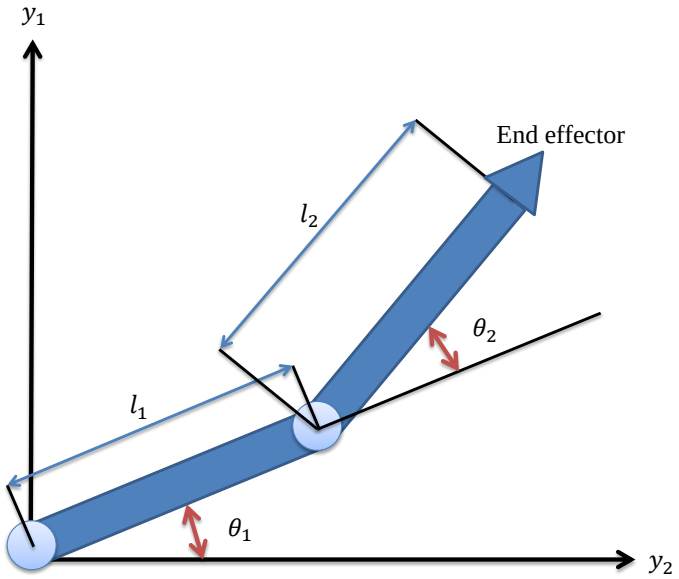
CHALMERS



A Study of Linear Complexity Particle Filter Smoothers

Ehsan Taghavi

Department of Signals and Systems
Signal Processing Group
CHALMERS UNIVERSITY OF TECHNOLOGY
Sweden, 2012
EX020/2012

Thesis for the degree of Master of Science

A Study of Linear Complexity Particle Filter Smoothers

by

Ehsan Taghavi

Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2012

A Study of Linear Complexity Particle Filter Smoothers
Ehsan Taghavi
EX020/2012

This thesis has been prepared using LaTeX.

Copyright © Ehsan Taghavi, 2012.
All rights reserved.

Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden

email: ehsant@student.chalmers.se and ehsan.taghavie@gmail.com

Cover:
A robot arm with planar tow-link manipulator.

To my family

A Study of Particle Filter Smoother

Ehsan Taghavi
Department of Signals and Systems
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology

Abstract

In the recent decade, sequential Monte Carlo (SMC) methods emerged as one of the solutions to non-linear and/or non-Gaussian state-space models (SSM). Smoothing, because of the twist in factorizing the problem, was not receiving enough attention. In recent years, researchers tried to find better solutions for the smoothing and proposed interesting smoothing algorithms with properties that makes them applicable for many of the problems. The most important property of recent proposed smoothing algorithms is that the complexity of these methods are linear in term of number of the particles. The goal of this thesis is to give a brief review of the available smoothing methods in a comprehensive way.

In this thesis we investigate some important advantages and disadvantages of existing smoothing algorithms with linear complexity and discuss about them in detail such as optimality of proposal choices and problems of methods which use rejection sampling to target the smoothing. Moreover, there is a discussion about the design variables for each of the algorithms as well as possible choices for these parameters. We also proposed solution to some of the problems of these algorithms, like the situations when fast forward-filtering backward-simulation (FFBSi) algorithm spend a lot of time to just sample one particle. Other than that we introduce methods to empower us for implementing new linear smoother algorithms and give one such example by using the backward information filter.

Keywords: Kalman filter, Kalman smoother, particle filter, particle smoother, complexity.

Acknowledgments

First and foremast, I would like to express my sincere gratitude to my supervisor Dr. Lennart Svensson for his great support during the last year in my study and research and for his kindness and patience. His guidance helped me a great deal throughout the research and for writing this thesis. Besides my supervisor, I would like to thank my father, mother and my brother Hamidreza for their kind and unending support during my studies. For sure without their support I wouldn't achieve this level of success.

Contents

Chapter 1

Introduction

One of the most interesting models that is used in a vast number of applications is state-space model. It has been for many years that researchers are trying to find a general solution for non-linear and/or non-Gaussian models by modifying the famous Kalman filter. In the recent decade, sequential Monte Carlo (SMC) methods emerged as one of the solutions to non-linear and/or non-Gaussian state-space models. Smoothing, because of the twist in factorizing the problem, was not receiving enough attention. In recent years, researchers tried to find better solutions for the smoothing and proposed interesting smoothing algorithms with properties that makes them applicable for many of the problems. The most important property of recent proposed smoothing algorithms is that the complexity of these methods are linear in term of number of the particles.

There are many problems both in engineering and statistics for which one needs to apply smoothing algorithms to target the desired density. Before running a smoothing algorithm, we need to run a particle filter, which has linear complexity in number of the particles. The linear complexity of particle filters helps us to run them with large number of the particles when it is necessary, like when we have a high dimension state-space model. But when it comes to smoothing algorithms, until recent years they were not able to satisfy researchers goals easily but at the cost of an algorithm with quadratic complexity in number of the particles.

Now that there are two algorithms available with linear complexity, it is an interesting topic to investigate them more in depth. One interesting aspect is that there is still no comparison between these two algorithms and by doing a comparison, it can be shown which of them performs better in

different scenarios. Moreover, each algorithm has its own design variables to be set. It is also important to understand how we can set these variables to be able to run the algorithms with higher performance.

1.1 Aim of study

The goal of this thesis is to review the methods and their implementations in detail as well as explaining them in a complete, comprehensive way. One of the important parts in reviewing the SMC algorithms is how to select the design variables like proposal densities. In this thesis, after explaining the different algorithms, we try to introduce different ways to find these choices which will give us a very good insight about the smoothing problem and how we can improve the algorithms.

One other important aspect in the implementation of the algorithms is how they perform for different problems. To cover the performance of the algorithms in different scenarios, a two-dimensional linear-Gaussian SSM is used to show how different algorithms will perform when one can use the optimal proposal densities. In addition, because the main goal of SMC smoothing algorithms are to face with non-linearities, a non-linear examples is also investigated to gain more insight on how and with which settings one can select the design variables.

1.2 Thesis outline

Outline of the thesis is as follows: In Chapter 2 we formulate a general state-space model and give the notations used in the next chapters. As Kalman filter/smoother gives us the optimal solution to the state estimation in the case of having a linear-Gaussian model, we describe this method briefly in Chapter 3 and use the results as benchmark for our comparisons.

In Chapter 4 the auxiliary particle filter is introduced as the standard sequential Monte Carlo (SMC) filtering method in this thesis. Then we describe filter-smoother (Kitagawa, 1996), fixed-interval smoothing (Doucet et al., 2000), forward-filtering backward-simulation (FFBSi) (Godsill et al., 2004), two-filter smoother (Briers et al., 2010) and new two-filter smoother (Fearnhead et al., 2010). Besides, the main two algorithms which are new linear complexity smoother (Fearnhead et al., 2010) and fast FFBSi (Douc et al., 2010) are introduced in detail. In Chapter 5, a different way of find-

ing the optimal proposal densities is given with complete detail as well as a comparison between these new optimal densities to the ones introduced in (Fearnhead et al., 2010). Chapter 6 is about introducing a new method which is an extension to backward information filter as the suggestion given in (Fearnhead et al., 2010) for the smoothing. This algorithm has the same linear complexity in our comparisons but further issues is discussed in the same section.

In Chapter 7 we discuss the design variables of the linear methods introduced recently and after that the results of simulations for a linear-Gaussian model and a simple non-linear model are shown in Chapter 9. Finally, in Chapter 10 a brief talk on the findings and open problems is given to conclude and discuss the main advantages and disadvantages of SMC smoother algorithms.

Chapter 2

Problem formulation

The general problem, which we work on, is a state-space model (SSM) in which the system is assumed to be a discrete Markov process with unobserved states. Figure 2.1 shows the general architecture of an SSM in which the arrows in the diagram denote conditional dependencies.

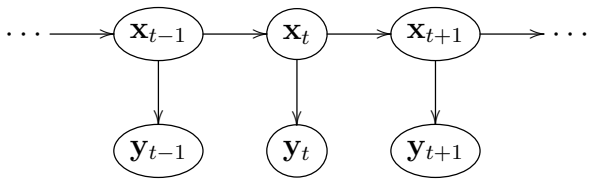


Figure 2.1: Graphical model of an SSM.

In Figure 2.1 the random variable $\mathbf{x}_t$ is the hidden state at time t and $\mathbf{y}_t$ is the observation at time t. This model can be formulated as

$$\mathbf{x}_{t+1} \mid (\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) \sim f(\cdot \mid \mathbf{x}_t) \tag{2.1}$$
$$\mathbf{y}_t \mid (\mathbf{x}_{1:t}, \mathbf{y}_{1:t-1}) \sim g(\cdot \mid \mathbf{x}_t), \tag{2.2}$$

where $f(\cdot)$ and $g(\cdot)$ denote arbitrary probability density functions (pdf) and $\mathbf{y}_{1:t} = \{y_1, y_2, ..., y_t\}$ and $\mathbf{x}_{1:t} = \{x_1, x_2, ..., x_t\}$ are the collection up to time t of the data and states respectively. This means here we do not have any restriction on having linear-Gaussian models (Roweis and Ghahramani, 1999). The model will be completed by defining an initial distribution for $\mathbf{x}_0$ at $t = 0$, i.e. $\mathbf{x}_0 \sim \nu(\mathbf{x}_0)$, where ν is an arbitrary density function.

Here our interest is to approximate $p(\mathbf{x}_t \mid \mathbf{y}_{1:T})$, where T denotes the last time step which we receive data from observation. This distribution is called

the smoothing density. Moreover to be able to approximate the smoothing density we need to have access to an approximation of the filtering that is $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$.

SSM is one of the general models that cover many of the problems in engineering and statistics as the densities and also the noises in the processes can be of any form. The problem arises here that how we can solve the filtering or the smoothing in the case of having a non-linear and/or non-Gaussian SSM? In the rest of this thesis we try to give a clear answer to this question. In Chapter 3 the classic method of solving this problem for a linear-Gaussian model (Kalman filter and smoother) is given with its derivations in a Bayesian framework. But in all other sections, our focus is to solve a general SSM and compare the different methods with each other from different perspectives such as accuracy and complexity.

Chapter 3

Kalman filter/smoother

3.1 Kalman filter

Kalman filtering (Kalman et al., 1960) is an iterative method of prediction-correction by using the measurements and state model available. In this method, the assumed model is a general linear state-space model and can be presented as

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{w}_{t-1} \tag{3.1}$$
$$\mathbf{y}_t = \mathbf{G}_t \mathbf{x}_t + \mathbf{v}_t, \tag{3.2}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the state vector at time t, $\mathbf{y}_t \in \mathbb{R}^n$ is the measurement vector at time t, $\mathbf{F}_t$ is transition matrix, $\mathbf{G}_t$ is measurement update matrix and $\mathbf{w}_t$ and $\mathbf{v}_t$ are independent random Gaussian samples with distribution $\mathcal{N}(0, \Sigma_{\mathbf{w}})$ and $\mathcal{N}(0, \Sigma_{\mathbf{v}})$ respectively. Because later on we will use Bayesian framework to derive any algorithm for particle filter/smoother, here we use the same framework to derive the formulas for Kalman filter.

Here we wish to recursively compute $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ given that we know $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$. To find the optimal solution for filtering which is $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, we start with factorizing it by using Bayes' rule as

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})}. \tag{3.3}$$

Assuming that we know the posterior distribution of previous time step, i.e. $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$, the joint distribution of $\mathbf{x}_t$, $\mathbf{x}_{t-1}$ given $\mathbf{y}_{1:t-1}$ can be

computed as

$$\begin{aligned} p(\mathbf{x}_t, \mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) &= p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) \\ &= f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}), \end{aligned} \tag{3.4}$$

which integrating over $\mathbf{x}_{t-1}$ gives the Chapman-Kolomogorov equation (Gardiner, 1985)

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \tag{3.5}$$

where normally we call it the prediction step of the optimal filter.

The posterior distribution can be computed by Bayes' rule

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) &= \frac{1}{Z_t} p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{y}_{1:t-1}) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) \\ &= \frac{1}{Z_t} g(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}), \end{aligned} \tag{3.6}$$

where $Z_t = \int p(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_t$ is a normalization factor. Equation (3.6) is called update step of the optimal filter.

Now if we want to drive analytically what are the prediction step and update step in a linear-Gaussian model, we can start with using the fact that

$$f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \mathbf{F}_t \mathbf{x}_{t-1}, \Sigma_{\mathbf{w}}) \tag{3.7}$$

$$g(\mathbf{y}_t \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{y}_t; \mathbf{G}_t \mathbf{x}_t, \Sigma_{\mathbf{v}}). \tag{3.8}$$

Furthermore we should assume that the posterior distribution of previous step is Gaussian with

$$p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_{t-1}; \hat{\mathbf{x}}_{t-1|t-1}, \hat{\mathbf{P}}_{t-1|t-1}). \tag{3.9}$$

Now the Chapman-Klomogorov equation gives

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) &= \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \\ &= \int \mathcal{N}(\mathbf{x}_t; \mathbf{F}_t \mathbf{x}_{t-1}, \Sigma_{\mathbf{w}}) \mathcal{N}(\mathbf{x}_{t-1}; \hat{\mathbf{x}}_{t-1|t-1}, \hat{\mathbf{P}}_{t-1|t-1}) d\mathbf{x}_{t-1}. \end{aligned} \tag{3.10}$$

Using the Gaussian distribution computation rules, we get the prediction step as

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t; \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1}, \mathbf{F}_t \hat{\mathbf{P}}_{t-1|t-1} \mathbf{F}^{\mathrm{T}} + \Sigma_{\mathbf{w}}), \tag{3.11}$$

where for simplicity in representing the equations we define $\mathbf{m}_t^- = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1}$ and $\mathbf{P}_t^- = \mathbf{F}_t \hat{\mathbf{P}}_{t-1|t-1} \mathbf{F}_t^{\mathrm{T}} + \Sigma_{\mathbf{w}}$.

The joint distribution of $\mathbf{y}_t$ and $\mathbf{x}_t$ is

$$\begin{aligned} p(\mathbf{x}_t, \mathbf{y}_t \mid \mathbf{y}_{1:t-1}) &= g(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) \\ &= \mathcal{N}(\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix}; \begin{bmatrix} \mathbf{m}_t^- \\ \mathbf{G}_t \mathbf{m}_t^- \end{bmatrix}, \begin{bmatrix} \mathbf{P}_t^- & \mathbf{P}_t^- \mathbf{G}_t^{\mathrm{T}} \\ \mathbf{G}_t \mathbf{P}_t^- & \mathbf{G}_t \mathbf{P}_t^- \mathbf{G}_t^{\mathrm{T}} + \Sigma_{\mathbf{v}} \end{bmatrix}). \end{aligned} \tag{3.12}$$

Finally, the conditional distribution of $\mathbf{x}_t$ given $\mathbf{y}_t$ is then given as

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_t, \mathbf{y}_{1:t-1}) &= p(\mathbf{x}_t; \mathbf{y}_{1:t}) \\ &= \mathcal{N}(\mathbf{x}_t; \mathbf{m}_t, \mathbf{P}_t), \end{aligned}$$

where

$$\begin{aligned} \mathbf{S}_t &= \mathbf{G}_t \mathbf{P}_t^- \mathbf{G}_t^{\mathrm{T}} + \Sigma_{\mathbf{v}} \\ \mathbf{K}_t &= \mathbf{P}_t^- \mathbf{G}_t^{\mathrm{T}} \mathbf{S}_t^{-1} \\ \mathbf{m}_t &= \mathbf{m}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{G}_t \mathbf{m}_t^- \right) \\ \mathbf{P}_t &= \mathbf{P}_t^- - \mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^{\mathrm{T}}. \end{aligned}$$

As result if we want to do the Kalman filtering we should start with initializing the states at t_0 with a Gaussian distribution with $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{m}_0, \mathbf{P}_0)$ and then iteratively do the following steps:

- Prediction step

$$\begin{aligned} \mathbf{m}_t^- &= \mathbf{F}_t \mathbf{m}_{t-1} \\ \mathbf{P}_t^- &= \mathbf{F}_t \mathbf{P}_{t-1} \mathbf{F}_t^{\mathrm{T}} + \Sigma_{\mathbf{w}}, \end{aligned}$$

- Update step

$$\begin{aligned} \mathbf{S}_t &= \mathbf{G}_t \mathbf{P}_t^- \mathbf{G}_t^{\mathrm{T}} + \Sigma_{\mathbf{v}} \\ \mathbf{K}_t &= \mathbf{P}_t^- \mathbf{G}_t^{\mathrm{T}} \mathbf{S}_t^{-1} \\ \mathbf{m}_t &= \mathbf{m}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{G}_t \mathbf{m}_t^- \right) \\ \mathbf{P}_t &= \mathbf{P}_t^- - \mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^{\mathrm{T}}. \end{aligned}$$

3.2 Kalman smoother

The derivation of Kalman smoother can be found in (Anderson and Moore, 1979; Shumway and Stoffer, 1982). The Kalman smoother calculates recursively the state posterior distributions $p(\mathbf{x}_t \mid \mathbf{y}_{1:T})$. The formulation in backward step is as follows

$$
\begin{aligned}
\mathbf{P}_{t,t+1} &= \mathbf{F}_t \mathbf{P}_t \mathbf{F}_t^T + \Sigma_{\mathbf{w}} && (3.13)\\
\mathbf{C}_t &= \mathbf{P}_t \mathbf{F}_t^T \mathbf{P}_{t,t+1}^{-1} && (3.14)\\
\hat{\mathbf{x}}_t^{smooth} &= \hat{\mathbf{x}}_t + \mathbf{C}_t \left(\hat{\mathbf{x}}_{t+1}^{smooth} - F_t \hat{\mathbf{x}}_t \right) && (3.15)\\
\mathbf{P}_t^{smooth} &= \mathbf{P}_t + \mathbf{C}_t \left(\mathbf{P}_{t+1}^{smooth} - \mathbf{P}_{t,t+1} \right) \mathbf{C}_t^T, && (3.16)
\end{aligned}
$$

starting from last step T, with $\hat{\mathbf{x}}_T^{smooth} = \hat{\mathbf{x}}_T$ and $\mathbf{P}_t^{smooth} = \mathbf{P}_t$ and going backward to $t = 1$.

