batch of $T = 100$ observations from the time series. We then run 100 independent copies of Algorithms 14 and 15 (all using the same data), each for $R = 5\,000$ iterations. We use $N = 500$ particles for the SMC samplers and $M = 25$ backward trajectories in Algorithm 15. For each run, we compute three estimates of the posterior mean of $x_{1:T}$ according to Equations (5.21), (5.22) and (5.23), respectively. Figure 5.1 shows the estimator variances for each time point t. Close to the final time point T, the Rao–Blackwellization in Equation (5.22) has a clear impact on the estimator variance, which is reduced by about two orders of

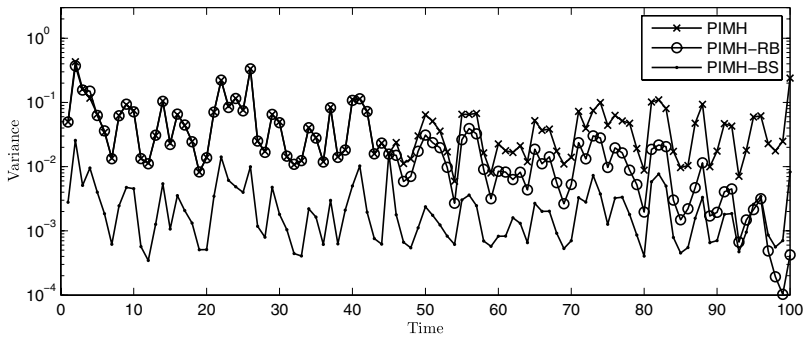

Fig. 5.1 Estimator variances for PIMH, PIMH-RB and PIMH-BS, respectively, for the estimates of the posterior mean of $x_{1:100}$.

magnitude compared to the crude PIMH estimator. However, due to path degeneracy, the variance reduction gets less pronounced for times t far from T. For $t < 50$, there is more or less no reduction at all. For the backward-simulation-based estimator (Equation (5.23)), however, we obtain a clear variance reduction for all time points.

5.4 Particle Gibbs with Backward Simulation

PMMH is a Metropolis–Hastings sampler, targeting Equation (5.10) with the specific proposal given in Equation (5.9). However, now that we have the extended target (Equation (5.10)) in place, we can think about other MCMC samplers targeting this distribution, leading to other members of the PMCMC family. One possibility is to design a multi-stage Gibbs sampler for ϕ, using the following sweep (note that $b_{1:T} = \{a_{2:T}^{b_{2:T}}, b_T\}$):

(i) Draw $\theta' \sim \phi(\theta \mid x_{1:T}^{b_{1:T}}, b_{1:T})$;
(ii) Draw $\{\mathbf{x}'^{,-b_{1:T}}_{1:T}, \mathbf{a}'^{,-b_{2:T}}_{2:T}\} \sim \phi(\mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}} \mid \theta', x_{1:T}^{b_{1:T}}, b_{1:T})$;
(iii) Draw $k' \sim \phi(k \mid \theta', \mathbf{x}'^{,-b_{1:T}}_{1:T}, \mathbf{a}'^{,-b_{2:T}}_{2:T}, x_{1:T}^{b_{1:T}}, a_{2:T}^{b_{2:T}})$;

This is the particle Gibbs (PG) sampler, proposed in [3]. The first step of the procedure is a partially collapsed Gibbs step (see Section 2.2.3 or [97, 137]), which leaves the target distribution invariant. Alternatively, we may view Steps (i)–(ii) as a grouped Gibbs step for the variables $\{\theta, \mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}}\}$.

Later in this section, we will consider an alternative Gibbs sweep which makes use of backward simulation to improve the mixing of the sampler. However, to be able to see why backward simulation can be useful in this context, we first need to understand the properties of the PG sampler. To get a better picture of what this method does, let us therefore go through the three steps of the procedure and discuss how they can be implemented.

For Step (i) we have, by the construction of the extended target distribution ϕ in Equation (5.10),

$$\phi(\theta \mid x_{1:T}^{b_{1:T}}, b_{1:T}) = \gamma_T(\theta \mid x_{1:T}) \triangleq \frac{\gamma_T(\theta, x_{1:T})}{\int \gamma_T(\theta, x_{1:T})\, d\theta}. \tag{5.25}$$

As argued before, it is in general much easier to sample the parameter θ conditionally on the latent variables $x_{1:T}$, than from the marginal density (Equation (5.18)). We shall thus assume that this step can be carried out exactly. This is possible when a conjugate prior is used for θ. For a non-conjugate model, we can instead sample θ' from a Metropolis–Hastings kernel, leaving Equation (5.25) invariant.

The conditional density used in Step (ii) is also given by construction,

$$\phi(\mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}} \mid \theta, x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1 \\ i \neq b_1}}^{N} r_1^\theta(x_1^i) \prod_{t=2}^{T} \prod_{\substack{i=1 \\ i \neq b_t}}^{N} M_t^\theta(a_t^i, x_t^i). \quad (5.26)$$

It is interesting to note that this expression does not depend explicitly on the target density $\gamma_T(\theta, x_{1:T})$, but only on the proposal kernels used in the SMC sampler (cf. Equation (5.8)). Hence, we can sample from Equation (5.26) by using a procedure reminiscent of the SMC sampler in Algorithm 11 on page 72. The difference is that in Equation (5.26), the factors corresponding to the indices $b_{1:T}$ have been excluded. Hence, to sample from Equation (5.26) we first generate $x_1^i \sim r_1^\theta(x_1)$ for $i \in \{1, \ldots, N\} \setminus b_1$, then $\{a_2^i, x_2^i\} \sim M_2^\theta(a_2, x_2)$ for $i \in \{1, \ldots, N\} \setminus b_2$, etc. Throughout this sampling process, the trajectory $x_{1:T}^{b_{1:T}}$ is kept fixed.

Before stating an algorithm corresponding to this sampling procedure, however, we note that the indices $b_{1:T}$ are nuisance variables. That is, in practice we are not interested in the values of these variables and they are only introduced to aid in the derivation of the algorithm. Furthermore, the SMC sampler is invariant to permutations of the particle indices, i.e., it does not matter in which order we enumerate the particles. Consequently, the actual values of the indices $b_{1:T}$ are of no significance when sampling from Equation (5.26), meaning that we do not have to keep track of these variables. Instead we can fix $b_{1:T}$ to some arbitrary sequence which we find more convenient, e.g., $b_{1:T} = (N, \ldots, N)$. With this convention, a method for sampling from Equation (5.26), referred to as a conditional SMC (CSMC) sampler, is given in Algorithm 16.

Finally, for Step (iii), we note that

$$\phi(k \mid \theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}) \propto \phi(\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k) \overset{(5.16)}{\propto} w_T^k. \quad (5.27)$$

Algorithm 16 Conditional SMC (conditioned on $x'_{1:T}$)

1: Draw $x_1^i \sim r_1^\theta(x_1)$ for $i = 1, \ldots, N-1$ and set $x_1^N = x'_1$.
2: Compute $\bar{w}_1^i = W_1(x_1^i)$ for $i = 1, \ldots, N$.
3: Normalize the weights $w_1^i = \bar{w}_1^i / \sum_l \bar{w}_1^l$ for $i = 1, \ldots, N$.
4: **for** $t = 2$ **to** T **do**
5: Draw $\{a_t^i, x_t^i\} \sim M_t^\theta(a_t, x_t)$ for $i = 1, \ldots, N-1$.
6: Set $a_t^N = N$ and $x_t^N = x'_t$.
7: Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$ for $i = 1, \ldots, N$.
8: Compute $\bar{w}_t^i = W_t^\theta(x_{1:t}^i)$ for $i = 1, \ldots, N$.
9: Normalize the weights $w_t^i = \bar{w}_t^i / \sum_l \bar{w}_t^l$ for $i = 1, \ldots, N$.
10: **end for**

Algorithm 17 Particle Gibbs [3]

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily.
2: **for** $r \geq 1$ **do**
3: Draw $\theta[r] \sim \gamma_T(\theta \mid x_{1:T}[r-1])$.
4: Run a CSMC sampler (Algorithm 16) targeting $\gamma_T^{\theta[r]}(x_{1:T})$, conditioned on $x_{1:T}[r-1]$.
5: Sample k with $P(k = i) = w_T^i$ and trace the ancestral path of particle x_T^k, i.e., set $x_{1:T}[r] = x_{1:T}^k$.
6: **end for**

Hence, as in PMMH, we sample the index k with $P(k = i) = w_T^i$. We can now reinterpret the three steps of the PG sampler as in Algorithm 17.

As can be seen in Theorem 5.2 below, ergodicity of PG holds under similar assumptions as for PMMH. See also [29] for a uniform ergodicity result for the PG sampler.

(A5) The ideal Gibbs sampler, defined by alternating between sampling from $\gamma_T(\theta \mid x_{1:T})$ and $\gamma_T^\theta(x_{1:T})$, is irreducible and aperiodic.

Theorem 5.2. Assume (A3) and (A5). Then, for any $N \geq 2$, the PG sampler generates a sequence $\{\theta[r], x_{1:T}[r]\}_{r \geq 0}$ whose marginal

distributions $\mathcal{L}^N(\{\theta[r], x_{1:T}[r]\} \in \cdot)$ satisfy,

$$\|\mathcal{L}^N(\{\theta[r], x_{1:T}[r]\} \in \cdot) - \gamma_T(\cdot)\|_{\mathrm{TV}} \to 0 \qquad \text{as } r \to \infty,$$

where $\|\cdot\|_{\mathrm{TV}}$ is the total variation norm.

Proof. See [3, Theorem 5]. □

As pointed out above, a key property of PMCMC methods is that they do not rely on asymptotics in N to be valid MCMC samplers. However, for the PMMH sampler we found that the acceptance probability (Equation (5.17)) depends on the SMC estimate of the normalization constant Equation (5.15). Hence, to obtain a reasonable probability of acceptance, we have to take N large enough to get sufficiently small variance of this estimator.

For the PG sampler, on the other hand, the dependence on N is not as obvious. Since it is a Gibbs sampler, all generated samples will in fact be accepted. To investigate how the method is affected by different values of N, we apply it to a simple toy problem in the example below.

Example 5.3 (PG for stochastic volatility model). Consider a simple stochastic volatility model on state-space form,

$$\begin{aligned} x_{t+1} &= ax_t + v_t, & v_t &\sim \mathcal{N}(0, \theta), \\ y_t &= e_t \exp\left(\tfrac{1}{2} x_t\right), & e_t &\sim \mathcal{N}(0, 1). \end{aligned}$$

For brevity, we keep $a = 0.9$ fixed, but assume that the variance θ of the latent process is unknown. We put a conjugate inverse gamma prior on θ, with hyperparameters $a = b = 0.01$. We then generate a batch of $T = 100$ measurements with the true value $\theta = 0.5^2$, and employ the PG sampler to find the posterior $p(\theta \mid y_{1:T})$ of the parameter.