Chapter 4

Sequential Monte Carlo methods

The main goal of this thesis is to review the particle smoother algorithms in detail, therefore, we need to start with some basic informations about particle filtering. A particle filter is an SMC method which gives a general solution to the filtering problem. To apply any smoothing algorithm to any problem, we need first to run a particle filter. Due to that, this chapter starts with giving some information about particle filter and specially auxiliary particle filter which is a powerful particle filtering approach introduced by Pitt and Shephard (1999).

After particle filter, we need to know how particle smoother algorithms work. In the rest of the chapter we try to introduce and give detailed information about these algorithms in the order of their appearance in the history of particle filter/smoother algorithms. Moreover, after introducing each algorithm, we talk about the advantages, disadvantages, complexity issues and drawback of them.

4.1 Particle filter

The goal of particle filtering is to approximate the joint smoothing distribution using a sequence of weighted particle systems. As result, the target distribution is $p(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$. To draw samples from the target distribution by using importance sampling, first we should introduce a proposal density such that it can be factorized according to

$$q_t(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t}) = h_t(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t}) q_{t-1}(\mathbf{x}_{1:t-1} \mid \mathbf{y}_{1:t-1}), \tag{4.1}$$

where $q_{t-1}(\mathbf{x}_{1:t-1} \mid \mathbf{y}_{1:t-1})$ is the proposal density at $t-1$. This factorization allows us to run a sequential method for approximating the target distribu-

tion. If we assume that we have a set of particles $\left\{\mathbf{x}_{1:t-1}^{(i)}\right\}_{i=1}^{N}$ at time $t-1$, where N is number of the particles, then by sampling N new particles from $h_t(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})$ at time t and add them to the joint distribution, we have

$$\mathbf{x}_{1:t}^{(i)} \;:=\; \left\{\mathbf{x}_{1:t-1}^{(i)}, \mathbf{x}_{t}^{(i)}\right\}. \tag{4.2}$$

Here we sample from a proposal density but not the target density. Therefore, the samples must be weighted in order to approximate the target distribution with

$$w_t^{(i)} \;=\; \frac{p_t(\mathbf{x}_{1:t}^{(i)} \mid \mathbf{y}_{1:t}^{(i)})}{q_t(\mathbf{x}_{1:t}^{(i)} \mid \mathbf{y}_{1:t}^{(i)})}, \tag{4.3}$$

where it can be factorized according to

$$\begin{aligned} w_t^{(i)} &= \frac{g(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) f(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{h_t(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t})} \frac{p_t(\mathbf{x}_{1:t}^{(i)} \mid \mathbf{y}_{1:t}^{(i)})}{q_t(\mathbf{x}_{1:t}^{(i)} \mid \mathbf{y}_{1:t}^{(i)})} \\ &= \frac{g(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) f(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{h_t(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t})} w_{t-1}^{(i)}. \end{aligned} \tag{4.4}$$

Now that we have a sequential updating formula for the importance weights as well as the importance sampling, we can perform an algorithm to draw samples and also update the weights recursively in forward direction. For more details on particle filters and their various implementations, see also (Doucet et al., 2000; Arulampalam et al., 2002; Fearnhead et al., 2010; Lindsten, 2011).

4.2 Auxiliary particle filter (Pitts & Shepherd)

In many Bayesian filtering problems, we are interested to find the solution of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ by receiving the observation data $\mathbf{y}_t$. The analytical solution to this problem is given in Equation (3.3). If we cross out the denominator, which is a normalizing factor and include

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \tag{4.5}$$

into Equation (3.3), we have

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = g(\mathbf{y}_t \mid \mathbf{x}_t) \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \tag{4.6}$$

which is a relation between $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$, $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ and $\mathbf{y}_t$. We know that this recursion is intractable in general.

Here we use particle filters (Doucet et al., 2000) and specially auxiliary particle filter (Pitt and Shephard, 1999) to overcome this problem. In this framework if we approximate the distribution $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ with a set of particles $\left\{\mathbf{x}_{t-1}^{(i)}\right\}_{i=1}^{N}$ and their related weights $\left\{w_{t-1}^{(i)}\right\}_{i=1}^{N}$, then we can replace $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ with its approximation $\sum_{i=1}^{N} w_{t-1}^{(i)} \delta(\mathbf{x} - \mathbf{x}_{t-1}^{(i)})$. Putting this approximation it into Equation (4.6) gives

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \simeq c g(\mathbf{y}_t \mid \mathbf{x}_t) \sum_{i=1}^{N} f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)}, \tag{4.7}$$

where c is a normalizing constant.

As mentioned, here we use auxiliary particle filter of (Pitt and Shephard, 1999). In this method we intend to sample $\mathbf{x}_t$ and i jointly. Therefore, Equation (4.7) can be viewed as the marginal distribution of $\mathbf{x}_t$ when $(\mathbf{x}_t, i)$ are generated from Equation (4.8) that is

$$c g(\mathbf{y}_t \mid \mathbf{x}_t) f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)}. \tag{4.8}$$

Finally, we should approximate Equation (4.8) with $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)\beta_t^{(i)}$, where $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)$ is a proposal distribution that we can draw samples from, $\left\{\beta_t^{(i)}\right\}_{i=1}^{N}$ are weights of a set of indices which can be depended on $\mathbf{y}_t$ and are normalized weights that sum to 1. The reason to use a proposal density to draw sample form is that we are not able to sample from Equation (4.8) directly. In hindsight, we should use importance sampling to propagate from a density which is easy to draw sample form. Algorithm 1 from (Fearnhead et al., 2010) gives a general algorithm for approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ from set of particles and weights $\left\{\mathbf{x}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}$.

Algorithm 1 Auxiliary particle filter (Pitt and Shephard, 1999)

1. **Initialize:** Sample $\left\{\mathbf{x}_0^{(i)}\right\}$ from prior $p(\mathbf{x}_0)$ and set $w_0^{(i)} = \frac{1}{N}$ for all i.
2. **For** $t = 1, ..., T$
   (a) **Resample:** Use $\left\{\beta_t^{(i)}\right\}$ as probabilities to sample N indices $j_1, ..., j_N$ from set $\{1, ..., N\}$.
   (b) **Propagate:** Sample the new particles $\mathbf{x}_t^{(i)}$ independently from $q(\cdot \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)$.
   (c) **Reweight:** Assign each particle $\mathbf{x}_t^{(i)}$ the corresponding importance weights

$$w_t^{(i)} \propto \frac{g(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) f(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(j_i)}) w_{t-1}^{(j_i)}}{q(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(j_i)}, \mathbf{y}_t) \beta_t^{(j_i)}}.$$

We know that the efficiency of the particle filter is directly related to the choice of proposal density $q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)$ and resampling probability $\beta_t^{(i)}$. Few important choices are introduced in (Fearnhead et al., 2010), but when it comes to a linear-Gaussian model, one can analytically finds the optimal densities for propagation and resampling probabilities. The results of such an approach is given as an appendix in (Fearnhead et al., 2010) and are as follows for linear-Gaussian model given in Equation (2.1) and Equation (2.2) with $\Sigma_{\mathbf{w}} = Q$ and $\Sigma_{\mathbf{v}} = R$. The derivation for these proposal densities and the ones that come in the next sections are given in Chapter 8.

$$\begin{aligned} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(j)}, \mathbf{y}_t)\beta_t^{(j)} &= f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(j)}) g(\mathbf{y}_t \mid \mathbf{x}_t) w_{t-1}^{(j)} \\ &= \mathcal{N}\left(\mathbf{x}_t \mid \mu_{t|t-1}^{(j)}, \Sigma_{t|t-1}\right) \times \\ &\quad \mathcal{N}\left(\mathbf{y}_t \mid \mathbf{G}\mathbf{F}\mathbf{x}_{t-1}^{(j)}, \mathbf{R} + \mathbf{G}\mathbf{Q}\mathbf{G}^T\right) w_{t-1}^{(j)} \end{aligned} \tag{4.9}$$

where $\Sigma_{t|t-1} = \mathbf{Q}^{-1} + \mathbf{G}^T\mathbf{R}^{-1}\mathbf{G}$ and

$$\mu_{t|t-1}^{(j)} = \Sigma_{t|t-1}\left(\mathbf{Q}^{-1}\mathbf{F}\mathbf{x}_{t-1}^{(j)} + \mathbf{G}^T\mathbf{R}^{-1}\mathbf{y}_t\right).$$

This can be used for any algorithm when the model is linear and Gaussian.

4.3 The filter-smoother (Kitagawa)

The filter-smoother (Kitagawa, 1996) is a simple extension to particle filter. We can easily show that the recursive solution for joint smoothing distribution is

$$p(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t}) \propto g(\mathbf{y}_t \mid \mathbf{x}_t) f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{1:t-1} \mid \mathbf{y}_{1:t-1}). \tag{4.10}$$

Here we can use the particle filter steps to the whole paths and their weights. In this case we can set $\mathbf{x}_{1:t}^{(i)} = \left(\mathbf{x}_{1:t-1}^{(j_i)}, \mathbf{x}_t^{(i)}\right)$ and use the final available weights $\left\{w_t^{(i)}\right\}_{i=1}^{N}$, so in hindsight we can use any filtering algorithm with $O(N)$ computational complexity for this problem. The drawback of this method is given in (Chopin, 2004), which shows $p(\mathbf{x}_t \mid \mathbf{y}_{1:T})$ will degenerate exponentially fast as $T - t$ grows. Recently a modification step is suggested in (Dubarry and Douc, 2011). The drawback of this modification is that in high dimensional problems the complexity grows fast and for good results we need to run the modification step for too many times.

4.4 The forward-backward smoother (Doucet et al.)

The forward-backward smoother proposed by Doucet et al. (2000) is based on the backward recursion

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) = p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \int \frac{f(\mathbf{x}_{t+1} \mid \mathbf{x}_t)}{p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t})} p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t+1}. \tag{4.11}$$

The derivation of Equation (4.11) is as follows,

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &= \int p(\mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t+1} \\ &= \int p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) p(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:T}) d\mathbf{x}_{t+1} \\ &= \int p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) p(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:t}) d\mathbf{x}_{t+1}, \end{aligned} \tag{4.12}$$

which in Equation (4.12) we used the fact that knowledge of $\mathbf{x}_{t+1}$ is enough to use observations up to time t and not more. By using Bayes' rule we have

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &= \int \frac{p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{y}_{1:t}) p(\mathbf{x}_t \mid \mathbf{y}_{1:t})}{p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t})} p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t+1} \\ &= p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \int \frac{p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, y_{1:t})}{p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t})} p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t+1}, \end{aligned} \tag{4.13}$$

and by this the derivation of Equation (4.11) is completed.

If we assume that we have the set $\left\{\mathbf{x}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}$ from filtering for $t = 1, ..., T$, then we can use the algorithm of the forward-backward smoother in (Doucet et al., 2000) to calculate the weights for the particle smoother as in Algorithm 2.

Algorithm 2 The forward-backward smoother (Doucet et al., 2000)

---

1. **Initialize:** Set all the weights at time T as $\tilde{w}_{T|T}^{(i)} = w_T^{(i)}$ for $i = 1, ..., N$.
2. **For** $t = T - 1, ..., 1$

   Update the new weights as follows for $i = 1, ..., N$

$$\tilde{w}_{t|T}^{(i)} = \sum_{j=1}^{N} \tilde{w}_{t+1|T}^{(j)} \frac{w_t^{(i)} f(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t^{(i)})}{\left[\sum_{l=1}^{N} w_t^{(l)} f(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t^{(l)})\right]}.$$

In Algorithm 2 we used the fact that if we have a set of weights and particles $\left\{\mathbf{x}_{t+1}, \tilde{w}_{t+1|T}^{(i)}\right\}_{i=1}^{N}$ at time $t+1$ in backward direction, then the approximation of the integration in Equation (4.11) will be

$$\int \frac{f(\mathbf{x}_{t+1} \mid \mathbf{x}_t)}{p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t})} p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t+1} \simeq \sum_{i=1}^{N} \tilde{w}_{t+1|T}^{(i)} \frac{f(\mathbf{x}_{t+1}^{(i)} \mid \mathbf{x}_t)}{p(\mathbf{x}_{t+1}^{(i)} \mid \mathbf{y}_{1:t})}, \tag{4.14}$$

where $p(\mathbf{x}_{t+1}^{(i)} \mid \mathbf{y}_{1:t})$ can be approximated by

$$\begin{aligned} p(\mathbf{x}_{t+1}^{(i)} \mid \mathbf{y}_{1:t}) &= \int f(\mathbf{x}_{t+1}^{(i)} \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) d\mathbf{x}_t \\ &\simeq \sum_{l=1}^{N} w_t^{(l)} f(\mathbf{x}_{t+1}^{(i)} \mid \mathbf{x}_t^{(l)}), \end{aligned} \tag{4.15}$$

which here we used the fact that we have the set $\left\{\mathbf{x}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}$ from filtering to approximate $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. As result, an approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:T})$ will

be

$$
\begin{aligned}
p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &\simeq \left[\sum_{i=1}^{N} w_t^{(i)} \delta(\mathbf{x} - \mathbf{x}_t^{(i)})\right] \sum_{j=1}^{N} \tilde{w}_{t+1|T}^{(j)} \frac{f(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t)}{\left[\sum_{l=1}^{N} w_t^{(l)} f(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t^{(l)})\right]} \\
&= \sum_{i=1}^{N} w_t^{(i)} \left[\sum_{j=1}^{N} \tilde{w}_{t+1|T}^{(j)} \frac{w_t^{(i)} f(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t^{(i)})}{\left[\sum_{l=1}^{N} w_t^{(l)} f(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t^{(l)})\right]}\right] \delta(\mathbf{x} - \mathbf{x}_t^{(i)}).
\end{aligned}
\tag{4.16}
$$

Therefore, Algorithm 2 follows Equation (4.16).

Equation (4.16) indicates that at each time step we need to do N^3 operations. In more detail, the denominator in the equation can be calculated once in each time step and as result the complexity of this algorithm can be reduced to the order of $O(N^2)$. Algorithm 2 has a quadratic complexity in number of the particles due to using Equation (4.16) for updating the weights. This is a huge drawback in SMC algorithms. The reason is that in many situations we need to run the algorithms with large number of the particles to approximate the target density more accurately. On the other hand there is no design variable to choose or set for particle smoother algorithm. In this algorithm we just simply update the weights for the samples from particle filter in each time step, which makes the algorithm easy to implement.

4.5 Standard forward-filtering backward-simulation (FFBSi) (Godsill et al.)

This method that was proposed by Godsill et al. (2004) is an extension to the method introduced by Doucet et al. (2000) as fixed-interval smoothing. Here we assume that we have access to data from filtering, which is a set of weighted particles $\left\{\left(\mathbf{x}_t^{(i)}, w_t^{(i)}\right)\right\}_{i=1}^{N}$ for $t = 1, ..., T$. Further if we factorize $p(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T})$ as

$$
\begin{aligned}
p(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_{1:T-1}, \mathbf{x}_T \mid \mathbf{y}_{1:T}) \\
&= p(\mathbf{x}_{1:T-1} \mid \mathbf{x}_T, \mathbf{y}_{1:T}) p(\mathbf{x}_T \mid \mathbf{y}_{1:T}) \\
&\vdots \\
&= p(\mathbf{x}_T \mid \mathbf{y}_{1:T}) \prod_{t=1}^{T-1} p(\mathbf{x}_t \mid \mathbf{x}_{t+1:T}, \mathbf{y}_{1:T}),
\end{aligned}
\tag{4.17}
$$

where, by using the properties of HMM and using Equation (4.5) and Equation (4.6), we can write

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{x}_{t+1:T}, \mathbf{y}_{1:T}) &= p(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{x}_{t+2:T}, \mathbf{y}_{1:T}) \\ &= \frac{p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2:T} \mid \mathbf{x}_t, \mathbf{y}_{1:T}) p(\mathbf{x}_t \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2:T} \mid \mathbf{y}_{1:T})} \\ &= \frac{\cancel{p(\mathbf{x}_{t+2:T} \mid \mathbf{x}_{t+1}, \mathbf{x}_t, \mathbf{y}_{1:T})} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{y}_{1:T}) p(\mathbf{x}_t \mid \mathbf{y}_{1:T})}{\cancel{p(\mathbf{x}_{t+2:T} \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:T})} p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})} \\ &= p(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:t}), \end{aligned}$$

which is a relation between $\mathbf{x}_t$, $\mathbf{x}_{t+1}$ and $\mathbf{y}_{1:t}$. Further we can reformulate $p(\mathbf{x}_t \mid \mathbf{x}_{t+1:T}, \mathbf{y}_{1:T})$ as

$$p(\mathbf{x}_t \mid \mathbf{x}_{t+1:T}, \mathbf{y}_{1:T}) = \frac{p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t)}{p(\mathbf{x}_{t+1} | \mathbf{y}_{1:t})}. \tag{4.18}$$

We know that $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ can be approximated by a forward filter so that we have set of weighted particles as said before. Furthermore, by using Equation (4.18), we can approximate $p(\mathbf{x}_t \mid \mathbf{x}_{t+1:T}, \mathbf{y}_{1:T})$ by

$$p(\mathbf{x}_t \mid \mathbf{x}_{t+1:T}, \mathbf{y}_{1:T}) \approx \sum_{i=1}^{N} \frac{w_t^{(i)} f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)})}{\sum_{j=1}^{N} w_t^{(j)} f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(j)})} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}) \tag{4.19}$$

where we used the fact that we can approximate $p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t}) = \int p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) d\mathbf{x}_t$ in Equation (4.18) with $\sum_{j=1}^{N} w_t^{(j)} f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(j)})$. To simplify the algorithm we can further use

$$\tilde{w}_{t|t+1}^{(i)} = \frac{w_t^{(i)} f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)})}{\sum_{j=1}^{N} w_t^{(j)} f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(j)})}, \tag{4.20}$$

so

$$p(\mathbf{x}_t \mid \mathbf{x}_{t+1:T}, \mathbf{y}_{1:T}) \approx \sum_{i=1}^{N} \tilde{w}_{t|t+1}^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}). \tag{4.21}$$

By using Equation (4.17), Equation (4.20) and Equation (4.21) we can perform an algorithm to compute the weights for our smoother. The algorithm is as follows (Algorithm 3).

Algorithm 3 Standard FFBSi (Godsill et al., 2004)

1. **Forward filtering:** Run a particle filter to obtain the set $\left\{\left(\mathbf{x}_t^{(i)}, w_t^{(i)}\right)\right\}$, which approximates $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ for $t = 1, ..., T$.

2. **Initialize:** Sample the indices $i_1, ..., i_m$ from $\{1, ..., N\}$ with probabilities $\{w_T^{(i)}\}_{i=1}^N$ and set $\tilde{\mathbf{x}}_T^{(i)} = \mathbf{x}_T^{(i_m)}$.

3. **Smoothing: For** $t = T-1, ..., 1$

   For $j = 1, ..., N$

   (a) **Reweight:** Update the new weights for each pair of $\{(i,j)\}_{i=1}^N$ as follows
   $$\tilde{w}_{t|T}^{(i,j)} = \frac{w_t^{(i)} f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t^{(i)})}{\sum_{l=1}^N w_t^{(l)} f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t^{(l)})}.$$

   (b) **Resample:** Use $\left\{\tilde{w}_{t|T}^{(i,j)}\right\}_{i=1}^N$ to sample J from set $\{1, ..., N\}$ and set $\tilde{\mathbf{x}}_t^{(j)} = \mathbf{x}_t^{(J)}$.

   (c) **Append:** Append new sample to the backward trajectory, $\tilde{\mathbf{x}}_{t:T}^{(j)} = \left\{\tilde{\mathbf{x}}_t^{(j)}, \tilde{\mathbf{x}}_{t+1:T}^{(j)}\right\}$.

Although updating the weights does not have a quadratic complexity, but this must be done for all the N particles at each time step that results in an algorithm with quadratic complexity. Here after updating the weights for the whole N particles, we resample them once according to the weights which is a drawback because we just use all the weights once and discard them. On the other hand, it is very simple to run this algorithm for any problem, even those which suffer from severe non-linearities. But the point is the quadratic complexity nature of the algorithm that is an obstacle in front of using large number of the particles. Hence it cannot give an accurate approximation of the smoothing distribution.

4.6 The two-filter smoother (Briers et al.)

The two-filter smoother that is introduced in (Briers et al., 2010) is based on using samples from particle filter and also new samples from the backward information filter. In this approach we combine the output of two indepen-

dent filters: the standard (forward) filter and the backward information filter, which is an approximation of $p(\mathbf{y}_{t:T} \mid \mathbf{x}_t)$. But we know that the backward information filter is not a probability measure on the space of the SSM. In cases where backward information filter can be computed in close form, this problem is not critical. But in non-linear and/or non-Gaussian SSM where we are not able to analytically solve the backward information filter integrations, SMC methods cannot be used for approximation. However, two-filter smoother gives a solution for the smoothing by using forward filter and the backward information filter which only requires approximating probability distributions and as result it can be applied to a general SSM by using SMC methods.

The desired smoother can be factorized as

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}, \mathbf{y}_{t:T}) \\ &= \frac{p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) p(\mathbf{y}_{t:T} \mid \mathbf{y}_{1:t-1}, \mathbf{x}_t)}{p(\mathbf{y}_{t:T} \mid \mathbf{y}_{1:t-1})} \\ &\propto p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) p(\mathbf{y}_{t:T} \mid \mathbf{x}_t), \end{aligned} \tag{4.22}$$

It is clear that we can apply any filtering algorithm to reach the prediction part that is

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}. \tag{4.23}$$

But $p(\mathbf{y}_{t:T} \mid \mathbf{x}_t)$ is not a density function in $\mathbf{x}_t$, so we must find a way to represent it as a density function to be able to apply SMC method for approximating it. To do that Briers et al. (2010) introduced an artificial distribution which we will explain it in the following.

Let us consider a sequence of probability densities $\{\gamma_t(\mathbf{x}_t)\}$ where $t = 1, ..., T$ and are defined such that

$$\text{if} \qquad p(\mathbf{y}_{t:T} \mid \mathbf{x}_t) > 0 \qquad \text{then} \qquad \gamma_t(\mathbf{x}_t) > 0. \tag{4.24}$$

In (Briers et al., 2010) these are defined through first defining a prior $\gamma_0(\mathbf{x}_0)$. Then they calculated the artificial prior at future time steps, i.e. $t = 1, ..., T$ recursively as

$$\gamma_t(\mathbf{x}_t) = \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \gamma_{t-1}(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1}. \tag{4.25}$$

It is clear that this formulation requires a close form expression of $\gamma_t(\mathbf{x}_t)$ which restricts the applicability of the method.

Generally by defining $\gamma_t(\mathbf{x}_t)$ as in Equation (4.24) we can change $p(\mathbf{y}_{t:T} \mid \mathbf{x}_t)$ into a term which consists of density functions with respect to $\mathbf{x}_t$ as

$$p(\mathbf{y}_{t:T} \mid \mathbf{x}_t) = \tilde{p}(\mathbf{y}_{t:T}) \frac{\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T})}{\gamma_t(\mathbf{x}_t)}, \tag{4.26}$$

or

$$\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T}) \propto \gamma_t(\mathbf{x}_t) p(\mathbf{y}_{t:T} \mid \mathbf{x}_t). \tag{4.27}$$

Equation (4.27) will then be used to perform the backward information filter with a new representation. The steps to perform the new representation of the backward information filter are as follows

$$\begin{aligned}
p(\mathbf{y}_{t:T} \mid \mathbf{x}_t) &= \int p(\mathbf{y}_{t:T}, \mathbf{x}_{t+1} \mid x_t) d\mathbf{x}_{t+1} \\
&= \int p(\mathbf{y}_t, \mathbf{y}_{t+1:T}, \mathbf{x}_{t+1} \mid \mathbf{x}_t) d\mathbf{x}_{t+1} \\
&= \int p(\mathbf{y}_{t+1:T} \mid \mathbf{y}_t, \mathbf{x}_{t+1}, \mathbf{x}_t) p(\mathbf{y}_t, \mathbf{x}_{t+1} \mid \mathbf{x}_t) d\mathbf{x}_{t+1} \\
&= \int p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_{t+1}) P(\mathbf{x}_{t+1} \mid \mathbf{y}_t, \mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) d\mathbf{x}_{t+1} \\
&\overset{(4.26)}{=} \int \frac{\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})}{\gamma_{t+1}(\mathbf{x}_{t+1})} \tilde{p}(\mathbf{y}_{t+1:T}) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) d\mathbf{x}_{t+1}.
\end{aligned} \tag{4.28}$$

If we replace $p(\mathbf{y}_{t:T} \mid \mathbf{x}_t)$ with $\tilde{p}(\mathbf{y}_{t:T}) \frac{\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T})}{\gamma_t(\mathbf{x}_t)}$ on the left hand side of Equation (4.28), then we have

$$\begin{aligned}
\tilde{p}(\mathbf{y}_{t:T}) \frac{\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T})}{\gamma_t(\mathbf{x}_t)} &= \frac{\tilde{p}(\mathbf{y}_{t+1:T})}{\tilde{p}(\mathbf{y}_{t:T})} \left(\int \frac{\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})}{1} \times \right. \\
&\qquad \left. \frac{f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t)}{\gamma_{t+1}(\mathbf{x}_{t+1})} d\mathbf{x}_{t+1} \right),
\end{aligned} \tag{4.29}$$

which can be represented as a proportion like

$$\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T}) \propto \gamma_t(\mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) \int f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \frac{\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})}{\gamma_{t+1}(\mathbf{x}_{t+1})} d\mathbf{x}_{t+1}. \tag{4.30}$$

Further if we assume that we have a set of weighted particles $\left\{ \left(\tilde{\mathbf{x}}_{t+1}^{(j)}, \tilde{w}_{t+1}^{(j)} \right) \right\}_{j=1}^{N}$, that approximate $\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})$, we can show that

$$\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T}) \propto \gamma_t(\mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) \sum_{i=1}^{N} \frac{f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t)}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \tilde{w}_{t+1}^{(j)}. \tag{4.31}$$

By this representation we can form an algorithm like forward filtering to recursively calculate Equation (4.28) in backward direction. To do this we can, for example, use an auxiliary backward filter similar to (Pitt and Shephard, 1999). For doing this we should find the distributions $\tilde{q}(\cdot \mid \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)})$ and $\tilde{\beta}_t^{(j)}$ which we can sample from such that

$$\tilde{q}(\mathbf{x}_t \mid \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(i)})\tilde{\beta}_t^{(j)} \simeq \gamma_t(\mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) \frac{\tilde{w}_{t+1}^{(j)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})}. \tag{4.32}$$

Now that we have an approximation to the backward information filter with a set of particles and weights we can easily target the smoothing just in a few steps. Back to Equation (4.22) the two-filter smoother is based on

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) \propto \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \frac{\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{1:T})}{\gamma_t(\mathbf{x}_t)},$$

thus filter particles $\left\{ \left(\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)} \right) \right\}_{i=1}^{N}$ approximating $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ and backward information filter particles $\left\{ \left(\tilde{\mathbf{x}}_{t+1}^{(j)}, \tilde{w}_{t+1}^{(j)} \right) \right\}_{j=1}^{N}$ approximating $\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{1:T})$ can be used to calculate the desired density as

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &\simeq \sum_{i=1}^{N} f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)} \times \sum_{j=1}^{N} \frac{\tilde{w}_t^{(j)}}{\gamma_t(\tilde{\mathbf{x}}_t^{(j)})} \delta(\tilde{\mathbf{x}}_t - \tilde{\mathbf{x}}_t^{(j)}) \\ &= \sum_{j=1}^{N} \delta(\tilde{\mathbf{x}}_t - \tilde{\mathbf{x}}_t^{(j)}) \tilde{w}_{t|T}^{(j)}, \end{aligned} \tag{4.33}$$

where

$$\tilde{w}_{t|T} = \frac{\tilde{w}_t^{(j)}}{\gamma_t(\tilde{\mathbf{x}}_t^{(j)})} \sum_{i=1}^{N} f(\tilde{\mathbf{x}}_t^{(j)} \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)}. \tag{4.34}$$

Now we can perform an algorithm as follows to calculate the weights and samples we need for our desired density (Algorithm 4).

Algorithm 4 The two-filter smoother (Briers et al., 2010)

1. **Initialize:** Set all the weights at time T as $\tilde{w}_T^{(i)} = w_T^{(i)}$ for $i = 1, ..., N$.
2. **For** $t = T - 1, ..., 1$
   (a) **Resample:** Use $\left\{\tilde{\beta}_t^{(j)}\right\}$ as probabilities to sample N indices $m_1, ..., m_N$ from set $\{1, ..., N\}$.
   (b) **Propagate:** Sample the new particles $\tilde{\mathbf{x}}_t^{(j)}$ independently from $\tilde{q}(\cdot \mid \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)})$.
   (c) **Reweight:** Assign each particle $\tilde{\mathbf{x}}_t^{(j)}$ the corresponding weights
   $$\tilde{w}_t^{(j)} \propto \frac{\gamma_t(\tilde{\mathbf{x}}_t^{(j)}) g(\mathbf{y}_t \mid \tilde{\mathbf{x}}_t^{(j)}) f(\tilde{\mathbf{x}}_{t+1}^{(m_j)} \mid \mathbf{x}_t) \tilde{w}_{t+1}^{(m_j)}}{\tilde{q}(\mathbf{x}_t^{(j)} \mid \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(m_j)}) \tilde{\beta}_t^{(j)} \gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(m_j)})}.$$
3. **For** $t = 1, ..., T$
   (a) **Reweight:** Assign new weights to the smoother particles $\left\{\tilde{\mathbf{x}}_t^{(j)}\right\}$ as follows
   $$\tilde{w}_{t|T}^{(j)} \propto \frac{\tilde{w}_t^{(j)}}{\gamma_t(\tilde{\mathbf{x}}_t^{(j)})} \sum_{i=1}^{N} f(\tilde{\mathbf{x}}_t^{(j)} \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)}.$$

Although this approach is using the information from the backward information filter to obtain the smoothing distribution, it has the same problems as previous algorithms. Firstly, the complexity of this algorithm is again at the order of $O(N^2)$. This means that it is computationally demanding to run this algorithm with large number of the particles. Moreover, again it didn't use independent samples for the smoothing but samples form the backward information filter. These drawbacks put this algorithm at the same level as the older ones. In addition, the performance of this algorithm is highly dependent on the choice of artificial density and need to be taken care of.

4.7 The two-filter smoother (Fearnhead et al.)

The aim of this new method is to overcome the weaknesses of the one proposed by Briers et al. (2010). As mentioned before in the original two-filter smoother, we do not use any new sample for the smoothing density. But here

we draw new particles from marginal smoothing densities directly, rather that reweighing those drawn from another distribution. To get more insight into this new method we will try to explain and derive the formulas from the begging. Firstly, it is important to see how the facotization should be done for this new algorithm. We can represent the desired smoothing density as

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}, \mathbf{y}_t, \mathbf{y}_{t+1:T}) \\ &\propto p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) g(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t) \\ &\propto p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) g(\mathbf{y}_t \mid \mathbf{x}_t) \int p(\mathbf{y}_{t+1:T}, \mathbf{x}_{t+1} \mid \mathbf{x}_t) d\mathbf{x}_{t+1} \\ &\propto \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} g(\mathbf{y}_t \mid \mathbf{x}_t) \times \\ &\quad \int f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \frac{\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})}{\gamma_{t+1}(\mathbf{x}_{t+1})} d\mathbf{x}_{t+1}, \end{aligned} \tag{4.35}$$

which $\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})$ and $\gamma_{t+1}(\mathbf{x}_{t+1})$ previously defined in Equations (4.26) and (4.27). Clearly, it is combination of the particle filter and the backward information filter. Now if we assume that we have a set of weighted particles $\left\{\left(\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\right)\right\}_{i=1}^{N}$ from the forward filtering and $\left\{\left(\tilde{\mathbf{x}}_{t+1}^{(j)}, \tilde{w}_{t+1}^{(j)}\right)\right\}_{j=1}^{N}$ from the backward information filter, we can show that the smoothing density can be approximated with

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) \simeq c \sum_{i=1}^{N} \sum_{j=1}^{N} f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)} g(\mathbf{y}_t \mid \mathbf{x}_t) \frac{f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t)}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \tilde{w}_{t+1}^{(j)}, \tag{4.36}$$

where c is a normalizing constant.

To sample from this approximation we can use the concept of auxiliary particle filter given in (Pitt and Shephard, 1999) and sample $(\mathbf{x}_t, i, j)$ jointly. As here we have two indices i and j, we need to sample them first. After sampling them, we can use a propagation density to sample $\mathbf{x}_t$ from it according to the indices we have drawn before. Therefore, if we can find a propagation density $\bar{q}$ and a resampling probability density $\bar{\beta}_t^{(i,j)}$ such that they satisfy the approximation

$$\bar{q}(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) \bar{\beta}_t^{(i,j)} \simeq f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) \frac{\tilde{w}_{t+1}^{(j)} w_{t-1}^{(i)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})}, \tag{4.37}$$

then we can form an algorithm to find the proper weights and samples for our desired density function as in Algorithm 5, i.e. first sampling the indices from a joint distribution and then drawing samples from $\bar{q}$ to target

the smoothing density.

This is important to know what are the optimal proposal densities. To clarify what is the optimal choice of propagation density, we can think of that as

$$\bar{q}(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}), \tag{4.38}$$

so the optimal resampling density will be

$$\bar{\beta}_t^{(i,j)} \propto \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) d\mathbf{x}_t \frac{\tilde{w}_{t+1}^{(j)} w_{t-1}^{(i)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})}. \tag{4.39}$$

Algorithm 5 The two-filter smoother (Fearnhead et al., 2010)

1. **Forward filtering:** Run a particle filter to obtain the set $\left\{\left(\mathbf{x}_t^{(i)}, w_t^{(i)}\right)\right\}_{i=1}^N$, which approximates $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ for $t = 1, ..., T$.
2. **Backward filtering:** Run a backward information filter to obtain the set $\left\{\left(\tilde{\mathbf{x}}_t^{(j)}, \tilde{w}_t^{(j)}\right)\right\}_{j=1}^N$, which approximates $\tilde{p}(x_t \mid y_{t:T}) \propto \gamma_t(x_t) p(y_{t:T} \mid x_t)$ for $t = T-1, ..., 1$.
3. **Smoothing:** For $t = 1, ..., T-1$
   (a) **Resample:** Use the joint distribution $\bar{\beta}_t^{(i,j)}$ to sample N pairs $\{(i_m, j_m)\}_{m=1}^N$.
   (b) **Propagate:** Sample the new particles $\bar{\mathbf{x}}_t^{(m)}$ independently from $\bar{q}(\cdot \mid \mathbf{x}_{t-1}^{(i_m)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j_m)})$.
   (c) **Reweight:** For each particle $\bar{\mathbf{x}}_t^{(m)}$ use the weight as follows

$$\bar{w}_t^{(m)} \propto \frac{f(\mathbf{x}_t^{(m)} \mid \mathbf{x}_{t-1}^{(i_m)}) g(\mathbf{y}_t \mid \mathbf{x}_t^{(m)}) f(\tilde{\mathbf{x}}_{t+1}^{(j_m)} \mid \mathbf{x}_t^{(m)}) \tilde{w}_{t+1}^{(j_m)} w_{t-1}^{(i_m)}}{\bar{q}(\tilde{\mathbf{x}}_t^{(m)} \mid \mathbf{x}_{t-1}^{(i_m)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j_m)}) \bar{\beta}_t^{(i_m, j_m)} \gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j_m)})}.$$

This is a very important improvement to draw samples for the approximation of the smoothing density independently as it helps to have a better support of the smoothing. But the point is that the cost of running this algorithm is still at the order of $O(N^2)$ and it comes from the fact that we need to sample the indices jointly. Sampling the indices jointly not only have

a quadratic complexity calculation but also needs a large amount of memory. In hindsight, this algorithm is not practical even for a moderate size of N.