To see how the sampler is affected by the number of particles, we consider four independent runs with $N = 5$, 20, 100 and 1 000, respectively. We run the samplers for 100 000 iteration, discarding the first 10 000 iterations as burn-in. The empirical ACFs for the residuals $\theta[r] - \mathbb{E}[\theta \mid y_{1:T}]$ are reported in the left panel of Figure 5.2. For $N = 1\,000$, we get a fairly sharp drop in autocorrelation, indicating a

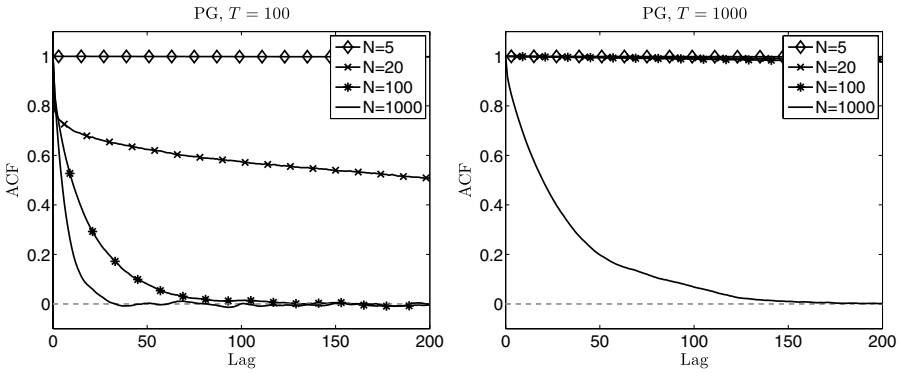

Fig. 5.2 Empirical ACF for θ for the PG samplers using different number of particles N.

sampler that mixes well. However, as we decrease the number of particles, there is a dramatic change in the ACF. For $N = 20$, the ACF drops off very slowly and for $N = 5$ it is almost constant, meaning that the sampler is more or less stuck at a single point in the parameter space for all 100 000 iterations.

Furthermore, to see how the sampler is affected by the size of the data set, we repeat the experiment with $T = 1\,000$. The ACFs are given in the right panel of Figure 5.2. The same effect is clearly visible and even more severe in this scenario. Even for $N = 1\,000$ the method struggles, and as we reduce the number of particles further, the sampler gets stuck.

To get a better insight into the poor mixing of the PG kernel for small N and/or large T, let us analyze two consecutive iterations of the sampler in more detail. For clarity, we use a small number of particles and time steps, $N = 20$ and $T = 50$, respectively.

Figure 5.3 (top) shows the particle system generated by the CSMC sampler at iteration r. Due to path degeneracy, there is only one distinct particle trajectory for $t \leq 27$. The extracted trajectory $x_{1:T}[r]$, corresponding to the ancestral path of particle x_T^k, is illustrated by a thick black line. At the next iteration of the PG sampler we run CSMC conditioned on $x_{1:T}[r]$. This results in the system shown in Figure 5.3 (bottom). Due to path degeneracy, we once again obtain only a single distinct particle trajectory for time points far from T, here for $t \leq 35$. However, the particle system generated by the CSMC sampler must

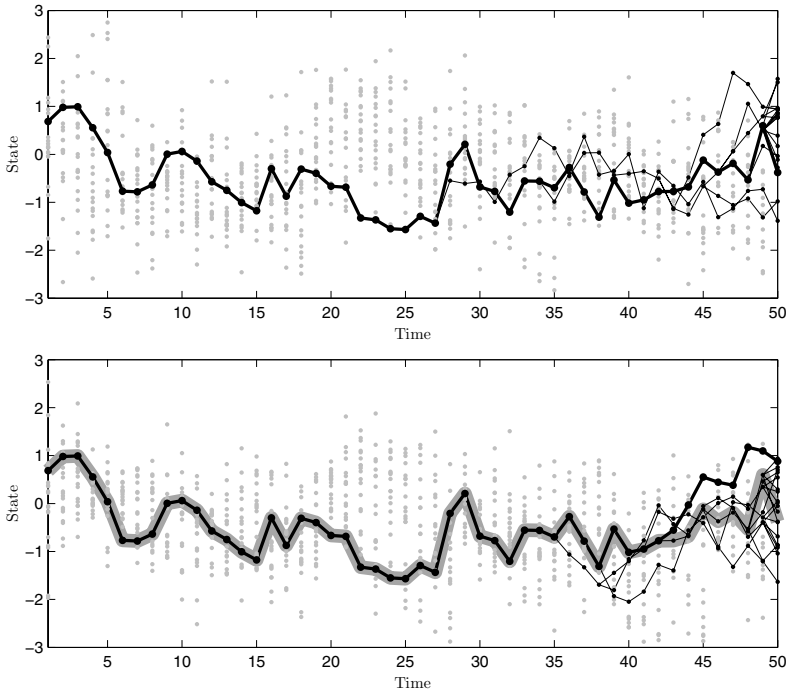

Fig. 5.3 Particle system generated by CSMC at iterations r (top) and $r+1$ (bottom) of the PG sampler. The dots show the particle positions, the thin black lines show the ancestral dependence of the particles and the thick black lines show the sampled trajectories $x_{1:T}[r]$ and $x_{1:T}[r+1]$, respectively. In the bottom pane, the thick gray line illustrates the conditioned path at iteration $r+1$, which by construction equals $x_{1:T}[r]$. Note that, due to path degeneracy, the particles shown as gray dots are not reachable by tracing any of the ancestral lineages from time T and back.

contain the conditioned path. Hence, all particle trajectories available at iteration $r+1$ are identical to $x_{1:T}[r]$ up to time $t = 27$. Consequently, when we sample $x_{1:T}[r+1]$ at iteration $r+1$, this trajectory will to a large extent be identical to $x_{1:T}[r]$. This results in a poor exploration of the state-space, which in turn means that the Gibbs kernel will mix slowly.

Based on this argument we note that the mixing will be particularly poor when the path degeneracy is severe. This will be the case if the length of the data record T is large and/or if the number of particles N is small. This is in agreement with the results reported in Figure 5.2.

From the discussion in the example above we conclude that the PG sampler suffers from poor mixing as a result of path degeneracy. Hence, for this method to work, we need to take N large enough to tackle the degeneracy, which for many problems is unrealistic from a computational point of view.

An alternative interpretation of this issue is that it is caused by the fact that the Gibbs sweep defining the PG sampler, Steps (i)–(iii) above, is incomplete. More precisely, if we collect the variables from the three steps of the procedure, we note that we never sample new values for $\{x_{1:T}^{b_{1:T}}, b_{1:T-1}\}$ in this sweep. Despite this incompleteness of the Gibbs sweep, it holds under the assumptions of Theorem 5.2 that PG is ergodic. Intuitively, this can be explained by the fact that the collection of variables that is left out is chosen randomly at each iteration. In the long run, we will thus include all the variables with probability one. However, due to degeneracy of the SMC sampler, the collections of variables that are left out at any two consecutive iterations will be strongly dependent, resulting in poor mixing.

A way to address this problem was proposed by Whiteley [141] and further explored in [94, 142]. The idea is to complement the PG sampler with a backward simulator to mitigate the path degeneracy problem, and thus obtain a faster mixing sampler. Alternatively, based on the interpretation above, this modification can be seen as a way to include the index variables $b_{1:T-1}$ in the Gibbs sampler.[1] As noted above, we are not interested in these index variables *per se*, but including them in the Gibbs sweep can lead to a general improvement in mixing, affecting all the variables of the model. The idea is to sample the variables b_t backward in time, using a sequence of Gibbs steps. More precisely, we replace Step (iii) of the PG sampler with the following,

[1] Ideally, we would like to include the variables $x_{1:T}^{b_{1:T}}$ as well, but this is in general not possible since it would be similar to sampling from the original target density, which we assume is infeasible.

(iii)′ Draw,

$$
\begin{aligned}
b'_T &\sim \phi(b_T \mid \theta', \mathbf{x}'^{,-b_{1:T}}_{1:T}, \mathbf{a}'^{,-b_{2:T}}_{2:T}, x^{b_{1:T}}_{1:T}, i^{b_{2:T}}_{2:T}); \\
&\vdots \\
b'_t &\sim \phi(b_t \mid \theta', \mathbf{x}'^{,-b_{1:t}}_{1:t}, \mathbf{a}'^{,-b_{2:t}}_{2:t}, x^{b_{1:t}}_{1:t}, i^{b_{2:t}}_{2:t}, x^{b'_{t+1:T}}_{t+1:T}, b'_{t+1:T}); \\
&\vdots \\
b'_1 &\sim \phi(b_1 \mid \theta', \mathbf{x}'^{,-b_1}_{1}, x^{b_1}_1, x^{b'_{2:T}}_{2:T}, b'_{2:T}).
\end{aligned}
$$

Note that the densities involved in this sampling scheme are conditionals, not under the full joint density $\phi(\theta, \mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k)$, but under marginals thereof. That is, Step (iii)′ corresponds to a sequence of partially collapsed Gibbs steps. As we have noted before, collapsing, or marginalization, is often used within Gibbs sampling and it does not violate the invariance of the sampler.

As hinted at above, this backward sampling of the ancestor indices corresponds to a run of a backward simulator. We call the resulting method, defined by the Gibbs Steps (i),(ii) and (iii)′, particle Gibbs with backward simulation (PGBS). Note that the first row of Step (iii)′ is identical to Step (iii) of the original PG sampler, where we sample the variable $k = b_T$ from its full conditional. However, in PGBS we do not stop there, but continue to draw new values for the indices b_t down to $t = 1$. This will break the strong dependencies caused by the SMC path degeneracy, and allow for a much faster mixing Gibbs kernel.