4.8 The new linear complexity smoother (Fearnhead et al.)

Here we explain the method introduced in (Fearnhead et al., 2010), which has a linear complexity in the number of the particles. As we can see the only problem in Algorithm 5 is the joint distribution which we draw indices (i, j) from. If one can find a solution so that i and j can be sampled independently, then the algorithm will not be computationally demanding anymore. To do that we should factorize $\bar{\beta}_t^{(i,j)}$ to $\beta_t^{(i)}$ and $\tilde{\beta}_t^{(j)}$ which we can sample the indices independently from them. This work is done in (Fearnhead et al., 2010), which here we explain it in more detail.

To find what is $\beta_t^{(i)}$ we can start with marginalizing $\bar{\beta}_t^{(i,j)}$ over j. Marginalizing over j gives

$$
\begin{aligned}
\beta_t^{(i)} &= \sum_{j=1}^{N} \bar{\beta}_t^{(i,j)} \\
&\propto \sum_{j=1}^{N} \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) d\mathbf{x}_t \frac{\tilde{w}_{t+1}^{(j)} w_{t-1}^{(i)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \\
&\approx \iint f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) d\mathbf{x}_t \frac{\tilde{w}_{t+1} w_{t-1}^{(i)}}{\gamma_{t+1}(\mathbf{x}_{t+1})} d\mathbf{x}_{t+1} \\
&\approx \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)} \times \\
&\quad \left(\int g(\mathbf{y}_t \mid \mathbf{x}_t) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \frac{\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})}{\gamma_{t+1}(\mathbf{x}_{t+1})} d\mathbf{x}_{t+1} \right) d\mathbf{x}_t,
\end{aligned}
$$

which instead a set of weighted samples $\left\{ \left(\tilde{\mathbf{x}}_{t+1}^{(j)}, \tilde{w}_{t+1}^{(j)} \right) \right\}$, we assumed that we can analytically compute the integration over $\mathbf{x}_{t+1}$. To do that we must change the summation with integration, and weights at time $t+1$ with

$\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})$ as we did above. So,

$$\begin{aligned}
\beta_t^{(i)} &\approx \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)} \left(g(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t)\right) d\mathbf{x}_t \\
&\overset{(4.27)}{\propto} w_{t-1}^{(i)} \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) \left(p(\mathbf{y}_{t:T} \mid \mathbf{x}_t)\right) d\mathbf{x}_t \\
&\propto w_{t-1}^{(i)} \int p(\mathbf{y}_{t:T} \mid \cancel{\mathbf{x}_{t-1}^{(i)}}, \mathbf{x}_t) f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) d\mathbf{x}_t \\
&\propto w_{t-1}^{(i)} \int p(\mathbf{y}_{t:T}, \mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) d\mathbf{x}_t \\
&\propto w_{t-1}^{(i)} p(\mathbf{y}_{t:T} \mid \mathbf{x}_{t-1}^{(i)}). \qquad (4.40)
\end{aligned}$$

It is obvious that calculating Equation (4.40) analytically is impossible. But Fearnhead et al. (2010) suggested that we should use two simple approximations. First is to sample particles at time $t-1$ according to their filtering weights $w_{t-1}^{(i)}$. Another approach that is suggested in (Fearnhead et al., 2010) is to sample from an approximation of $p(\mathbf{y}_{t:T} \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)}$ which is $p(\mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)}$. Clearly, the second approximation is better as it includes the information from $\mathbf{y}_t$ as well.

If we look into our particle filter, we used the auxiliary filter, which gave us the set of particles $\left\{\mathbf{x}_t^{(i)}\right\}_{i=1}^N$ with probability $\left\{\beta_t^{(i)}\right\}_{i=1}^N$. In this new smoothing approach we can use these probabilities as an approximation to $p(\mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)}$, therefore there is no need for further computations as we already have them from forward filtering.

We can use the same approach for $\tilde{\beta}_t^{(j)}$. To find what is $\tilde{\beta}_t^{(j)}$ we can marginalize $\bar{\beta}_t^{(i,j)}$ over i this time. As result we have

$$\begin{aligned}
\tilde{\beta}_t^{(j)} &= \sum_{i=1}^N \bar{\beta}_t^{(i,j)} \\
&\propto \sum_{i=1}^N \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) d\mathbf{x}_t \frac{\tilde{w}_{t+1}^{(j)} w_{t-1}^{(i)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \\
&\approx \frac{\tilde{w}_{t+1}^{(j)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \iint \left(p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) g(\mathbf{y}_t \mid \mathbf{x}_t) d\mathbf{x}_{t-1}\right) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) d\mathbf{x}_t,
\end{aligned}$$

which instead of a set of weighted samples $\left\{\left(\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\right)\right\}$, we assumed that we can use their analytical expression. To do that we changed the summation

into an integration and $f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})w_{t-1}^{(i)}$ with $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$, which gives

$$
\begin{aligned}
\bar{\beta}_t^{(j)} &\approx \frac{\tilde{w}_{t+1}^{(j)}}{\gamma_{t+1}(\mathbf{x}_{t+1}^{(j)})} \int p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) d\mathbf{x}_t \\
&\propto \frac{\tilde{w}_{t+1}^{(j)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \int p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t, \mathbf{y}_{1:t}) d\mathbf{x}_t \\
&\overset{Bayes'}{\propto} \frac{\tilde{w}_{t+1}^{(j)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \int p(\mathbf{x}_t, \tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{y}_{1:t}) d\mathbf{x}_t \\
&\propto \left(\frac{p(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{y}_{1:t})}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \right) \tilde{w}_{t+1}^{(j)} \\
&\overset{(4.27)}{\propto} \tilde{p}(\mathbf{y}_{1:t} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}) \tilde{w}_{t+1}^{(j)}. \qquad (4.41)
\end{aligned}
$$

Similar to Equation (4.40), it is impossible to compute Equation (4.41) analytically but this time to draw indices for sampling from the backward information filter at time $t+1$. Instead, we can use an approximation of that, which is $\tilde{p}(\mathbf{y}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)})\tilde{w}_{t+1}^{(j)}$. By this approximation we can use $\tilde{\beta}_t^{(j)}$ that we have had from the backward information filter to use them as an approximation to $\tilde{p}(\mathbf{y}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)})\tilde{w}_{t+1}^{(j)}$.

Drawing indices independently by using the approximations above allows to perform a simple algorithm with linear complexity in the number of the particles that is described in Algorithm 6. Clearly, the main difference between Algorithm 6 and Algorithm 5 is that instead of sampling from a joint distribution to draw indices i, j jointly, in Algorithm 6 we sample the indices independently. Moreover, because we already have a good approximation for $\beta_t^{(i)}$ and $\bar{\beta}_t^{(j)}$ from the forward filter and the backward information filter respectively, we can use them again without any further modification.

Algorithm 6 The new linear complexity smoother (Fearnhead et al., 2010)

1. **Forward filtering:** Run a particle filter to obtain the set $\left\{\left(\mathbf{x}_t^{(i)}, w_t^{(i)}\right)\right\}_{i=1}^{N}$, which approximates $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ for $t = 1, ..., T$.
2. **Backward filtering:** Run a backwards information filter to obtain the set $\left\{\left(\tilde{\mathbf{x}}_t^{(j)}, \tilde{w}_t^{(j)}\right)\right\}_{j=1}^{N}$, which approximates $p(\mathbf{x}_t \mid \mathbf{y}_{t:T}) \propto \gamma_t(\mathbf{x}_t)p(\mathbf{y}_{t:T} \mid \mathbf{x}_t)$ for $t = T-1, ..., 1$.
3. **Smoothing: For** $t = 1, ..., T-1$
   (a) **Resample:** Use $\left\{\beta_t^{(i)}\right\}$ from forward filter to sample $i_1, ..., i_m$ and $\left\{\tilde{\beta}_t^{(j)}\right\}$ from backwards filter to sample $j_1, ..., j_m$ from the set $\{1, ..., N\}$.
   (b) **Propagate:** Sample the new particles $\bar{\mathbf{x}}_t^{(m)}$ independently from $\bar{q}(\cdot \mid \mathbf{x}_{t-1}^{(i_m)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j_m)})$.
   (c) **Reweight:** For each particle $\bar{\mathbf{x}}_t^{(m)}$ use the weight as follows

$$\bar{w}_t^{(m)} \propto \frac{f(\mathbf{x}_t^{(m)} \mid \mathbf{x}_{t-1}^{(i_m)})g(\mathbf{y}_t \mid \mathbf{x}_t^{(m)})f(\tilde{\mathbf{x}}_{t+1}^{(j_m)} \mid \mathbf{x}_t^{(m)})\tilde{w}_{t+1}^{(j_m)}w_{t-1}^{(i_m)}}{\bar{q}(\tilde{\mathbf{x}}_t^{(m)} \mid \mathbf{x}_{t-1}^{(i_m)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j_m)})\beta_t^{(i_m)}\tilde{\beta}_t^{(j_m)}\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j_m)})}.$$

It is clear that there is just a slight difference between these two methods, which introduced by Fearnhead et al. (2010). Still, the linear complexity of the second method is a huge reduction in complexity that makes it possible to use many more particles in our smoothing algorithm.

One of the drawbacks of this algorithm is number of the design variables that must be designed. For example, the choice of artificial density that can change the performance of this algorithm by a large factor. In addition, it is not always easy to use proposal densities such as $\bar{q}(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i_m)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j_m)})$ which are conditioned on three variables specially when the SSM is severely non-linear and also the perturbations are not Gaussian. But if one can use very good proposal densities and also could find a good choice for artificial density, then it can outperform any other algorithm as it is shown in simulation results.

4.9 Fast FFBSi (Douc et al.)

In the last part of this chapter we are going to introduce another method, which has asymptotically linear complexity in number of the particles. Here we will mostly refer to the licentiate thesis that is defended by Fredrik Lindsten in Linköping University (Lindsten, 2011). The main paper of this new method is (Douc et al., 2010), which covered the method and algorithm in more details as well as asymptotic convergence results. The idea of this algorithm is directly related to FFBSi which is introduced previously. In FFBSi we evaluate all the smoothing weights to be able to sample from the smoothing distribution which is the main reason to have a quadratic complexity. Moreover, after drawing one sample from categorical distribution, we discard all the weights. In (Douc et al., 2010) it is claimed that we do not need to evaluate all those weights at each time step to be able to sample from the smoothing and instead we can use a rejection sampling approach. Because the main idea in this new method is related to rejection sampling, first, we explain briefly what it is.

4.9.1 Rejection sampling

Suppose we wish to sample from a target density $f(x)$ that is difficult or impossible to sample from directly. If we assume that we have a proposal density $q(x)$ that we know how to draw samples from and probably a known distribution like uniform probability density function, then if there is a constant k such that

$$\forall x \quad kg(x) \geq f(x),$$

then accepting samples draws in succession from $kg(x)$ with ratio $\frac{f(x)}{kg(x)}$ close to 1 will yield a sample that follows the target distribution $f(x)$. On the other hand we would reject the samples if the ratio is not close to 1. Figure 4.1 shows the pdf of $f(x)$ (target distribution) and $kg(x)$ (proposal distribution).

The proof of this fact is straight-forward. From Bayes' theorem we have

$$\Pr(X \mid accept) = \frac{\Pr(accept \mid X) \times Pr(X)}{\Pr(accept)}.$$

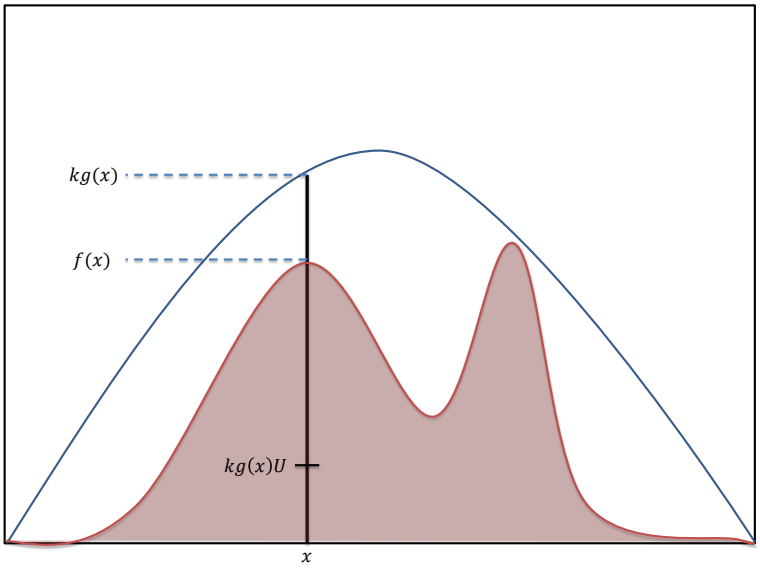


Figure 4.1: An example of a possible distribution of $f(x)$ and $kg(x)$. A sample x is proposed from density $kg(x)$. The sample is accepted if $kg(x)U \leq f(x)$.

Furthermore, we know that $Pr(accept \mid X) = \frac{f(x)}{kg(x)}$ and $Pr(X) = g(x)$. So,

$$\begin{aligned} Pr(accept) &= \int_x Pr(accept \mid X) Pr(X) dx \\ &= \int_x \frac{f(x)}{kg(x)} g(x) dx \\ &= \frac{1}{k}. \end{aligned}$$

In hindsight, we can show that

$$\begin{aligned} Pr(X \mid accept) &= \frac{\frac{f(x)}{kg(x)} g(x)}{\frac{1}{k}} \\ &= f(x), \end{aligned}$$

as required. Rejection sampling algorithm can be performed easily and is shown in Algorithm 7.

Algorithm 7 Rejection sampling

1. **Initialize:** $L \leftarrow \{1, ..., N\}$.
2. **While** L is not empty
   (a) $n \leftarrow \text{card}(L)$.
   (b) $\delta \leftarrow \phi$.
   (c) **Sample** independently $\{X'_t\}_{t=1}^n \sim g$.
   (d) **Sample** independently $\{U_t\}_{t=1}^n \sim \mathcal{U}([0,1])$.
   (e) **For** $t = 1, ..., n$
      i. **If** $U_t \leq \frac{f(x)}{kg(x)}$ **then** $X_{L(t)} \leftarrow X'_t$ and $\delta \leftarrow \delta \cup \{L(t)\}$.
   (f) $L \leftarrow L \setminus \delta$.

4.9.2 Implementation of fast FFBSi

If we look at Algorithm 3 in more detail, it can be understood that we are using $\left\{w_{t|T}^{(i,j)}\right\}_{i=1}^N$ to just sample one index. Douc et al. (2010) suggested that we can reduce this by using a rejection sampling approach instead (see also section 2.4.3 of (Lindsten, 2011)). To do this, we should assume that our transition density function is bounded from above,

$$f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \leq \rho, \tag{4.42}$$

which is true in many situations.

Here we want to sample index $I(j)$, according to the forward filtering and then append it to the j^{th} backward trajectory. To do that we should sample an index from set $\{1, ..., N\}$ with corresponding probabilities $\left\{w_{t|T}^{(i,j)}\right\}_{i=1}^N$, which we don't have them yet. So, we will do that with $\left\{w_t^{(i)}\right\}_{i=1}^N$ instead. That is, we propose samples based on the filter weights rather than any smoothing weights. If we assume that we have the index $I(j)$, we should compute its acceptance probability. This can be done by considering the quotient between the target and the proposal. Using the definition of the smoothing weights from Equation (4.19) , we have

$$\frac{\tilde{w}_{t|T}^{(I(j),j)}}{w_t^{(I(j))}} = \frac{1}{\sum_{l=1}^N w_t^{(l)} f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t^{(l)})} f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t^{(I(j))}), \tag{4.43}$$

which implies that the sample should be accepted with probability $f(\mathbf{x}_{t+1} \mid \mathbf{x}_t)/\rho$. The fast FFBSi is given in Algorithm 8.

Algorithm 8 Fast FFBSi (Douc et al., 2010)

1. **Initialize:** Use $\left\{w_t^{(i)}\right\}_{i=1}^N$ to sample $I(1), ..., I(N)$ and then set $\tilde{\mathbf{x}}_T^{(j)} = \mathbf{x}_T^{(I(j))}$ for $j = 1, ..., N$.
2. **Resample:** **For** $t = T-1, ..., 1$ **do** the following steps
   (a) $L \leftarrow \{1, ..., N\}$.
   (b) **while** L is not empty **do**
      i. $M \leftarrow \text{card}(L)$
      ii. $\delta \leftarrow \phi$.
      iii. **Sample** independently $\{C(m)\}_{m=1}^M \sim \text{Cat}\left(\left\{w_t^{(i)}\right\}_{i=1}^N\right)$.
      iv. **Sample** independently $\{U(m)\}_{m=1}^M \sim \mathcal{U}([0,1])$.
      v. For $m = 1, ..., M$
         A. **If** $U(m) \leq f(\widetilde{\mathbf{x}}_{t+1}^{(L(m))} \mid \mathbf{x}_t^{(C(m))})/\rho$ **then** $I(L(m)) \leftarrow C(m)$ and $\delta \leftarrow \delta \cup \{L(t)\}$.
   (c) $L \leftarrow L \setminus \delta$
3. **Append:** For $j = 1, ..., N$ append the samples to the backward trajectories as follows

$$\begin{aligned} \tilde{\mathbf{x}}_t^{(j)} &= \mathbf{x}_t^{(I(j))} \\ \tilde{\mathbf{x}}_{t:T}^{(j)} &= \left\{\widetilde{\mathbf{x}}_t^{(j)}, \widetilde{\mathbf{x}}_{t+1:T}^{(j)}\right\}. \end{aligned}$$

The main problem of this algorithm is related to the fact that we used the rejection sampling method and as it is said in section 2.4.3 (Lindsten, 2011) that the applicability of the rejection sampling algorithm relies on a sufficiently high acceptance probability. In section 5.2.2 (Lindsten, 2011) it is indicated that most of the time required by Algorithm 8 is spent on just a few particles. In other words, after a few steps in the **while loop** the cardinality of L becomes very low but can linger for a long time close to zero. To overcome this problem, a "time out" check can be added to Algorithm 8.