To see that Step (iii)′ indeed corresponds to a run of a backward simulator, note first that by marginalizing Equation (5.26) over $\{\mathbf{x}^{-b_{t+1:T}}_{t+1:T}, \mathbf{a}^{-b_{t+1:T}}_{t+1:T}\}$ we get

$$
\phi(\mathbf{x}^{-b_{1:t}}_{1:t}, \mathbf{a}^{-b_{2:t}}_{2:t} \mid \theta, x^{b_{1:T}}_{1:T}, b_{1:T}) = \prod_{\substack{i=1 \\ i \neq b_1}}^{N} r^{\theta}_1(x^i_1) \prod_{s=2}^{t} \prod_{\substack{i=1 \\ i \neq b_s}}^{N} M^{\theta}_s(a^i_s, x^i_s). \quad (5.28)
$$

We can thus write

$$\begin{aligned}\phi(b_t \mid \theta, \mathbf{x}_{1:t}, \mathbf{a}_{2:t}, x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}) &\propto \phi(\theta, \mathbf{x}_{1:t}, \mathbf{a}_{2:t}, x_{t+1:T}^{b_{t+1:T}}, b_{t:T}) \\ &= \phi(\theta, x_{1:T}^{b_{1:T}}, b_{1:T}) \phi(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{2:t}^{-b_{2:t}} \mid \theta, x_{1:T}^{b_{1:T}}, b_{1:T}) \\ &= \frac{\bar{\gamma}_T^\theta(x_{1:T}^{b_{1:T}})}{\bar{\gamma}_t^\theta(x_{1:t}^{b_{1:t}})} \frac{\bar{\gamma}_t^\theta(x_{1:t}^{b_{1:t}})\pi(\theta)}{\mathcal{Z}_\mathcal{T} N^T} \prod_{\substack{i=1 \\ i \neq b_1}}^{N} r_1^\theta(x_1^i) \prod_{s=2}^{t} \prod_{\substack{i=1 \\ i \neq b_s}}^{N} M_s^\theta(a_s^i, x_s^i), \end{aligned} \tag{5.29}$$

where we have used Equations (5.1) and (5.28) for the last equality. By expanding $\bar{\gamma}_t^\theta(x_{1:t}^{b_{1:t}})$ in the numerator according to Equation (5.14), we obtain,

$$\phi(b_t \mid \theta, \mathbf{x}_{1:t}, \mathbf{a}_{2:t}, x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}) \propto w_t^{b_t} \frac{\bar{\gamma}_T^\theta(x_{1:T}^{b_{1:T}})}{\bar{\gamma}_t^\theta(x_{1:t}^{b_{1:t}})}, \tag{5.30}$$

which is exactly the expression for the backward simulation weights (Equation (4.14)). We summarize the PGBS sampler in Algorithm 18. On line 5 of the algorithm, the indices $b_{1:T}$ are generated according to Step (iii)$'$ of the sampler. However, as noted above, we are not interested in these indices themselves, but only in the trajectory $x_{1:T}^{b_{1:T}}$. Consequently, for practical convenience, the algorithm makes no explicit reference to $b_{1:T}$. Note that we only generate a single backward trajectory at each iteration of the sampler. Hence, the computational complexity is still linear in the number of particles. The effect of adding backward simulation to the sampler is quite substantial, which is illustrated on our toy problem in the example below.

Example 5.4 (PGBS for stochastic volatility model). We return to Example 5.3 and apply the PGBS sampler to the same batches of data, for $T = 100$ and $T = 1\,000$, respectively. As before, we consider four independent runs with $N = 5$, 20, 100 and 1 000, respectively. The empirical ACFs for the residuals $\theta[r] - \mathbb{E}[\theta \mid y_{1:T}]$, based on 100 000 iterations of the samplers, are given in Figure 5.4.

When compared with the corresponding plots for the PG sampler in Figure 5.2, we see a large improvement. The addition of a backward

Algorithm 18 Particle Gibbs with backward simulation [142, 94]

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily.
2: **for** $r \geq 1$ **do**
3: Draw $\theta[r] \sim \gamma_T(\theta \mid x_{1:T}[r-1])$.
4: Run a CSMC sampler (Algorithm 16) targeting $\gamma_T^{\theta[r]}(x_{1:T})$, conditioned on $x_{1:T}[r-1]$.
5: Run a backward simulator (Algorithm 12, page 74, with $M = 1$) to sample the trajectory $x_{1:T}[r]$.
6: **end for**

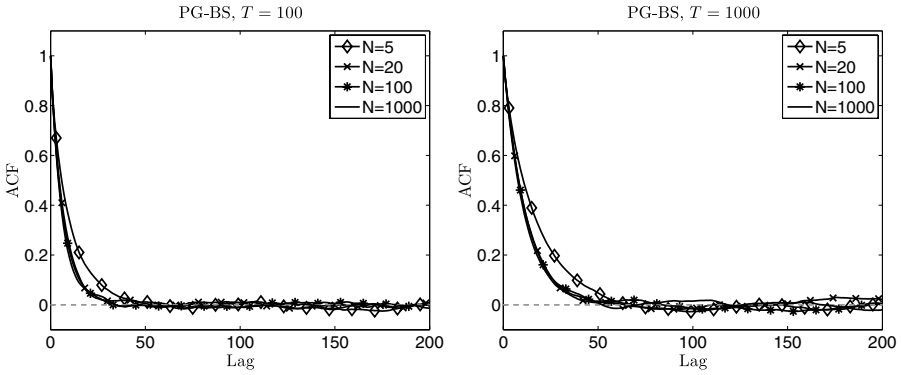

Fig. 5.4 Empirical ACF for θ for the PGBS samplers using different numbers of particles N.

simulation sweep appears to make the method much more robust to a small number of particles, as well as larger data sets. In fact, there is no noticeable improvement when increasing the number of particles above $N = 20$. Hence, for this specific example, we can conclude that PGBS with $N \gtrsim 20$ performs very close to an ideal Gibbs sampler, i.e., a Gibbs sampler in which $x_{1:T}$ is sampled from the exact JSD.

To provide further insight into the effect of the backward simulation step in PGBS, we make a similar inspection as in Example 5.3. That is, we look at two consecutive iterations of the sampler in more detail, using $N = 20$ and $T = 50$ for clarity.

Figure 5.5 (top) shows the particles generated by the CSMC sampler at iteration r, as well as the specific trajectory $x_{1:T}[r]$ which is sampled by the backward simulator. At the next iteration of the PGBS sampler

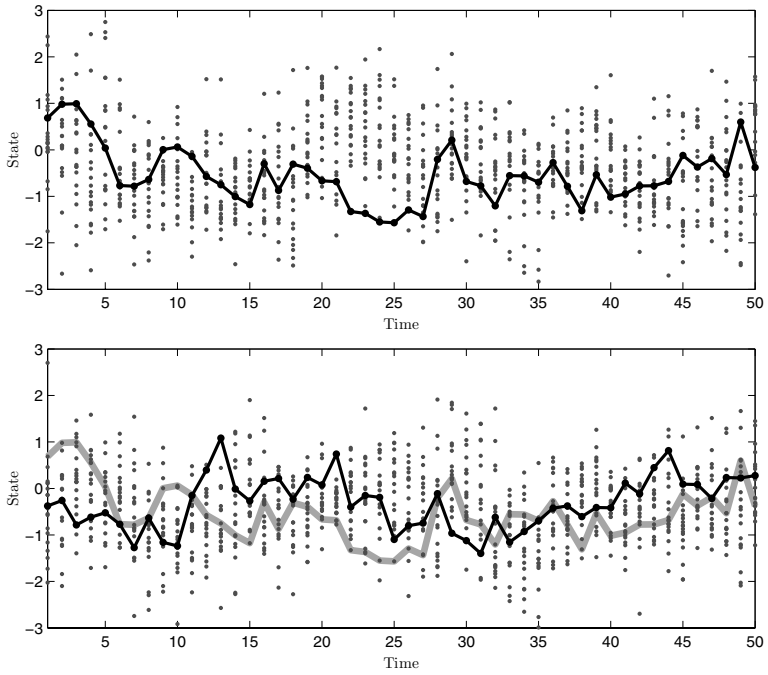

Fig. 5.5 Particles generated by CSMC at iterations r (top) and $r+1$ (bottom) of the PGBS sampler. The dots show the particle positions and the black lines show the trajectories $x_{1:T}[r]$ and $x_{1:T}[r+1]$, respectively, sampled by backward simulation. In the bottom panel, the thick gray line illustrates the conditioned path at iteration $r+1$, which by construction equals $x_{1:T}[r]$. Note that all the particles, at all time points, are reachable by the backward simulators.

we run CSMC conditioned on $x_{1:T}[r]$. This results in the particles shown in Figure 5.5 (bottom). The conditioned path is illustrated by a thick gray line, and the path $x_{1:T}[r+1]$ is shown as a black line.

As discussed in Example 5.3, the CSMC sampler will degenerate toward the conditioned path. However, since $x_{1:T}[r+1]$ is sampled using a backward simulator, we are not constrained to any of the ancestral lineages. Instead, we explore all the particles generated in the forward sweep, and not only those that survive to time T. This results in a path $x_{1:T}[r+1]$ which, with high probability, is to a large extent different from $x_{1:T}[r]$. The effect is that PGBS explores the state-space around the conditioned path much better than PG, resulting in a faster mixing Gibbs kernel.

In the example above, PGBS appeared to be quite robust to both large T and small N, performing close to an ideal Gibbs sampler even with very few particles. This has been experienced also in more challenging scenarios, see e.g., [142] for inference in jump Markov linear systems and [143] for multiple change-point problems.

Intuitively, there are two reasons for why PGBS can perform well, even for small N. First, the conditioning in the CSMC sampler is crucial. It can be thought of as guiding the particles toward the conditioned path. In that way we retain information from one MCMC iteration to the next. In this aspect, PMMH, for instance, is more blunt, since it is based on independent SMC samplers at each iteration. Second, the use of backward simulation to update the indices $b_{1:T}$ allows for a perturbation around the conditioned path. Due to this, the sampler can avoid getting stuck as a result of path degeneracy, and therefore explores the state-space efficiently around the conditioned path.

PGBS is reminiscent of the Gibbs sampler proposed by Neal et al. [110] for inference in SSMs. In this method, a pool of candidate states is generated, followed by a backward trajectory simulation. From this perspective, PGBS can be seen as a clever way of generating the candidate states by running an SMC sampler.

5.5 Particle Gibbs with Ancestor Sampling

The CSMC sampler that is used in PG and PGBS is a sequential method, progressing forward in time, generating the variables $\{\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t}\}$ for $t = 1, \ldots, T$. Similar to the backward simulation pass, this forward sweep can be interpreted as a sequence of partially collapsed Gibbs steps. To see this, we note that Equation (5.28) implies that

$$\phi(\mathbf{x}_1^{-b_1} \mid \theta, x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1 \\ i \neq b_1}}^{N} r_1^\theta(x_1^i), \tag{5.31a}$$

and, for $t = 2, \ldots, T$,

$$\begin{aligned} &\phi(\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t} \mid \theta, \mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}}, x_{1:T}^{b_{1:T}}, b_{1:T}) \\ &\quad = \frac{\phi(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{2:t}^{-b_{2:t}} \mid \theta, x_{1:T}^{b_{1:T}}, b_{1:T})}{\phi(\mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}} \mid \theta, x_{1:T}^{b_{1:T}}, b_{1:T})} = \prod_{\substack{i=1 \\ i \neq b_t}}^{N} M_t^\theta(a_t^i, x_t^i). \end{aligned} \tag{5.31b}$$

That is, we can interpret the CSMC sampler in Algorithm 16 as first sampling $\mathbf{x}_1^{-b_1}$ from Equation (5.31a) and then, for $t = 2, \ldots, T$, sampling $\{\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t}\}$ from Equation (5.31b).[2] This corresponds to a sequence of partially collapsed Gibbs steps.

In PGBS, this forward sweep is complemented with a backward sweep, in which the variables b_t are generated from the conditionals (Equation (5.30)) for $t = T, T-1, \ldots, 1$. However, it is possible to instead alternate between these two sequences and generate all the variables in a single forward sweep. That is, after an initial draw from Equation (5.31a), we alternate between sampling $\{\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t}\}$ from Equation (5.31b) and b_{t-1} from Equation (5.30). Finally, we sample b_T from its full conditional (Equation (5.27), analogously to the PG sampler. This leads to a related method proposed by Lindsten et al. [93], referred to as particle Gibbs with ancestor sampling (PGAS).

In summary, PGAS is a Gibbs sampler for the extended target distribution ϕ defined in Equation (5.10), using the following sweep:

(i) Draw $\theta' \sim \phi(\theta \mid x_{1:T}^{b_{1:T}}, b_{1:T})$;

(ii) Draw $\mathbf{x}_1^{\prime,-b_1} \sim \phi(\mathbf{x}_1^{-b_1} \mid \theta, x_{1:T}^{b_{1:T}}, b_{1:T})$ and, for $t = 2, \ldots, T$,

$$\{\mathbf{x}_t^{\prime,-b_t}, \mathbf{a}_t^{\prime,-b_t}\} \sim \phi(\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t} \mid \theta', \mathbf{x}_{1:t-1}^{\prime,-b_{1:t-1}}, \mathbf{a}'_{2:t-1}, x_{1:T}^{b_{1:T}}, b_{t-1:T});$$

$$(a_t^{\prime,b_t} =)\ b'_{t-1} \sim \phi(b_{t-1} \mid \theta', \mathbf{x}_{1:t-1}^{\prime,-b_{1:t-1}}, \mathbf{a}'_{2:t-1}, x_{1:T}^{b_{1:T}}, b_{t:T});$$

(iii) Draw $(k' =)\ b'_T \sim \phi(b_T \mid \theta', \mathbf{x}_{1:T}^{\prime,-b_{1:T}}, \mathbf{a}'_{2:T}, x_{1:T}^{b_{1:T}})$.

Hence, at each iteration of the CSMC sampler, we draw a new value for the ancestor index $a_t^{b_t}$, instead of setting this according to the conditioning. The resulting method, denoted CSMC with ancestor sampling, is given in Algorithm 19. Again, for convenience we put the conditioned particles in the Nth positions, i.e., we set $b_{1:T} = (N, \ldots, N)$. The PGAS sampler is summarized in Algorithm 20.

As for PG and PGBS, it can be verified that the above procedure is a partially collapsed Gibbs sampler and it will thus leave ϕ invariant. Furthermore, the ergodicity result from Theorem 5.2 applies without modification to PGAS. In practice, the ancestor sampling in the CSMC procedure gives rise to a considerable improvement over PG, comparable to that of backward simulation. The method has been successfully

[2] As pointed out above, Algorithm 16 fixates $b_{1:T} = (N, \ldots, N)$ for the conditioned path.

Algorithm 19 CSMC with ancestor sampling (conditioned on $x'_{1:T}$)

1: Draw $x_1^i \sim r_1^\theta(x_1)$ for $i = 1, \ldots, N-1$ and set $x_1^N = x'_1$.
2: Compute $\bar{w}_1^i = W_1(x_1^i)$ for $i = 1, \ldots, N$.
3: Normalize the weights $w_1^i = \bar{w}_1^i / \sum_l \bar{w}_1^l$ for $i = 1, \ldots, N$.
4: **for** $t = 2$ **to** T **do**
5: Draw $\{a_t^i, x_t^i\} \sim M_t^\theta(a_t, x_t)$ for $i = 1, \ldots, N-1$ and set $x_t^N = x'_t$.
6: Draw a_t^N with

$$P(a_t^N = i) \propto w_{t-1}^i \frac{\bar{\gamma}_T^\theta(\{x_{1:t-1}^i, x'_{t:T}\})}{\bar{\gamma}_{t-1}^\theta(x_{1:t-1}^i)}.$$

7: Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$ for $i = 1, \ldots, N$.
8: Compute $\bar{w}_t^i = W_t^\theta(x_{1:t}^i)$ for $i = 1, \ldots, N$.
9: Normalize the weights $w_t^i = \bar{w}_t^i / \sum_l \bar{w}_t^l$ for $i = 1, \ldots, N$.
10: **end for**

Algorithm 20 Particle Gibbs with ancestor sampling [93]

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily.
2: **for** $r \geq 1$ **do**
3: Draw $\theta[r] \sim \gamma_T(\theta \mid x_{1:T}[r-1])$.
4: Run CSMC with ancestor sampling (Algorithm 19) targeting $\gamma_T^{\theta[r]}(x_{1:T})$, conditioned on $x_{1:T}[r-1]$.
5: Sample k with $P(k = i) = w_T^i$ and trace the ancestral path of particle x_T^k, i.e., set $x_{1:T}[r] = x_{1:T}^k$.
6: **end for**

applied to challenging inference problems, such as Wiener system identification [95] and learning of nonparametric, nonlinear SSMs [55]. We illustrate the PGAS method in the following example.

Example 5.5 (PGAS for stochastic volatility model). We return again to the toy problem studied in Examples 5.3 and 5.4. We now apply the PGAS sampler to the same batches of data, for $T = 100$ and $T = 1\,000$, respectively. As before, we consider four independent runs with $N = 5$, 20, 100 and 1 000, respectively. The empirical ACFs for the

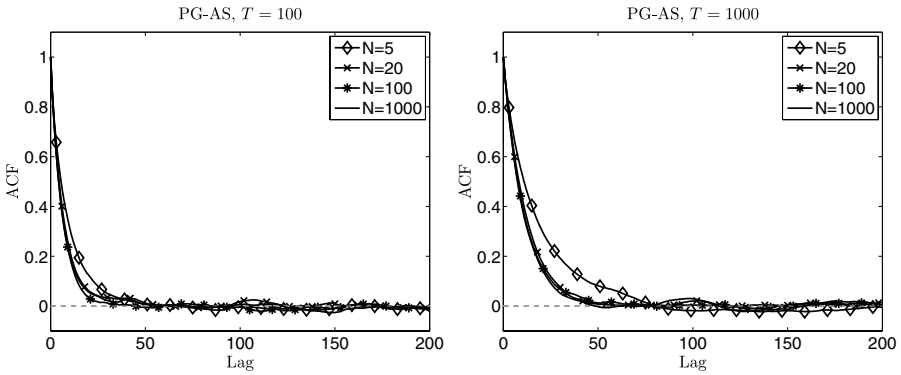

Fig. 5.6 Empirical ACF for θ for the PGAS samplers using different numbers of particles N.

residuals $\theta[r] - \mathbb{E}[\theta \mid y_{1:T}]$, based on 100 000 iterations of the samplers, are given in Figure 5.6.

The results are similar to those obtained by PGBS; see Figure 5.4. As for PGBS, in this specific example, the PGAS sampler performs very close to an ideal Gibbs sampler for $N \gtrsim 20$. As was discussed in the previous section, the source of the poor mixing of the basic PG sampler is path degeneracy. We have previously seen how backward simulation can be used to mitigate this problem, which explains why PGBS can outperform the basic PG sampler. However, it might be harder to see why the ancestor sampling used in PGAS gives the same effect. In fact, the paths generated by the CSMC procedure with ancestor sampling (Algorithm 19) will degenerate. The difference from the basic CSMC procedure (Algorithm 16, page 107), however, is that they do not degenerate to the conditioned path.

To understand the meaning of this, let us again look at two consecutive iterations of the sampler in more detail, using $N = 20$ and $T = 50$ as before. Figure 5.7 (top) shows the particles generated by the CSMC procedure with ancestor sampling, at iteration r. As pointed out above, due to path degeneracy, all the trajectories coincide for times t far from the final time T. The extracted trajectory $x_{1:T}[r]$, corresponding to the ancestral path of particle x_T^k, is illustrated by a thick black line.

At the next iteration of the PGAS sampler we run CSMC with ancestor sampling, conditioned on $x_{1:T}[r]$. This results in the particle

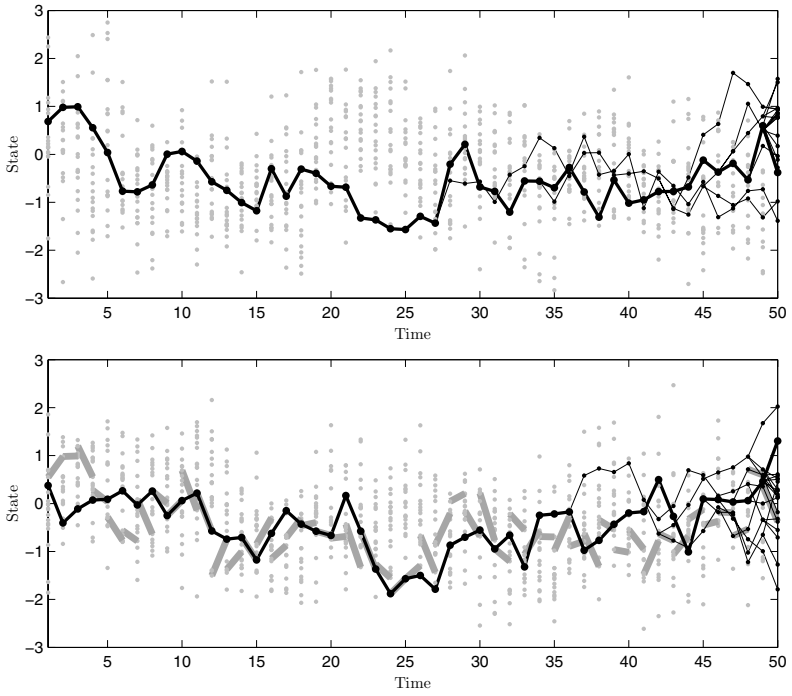

Fig. 5.7 Particle system generated by CSMC with ancestor sampling at iterations r (top) and $r+1$ (bottom) of the PGAS sampler. The dots show the particle positions, the thin black lines show the ancestral dependence of the particles and the thick black lines show the sampled trajectories $x_{1:T}[r]$ and $x_{1:T}[r+1]$, respectively. In the bottom pane, the thick gray line fragments illustrate the conditioned path at iteration $r+1$, with newly sampled ancestor indices. As an effect of ancestor sampling, the particle system at iteration $r+1$ does not degenerate to the conditioned path.

system shown in Figure 5.7 (bottom). As in Figures 5.3 and 5.5, the conditioned path is illustrated by a thick gray line. However, since, at each time point t, we sample a new ancestor for the conditioned particle $x_t[r]$, the conditioned path is broken up into pieces. This has the effect that the particle system generated at iteration $r+1$ of PGAS degenerates to a path which, to a large extent, is different from the conditioned path $x_{1:T}[r]$. Similar to using backward simulation, the effect is that PGAS explores the state-space around the conditioned path much better than PG, resulting in a faster mixing Gibbs kernel.