After for instance R_{max} iteration, if L is not empty, we can jump out from the **while loop** and the remaining weights for indices in L can be taken care of in Algorithm 3 to draw sample from. By doing this the execution time can be reduced by a large factor. A good value for R_{max} could be $N/2$ or $N/3$ but it depends on the problem and the number of the particles which we use. Further modifications are discussed in Chapter 7.

Chapter 5

On optimality of proposal distribution choices

5.1 Optimal proposals

One of the important parts in implementation of particle methods is to have insights about the optimal choices for proposal distributions. There are different ways to define the optimal densities and in each we can use different factorization that can help us to have a better understanding of the methods and also enable us to find new low complexity Monte Carlo solutions for smoothing problems. In this chapter we mostly introduce the optimal choices for a general smoothing algorithm by using a triple joint smoothing density. Here we start by finding the smoothing density $p(\mathbf{x}_t \mid \mathbf{y}_{1:T})$ by using $p(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$. In this way we derive the optimal proposal densities, in a different way as in (Fearnhead et al., 2010), because they use samples from a particle filter and a backward information filter at times $t-1$ and $t+1$ respectively. Then they directly factorized $p(\mathbf{x}_t \mid \mathbf{y}_{1:T})$ to use those particles to approximate the smoothing density. We know that the smoothing density can be expressed as [1]

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &= \iint p(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t-1} d\mathbf{x}_{t+1} \\ &\propto \iint p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_t, \mathbf{x}_{t+1}) p(\mathbf{x}_{t-1} \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:t}) \times \\ &\quad p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t-1} d\mathbf{x}_{t+1}, \end{aligned} \tag{5.1}$$

[1] Note that an alternative factorization is obtained by simply exchanging the roles of x_{t-1} and x_{t+1} in Equation (5.1).

where we assume that we are going backward in time so that we have access to the approximation of $p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$ with a set of particles and their related weights $\left\{\tilde{\mathbf{x}}_{t+1}^{(j)}, \tilde{w}_{t+1}^{(j)}\right\}_{j=1}^{N}$. Then instead of $p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$ we can use its approximation $\sum_{j=1}^{N} \tilde{w}_{t+1}^{(j)} \delta\left(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}^{(j)}\right)$ in the integral. As result, we have

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &\propto \iint p(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t-1} d\mathbf{x}_{t+1} \\ &\propto \sum_{j=1}^{N} \int \tilde{w}_{t+1}^{(j)} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t}) d\mathbf{x}_{t-1}. \end{aligned}$$

As a matter of fact we must approximate $p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t})$ as well. Unfortunately, when $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t+1}$ are significantly correlated, it is typically very difficult to approximate this density accurately. This problem can be illustrated easily with a toy scalar Gaussian model.

5.2 Illustration of the problem

To be able to understand the situation when there is a correlation between the states through the time, i.e. $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t+1}$, we try to use a toy example to be able to investigate it easily. Here we assume a simple two dimensional pdf with variables x_1 and x_2, which we want to draw samples from. Clearly, we also obtain information from the observation. Our goal is to have samples from $p(x_1, x_2)$ that suggests to first sample from a proposal density $f(x_1, x_2)$ and then set the weights to $w \propto \frac{p(x_1,x_2)}{f(x_1,x_2)}$.

To obtain a method that addresses our problem, we should factorize the proposal density according to

$$f(x_1, x_2) = f(x_1 \mid x_2) f(x_2),$$

and first generate x_2 and then x_1. The optimal choice is to first sample x_2 form $f(x_2) = p(x_2) = \int p(x_1, x_2) dx_1$ and then select $f(x_1 \mid x_2) = p(x_1 \mid x_2)$. In many problems it is not easy to have access to $p(x_1 \mid x_2)$, but we can use an approximation instead. A simple example of this problem is a scalar linear-Gaussian model with different mean and covariances. If we assume that $p(x_2) = \mathcal{N}(x_2; 2, 1)$ and $p(x_1 \mid x_2) = \mathcal{N}(x_1; 2.5x_2; 0.3)$ then $p(x_1, x_2)$ has a pdf in (x_1, x_2) plane (see Figure 5.1).

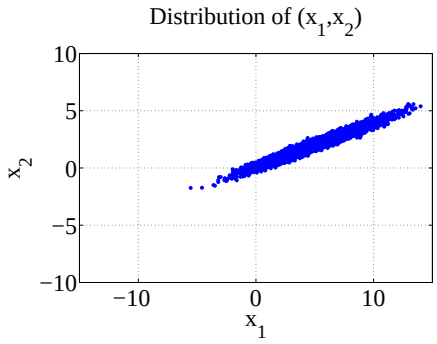


Figure 5.1: [•] Pdf of $p(x_1, x_2)$ over 2-dimensional space (x_1, x_2).

Here we have two different scenarios. First we will design our algorithm by having knowledge about the optimal proposal density $p(x_1 \mid x_2)$ and a fairly good approximation of $p(x_2)$ to draw samples form first. In this case we choose $f(x_2) = \mathcal{N}(x_2; 2, 3)$ and $f(x_1 \mid x_2) = p(x_1 \mid x_2)$. So to have samples from joint distribution $p(x_1, x_2)$, we should first draw samples independently from $f(x_2)$ and then sample from the conditional distribution $f(x_1 \mid x_2)$. In this special case as we used the optimal proposal density, the weights of the samples from $f(x_1 \mid x_2)$ will have the same value $1/M$ where M is number of the samples we have drawn. As the weights are evenly distributed we can omit the weights for the samples drawn from $f(x_1 \mid x_2)$ and plot $f(x_1, x_2)$ which is an approximation of $p(x_1, x_2)$ as in Figure 5.2.

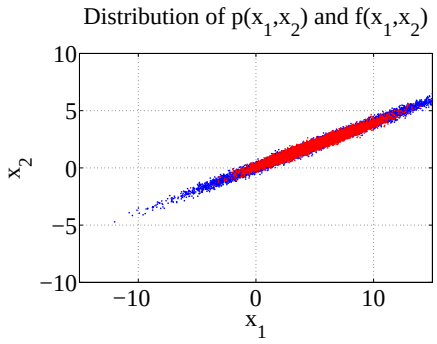


Figure 5.2: [•] Pdf of $p(x_1, x_2)$ and [•] pdf of $f(x_1, x_2)$ over 2-dimensional space (x_1, x_2).

A more interesting scenario is when we cannot draw samples from $p(x_1 \mid x_2)$. In this case we choose our proposal density to be similar to $p(x_1 \mid x_2)$ and easy to draw samples from. To illustrate how this choice will change the efficiency of our approximation we keep $f(x_2)$ the same as our first scenario.

But this time we will go to choose a few different proposal densities as $f(x_1 \mid x_2)$. The choices for $f(x_1 \mid x_2)$ will be

- $f(x_1 \mid x_2) = \mathcal{N}(x_1; 0.1x_2, 0.3)$
- $f(x_1 \mid x_2) = \mathcal{N}(x_1; 0.5x_2, 0.3)$
- $f(x_1 \mid x_2) = \mathcal{N}(x_1; 5x_2, 0.3)$
- $f(x_1 \mid x_2) = \mathcal{N}(x_1; 10x_2, 0.3)$

The results for these four different choices are shown in Figure 5.3. In this scenario, the samples and their according weights should approximate x_1, which is a unimodal distribution with the mean equal to $\bar{x}_1 = 5$.

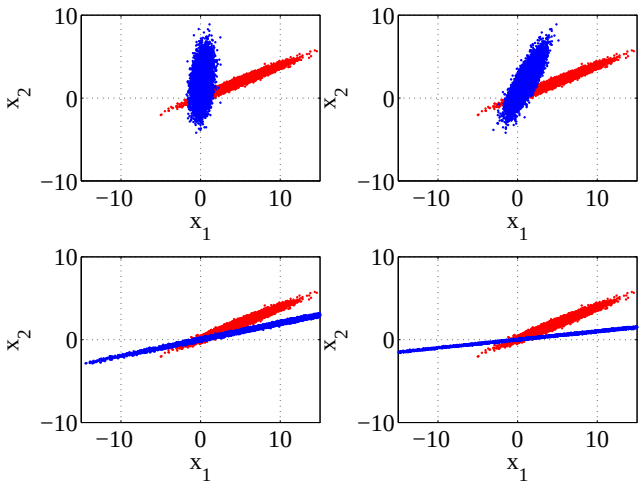


Figure 5.3: [•] Pdf of $p(x_1, x_2)$ and [•] its approximation $f(x_1, x_2)$ in four different scenarios ($\mathcal{N}(x_1; 0.1x_2, 0.3)$ top-left, $\mathcal{N}(x_1; 0.5x_2, 0.3)$ top-right, $\mathcal{N}(x_1; 5x_2, 0.3)$ bottom-left and $\mathcal{N}(x_1; 10x_2, 0.3)$ bottom-right).

Figure 5.3 shows where and for which samples the weights will have high value. Unfortunately, the intersection between $p(x_1 \mid x_2)$ and $f(x_1 \mid x_2)$ does not have a good support of the real mean. This results in a poor approximation of $\bar{x}_1$ in all four scenarios. To overcome this problem we can increase the variance of our proposal density to force it have a larger support. Then the intersection between $p(x_1 \mid x_2)$ and $f(x_1 \mid x_2)$ might be enough and samples with large weights can approximate the mean and the covariance of target density more accurately. The choices for $f(x_1 \mid x_2)$ in this scenario will be

- $f(x_1 \mid x_2) = \mathcal{N}(x_1; 0.1x_2, 5)$
- $f(x_1 \mid x_2) = \mathcal{N}(x_1; 0.5x_2, 5)$
- $f(x_1 \mid x_2) = \mathcal{N}(x_1; 5x_2, 5)$
- $f(x_1 \mid x_2) = \mathcal{N}(x_1; 10x_2, 5)$.

The results for these proposal densities are illustrated in Figure 5.4. Although the approximations are better than the four previous proposal densities but the mean and covariance approximation is good just in the second scenario due to the good support of the samples.

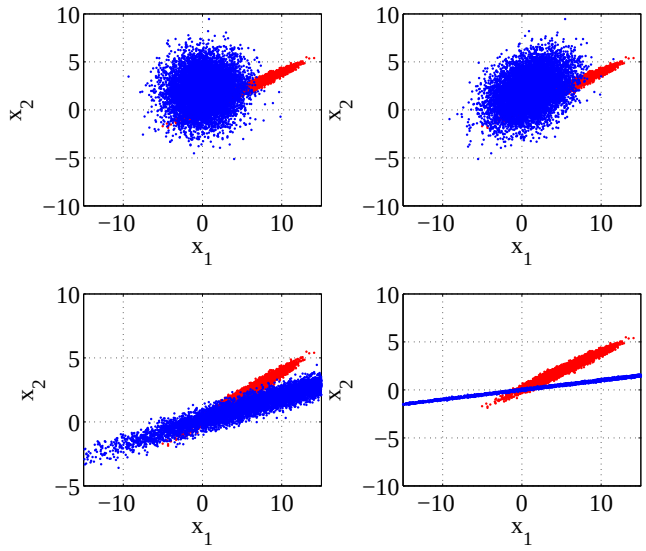


Figure 5.4: [•] Pdf of $p(x_1, x_2)$ and [•] its approximation $f(x_1, x_2)$ in four different scenarios ($\mathcal{N}(x_1; 0.1x_2, 5)$ top-left, $\mathcal{N}(x_1; 0.5x_2, 5)$ top-right, $\mathcal{N}(x_1; 5x_2, 5)$ bottom-left and $\mathcal{N}(x_1; 10x_2, 5)$ bottom-right).

Even in the second scenario, where we chose the variance to be larger (more that ten times), because of poor approximation of the mean in all cases, the ellipsoid which represents the joint distribution is rotated and the intersection between the optimal density and the proposal density has reduced. Clearly, if we do resampling to have evenly distributed weights, then we will have samples just in the region that two distributions have intersection. The important part in this problem is to find a very good proposal density that can have a good support of the desired density, which needs a good approximation of the mean at the first place. If we are not able

to choose a good mean close to the mean of the target density then with high probability we will not have enough samples with large weights. This was just simple scenario to show how important and tricky is to approximate a state given the information of that state in a future time, i.e. $p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t})$. If we are not able to approximate this density with high accuracy, then we cannot achieve good results from the optimal algorithm for smoothing. But if we are able to do that then we can approximate $p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t})$ with a set of particles and their related weights $\left\{\hat{\mathbf{x}}_{t-1}^{(j)}, \hat{w}_{t-1}^{(j)}\right\}_{j=1}^{N}$. Then instead of $p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t})$ we can use its approximation $\sum_{i=1}^{N} \hat{w}_{t-1}^{(i)} \delta\left(\mathbf{x}_{t-1} - \hat{\mathbf{x}}_{t-1}^{(i)}\right)$ in the integration. Thus we have

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &\propto \sum_{j=1}^{N} \int \tilde{w}_{t+1}^{(j)} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t}) d\mathbf{x}_{t-1} \\ &\propto \sum_{j=1}^{N} \sum_{i=1}^{N} \hat{w}_{t-1}^{(i)} \tilde{w}_{t+1}^{(j)} p(\mathbf{x}_t \mid \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) p(\hat{\mathbf{x}}_{t-1}^{(i)} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t}), \end{aligned}$$

which suggest first to sample the indices according to $\tilde{\beta}_{t+1}^{(j)}$ and $\hat{\beta}_{t-1}^{(i)}$ if we approximate them in an auxiliary particle filter manner, and then approximate $p(\mathbf{x}_t \mid \mathbf{y}_{1:T})$ with N particles by using

$$\hat{w}_{t-1}^{(i)} \tilde{w}_{t+1}^{(j)} p(\mathbf{x}_t \mid \hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) p(\hat{\mathbf{x}}_{t-1}^{(i)} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t}).$$

One other way to approximate $p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t})$ is to omit the information form the observation at time t and use $p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t-1})$ instead. In this way we have

$$\begin{aligned} p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t}) &\simeq p(\mathbf{x}_{t-1} \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:t-1}) \\ &\simeq \frac{p(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})}{p(x_{t+1} \mid y_{1:t-1})} \\ &\propto p(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}), \end{aligned}$$

where we are able to factorize it but at the cost of eliminating $\mathbf{y}_t$. Here, the problem is that by omitting the information from observation at time t we are loosing some valuable information and with a high probability we end up in a same problem as our toy example in approximating the mean and variance of the proposal densities.

5.3 Illustration of degeneracy in Fearnhead's linear smoother

One of the drawbacks mentioned in (Fearnhead et al., 2010) about the linear smoother is that this algorithm performs poorly when there is a correlation between the states through time. To get more insight, we can use our toy example and illustrate the problem visually. To do so we start with the fact that in Fearnhead's linear smoother we should sample $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t+1}$ independently which is the main reason for having a linear computational cost $O(N)$ algorithm. More precisely, in the approximations done in (Fearnhead et al., 2010), we should use a forward filter and a backward information filter to draw samples independently at times $t-1$ and $t+1$ respectively. This means if there is a high correlation between the states through times $t-1$, t and $t+1$, we do not include this information into the approximation of the smoothing distribution. To illustrate how poor is independent sampling at times $t-1$ and $t+1$, we can use our toy example with $x_1 = x_{t-1}$ and $x_2 = x_{t+1}$ and draw samples independently from these distributions. We should notice that there is a correlation between x_1 and x_2 but we do not use this information. To do a similar comparison, we use the same settings for the original data and also $f(x_2)$ as our previous setting, i.e.

$$\begin{aligned} p(x_2) &\sim p(x_2) = \mathcal{N}(x_2; 2, 1) \\ p(x_1 \mid x_2) &\sim \mathcal{N}(x_1; 2.5x_2; 0.3) \\ f(x_2) &\sim p(x_2) = \mathcal{N}(x_2; 2, 3). \end{aligned}$$

Here we use two different scenarios. In the first one, we draw samples from $f(x_1)$ independently with different means but with the same variance as $f(x_2)$. The choices for $f(x_1)$ in this scenario are

- $f(x_1) = \mathcal{N}(x_1; 0.2, 3)$
- $f(x_1) = \mathcal{N}(x_1; 2, 3)$
- $f(x_1) = \mathcal{N}(x_1; 10, 3)$
- $f(x_1) = \mathcal{N}(x_1; 20, 3)$.

The results for these proposal densities are shown in Figure 5.6.

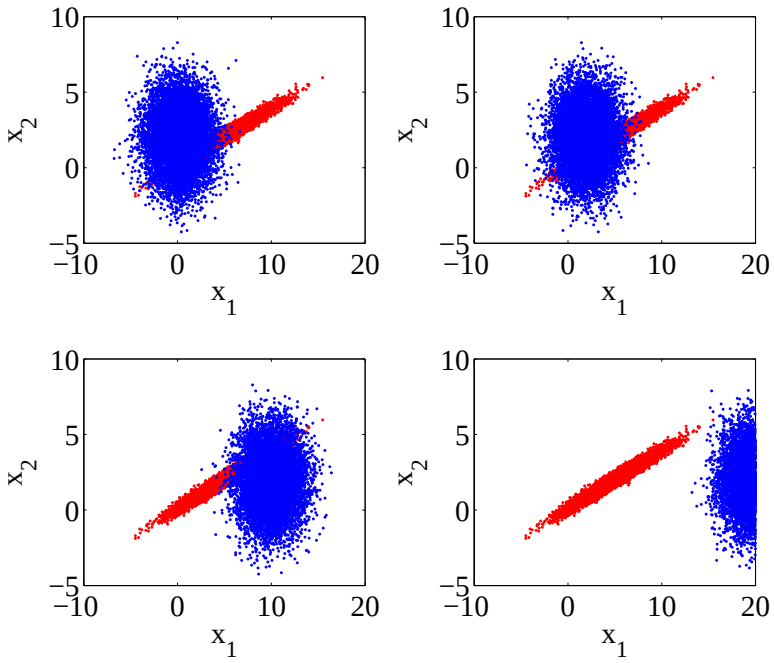


Figure 5.5: [•] Pdf of $p(x_1, x_2)$ and [•] its approximation $f(x_1, x_2)$ in four different scenarios ($\mathcal{N}(x_1; 0.2, 3)$ top-left, $\mathcal{N}(x_1; 2, 3)$ top-right, $\mathcal{N}(x_1; 10, 3)$ bottom-left and $\mathcal{N}(x_1; 20, 3)$ bottom-right).