As could be seen in the example above, the use of ancestor sampling in PGAS gives a similar improvement over the basic PG sampler, as the backward simulation pass used in PGBS. This is not surprising, since both methods in fact make use of the same collections of Gibbs steps to sample from the extended target ϕ. The difference between the methods lies in the order in which these Gibbs steps are carried out. Here, PGBS requires separate forward and backward passes, whereas PGAS only proceeds in the forward direction, without the need of an explicit backward pass.

In [93], some more practical differences between the methods are pointed out. Primarily, they derive the PGAS sampler with the class of non-Markovian latent variable models in mind. They then make use of a weight truncation as discussed in Section 4.5, and illustrate empirically that PGAS is more robust to the truncation error than alternative backward-simulation-based methods, such as PGBS. Additionally, they point out that PGAS can be implemented more straightforwardly than PGBS, especially in the case of non-Markovian models. The reason is that, since it only proceeds in the forward direction, there is no need to store or regenerate intermediate quantities from the SMC sampler, thus requiring less bookkeeping. For the same reason, PGAS can be more memory efficient than alternative backward-simulation-based methods, in particular when the state is high-dimensional or equipped with some high-dimensional statistic.

5.6 PMCMC for Maximum Likelihood Inference

The PMCMC framework provides efficient methods for Bayesian parameter inference, which rely on the notion of *exact approximation*. However, as noted by Donnet and Samson [34], Andrieu and Vihola [7], and Lindsten [91], these attractive methods are not exclusive to the Bayesian. Indeed, it is possible to make use of PMCMC kernels when addressing, for instance, maximum likelihood problems as well. In this section, we review the method by Lindsten[91], which is a procedure for maximum likelihood parameter inference in general SSMs, based on CSMC with ancestor sampling (Algorithm 19, page 118).

For ease of comparison with the PSEM methods discussed in Section 3.5 and to highlight the fact that we are indeed dealing with a maximum likelihood problem, we restrict our attention to SSMs. However, the same approach can straightforwardly be used for parameter inference in the class of general latent variable models considered above. Hence, consider an SSM, parameterized by $\theta \in \Theta$,

$$x_{t+1} \sim f_\theta(x_{t+1} \mid x_t), \tag{5.32a}$$

$$y_t \sim g_\theta(y_t \mid x_t), \tag{5.32b}$$

and $x_1 \sim \mu_\theta(x_1)$. We observe a batch of measurements $y_{1:T}$ and seek the maximum likelihood estimator,

$$\widehat{\theta}_{\mathrm{ML}} = \arg\max_{\theta \in \Theta} \log p_\theta(y_{1:T}). \tag{5.33}$$

In Section 3.5, we derived a PSEM algorithm to address this problem. However, we also noted that PSEM relies on asymptotics in the number of particles and backward trajectories to obtain a convergent sequence of parameter estimates. Hence, a computationally expensive particle smoother has to be run at each iteration of the algorithm, leading to a learning procedure with a very high computational cost.

This issue has been recognized as a problem with the basic Monte Carlo EM algorithm, of which PSEM can be seen as a generalization. To be able to make more efficient use of the simulated variables in Monte Carlo EM, a related method, referred to as stochastic approximation EM (SAEM) was proposed by Delyon et al. [32]. This method uses a stochastic approximation update of the auxiliary quantity of the EM algorithm. In the SSM setting, this quantity is defined in Equation (1.5), and we thus get the following approximation,

$$\widehat{Q}_r(\theta) = (1 - \alpha_r)\widehat{Q}_{r-1}(\theta) + \alpha_r \left(\frac{1}{M_r} \sum_{j=1}^{M_r} \log p_\theta(\widetilde{x}_{1:T}, y_{1:T}) \right). \tag{5.34}$$

In the vanilla form of SAEM, the samples $\{\widetilde{x}^j_{1:T}\}_{j=1}^M$ are independent draws from the JSD $p_{\theta[r-1]}(x_{1:T} \mid y_{1:T})$, parameterized by the current estimate $\theta[r-1]$ (assuming for the time being that these samples can be generated).

As for PSEM, the M-step of the EM algorithm remains unchanged, but now we maximize the stochastic approximation $\widehat{Q}_r(\theta)$ instead of $Q(\theta, \theta[r-1])$. In Equation (5.34), $\{\alpha_r\}_{r\geq 1}$ is a decreasing sequence of positive step sizes, satisfying the usual stochastic approximation conditions, $\sum_r \alpha_r = \infty$ and $\sum_r \alpha_r^2 < \infty$; see e.g., [88]. In SAEM, all simulated values contribute to $\widehat{Q}_{r-1}(\theta)$, but they are down-weighted using a forgetting factor given by the step size. Under appropriate assumptions, SAEM can be shown to converge for fixed M_r (e.g., $M_r \equiv 1$), as $r \to \infty$ [32, 23]. When the simulation step is computationally involved, there is a considerable computational advantage of SAEM over Monte Carlo EM [32].

To make use of this approach, we still need to generate samples from the JSD. One option is of course to apply a standard FFBSi to generate a collection of backward trajectories $\{\widetilde{x}_{1:T}^j\}_{j=1}^{M_r}$ which are used in Equation (5.34). Due to the fact that we now use a stochastic approximation update, a small number of backward trajectories (e.g., $M_r \equiv 1$), are sufficient. However, we still require these trajectories to be generated from a distribution with a small (and diminishing) discrepancy from the JSD. In other words, this approach still relies on asymptotics in the number of forward filter particles N_r, in order to obtain accurate backward kernel approximations. This is not satisfactory, since we aim for a method which enjoys the exact approximation property obtained in the PMCMC setting.

To enable this, and thus be able to further reduce the computational complexity, we will use a Markovian version of stochastic approximation [12, 5]. It has been recognized that it is not necessary to sample exactly from the posterior distribution of the latent variables, to assess convergence of the SAEM algorithm. Indeed, it is sufficient to simulate from a family of Markov kernels $\{K_\theta : \theta \in \Theta\}$, leaving the family of posteriors invariant [86]. This is where PMCMC comes into play. Using any one of the PMCMC samplers presented above, we can construct a family of Markov kernels on X^T, such that, for each $\theta \in \Theta$, $K_\theta(x'_{1:T} \mid x_{1:T})$ leaves the JSD $p_\theta(x_{1:T} \mid y_{1:T})$ invariant. In combination with Markovian SAEM, we refer to this approach as particle SAEM (PSAEM).

To be more specific, we follow [91] and employ the CSMC with ancestor sampling presented in Algorithm 19 on page 118. That is,

at iteration r, we first run a CSMC sampler with ancestor sampling, targeting $p_{\theta[r-1]}(x_{1:T} \mid y_{1:T})$. We then sample one of the particle trajectories, with probabilities given by the importance weights at the final time point. With a similar argument as was used to prove the validity of the PGAS sampler (see Section 5.5), this procedure will leave the JSD invariant.

In [91], it is suggested to reuse all the particle trajectories in the approximation of the auxiliary quantity Q. This amounts to a Rao–Blackwellization, similar to Equation (5.22). Hence, let $\{x^i_{1:T}, w^i_T\}_{i=1}^N$ be the weighted particle system generated by Algorithm 19. We then compute a stochastic approximation according to

$$\widehat{Q}_r(\theta) = (1 - \alpha_r)\widehat{Q}_{r-1}(\theta) + \alpha_r \sum_{i=1}^{N} w^i_T \log p_\theta(x^i_{1:T}, y_{1:T}). \tag{5.35}$$

This approximation is then maximized w.r.t. θ in the M-step of the EM algorithm. We emphasize that, due to the use of stochastic approximation updates and the invariance property of the CSMC with ancestor sampling, this approach does not rely on asymptotics in N.

We summarize the method in Algorithm 21, and illustrate its performance in Example 5.6 below.

Example 5.6 (PSAEM). We return to the nonlinear time-series model studied in Example 3.4. We use the same settings and the same batch of data with $T = 1500$ observations. The unknown parameters are given by the process noise and measurement noise variances, i.e., $\theta = (\sigma_v^2, \sigma_e^2)$. We apply Algorithm 21 with $N = 15$ particles for 2000 iterations, initialized at $\theta[0] = (2, 2)$. We let $\alpha_r \equiv 1$ for $r \leq 100$, and $\alpha_r \sim r^{-0.7}$ for $r > 100$. This allows for a rapid change in the parameter estimates during the initial iterations, followed by a convergent phase. The resulting parameter estimates $\theta[r]$ are shown in Figure 5.8. As can be seen, the estimates converge to values close to the true parameters, despite the fact that we use a fixed (and small) number of particles. Compared to PSEM (see Figure 3.4), there is a significant improvement in terms of variance of the estimates and, most notably, in computational complexity.

Algorithm 21 PSAEM using CSMC with ancestor sampling

1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily. Set $\widehat{Q}_0(\theta) \equiv 0$.
2: **for** $r \geq 1$ **do**
3: Run CSMC with ancestor sampling (Algorithm 19) targeting $p_{\theta[r-1]}(x_{1:T} \mid y_{1:T})$, conditioned on $x_{1:T}[r-1]$.
4: Compute $\widehat{Q}_r(\theta)$ according to Equation (5.35).
5: Compute $\theta[r] = \arg\max_{\theta \in \Theta} \widehat{Q}_r(\theta)$.
6: **if** convergence criterion is met **then**
7: **break**
8: **end if**
9: Sample k with $P(k = i) = w_T^i$ and trace the ancestral path of particle x_T^k, i.e., set $x_{1:T}[r] = x_{1:T}^k$.
10: **end for**
11: **return** $\hat{\theta}_{\text{PSAEM}} = \theta[r]$.

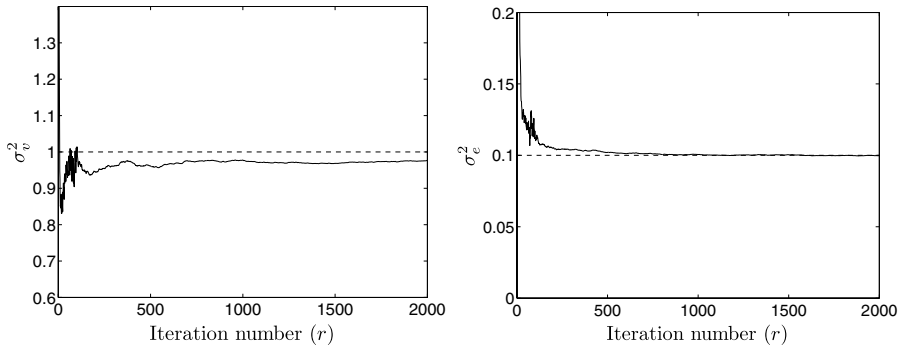

Fig. 5.8 Estimates of σ_v^2 (left) and σ_e^2 (right) vs. the iteration number k. The true parameter values are shown as dashed lines.

5.7 PMCMC for State Smoothing

Before we leave this section on PMCMC, it is worth to emphasize that all the PMCMC methods covered in this section can be used for state inference, as well as for parameter inference. The PMCMC samplers target the density $\gamma_T(\theta, x_{1:T})$, or, in the SSM setting, $p(\theta, x_{1:T} \mid y_{1:T})$. Consequently, if there are no unknown parameters in the models, the JSD $p(x_{1:T} \mid y_{1:T})$ can be obtained as a special case.

In fact, the addition of backward simulation to the PMMH sampler, as discussed in Section 5.3, is a way to improve the inference about the state, but does not alter the inferential performance for the parameter. Hence, this method should primarily be seen as a state smoother, and can be thought of as a way to Metropolise the backward simulator in Algorithm 12 (page 74) or the FFBSi in Algorithm 4 (page 38). See also Example 5.2.

Similarly, the PGAS and PGBS samplers can be applied equally well in the absence of any unknown parameter θ, by simply skipping the corresponding Gibbs step. These methods can thus also be used as state smoothers. In fact, these methods could prove to be valuable alternatives to the backward simulator in Algorithm 12. Since they require much fewer particles than a stand-alone backward simulator, they could potentially reach higher accuracy at the same computational cost, despite the fact that they need many iterations of the outer MCMC loop. To date, no exhaustive comparison of these different approaches to smoothing has been made. Furthermore, it is likely that the preference for one method over another has to be investigated on a case-by-case basis.

6

Discussion

The purpose of this tutorial has been to present and discuss various backward simulation methods for Monte Carlo statistical inference. We have focused on SMC-based backward simulators, which are useful for inference in analytically intractable models, such as nonlinear and/or non-Gaussian SSMs, but also in more general latent variable models. This is an active area of research. Inference for linear Gaussian and finite state-space models is by now well understood. For nonlinear models, on the other hand, there are still many challenges and open problems. This is apparent, for instance, by looking at the large number of related methods that have been presented in the literature during the past few years, many of which have been reviewed.

We have seen that there exist a large number of different approaches to backward simulation based on SMC. Indeed, backward simulation underlies a wide range of related particle smoothers, many of which were discussed in Sections 3–4 and in Section 5.7. Hence, when facing a certain state inference problem, we have many possible methods to choose from. Which method that is most suitable for the given problem has to be evaluated on a case-by-case basis, based on specific model properties, variance-bias trade-offs, etc. Computational complexities

and overheads, memory usage and possibilities for parallelization are also important factors that have to be taken into account. The backward simulators presented in Sections 3–4 are particularly well suited for parallelization, since the backward trajectories can be generated independently. To the best of the authors' knowledge, no exhaustive comparison of different particle smoothing algorithms has been conducted to date.

As we have seen, backward simulation is useful for both state inference and parameter inference. For instance, PMCMC (discussed in Section 5) enables Bayesian parameter learning in complicated latent variable models. In this context, the particle Gibbs samplers presented in Sections 5.4 and 5.5 used backward simulation as a way to do *exact approximate* grouping of the state variables, in order to improve the mixing over a single-state Gibbs sampler. We also made use of backward-simulation-based methods for maximum likelihood parameter inference through PSEM in Section 3.5 and particle SAEM in Section 5.6.

Both the Gibbs samplers and the EM-based methods for parameter inference rely on inherent state inference problems to be solved. In EM, the E-step amounts to solving a smoothing problem, and in the Gibbs sampler, the states are used as auxiliary variables to enable learning of the parameters. Hence, it is not surprising that state inference procedures can be used as components in these algorithms. However, there are alternative methods for parameter inference based on SMC, which do note make use of intermediate state inference steps in the same way. These include the PMMH sampler presented in Section 5.2 and direct maximum likelihood methods (e.g., based on zero-order stochastic optimization). For these methods, it is not clear that backward simulation can aid in solving the parameter inference problem. Indeed, in Section 5.3 we saw that the acceptance probability of the PMMH sampler was unaffected by the introduction of a backward simulator. Whether or not it is possible to exploit the backward simulation idea for these parameter inference methods as well, is a question which requires further investigation.

Many of the algorithms discussed throughout this tutorial can be thought of as (non-trivial) combinations of more basic algorithms. For

instance, the samplers presented in Section 3.4 make use of MCMC within SMC to generate backward trajectories. PMCMC goes the other way around and makes use of SMC within MCMC to construct specialized Markov kernels. Due to the large number of alternative methods that have been developed, it is possible to come up with many more composite algorithms in the same way. For instance, the MH-IPS discussed in Section 9 makes use of single-state Gibbs samplers to rejuvenate a degenerate particle system produced by a forward filter. Promising results have been reported for this method, but we have also experienced that single-state samplers can have slow convergence. An alternative is to make use of PMCMC samplers (e.g., PGAS, see Section 5.5) to update the particle trajectories in MH-IPS instead. This will, among other things, open up for easy parallelization of PMCMC samplers. It is not unlikely that such composite methods can prove to be serious competitors to the more basic ones, since they are able to exploit the strengths of each of the components. An obvious drawback, however, is that by combining more and more advanced methods, the implementation and analysis of the resulting algorithms will become increasingly more complex.

Another direction of future work is to tailor backward simulators for specific model classes. We saw an example of this with the Rao–Blackwellized FFBSi for conditionally linear Gaussian models in Section 4.4. Other models of central interest are those discussed in Section 4.1. For instance, the tree sampler for undirected graphical models discussed in Section 4.1.1 exploited a partitioning of the graph into disjoint chains. Under this partitioning, a sequential structure of the latent variables can be identified which makes the problem well suited for SMC and backward simulation. However, this partitioning is only one possibility. In [69, 68] other alternatives are considered, e.g., to partition the graph into branching trees. Application of backward simulation to these models, or more generally to graphical models containing loops, requires further generalizations of the algorithmic framework. Interesting developments in this direction are made in [19, 76, 130] where SMC is used for loopy belief propagation in general undirected graphical models.

The example considered in Section 4.1.2, i.e., to use inference strategies to solve optimal control problems, is also an interesting topic for future work. As previously mentioned, this problem has been addressed using transdimensional MCMC [75, 42]. By combining this approach with PMCMC, it might be possible to design efficient samplers for the optimal control problem in a rather general setting. If fact, the combination of transdimensional sampling and PMCMC is interesting in its own right, with many potential applications.

The challenging classes of non-Markovian models discussed in Section 4.6 are also of key relevance. Although it is possible to apply the general backward simulator of Algorithm 12 (page 74) to these models, this is plagued by a high computational cost. To find efficient samplers for these models is a topic for future work. There are also open challenges in applying backward-simulation-based methods to Bayesian nonparametric models (see e.g., [73]), such as Dirichlet process mixture models [9, 52]. The Chinese restaurant process [1, 14] can be thought of as a non-Markovian, sequential latent variable model, suggesting that the inference methods discussed in this tutorial can be applied to models containing this structure. However, it is not obvious how to correctly represent the backward kernel in such nonparametric settings.

Acknowledgments

This work was supported by the project Calibrating Nonlinear Dynamical Models (Contract number: 621-2010-5876) funded by the Swedish Research Council and CADICS, a Linnaeus Center also funded by the Swedish Research Council.

Notations and Acronyms

Abbreviation	Meaning
ACF	Autocorrelation function
APF	Auxiliary particle filter
CLGSS	Conditionally linear Gaussian state-space
CLT	Central limit theorem
CSMC	Conditional sequential Monte Carlo
ESS	Effective sample size
EM	Expectation maximization
FFBSi	Forward filter/backward simulator
FFBSm	Forward filter/backward smoother
GP	Gaussian process
HMM	Hidden Markov model
JMLS	Jump Markov linear system
JSD	Joint smoothing density
LGSS	Linear Gaussian state-space
MCMC	Markov chain Monte Carlo
MH	Metropolis–Hastings
MH-FFBP	Metropolis–Hastings forward filter/backward proposing
MH-FFBSi	Metropolis–Hastings forward filter/backward simulator
MH-IPS	Metropolis–Hastings improved particle smoother
MLE	Maximum-likelihood estimator
MRF	Markov random field
PF	Particle filter
PG	Particle Gibbs
PGAS	Particle Gibbs with ancestral sampling
PGBS	Particle Gibbs with backward simulation
PIMH	Particle independent Metropolis–Hastings

PMCMC	Particle Markov chain Monte Carlo
PMMH	Particle marginal Metropolis–Hastings
PSAEM	Particle stochastic approximation expectation maximization
PSEM	Particle smoother expectation maximization
RBPF	Rao–Blackwellized particle filter
RMSE	Root-mean-square error
RS	Rejection sampling
RS-FFBSi	Rejection sampling forward filter/backward simulator
SAEM	Stochastic approximation expectation maximization
SMC	Sequential Monte Carlo
SSM	State-space model
TV	Total variation

References

[1] D. Aldous, "Exchangeability and related topics," in *École d'Été de Probabilités de Saint-Flour XIII–1983*, (P. L. Hennequin, ed.), Springer, 1985.

[2] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine Learning*, vol. 50, no. 1, pp. 5–43, 2003.

[3] C. Andrieu, A. Doucet, and R. Holenstein, "Particle Markov chain Monte Carlo methods," *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 3, pp. 269–342, 2010.