Figure 5.6 shows that by independent sampling, we are not able to give a good support of the actual distribution $p(x_1, x_2)$. It can be seen that the performance of independent sampling is highly dependent of the approximation of the conditional mean which is poor in most of the cases.

To overcome this problem we can use a larger variance to draw samples form $f(x_1)$. In this case the approximation of the conditional mean has less impact on the support of $p(x_1, x_2)$. The choices for $f(x_1)$ in this scenario are

- $f(x_1) = \mathcal{N}(x_1; 0.2, 10)$
- $f(x_1) = \mathcal{N}(x_1; 2, 10)$
- $f(x_1) = \mathcal{N}(x_1; 10, 10)$
- $f(x_1) = \mathcal{N}(x_1; 20, 10)$.

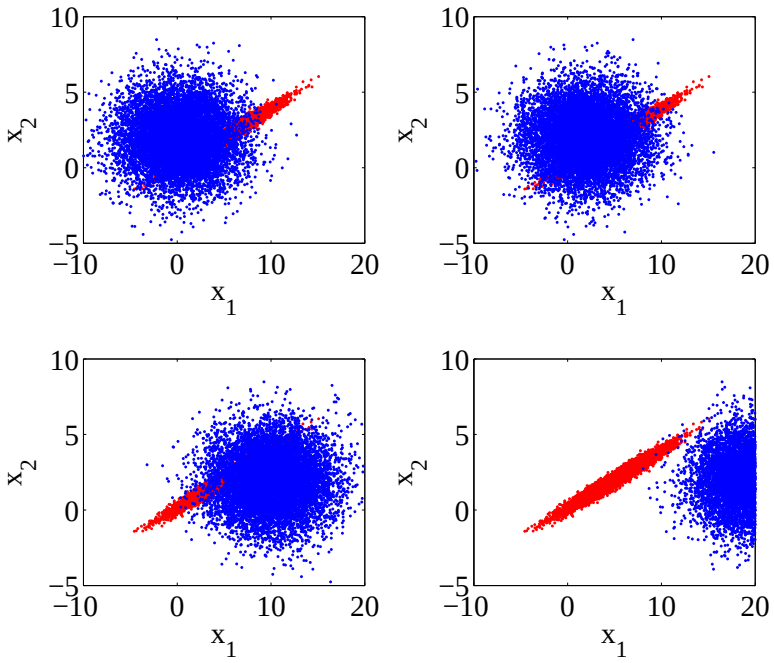


Figure 5.6: [•] Pdf of $p(x_1, x_2)$ and [•] its approximation $f(x_1, x_2)$ in four different scenarios ($\mathcal{N}(x_1; 0.2, 10)$ top-left, $\mathcal{N}(x_1; 2, 10)$ top-right, $\mathcal{N}(x_1; 10, 10)$ bottom-left and $\mathcal{N}(x_1; 20, 10)$ bottom-right).

Figure 5.6 shows the results when we use a large variance to draw sample from $f(x_1)$. This gives us the opportunity to have a better support of the desired distribution which is $p(x_1, x_2)$. Clearly, in situations where the mean of the proposal density is a fairly good estimate of the actual mean of $p(x_1 \mid x_2)$, we have a good support of desired distribution. But if we use a large variance in our proposal densities, we need to draw more samples to have enough samples in all the regions. Moreover, if the mean of $f(x_1)$ is not a good approximation of the conditional mean, the support of $f(x_1, x_2)$ is very poor even if we draw large number of samples.

In the cases that there is not a high correlation between the states through time, combining the information (samples) from the forward filter and the backward information filter in Fearnhead's linear smoother gives a very good approximation of the smoothing distribution. On the other hand, weak approximation of $p(x_1, x_2)$,or equivalently $p(\mathbf{x}_{t-1}, \tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{y}_{1:T})$, gives a very poor approximation of the smoothing distribution. This situation is happening when there is a correlation between the states through time and we use independent samples at times $t-1$ and $t+1$, i.e. using samples from

$p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ and $p(\tilde{\mathbf{x}}_{t+1} \mid \mathbf{y}_{t+1:T})$ respectively, the same as Fearnhead's linear smoother.

Chapter 6

Backward information smoothing

The goal of this chapter is to give a new approach for particle smoothing by using the suggestion given in (Fearnhead et al., 2010). The main concept here is to use the backward information filter and turn it into a smoothing algorithm. This needs a special choice of design variables as is explained in this chapter. The algorithm proposed here is generally approximated since the forward filter is at one point approximated as a Gaussian mixture. In scenarios where this approximation is reasonably accurate, our initial studies indicate that it is more efficient than the other particle smoothing algorithms. Also, like the other algorithms, it is (asymptotically) exact for linear and Gaussian settings.

The backwards information filter, used in Fearnhead's algorithm, has a design parameter in terms of the artificial prior $\gamma_t(\mathbf{x}_t)$. The optimal choice for that density is given by the forward (particle) filter, $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$, for which the backwards information filter coincides with the FFBSi algorithm (Algorithm 3). It is therefore possible to use the optimal artificial prior, but at the price of the complexity $O(N^2)$.

In this chapter we consider using a Gaussian approximation of the optimal artificial prior. In other words, we find $\hat{\mathbf{x}}_{t|t-1}$ and $\hat{\mathbf{P}}_{t|t-1}$ such that

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \simeq \mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_{t|t-1}, \hat{\mathbf{P}}_{t|t-1}), \tag{6.1}$$

and set

$$\gamma_t(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \hat{\mathbf{x}}_{t|t-1}, \hat{\mathbf{P}}_{t|t-1}). \tag{6.2}$$

The output from this filter is hence an approximation of the optimal backward filter. However, it is also an approximation of the FFBSi (Algorithm 3) which means that the output is an approximation of the smoothing density of interest itself.

In more detail, to form this smoothing algorithm we start with formulating the target distribution as

$$\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{1:T}) \propto \gamma_t(\mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) \int f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \frac{\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})}{\gamma_{t+1}(\mathbf{x}_{t+1})} d\mathbf{x}_{t+1}, \quad (6.3)$$

where the choice of artificial density is the same as Equation (6.2). The reason for using Equation (6.3) is that by this special choice of the artificial density we have

$$\begin{aligned} \tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T}) &\propto \gamma_t(\mathbf{x}_t) p(\mathbf{y}_{t:T} \mid \mathbf{x}_t) \\ &\propto p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) p(\mathbf{y}_{t:T} \mid \mathbf{x}_t) \\ &\propto p(\mathbf{x}_t \mid \mathbf{y}_{1:T}), \end{aligned}$$

which indicates that $\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T})$ is an approximation of the smoothing distribution itself and as result we can use the same formulation as in the backward information filter for our smoothing algorithm. Further if we assume that we have a set of weighted particles $\left\{ \left(\tilde{\mathbf{x}}_{t+1}^{(j)}, \tilde{w}_{t+1}^{(j)} \right) \right\}_{j=1}^{N}$ at $t+1$, which approximates $\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})$ or equivalently $\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$, it gives

$$\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{1:T}) \propto \gamma_t(\mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) \sum_{i=1}^{N} \frac{f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t)}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \tilde{w}_{t+1}^{(j)}. \quad (6.4)$$

Here we use the concept of auxiliary particle filter to sample $\mathbf{x}_{t+1}$ and j jointly in backward direction. This means we need to propose densities for propagation and resampling probability densities such that

$$\tilde{q}(\mathbf{x}_t \mid \mathbf{x}_{t+1}^{(j)}, \mathbf{y}_t) \tilde{\beta}_t^{(j)} \simeq \gamma_t(\mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) \frac{f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t)}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \tilde{w}_{t+1}^{(j)}. \quad (6.5)$$

Therefore, in the algorithm we first sample the indices according to $\tilde{\beta}_t^{(j)}$ and then propagate the samples with $\tilde{q}(\mathbf{x}_t \mid \mathbf{x}_{t+1}^{(j)}, \mathbf{y}_t)$. This procedure allows to perform Algorithm (9) to approximate the smoothing distribution.

It should be noticed again that the choice of artificial density make this algorithm suffers from quadratic complexity in number of the particles. In fact a more precisely representation of the artificial density would be

$$\gamma_t(\mathbf{x}_t) = \sum_{k=1}^{K} \mathcal{N}\left(\cdot \mid \widehat{\mathbf{x}}_{t|t-1}^{(k)}, \widehat{\mathbf{P}}_{t|t-1}^{(k)} \right), \quad (6.6)$$

where K is total number of the Gaussian mixtures we used to approximate $\gamma_t(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$. In its simplest setting $K = N$, this approximation

results an algorithm similar to Algorithm 3 with a quadratic complexity in number of the particles. This means for an efficient implementation of this algorithm we need to use less number of Gaussian mixtures. The other issue in the design of this algorithm is the dimension of the SSM. The higher the dimension of the states are, the more complex is to find a suitable Gaussian mixture for it. Although, there are new methods proposed recently to approximate the Gaussian mixture with linear complexity in number of the dimensions. Aside form all these complexities in performing this algorithm, there are many problems that the artificial density can be approximated with just a few number of the Gaussian mixtures. As it is said before, when we are able to approximate $\gamma_t(\mathbf{x}_t)$ accurately, this algorithm is more efficient that other particle smoothing methods but at the cost of $O(KN)$. In hindsight, the applicability of the backward information smoothing (see Algorithm (9)) is directly related to the approximation of $\gamma_t(\mathbf{x}_t)$ and the number of the Gaussian mixtures we use for approximation.

Algorithm 9 Backward information smoothing

1. **Forward filtering:** Run a particle filter to obtain the set $\left\{\left(\mathbf{x}_t^{(i)}, w_t^{(i)}\right)\right\}_{i=1}^{N}$, which approximates $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ for $t = 1, ..., T$.
2. **Gaussian mixture approximation:** Find the proper Gaussian mixture to approximate $\gamma_t(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ and set $\gamma_t(x_t) = \sum_{k=1}^{K} \mathcal{N}\left(\cdot \mid \widehat{\mathbf{x}}^{(k)}, \widehat{\mathbf{P}}^{(k)}\right)$ for $t = 1, ..., T$.
3. **Initializing:** Initialize Backward Filter at time T.
4. **Backward filtering/smoothing: For** $t = T-1, ..., 1$
   (a) **Resample:** Use the distribution $\tilde{\beta}_t$ to sample N indices $\{(j_m)\}_{m=1}^{N}$.
   (b) **Propagate:** Sample the new particles $\tilde{\mathbf{x}}_t^{(m)}$ independently from $\tilde{q}(\cdot \mid \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j_m)})$.
   (c) **Reweight:** For each particle $\tilde{\mathbf{x}}_t^{(m)}$ use the weight as follows by using the new choice of artificial density

$$\tilde{w}_t^{(m)} \propto \frac{\gamma_t(\tilde{\mathbf{x}}_t^{(m)}) g(\mathbf{y}_t \mid \tilde{\mathbf{x}}_t^{(m)}) f(\tilde{\mathbf{x}}_{t+1}^{(j_m)} \mid \tilde{\mathbf{x}}_t^{(m)}) \tilde{w}_{t+1}^{(j_m)}}{\bar{q}(\tilde{\mathbf{x}}_t^{(m)} \mid \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j_m)}) \tilde{\beta}_t^{(j_m)} \gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j_m)})}.$$

The backward information smoothing has some interesting properties. One of them is that it has one step less than Fearnhead's linear smoother and therefore design variables. This is important when the problems are non-linear and/or non-Gaussian and we cannot use optimal proposal densities. The second one is that if we can approximate the artificial density with a few Gaussian mixtures then it can approximately be a linear algorithm in number of the particles, therefore, can be run with large N. Third, it is optimal in the sense that we are using samples from the approximation of the smoothing at time $t+1$ for the approximation of the states at time t. More detail of optimality of this algorithm is given in the next section. And finally, it does not have the problem of Douc's algorithm (remaining for a long time to sample from a small set).

6.1 Optimality of the backward information smoother

In this section we show the reason of optimality of this new algorithm. To show it, we should start with formulating the smoother as

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) = \int p(\mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t+1}, \tag{6.7}$$

where both $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ are from the smoothing distribution. Further if we factorize the integrand in Equation (6.7), it gives

$$p(\mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) = p(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:T}) p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}), \tag{6.8}$$

so the optimal choices for propagation and resampling probability densities are

$$\tilde{q}(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:T}) = p(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:T}), \tag{6.9}$$

and

$$\tilde{\beta}_t^{(j)} = p(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{y}_{1:T}), \tag{6.10}$$

which results in

$$\tilde{q}(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:T}) \tilde{\beta}_t^{(j)} \propto p(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:T}) p(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{y}_{1:T}). \tag{6.11}$$

The final step to show that the backward information smoothing is the optimal smoother that one can derive form the backward information filter,

we should show that the propagation and resampling probability densities in Algorithm (9) are exactly the same as Equations (6.9) and (6.10). Clearly, the choice of propagation density is exactly the same as in Equation (6.9). But for resampling probability density, we start from the fact that in the backward information filter with $\gamma_t(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$, we have

$$\tilde{q}(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_{1:T})\tilde{\beta}_t^{(j)} \propto \frac{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:T})\gamma_t(\mathbf{x}_t)p(\mathbf{y}_t \mid \mathbf{x}_t)p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t)}{\gamma_{t+1}(\mathbf{x}_{t+1}^{(j)})}. \tag{6.12}$$

To find the optimal resampling probability density, we can marginalize Equation (6.12) over $\mathbf{x}_t$, which gives

$$\begin{aligned}
\tilde{\beta}_t^{(j)} &\propto \int \frac{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:T})\gamma_t(\mathbf{x}_t)p(\mathbf{y}_t \mid \mathbf{x}_t)p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t)}{\gamma_{t+1}(\mathbf{x}_{t+1}^{(j)})} d\mathbf{x}_t \\
&\propto \frac{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:t})} \int p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})p(\mathbf{y}_t \mid \mathbf{x}_t)p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t) d\mathbf{x}_t \\
&\propto \frac{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:t})} \int p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{x}_t, \mathbf{y}_{1:t})p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) d\mathbf{x}_t \\
&\propto \frac{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:t})} \int p(\mathbf{x}_t, \mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:t}) d\mathbf{x}_t \\
&\propto \frac{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:t})} \times p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:t}) \\
&\propto p(\mathbf{x}_{t+1}^{(j)} \mid \mathbf{y}_{1:T}),
\end{aligned} \tag{6.13}$$

and clearly it is the same as Equation (6.10).

Chapter 7

Design variables

In this chapter we go through design variables of Fearnhead's linear smoother, the backward information smoothing and fast FFBSi. The main goal of this chapter is to give some useful information about the parameters we can control in particle smoothing algorithms mentioned above. This investigation will give us the opportunity to understand if there are modifications available for a better approximation of the smoothing. In addition, it gives us the opportunity to understand how sensitive are these algorithms to change their design variables and if their complexities remain linear in all cases and problems.

7.1 Design variables of Fearnhead's linear smoother

As mentioned in previous chapters, this algorithm uses samples from one forward filtering and one backward information filter to approximate the smoothing density at time t. This approximation is done with a set of particles and their related weights $\left\{\bar{\mathbf{x}}_t^{(k)}, \bar{w}_t^{(k)}\right\}_{k=1}^{N}$. In this section we are going to elucidate the design choices, trade-offs and the amount of their impact in the performance of this algorithm in detail. From the results given in (Fearnhead et al., 2010) and also our results for the same example, it is clear that this algorithm performs poorly in the first and last time steps. To clarify how the sampling can be done in a different way, we start with explaining the factorization in (Fearnhead et al., 2010) once more. In (Fearnhead et al.,

2010), they factorized the target density $p(\mathbf{x}_t \mid \mathbf{y}_{1:T})$ as

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}, \mathbf{y}_t, \mathbf{y}_{t+1:T}) \\ &\propto p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) g(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{y}_{t+1:T} \mid \mathbf{x}_t) \\ &\propto \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} g(\mathbf{y}_t \mid \mathbf{x}_t) \times \\ &\quad \int f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \frac{\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})}{\gamma_{t+1}(\mathbf{x}_{t+1})} d\mathbf{x}_{t+1}, \end{aligned}$$

where they used particle approximation $\sum_{i=1}^{N} w_{t-1}^{(i)} \delta\left(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^{(i)}\right)$ for approximating the filtering distribution $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ at time $t-1$ and particle approximation $\sum_{j=1}^{N} \tilde{w}_{t-1}^{(j)} \delta\left(\mathbf{x}_{t+1} - \mathbf{x}_{t+1}^{(j)}\right)$ for approximating the backward information filter $p(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})$ at time $t+1$.

A more natural choice is to use the triple joint smoothing distribution at times $t-1$, t and $t+1$ i.e., $p(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$. By studying the factorization

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) = \int p(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) d\mathbf{x}_{t-1} d\mathbf{x}_{t+1},$$

we conclude that $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t+1}$ should ideally be generated from the joint smoothing density $p(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$. This suggestion comes from the fact that we can further factorize $p(\mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$ as

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_t, \mathbf{x}_{t+1}) p(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}).$$

A natural approximation of $p(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$ could be

$$p(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) \simeq p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:T}) p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}),$$

i.e., from the independent smoothing densities. This approximation allows to perform an algorithm with linear computational cost in terms of number of the particles. But this approximation has two drawbacks. Firstly, as we are going backward in time to approximate the smoothing distribution, we do not have access to a particle approximation of $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:T})$. In addition, the exact factorization of $p(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$ is

$$p(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) \simeq p(\mathbf{x}_{t-1} \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:T}) p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}),$$

which indicates that we should sample $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t+1}$ jointly. This yields to an algorithm with quadratic complexity in number of the particles because

we cannot factorize $p(\mathbf{x}_{t-1} \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:T})$. This may be one of the reasons for degeneracy of Fearnhead's linear smoother and also other particle smoother algorithms when there is a high correlation between the states through time. Non of the algorithms includes this distribution because of the complexity it adds to the algorithms.