[4] C. Andrieu and S. J. Godsill, "A particle filter for model based audio source separation," in *Proceedings of the 2000 International Workshop on Independent Component Analysis and Blind Signal Separation (ICA)*, Helsinki, Finland, June 2000.

[5] C. Andrieu, E. Moulines, and P. Priouret, "Stability of stochastic approximation under verifiable conditions," *SIAM Journal on Control and Optimization*, vol. 44, no. 1, pp. 283–312, 2005.

[6] C. Andrieu and G. O. Roberts, "The pseudo-marginal approach for efficient Monte Carlo computations," *The Annals of Statistics*, vol. 37, no. 2, pp. 697–725, 2009.

[7] C. Andrieu and M. Vihola, "Markovian stochastic approximation with expanding projections," arXiv.org, arXiv:1111.5421, November 2011.

[8] C. Andrieu and M. Vihola, "Convergence properties of pseudo-marginal Markov chain Monte Carlo algorithms," arXiv.org, arXiv:1210.1484, October 2012.

[9] C. E. Antoniak, "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems," *The Annals of Statistics*, vol. 2, no. 6, pp. 1152–1174, 1974.

[10] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[11] M. A. Beaumont, "Estimation of population growth or decline in genetically monitored populations," *Genetics*, vol. 164, no. 3, pp. 1139–1160, 2003.

[12] A. Benveniste, M. Métivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations*. New York, USA: Springer-Verlag, 1990.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*, Information Science and Statistics. New York, USA: Springer, 2006.

[14] D. Blackwell and J. B. MacQueen, "Ferguson distributions via Polya urn schemes," *The Annals of Statistics*, vol. 1, no. 2, pp. 353–355, 1973.

[15] A. Blake, P. Kohli, and C. Rother, eds., *Markov Random Fields For Vision And Image Processing*. MIT Press, 2011.

[16] A. Bouchard-Côté, S. Sankararaman, and M. I. Jordan, "Phylogenetic inference via sequential Monte Carlo," *Systematic Biology*, vol. 61, no. 4, pp. 579–593, 2012.

[17] Y. Bresler, "Two-filter formulae for discrete-time non-linear bayesian smoothing," *International Journal of Control*, vol. 43, no. 2, pp. 629–641, 1986.

[18] M. Briers, A. Doucet, and S. Maskell, "Smoothing algorithms for state-space models," *Annals of the Institute of Statistical Mathematics*, vol. 62, no. 1, pp. 61–89, February 2010.

[19] M. Briers, A. Doucet, and S. S. Singh, "Sequential auxiliary particle belief propagation," in *Proceedings of the International Conference on Information Fusion (FUSION)*, July 2005.

[20] S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, eds., *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011.

[21] P. Bunch and S. Godsill, "Improved particle approximations to the joint smoothing distribution using Markov chain Monte Carlo," *IEEE Transactions on Signal Processing (submitted)*, 2012.

[22] O. Cappé, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[23] O. Cappé, E. Moulines, and T. Rydén, *Inference in Hidden Markov Models*. Springer, 2005.

[24] C. K. Carter and R. Kohn, "On Gibbs sampling for state space models," *Biometrika*, vol. 81, no. 3, pp. 541–553, 1994.

[25] R. Chen and J. S. Liu, "Mixture Kalman filters," *Journal of the Royal Statistical Society: Series B*, vol. 62, no. 3, pp. 493–508, 2000.

[26] R. Chen, X. Wang, and J. S. Liu, "Adaptive joint detection and decoding in flat-fading channels via mixture Kalman filtering," *IEEE Transactions on Information Theory*, vol. 46, no. 6, pp. 2079–2094, 2000.

[27] M. Chesney and L. Scott, "Pricing European currency options: A comparison of the modified Black-Scholes model and a random variance model," *The Journal of Financial and Quantitative Analysis*, vol. 24, no. 3, pp. 267–284, 1989.

[28] N. Chopin, "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference," *The Annals of Statistics*, vol. 32, no. 6, pp. 2385–2411, 2004.

[29] N. Chopin and S. S. Singh, "On the particle Gibbs sampler," arXiv.org, arXiv:1304.1887, April 2013.

[30] P. de Jong and N. Shephard, "The simulation smoother for time series models," *Biometrika*, vol. 82, no. 2, pp. 339–350, 1995.

[31] P. Del Moral, *Feynman-Kac Formulae — Genealogical and Interacting Particle Systems with Applications*, Probability and its Applications. Springer, 2004.

[32] B. Delyon, M. Lavielle, and E. Moulines, "Convergence of a stochastic approximation version of the EM algorithm," *The Annals of Statistics*, vol. 27, no. 1, pp. 94–128, 1999.

[33] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[34] S. Donnet and A. Samson, "EM algorithm coupled with particle filter for maximum likelihood parameter estimation of stochastic differential mixed-effects models," Technical Report hal-00519576, v2, Université Paris Descartes, MAP5, 2011.

[35] R. Douc, A. Garivier, E. Moulines, and J. Olsson, "Sequential Monte Carlo smoothing for general state space hidden Markov models," *Annals of Applied Probability*, vol. 21, no. 6, pp. 2109–2145, 2011.

[36] R. Douc and E. Moulines, "Limit theorems for weighted samples with applications to sequential Monte Carlo," *The Annals of Statistics*, vol. 36, no. 5, pp. 2344–2376, 2008.

[37] R. Douc, E. Moulines, and J. Olsson, "Optimality of the auxiliary particle filter," *Probability and Mathematical Statistics*, vol. 29, pp. 1–28, 2009.

[38] A. Doucet, N. de Freitas, and N. Gordon, eds., *Sequential Monte Carlo Methods in Practice*. New York, USA: Springer Verlag, 2001.

[39] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[40] A. Doucet, S. J. Godsill, and M. West, "Monte Carlo filtering and smoothing with application to time-varying spectral estimation," in *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Istanbul, Turkey, June 2000.

[41] A. Doucet and A. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*, (D. Crisan and B. Rozovsky, eds.), Oxford University Press, 2011.

[42] A. Doucet, A. M. Johansen, and V. B. Tadić, "On solving integral equations using Markov chain Monte Carlo methods," *Applied Mathematics and Computation*, vol. 216, pp. 2869–2880, 2010.

[43] A. Doucet, M. K. Pitt, and R. Kohn, "Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator," arXiv.org, arXiv:1210.1871, October 2012.

[44] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[45] C. Dubarry and S. L. Corff, "Non-asymptotic deviation inequalities for smoothed additive functionals in non-linear state-space models," arXiv.org, arXiv:1012.4183v2, April 2012.

[46] C. Dubarry and R. Douc, "Particle approximation improvement of the joint smoothing distribution with on-the-fly variance estimation," arXiv.org, arXiv:1107.5524, July 2011.

[47] J. Durbin and S. J. Koopman, "A simple and efficient simulation smoother for state space time series analysis," *Biometrika*, vol. 89, no. 3, pp. 603–616, 2002.

[48] P. Fearnhead, "Particle filters for mixture models with an unknown number of components," *Statistics and Computing*, vol. 14, pp. 11–21, 2004.

[49] P. Fearnhead, "Using random quasi-Monte-Carlo within particle filters, with application to financial time series," *Journal of Computational and Graphical Statistics*, vol. 14, no. 4, pp. 751–769, 2005.

[50] P. Fearnhead, O. Papaspiliopoulos, and G. O. Roberts, "Particle filters for partially observed diffusions," *Journal of the Royal Statistical Society: Series B*, vol. 70, no. 4, pp. 755–777, 2008.

[51] P. Fearnhead, D. Wyncoll, and J. Tawn, "A sequential smoothing algorithm with linear computational cost," *Biometrika*, vol. 97, no. 2, pp. 447–464, 2010.

[52] T. S. Ferguson, "A Bayesian analysis of some nonparametric problems," *The Annals of Statistics*, vol. 1, no. 2, pp. 209–230, 1973.

[53] W. Fong, S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing with application to audio signal enhancement," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 438–449, February 2002.

[54] G. Fort and E. Moulines, "Convergence of the Monte Carlo expectation maximization for curved exponential families," *The Annals of Statistics*, vol. 31, no. 4, pp. 1220–1259, 2003.

[55] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian inference and learning in Gaussian process state-space models with particle MCMC," arXiv.org, arXiv:1306.2861, June 2013.

[56] S. Frühwirth-Schnatter, "Data augmentation and dynamic linear models," *Journal of Time Series Analysis*, vol. 15, no. 2, pp. 183–202, 1994.

[57] A. E. Gelfand and A. F. M. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American Statistical Association*, vol. 85, no. 410, pp. 398–409, 1990.

[58] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, 1984.

[59] R. Gerlach, C. Carter, and R. Kohn, "Efficient Bayesian inference for dynamic mixture models," *Journal of the American Statistical Association*, vol. 95, no. 451, pp. 819–828, 2000.

[60] W. R. Gilks and C. Berzuini, "Following a moving target — Monte Carlo inference for dynamic Bayesian models," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 63, no. 1, pp. 127–146, 2001.

[61] M. Girolami and B. Calderhead, "Riemann manifold Langevin and Hamiltonian Monte Carlo methods," *Journal of the Royal Statistical Society: Series B*, vol. 73, no. 2, pp. 1–37, 2011.

[62] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 156–168, March 2004.

[63] A. Golightly and D. J. Wilkinson, "Bayesian inference for nonlinear multivariate diffusion models observed with error," *Computational Statistics & Data Analysis*, vol. 52, no. 3, pp. 1674–1693, 2008.

[64] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEEE Proceedings F*, vol. 140, no. 2, pp. 107–113, April 1993.

[65] P. J. Green, "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," *Biometrika*, vol. 82, no. 4, pp. 711–732, 1995.

[66] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[67] J. Hall, M. K. Pitt, and R. Kohn, "Bayesian inference for nonlinear structural time series models," arXiv.org, arXiv:1209.0253v2, September 2012.

[68] F. Hamze and N. de Freitas, "From fields to trees," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

[69] F. Hamze, J.-N. Rivasseau, and N. de Freitas, "Information theory tools to rank MCMC algorithms on probabilistic graphical models," in *Proceedings of the UCSD Information Theory Workshop*, 2006.

[70] J. Handschin and D. Mayne, "Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering," *International Journal of Control*, vol. 9, no. 5, pp. 547–559, May 1969.

[71] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[72] S. E. Hills and A. F. M. Smith, "Parameterization issues in Bayesian inference (with discussion)," in *Bayesian Statistics 4*, (J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, eds.), pp. 641–649, Oxford University Press, 1992.