One possible way to approximate $p(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$ could be

$$p(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{y}_{1:T}) \quad \simeq \quad p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T}),$$

which seems a possible approximation in general and it is used by Fearnhead et al. (2010). But here we face with another problem. The problem comes form the fact that in different settings and problems, $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ gives a poor approximation of the filtering distribution in times near $t = 1$. In addition, approximation of $\tilde{p}(\mathbf{x}_{t+1} \mid \mathbf{y}_{t+1:T})$ is not accurate at times near $t = T$. These poor approximations cause the smoothing distribution degenerates or be likely to degenerate at times near to 1 and T. The reason is that the particles from a forward filter and a backward information filter do not always have a good support of the smoothing distribution at these time steps.

One way to improve part of the weaknesses of Feanhead's linear smoother is to choose the optimal artificial density which enables the backward information filter targets the smoothing distribution directly. In Subsection 7.1.2, we explain how this special choice of artificial density changes the settings in the backward information filter in more detail.

7.1.1 Divergence due to time correlation

The second issue with this algorithm and almost all other smoothing algorithms is the poor performance of them when there is a high correlation between the states through the time. We have already investigated this problem in Chapter 5. If we want to overcome this weakness we need to use the joint smoothing distribution at times $t-1$, t and $t+1$ or at least the relation between the states at these three time steps, even if they are not from the same density. This problem appears even for the seemingly accurate proposal i.e., $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:T})p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:T})$, because again it does not using any condition between the states through time.

7.1.2 Design variables of backward information filter

Backward information filter as a smoother

In Chapter 6 we talked about a new choice for the artificial density and we chose $\gamma_t(\mathbf{x}_t)$ to be $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$. As mentioned in Chapter 6, this is the

optimal choice for the artificial density in backward information filter and also it changes this filter to a smoother. Moreover, we showed in the same chapter that this new smoothing algorithm is optimal in the sense that we are using a pair joint smoothing distribution so that we are using all the information available in the backward direction. But we did not discuss about how this choice will affect the algorithm in detail. We know that in a linear-Gaussian model we can choose $\gamma_t(\mathbf{x}_t) \propto \mathcal{N}\left(\cdot \mid \widehat{\mathbf{x}}, \widehat{\mathbf{P}}\right)$, where $\widehat{\mathbf{x}}$ and $\widehat{\mathbf{P}}$ are the mean and covariance matrix of a Gaussian distribution approximating $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$. But this is not possible in a general state-space model with non-linearity in the states. In a general case we can go for two different approaches. First if we use an approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ i.e. $\sum_{i=1}^{N} w_{t-1}^{(i)} f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$, which here we used the fact that a particle filter preformed us with a set of particles and weights $\left\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\right\}_{i=1}^{N}$ to approximate $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$, then we have

$$\begin{aligned}\tilde{p}(\mathbf{x}_t \mid \mathbf{y}_{t:T}) &\propto p(\mathbf{x}_t \mid \mathbf{y}_{1:T}) \\ &\propto \sum_{i=1}^{N} w_{t-1}^{(i)} f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) g(\mathbf{y}_t \mid \mathbf{x}_t) \sum_{i=1}^{N} \frac{f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t)}{\sum_{k=1}^{N} w_{t-1}^{(k)} f(\tilde{\mathbf{x}}_{t+1}^{(k)} \mid \mathbf{x}_t^{(k)})} \tilde{w}_{t+1}^{(j)},\end{aligned}$$

which suffer from a quadratic complexity in number of the particles and is similar to the FFBSi (see Algorithm 3).

Second approach that has more interesting properties is to use a Gaussian mixture approximation to $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$. One possible question here is that why we do not go for other smoother algorithms when we can approximate the densities with Gaussian mixture. The answer relies on the fact that this algorithm has a very good performance if we are able to approximate $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ accurately with a Gaussian mixture. If we are able to do it with less number of mixtures with linear complexity in high dimension problems which is a hot research topic in recent years then we have a really good smoother algorithm which can give us a good approximation with a complexity order $O(KTN)$, where K is the number of mixtures, T is number of time steps and N is number of the particles.

Relation between the backward information filter and Fearnhead's linear smoother

One interesting question is what does happen if we choose $\gamma_t(\mathbf{x}_t) = p(\mathbf{x}_t)$? This is a huge reduction in the amount of information we use for the choice of artificial density and it is far away from being the optimal one. Without any doubt, the backward information smoother will not perform good with this

choice. The reason is that the only choice to make the backward information filter targets the smoothing distribution is to choose the artificial density to be $\gamma_t(\mathbf{x}_t) \propto p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$. But an important question here is how these choices of artificial density will change the performance of Fearnhead's linear smoother. From the studies that are done in the simulations, we found that in linear-Gaussian models when we are able to use the optimal proposal densities, if we choose the artificial density to be the actual prior $p(\mathbf{x}_t)$, which is a suboptimal choice, the results are acceptable and Fearnhead's linear smoother performs very well.

A more interesting result achieved when we used $\gamma_t(\mathbf{x}_t) \propto p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ for the backward information filter and also ran the third step of Fearnhead's linear smoother in a linear-Gaussian example. Although this special choice of the artificial density gives the smoothing distribution directly in the backward information filter (see Algorithm (9)), it is interesting to see the results of Fearnhead's linear smoother with different choices of the artificial density. Surprisingly, the results indicates that there is nearly no difference between the performance of the algorithm with different choices of the artificial density. The effects of changing the choice of the artificial density is still an open area to work on. As it is stated in (Fearnhead et al., 2010), the choice of artificial density has a large impact in the performance of particle smoother algorithms based on using this density and it is interesting to understand how this choice is affecting the general performance of particle smoother algorithms in SSM with non-linearities in the model and non-Gaussian perturbations.

7.2 Design variables of fast FFBSi

In this section we investigate available design variables of fast FFBSi algorithm. We already talk about the algorithm and rejection sampling in 4 and here we want to show how we can overcome the problem time demanding of this algorithm. As mentioned, one possible choice is to stop the while loop when ever number of the iterations reach a number like R_{max} which can be equal to $N/2$ or $N/3$. But to trust these choices we need to see how exactly rejection sampling step of the algorithm performs. To do this one can go for calculating the acceptance probability for a fixed $\mathbf{x}_t$. As it is infeasible to calculate this probability in general case we go for simplest case where the model is scalar and linear-Gaussian. If we assume $x_t = ax_{t-1} + w_{t-1}$ and $w_{t-1} \sim \mathcal{N}(0, \sigma_w^2)$ and also assuming that we have access to a perfect approximation to the states at time $t-1$ with $x_{t-1|t-1}$ and $\sigma^2_{t-1|t-1}$, then using the

fact that the acceptance probability is given by

$$U(0,1) \leq f(x_{t+1} \mid x_t)/\rho,$$

we can show that this is equal to

$$\begin{aligned}
\Pr(accept) &= \int \mathcal{N}(x_{t-1}; x_{t-1|t-1}, \sigma^2_{t-1|t-1})\mathcal{N}(x_t; ax_{t-1}, \sigma^2_w)dx_{t-1} \\
&= \frac{1}{\sqrt{2\pi\left(\sigma^2_w + a^2\sigma^2_{t-1|t-1}\right)}} \exp\left(-\frac{\left(x_t - ax_{t-1|t-1}\right)^2}{2\left(\sigma^2_w + a^2\sigma^2_{t-1|t-1}\right)}\right),
\end{aligned}$$

which is a Gaussian distribution over x_t with new mean and covariance. This Gaussian density shows that the samples around the mean that is $ax_{t-1|t-1}$ will be accepted with high probability and those that are away from the mean will have a less chance to be accepted. But the problem arises when there is a sample in the set $\left\{x_t^{(i)}\right\}_{i=1}^N$ where it is far away from the mean. In this case it takes a long time for the algorithm to accept this sample. For better understanding, we can use the inverse of $Pr(accept)$, which has a linear relationship with the expected number of iterations in fast FFBSi algorithm. To illustrate it easier we assume that $x_{t-1|t-1} = 0$, $\sigma^2_{t-1|t-1} = 4$, $a = 2$ and $\sigma^2_w = 3$. Figure 7.1 shows the relation between the probability and number of iterations we need to have a successful sample. This figure shows that R_{max} is highly dependent on a problem that we are working on and can be chosen wisely. Of course one must be able to find the probability of acceptance analytically to be able to choose a good R_{max} but still it is an important design variable to choose in this algorithm.

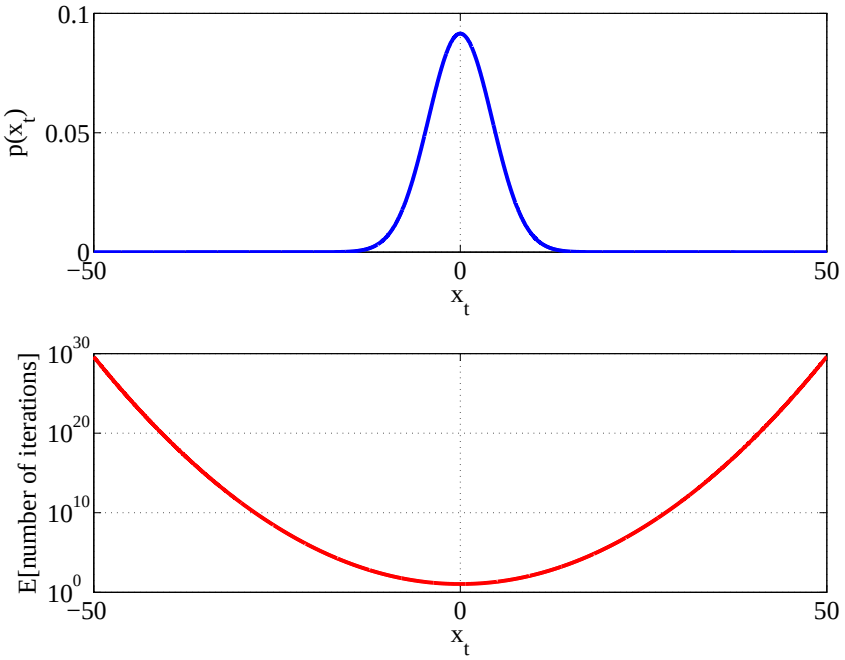


Figure 7.1: [—] Pdf and [—] expected number of iterations for a simple linear-Gaussian example.

Chapter 8

On derivation of optimal proposal densities in linear-Gaussian model

When we are dealing with a linear-Gaussian model, we already has the optimal solution for the filtering and smoothing as they are given in (Anderson and Moore, 1979; Kalman et al., 1960). Although any other method is not comparable with the results form the optimal solutions, still it is a good attempt to derive the optimal proposal densities for this special case and for all the algorithms as these derivations will give us an insight about how we can really find the proposal densities. In addition, there are problems, which are linear-Gaussian in part of the state or observation and in fact these methods can be applied for them as well.

Here we start with giving the optimal proposal densities for auxiliary particle filter and then for all other smoothing algorithms. So, the sections will come in the same order as in Chapter 4.

8.1 Auxiliary particle filter

In auxiliary particle filter we know that the optimal choice must be

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)\beta_t^{(i)} = g(\mathbf{y}_t \mid \mathbf{x}_t) f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) w_{t-1}^{(i)}.$$

By using general refactorization lemma (GRL) for a linear-Gaussian model we have

$$\begin{aligned} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t)\beta_t^{(i)} &= \mathcal{N}(\mathbf{x}_t; \mathbf{F}\mathbf{x}_{t-1}^{(i)}, \mathbf{Q})\mathcal{N}(\mathbf{y}_t; \mathbf{G}\mathbf{x}_t, \mathbf{R}) w_{t-1}^{(i)} \\ &= \mathcal{N}(\mathbf{x}_t; \mu_{t|t-1}, \Sigma_{t|t-1})\mathcal{N}(\mathbf{y}_t; \mathbf{G}\mathbf{F}\mathbf{x}_{t-1}^{(i)}, \mathbf{R} + \mathbf{G}\mathbf{Q}\mathbf{G}^{\mathrm{T}}), \end{aligned}$$

where $\Sigma_{t|t-1}^{-1} = \mathbf{Q}^{-1} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}$ and $\mu_{t|t-1} = \Sigma_{t|t-1}\left(\mathbf{Q}^{-1}\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_t\right)$.

8.2 Backward information filter

For the backward information filter we can use the actual prior $\gamma_t(\mathbf{x}_t) = p(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t \mid \mu_t, \Sigma_t)$, where we should start with μ_0 and Σ_0 as initial mean and covariance matrix and then use the state transition $\mu_{t+1} = \mathbf{F}\mu_t$ for the mean and $\Sigma_{t+1} = \mathbf{F}\mathbf{Q}\mathbf{F}^{\mathrm{T}} + \mathbf{Q}$ for the covariance matrix update. Here we assume that the transition and covariance matrices are not varying over the time. In addition to the prior, we need to have access to $p(\mathbf{x}_t \mid \mathbf{x}_{t+1})$. One way to find the exact distribution of this density is to use the concepts similar to reverse Kalman filter (Taplin, 1998). If the state space model equations defined as

$$\begin{cases} \mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{w}_{t-1} \\ \mathbf{y}_t = \mathbf{G}\mathbf{x}_t + \mathbf{v}_t \end{cases},$$

then the reverse formulation for that will be

$$\begin{cases} \mathbf{x}_t = \mathbf{H}_t\mathbf{x}_{t+1} + \mathbf{D}_t + \eta_t \\ \mathbf{H}_t = \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1},\ \mathbf{D}_t = \mu_t - \mathbf{H}_t\mu_{t+1} \end{cases},$$

where $\eta_t \sim \mathcal{N}(0, \mathbf{u}_t)$ and $\mathbf{u}_t = \Sigma_t - \mathbf{H}_t\Sigma_{t+1}\mathbf{H}_t^{\mathrm{T}}$. If we represent $p(\mathbf{x}_t \mid \mathbf{x}_{t+1})$ with $\mathcal{N}(\mathbf{x}_t \mid m_t, M_t)$, then for m_t

$$\begin{aligned} m_t &= \mathbf{H}_t\mathbf{x}_{t+1}\mathbf{D}_t \\ &= \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1}\mathbf{x}_{t+1} + \mu_t - \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1}\mu_{t+1}, \end{aligned}$$

now if we use $\mu_{t+1} = \mathbf{F}\mu_t$, form $\Sigma_{t+1} = \mathbf{F}\Sigma_t\mathbf{F}^{\mathrm{T}} + \mathbf{Q}$ we use $\mathbf{F} = (\Sigma_{t+1} - \mathbf{Q})\,\mathbf{F}^{-\mathrm{T}}\Sigma_t^{-1}$ and $\tilde{\mathbf{F}} = \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1}$ in the formula, then we have

$$\begin{aligned} m_t &= \tilde{\mathbf{F}}\mathbf{x}_{t+1} + \overbrace{\Sigma_t\mathbf{F}^T\Sigma_{t+1}^{-1}\mathbf{Q}\mathbf{F}^{-\mathrm{T}}}^{\tilde{\mathbf{Q}}}\Sigma_t^{-1}\mu_t \\ &= \tilde{\mathbf{F}}\mathbf{x}_{t+1} + \tilde{\mathbf{Q}}\Sigma_t^{-1}\mu_t. \end{aligned}$$

For covariance matrix we have

$$\begin{aligned} M_t &= \mathbf{u}_t \\ &= \Sigma_t - \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1}\Sigma_{t+1}\Sigma_{t+1}^{-\mathrm{T}}\mathbf{F}\Sigma_t^{\mathrm{T}} \\ &\overset{\text{(section 4, Taplin 1998)}}{=} \Sigma_t - \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1}\mathbf{F}\Sigma_t \\ &= \Sigma_t - \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1}\left(\Sigma_{t+1} - \mathbf{Q}\right)\mathbf{F}^{-\mathrm{T}}\Sigma_t^{-1}\Sigma_t \\ &= \Sigma_t - \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1}\Sigma_{t+1}\mathbf{F}^{-\mathrm{T}}\Sigma_t^{-1}\Sigma_t^{\mathrm{T}} + \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-\mathrm{T}}\Sigma_{t+1}\mathbf{Q}\mathbf{F}^{-\mathrm{T}}\Sigma_t^{-1}\Sigma_t \\ &= \Sigma_t - \Sigma_t + \Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-\mathrm{T}}\Sigma_{t+1}\mathbf{Q}\mathbf{F}^{-\mathrm{T}} \\ &= \tilde{\mathbf{Q}}, \end{aligned}$$

so

$$p(\mathbf{x}_t \mid \mathbf{x}_{t+1}) = \mathcal{N}(\mathbf{x}_t; \tilde{\mathbf{F}}\mathbf{x}_{t+1} + \tilde{\mathbf{Q}}\Sigma_t^{-1}\mu_t, \tilde{\mathbf{Q}}).$$

Now for backward information filter we have

$$\begin{aligned}
\tilde{q}(\mathbf{x}_t \mid \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(i)})\tilde{\beta}_t^{(j)} &\simeq \gamma_t(\mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) \frac{\tilde{w}_{t+1}^{(j)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \\
&\simeq p(\mathbf{x}_t) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) \frac{\tilde{w}_{t+1}^{(j)}}{p(\tilde{\mathbf{x}}_{t+1}^{(j)})} \\
&= g(\mathbf{y}_t \mid \mathbf{x}_t) p(x_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}) \tilde{w}_{t+1}^{(j)} \\
&= \mathcal{N}(\mathbf{y}_t; \mathbf{G}\mathbf{x}_t, \mathbf{R}) \mathcal{N}(\mathbf{x}_t; \tilde{\mathbf{F}}\tilde{\mathbf{x}}_{t+1}^{(j)} + \tilde{\mathbf{Q}}\Sigma_t^{-1}\mu_t, \tilde{\mathbf{Q}}) \tilde{w}_{t+1}^{(j)}.
\end{aligned}$$