[73] N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker, eds., *Bayesian Nonparametrics*. Cambridge University Press, 2010.

[74] M. Hoffman, N. de Freitas, A. Doucet, and J. Peters, "An expectation maximization algorithm for continuous Markov decision processes with arbitrary rewards," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, Clearwater Beach, FL, USA, 2009.

[75] M. Hoffman, H. Kueck, N. de Freitas, and A. Doucet, "New inference strategies for solving Markov decision processes using reversible jump MCMC," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 223–231, Corvallis, OR, USA, 2009.

[76] A. Ihler and D. McAllester, "Particle belief propagation," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, Clearwater Beach, FL, USA, 2009.

[77] M. Isard and A. Blake, "Condensation — conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[78] H. Ishwaran, "Applications of hybrid Monte Carlo to Bayesian generalized linear models: Quasicomplete separation and neural networks," *Journal of Computational and Graphical Statistics*, vol. 8, no. 4, pp. 779–799, 1999.

[79] A. M. Johansen and A. Doucet, "A note on auxiliary particle filters," *Statistics & Probability Letters*, vol. 78, no. 12, pp. 1498–1504, 2008.

[80] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. Upper Saddle River, NJ, USA: Prentice Hall, 2000.

[81] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[82] R. Karlsson and F. Gustafsson, "Particle filter for underwater navigation," in *Proceedings of the 2003 IEEE Workshop on Statistical Signal Processing (SSP)*, pp. 509–512, St. Louis, USA, September 2003.

[83] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[84] M. Klaas, M. Briers, N. de Freitas, A. Doucet, S. Maskell, and D. Lang, "Fast particle smoothing: if I had a million particles," in *Proceedings of the International Conference on Machine Learning*, Pittsburgh, USA, June 2006.

[85] M. Klaas, D. Lang, and N. de Freitas, "Fast maximum a posteriori inference in Monte Carlo state spaces," in *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[86] E. Kuhn and M. Lavielle, "Coupling a stochastic approximation version of EM with an MCMC procedure," *ESAIM: Probability and Statistics*, vol. 8, pp. 115–131, 2004.

[87] H. R. Künsch, "Recursive Monte Carlo filters: Algorithms and theoretical analysis," *The Annals of Statistics*, vol. 33, no. 5, pp. 1983–2021, 2005.

[88] H. J. Kushner and G. G. Yin, *Stochastic Approximation Algorithms and Applications*. Springer, 1997.

[89] D. Lang and N. de Freitas, "Beat tracking the graphical model way," in *Proceedings of the 2004 Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2004.

[90] E. L. Lehmann and G. Casella, *Theory of Point Estimation*. Springer Texts in Statistics. New York, USA: Springer, 2nd ed., 1998.

[91] F. Lindsten, "An efficient stochastic approximation EM algorithm using conditional particle filters," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

[92] F. Lindsten, P. Bunch, S. J. Godsill, and T. B. Schön, "Rao-Blackwellized particle smoothers for mixed linear/nonlinear state-space models," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

[93] F. Lindsten, M. I. Jordan, and T. B. Schön, "Ancestor sampling for particle Gibbs," in *Advances in Neural Information Processing Systems 25*, (P. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 2600–2608, 2012.

[94] F. Lindsten and T. B. Schön, "On the use of backward simulation in the particle Gibbs sampler," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, March 2012.

[95] F. Lindsten, T. B. Schön, and M. I. Jordan, "Bayesian semiparametric Wiener system identification," *Automatica*, vol. 49, no. 7, pp. 2053–2063, 2013.

[96] F. Lindsten, T. B. Schön, and J. Olsson, "An explicit variance reduction expression for the Rao-Blackwellised particle filter," in *Proceedings of the 18th IFAC World Congress*, Milan, Italy, August 2011.

[97] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

[98] J. S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[99] S. N. MacEachern, M. Clyde, and J. S. Liu, "Sequential importance sampling for nonparametric Bayes models: The next generation," *The Canadian Journal of Statistics*, vol. 27, no. 2, pp. 251–267, 1999.

[100] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. Whiley Series in Probability and Statistics. New York, USA: John Wiley & Sons, second ed., 2008.

[101] A. Melino and S. M. Turnbull, "Pricing foreign currency options with stochastic volatility," *Journal of Econometrics*, vol. 45, no. 1–2, pp. 239–265, 1990.

[102] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[103] N. Metropolis and S. Ulam, "The Monte Carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[104] S. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. Cambridge University Press, 2nd ed., 2009.

[105] M. Montemerlo and S. Thrun, *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*. Berlin, Germany: Springer, 2007.

[106] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.

[107] P. D. Moral, A. Doucet, and A. Jasra, "Sequential Monte Carlo samplers," *Journal of the Royal Statistical Society: Series B*, vol. 68, no. 3, pp. 411–436, 2006.

[108] L. M. Murray, E. M. Jones, and J. Parslow, "On collapsed state-space models and the particle marginal Metropolis-Hastings sampler," arXiv.org, arXiv:1202.6159v1, February 2012.

[109] R. M. Neal, "MCMC using Hamiltonian dynamics," in *Handbook of Markov Chain Monte Carlo*, (S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, eds.), pp. 113–162, Chapman & Hall/CRC, 2011.

[110] R. M. Neal, M. J. Beal, and S. T. Roweis, "Inferring state sequences for non-linear systems with embedded hidden Markov models," in *Proceedings of the 2003 Conference on Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2003.

[111] M. L. A. Netto, L. Gimeno, and M. J. Mendes, "A new spline algorithm for non-linear filtering of discrete time systems," in *Proceedings of the 7th Triennial World Congress*, pp. 2123–2130, Helsinki, Finland, 1979.

[112] J. Olsson, R. Douc, O. Cappé, and E. Moulines, "Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state-space models," *Bernoulli*, vol. 14, no. 1, pp. 155–179, 2008.

[113] J. Olsson and T. Rydén, "Rao-Blackwellization of particle Markov chain Monte Carlo methods using forward filtering backward sampling," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4606–4619, 2011.

[114] O. Papaspiliopoulos, G. O. Roberts, and M. Sköld, "Non-centered parameterisations for hierarchical models and data augmentation," in *Bayesian Statistics 7*, (J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, eds.), pp. 307–326, Oxford University Press, 2003.

[115] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausble Inference*. San Francisco, CA, USA: Morgan Kaufmann, 2nd ed., 1988.

[116] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999.

[117] M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn, "On some properties of Markov chain Monte Carlo simulation methods based on the particle filter," *Journal of Econometrics*, vol. 171, pp. 134–151, 2012.

[118] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[119] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, August 1965.

[120] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2004.

[121] G. O. Roberts and S. K. Sahu, "Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler," *Journal of the Royal Statistical Society: Series B*, vol. 59, no. 2, pp. 291–317, 1997.

[122] D. B. Rubin, "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 algorithm," *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 543–546, June 1987. Comment to Tanner and Wong: The Calculation of Posterior Distributions by Data Augmentation.

[123] S. Särkkä, P. Bunch, and S. Godsill, "A backward-simulation based Rao-Blackwellized particle smoother for conditionally linear Gaussian models," in *Proceedings of the 16th IFAC Symposium on System Identification*, Brussels, Belgium, July 2012.

[124] M. N. Schmidt, "Function factorization using warped Gaussian processes," in *Proceedings of the International Conference on Machine Learning*, pp. 921–928, 2009.

[125] T. Schön, F. Gustafsson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2279–2289, July 2005.

[126] T. B. Schön and F. Lindsten, *Computational Learning in Dynamical Systems*. 2013. (forthcoming, draft manuscript is available from the authors).

[127] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

[128] J. C. Spall, "Estimation via Markov chain Monte Carlo," *IEEE Control Systems Magazine*, vol. 23, no. 2, pp. 34–45, 2003.

[129] L. Stewart and P. McCarty, "The use of Bayesian belief networks to fuse continuous and discrete information for target recognition, tracking, and situation assessment," in *Proceedings of the SPIE 1699, Signal Processing, Sensor Fusion, and Target Recognition*, 1992.

[130] E. B. Sudderth, A. T. Ihler, M. Isard, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," *Communications of the ACM*, vol. 53, no. 10, pp. 95–103, 2010.

[131] E. Taghavi, F. Lindsten, L. Svensson, and T. B. Schön, "Adaptive stopping for fast particle smoothing," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

[132] M. A. Tanner and W. H. Wong, "The calculation of posterior distributions by data augmentation," *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 528–540, June 1987.

[133] Y. W. Teh, H. Daumé III, and D. Roy, "Bayesian agglomerative clustering with coalescents," *Advances in Neural Information Processing*, pp. 1473–1480, 2008.

[134] L. Tierney, "Markov chains for exploring posterior distributions," *The Annals of Statistics*, vol. 22, no. 4, pp. 1701–1728, 1994.

[135] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state Markov decision processes," in *Proceedings of the International Conference on Machine Learning*, Pittsburgh, PA, USA, 2006.

[136] D. A. van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, pp. 1–50, March 2001.

[137] D. A. Van Dyk and T. Park, "Partially collapsed Gibbs samplers: Theory and methods," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 790–796, 2008.

[138] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[139] X. Wang, R. Chen, and D. Guo, "Delayed-pilot sampling for mixture Kalman filter with application in fading channels," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 241–254, 2002.

[140] G. C. G. Wei and M. A. Tanner, "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 699–704, 1990.

[141] N. Whiteley, "Discussion on Particle Markov chain Monte Carlo methods," *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 3, pp. 306–307, 2010.

[142] N. Whiteley, C. Andrieu, and A. Doucet, "Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods," Technical report, Bristol Statistics Research Report 10:04, 2010.

[143] N. Whiteley, C. Andrieu, and A. Doucet, "Bayesian computational methods for inference in multiple change-points models," *Submitted*, 2011.

[144] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, pp. 65–85, 1994.

[145] P. Wild and W. R. Gilks, "Algorithm AS 287: Adaptive rejection sampling from log-concave density functions," *Journal of the Royal Statistical Society: Series C*, vol. 42, no. 4, pp. 701–709, 1993.

[146] D. J. Wilkinson and S. K. H. Yeung, "Conditional simulation from highly structured Gaussian systems, with application to blocking-MCMC for the Bayesian analysis of very large linear models," *Statistics and Computing*, vol. 12, no. 3, pp. 287–300, July 2002.

[147] A. Wills, T. B. Schön, L. Ljung, and B. Ninness, "Identification of Hammerstein–Wiener models," *Automatica*, vol. 49, no. 1, pp. 70–81, 2013.

[148] C. F. J. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1983.