Here the suggestion is again to use GRL. By using this lemma we have

$$\begin{aligned}
\tilde{q}(\mathbf{x}_t \mid \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(i)})\tilde{\beta}_t^{(j)} &\simeq \mathcal{N}\left(\mathbf{y}_t; \mathbf{G}\mathbf{x}_t, \mathbf{R}\right) \mathcal{N}\left(\mathbf{x}_t; \tilde{\mathbf{F}}\tilde{\mathbf{x}}_{t+1}^{(j)} + \tilde{\mathbf{Q}}\Sigma_t^{-1}\mu_t, \tilde{\mathbf{Q}}\right) \tilde{w}_{t+1}^{(j)} \\
&= \mathcal{N}\left(\mathbf{x}_t; \mu_{t|t+1}, \Sigma_{t|t+1}\right) \times \\
&\quad \mathcal{N}\left(\mathbf{y}_t; \mathbf{G}\left(\tilde{\mathbf{F}}\tilde{\mathbf{x}}_{t+1}^{(j)} + \tilde{\mathbf{Q}}\Sigma_t^{-1}\mu_t\right), \mathbf{R} + \mathbf{G}\tilde{\mathbf{Q}}\mathbf{G}^{\mathrm{T}}\right),
\end{aligned}$$

where

$$\begin{aligned}
\mu_{t|t+1} &= \Sigma_{t|t+1}\left(\tilde{\mathbf{Q}}^{-1}\left(\tilde{\mathbf{F}}\tilde{\mathbf{x}}_{t+1}^{(j)} + \tilde{\mathbf{Q}}\Sigma_t^{-1}\mu_t\right) + \mathbf{G}^T\mathbf{R}^{-1}\mathbf{y}_t\right) \\
&= \Sigma_{t|t+1}\Big(\mathbf{F}^T\mathbf{Q}^{-1}\Sigma_{t+1}\mathbf{F}^{-\mathrm{T}}\Sigma_t^{-1}\Sigma_t\mathbf{F}^{\mathrm{T}}\Sigma_{t+1}^{-1}\tilde{\mathbf{x}}_{t+1}^{(j)} + \\
&\quad \tilde{\mathbf{Q}}^{-1}\tilde{\mathbf{Q}}\Sigma_t^{-1}\mu_t + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_t\Big) \\
&= \Sigma_{t|t+1}\left(\mathbf{F}^T\mathbf{Q}^{-1}\tilde{\mathbf{x}}_{t+1}^{(j)} + \Sigma_t^{-1}\mu_t + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_t\right),
\end{aligned}$$

and

$$\Sigma_{t|t+1}^{-1} = \tilde{\mathbf{Q}}^{-1} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}.$$

8.3 Two-filter smoother (Fearnhead et al.)

The optimal densities for Fearnhead's two-filter smoother is given by

$$\bar{q}(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)})\bar{\beta}_t^{(i,j)} \simeq f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) \frac{\tilde{w}_{t+1}^{(j)} w_{t-1}^{(i)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})},$$

where we can optimally choose $\bar{q}(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)})$, which gives

$$
\begin{aligned}
\bar{q}(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) &= p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) \\
&= \frac{p(\tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_t \mid \mathbf{x}_t, \mathbf{x}_{t-1}^{(i)}) f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})}{p(\tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)})} \\
&= \frac{p(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{y}_t, \mathbf{x}_t, \mathbf{x}_{t-1}^{(i)}) p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{x}_{t-1}^{(i)}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})}{p(\tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)})} \\
&\propto p(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) p(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) \\
&\propto \mathcal{N}(\tilde{\mathbf{x}}_{t+1}^{(j)}; \mathbf{F}\mathbf{x}_t, \mathbf{Q}) \left(\mathcal{N}(\mathbf{y}_t; \mathbf{G}\mathbf{x}_t, \mathbf{R}) \mathcal{N}(\mathbf{x}_t; \mathbf{F}\mathbf{x}_{t-1}^{(i)}, \mathbf{Q}) \right) \\
&\propto \mathcal{N}\left(\tilde{\mathbf{x}}_{t+1}^{(j)}; \mathbf{F}\mathbf{x}_t, \mathbf{Q}\right) \cancel{\mathcal{N}\left(\mathbf{y}_t; \mathbf{G}\mathbf{F}\mathbf{x}_{t-1}^{(i)}, \mathbf{R} + \mathbf{G}\mathbf{Q}\mathbf{G}^{\mathrm{T}}\right)} \\
&\quad \mathcal{N}\left(\mathbf{x}_t; \Lambda\left(\mathbf{Q}^{-1}\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_t\right), \right. \\
&\quad \left. \left(\Lambda^{-1} = \mathbf{Q}^{-1} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G}\right)\right) \\
&\propto \mathcal{N}\left(\tilde{\mathbf{x}}_{t+1}^{(j)}; \mathbf{F}\Lambda\left(\mathbf{Q}^{-1}\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_t\right)\right. \\
&\quad \left., \mathbf{Q} + \mathbf{F}\Lambda\mathbf{F}^{\mathrm{T}}\right) \\
&\quad \mathcal{N}\left(\mathbf{x}_t; \left(\tilde{\Lambda}\cancel{\Lambda^{-1}}\Lambda\left(\mathbf{Q}^{-1}\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_t\right)\right.\right. \\
&\quad \left.\left. + \mathbf{F}^{\mathrm{T}}\mathbf{Q}^{-1}\tilde{\mathbf{x}}_{t+1}^{(j)}\right), \tilde{\Lambda}^{-1} = \mathbf{Q}^{-1} + \mathbf{G}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{G} + \mathbf{F}^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{F}\right).
\end{aligned}
$$

The last density is the resampling probability of this algorithm and can be calculated as

$$
\begin{aligned}
\bar{\beta}_t^{(i,j)} &\propto \int f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) g(\mathbf{y}_t \mid \mathbf{x}_t) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) d\mathbf{x}_t \frac{\tilde{w}_{t+1}^{(j)} w_{t-1}^{(i)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \\
&\propto \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) p(\tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)}) dx_t \frac{\tilde{w}_{t+1}^{(j)} w_{t-1}^{(i)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})} \\
&\propto p(\tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)}) \frac{\tilde{w}_{t+1}^{(j)} w_{t-1}^{(i)}}{\gamma_{t+1}(\tilde{\mathbf{x}}_{t+1}^{(j)})},
\end{aligned}
$$

which cannot be factorized. But we can have access to a joint Gaussian distribution that defines this density by a mean vector and a covariance matrix. First we start with defining the target density as

$$
p(\tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)}) \propto \mathcal{N}\left(\begin{bmatrix} \tilde{\mathbf{x}}_{t+1}^{(j)} \\ \mathbf{y}_t \end{bmatrix}; \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} C_{xx} & C_{xy} \\ C_{yx} & C_{yy} \end{bmatrix} \right).
$$

C_{xx} can be calculated as

$$\begin{aligned} C_{xx} &= \mathbb{E}\left[\left(\tilde{\mathbf{x}}_{t+1}^{(j)}\right)\left(\tilde{\mathbf{x}}_{t+1}^{(j)}\right)^{\mathrm{T}}\right] - \left(\mathbb{E}\left[\tilde{\mathbf{x}}_{t+1}^{(j)}\right]\right)^2 \\ &= \mathbb{E}\left[\left(\mathbf{F}\left(\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t\right) + \mathbf{w}_{t+1}\right)\left(\mathbf{F}\left(\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t\right) + \mathbf{w}_{t+1}\right)^{\mathrm{T}}\right] \\ &\quad - \left(\mathbb{E}\left[\mathbf{F}\left(\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t\right) + \mathbf{w}_{t+1}\right]\right)^2 \\ &= \mathbf{Q} + \mathbf{F}\mathbf{Q}\mathbf{F}^{\mathrm{T}}. \end{aligned}$$

C_{yy} can be driven as

$$\begin{aligned} C_{yy} &= \mathbb{E}\left[\mathbf{y}_t\mathbf{y}_t^{\mathrm{T}}\right] - \left(\mathbb{E}\left[\mathbf{y}_t\right]\right)^2 \\ &= \mathbb{E}\left[\left(\mathbf{G}\left(\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t\right) + \mathbf{v}_t\right)\left(\mathbf{G}\left(\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t\right) + \mathbf{v}_t\right)^{\mathrm{T}}\right] \\ &\quad - \left(\mathbb{E}\left[\left(\mathbf{G}\left(\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t\right) + \mathbf{v}_t\right)\right]\right)^2 \\ &= \mathbf{R} + \mathbf{G}\mathbf{Q}\mathbf{G}^{\mathrm{T}}. \end{aligned}$$

As for the cross covariance C_{xy} we can write that

$$\begin{aligned} C_{xy} &= \mathbb{E}\left[\left(\mathbf{F}\left(\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t\right) + \mathbf{w}_{t+1}\right)\left(\mathbf{G}\left(\mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{w}_t\right) + \mathbf{v}_t\right)^{\mathrm{T}}\right] \\ &\quad - \left(\mathbb{E}\left[\tilde{\mathbf{x}}_{t+1}^{(j)}\mathbf{y}_t\right]\right)^2 \\ &= \mathbb{E}\Big[\mathbf{F}\mathbf{F}\mathbf{x}_{t-1}^{(i)}\left(\mathbf{x}_{t-1}^{(i)}\right)^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}} + \mathbf{F}\mathbf{F}\mathbf{x}_{t-1}^{(i)}w_t^{\mathrm{T}}\mathbf{G}^{\mathrm{T}} + \\ &\quad \mathbf{F}\mathbf{F}\mathbf{x}_{t-1}^{(i)}\mathbf{v}_t^{\mathrm{T}} + \mathbf{F}\mathbf{w}_t\left(\mathbf{x}_{t-1}^{(i)}\right)^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}} + \\ &\quad \mathbf{F}\mathbf{w}_t\mathbf{w}_t^{\mathrm{T}}\mathbf{G}^{\mathrm{T}} + \mathbf{F}\mathbf{w}_t\mathbf{v}_t^{\mathrm{T}} + \mathbf{w}_{t+1}\left(\mathbf{x}_{t-1}^{(i)}\right)^{\mathrm{T}}\mathbf{G}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}} + \\ &\quad \mathbf{w}_{t+1}\mathbf{w}_t^{\mathrm{T}}\mathbf{G}^{\mathrm{T}} + \mathbf{w}_{t+1}\mathbf{v}_t^{\mathrm{T}}\Big] - \left(\mathbb{E}\left[\tilde{\mathbf{x}}_{t+1}^{(j)}\mathbf{y}_t\right]\right)^2 \\ &= \left(\mathbb{E}\left[\tilde{\mathbf{x}}_{t+1}^{(j)}\mathbf{y}_t\right]\right)^2 + \mathbf{F}\mathbf{Q}\mathbf{G}^{\mathrm{T}} - \left(\mathbb{E}\left[\tilde{\mathbf{x}}_{t+1}^{(j)}\mathbf{y}_t\right]\right)^2 \\ &= \mathbf{F}\mathbf{Q}\mathbf{G}^{\mathrm{T}}. \end{aligned}$$

In the same as for C_{xy} we can find $C_{yx} = \mathbf{G}\mathbf{Q}\mathbf{F}^{\mathrm{T}}$ and by this we derived all the densities needed for the implementation of smoothing algorithms for a linear-Gaussian model.

Chapter 9

Results and simulation studies

9.1 Linear Gaussian example

Here we use a linear-Gaussian model for our simulation study. This choice allows us to compare different particle smoother algorithms with each other and also with Kalman smoother (Anderson and Moore, 1979; Kalman et al., 1960) as a benchmark in this special case. Although the optimal solution is given by Kalman smoother in this case, this is the only fair comparison between different algorithms as we have access to optimal propagation and resampling probability densities.

To do that, we consider a model the same as the one used in Fearnhead et al. (2010) with state and observation models as

$$\mathbf{x}_{t+1} \mid (\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) \sim \mathcal{N}(F\mathbf{x}_t, Q) \tag{9.1}$$
$$\mathbf{y}_t \mid (\mathbf{x}_{1:t}, \mathbf{y}_{1:t-1}) \sim \mathcal{N}(G\mathbf{x}_t, R), \tag{9.2}$$

where $G = \begin{bmatrix} 1 & 0 \end{bmatrix}$, $R = \tau^2$, and

$$F = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad , \quad Q = \nu^2 \begin{bmatrix} \frac{1}{3} & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}.$$

Finally, the model is completed by choosing an initial state as $\mathbf{x}_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$.

The state is derived from a pair of stochastic differential equations $\mathrm{d}\mathbf{x}_{t,1} = \mathbf{x}_{t,2}\mathrm{d}t$. A noisy observation of the first component is made at each time step. The parameter ν^2 determines the smoothness of the state over time. With a large value of ν^2, the state can move freely and thus follows the observations. When ν^2 is small, the model makes a linear fit to the observations.[1]

[1] This paragraph is taken from Fearnhead et al. (2010).

Here we compare all the eight different smoother algorithms including the new one from different aspects. As the Kalman smoother gives the best possible solution, first we compare all the methods with Kalman smoother in terms of the number of effective samples in a way similar to (Fearnhead et al., 2010) and in addition for a running time of 20 sec. After that we compare the performance in terms of root mean square (RMS) between Kalman smoother and particle smoother. Comparing the performance of different smoother algorithms will give us an insightful view of how they behave if we run them for different amount of time.

We first compare smoothing algorithms using model parameters of $\nu^2 = \tau^2 = 1$ with $\mu_0 = (0, 0)^{\mathrm{T}}$ and $\Sigma_0 = I_2$ for the prior. We used 20 different data-sets of length 200 and ran all the algorithms 20 times to remove the effects caused by a single data-set and a single run. In comparison to the work done in Fearnhead et al. (2010), after comparing the performance for a constant number of particles which are $10000, 300, 300, 300, 300, 1000, 3000, 3000$, respectively for filter-smoother, fixed-interval smoother, FFBSi, two-filter smoother of Briers, two-filter smoother of Fearnhead, fast FFBSi, linear smoother of Fearnhead and backward information smoother, we tried to run all the algorithms for maximum of 20 sec and compare them with each other for different computational time.

In our first type of comparisons, for the accuracy of our smoothing algorithms' estimates of the states, we estimate the effective sample size the same as the method used in Fearnhead et al. (2010). Comes from the fact that

$$\mathbb{E}\left\{\frac{(\overline{\mathbf{x}} - \mu)^2}{\sigma^2}\right\} = \frac{1}{N}, \tag{9.3}$$

where $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}$ are independent and identically distributed with $\mathcal{N}(\mu, \sigma^2)$, and $\overline{\mathbf{x}}$ is their sample mean. Here we use

$$N_{eff}(\mathbf{x}_{t,d}) = \left[\mathbb{E}\left\{\frac{(\hat{\mathbf{x}}_{t,d} - \mu_{t,d})^2}{\sigma_{t,d}^2}\right\}\right]^{-1},$$

where $\mu_{t,d}$ and $\sigma_{t,d}^2$ are the true mean and covariance of the smoother obtained from Kalman smoother and $\hat{\mathbf{x}}_{t,d}$ is the random estimate form a particle smoother. By this, we can say that our algorithm is more accurate if it has a larger $N_{eff}(\mathbf{x}_{t,d})$.

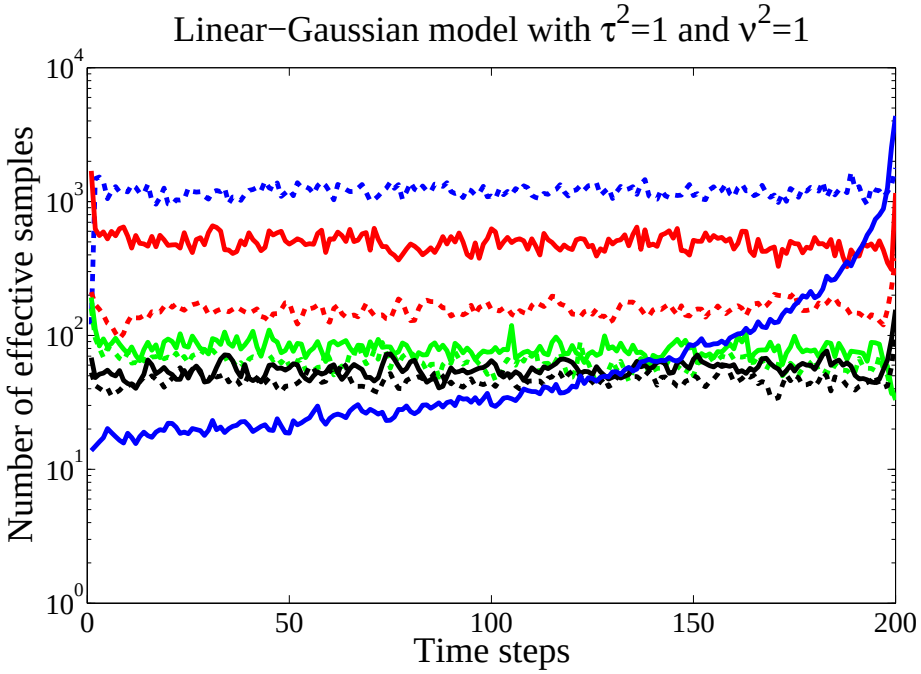


Figure 9.1: N_{eff} for fixed number of particles.[−−] Backward information smoother (N=3000**),** [−] Linear Smoother (Fearnhead, N=3000), [−−] Fast FF-BSi (N=1000), [−] Two-Filter Smoother (Fearnhead, N=300), [−−] Two-Filter Smoother (Briers, N=300), [−] Forward-Backward Smoother (N=300), [−−] FF-BSi (N=300), [−] Filter Smoother (Kitagawa 1996, N=10000).

Figure 9.1 shows how the average effective number of particles for estimating $\mathbf{x}_{t,1}$ varies through time steps for the eight algorithms in an approximately fixed running time. It is clear that filter-smoother does very well for times close to $T = 200$. But this algorithm gets progressively worse as it goes backward. This is not the case in other algorithms and their number of effective samples mainly remain constant in all time steps except the first and the last ones. In addition, although fast FFBSi is claimed to be linear (at least asymptotically), it does not compare favourably with Fearnhead's linear smoother and also backward information smoother as it is much slower than these two algorithm. On the other hand, fast FFBSi has a simpler implementation and design parameters are just related to the filtering, i.e. for smoothing there is no need to introduce any new proposal density. But Figure 9.1 cannot give us a full overview of the accuracy of these algorithms as we should have the results for different computational time. To do that, we ran all the algorithms for 20 sec and then compared them with each other.

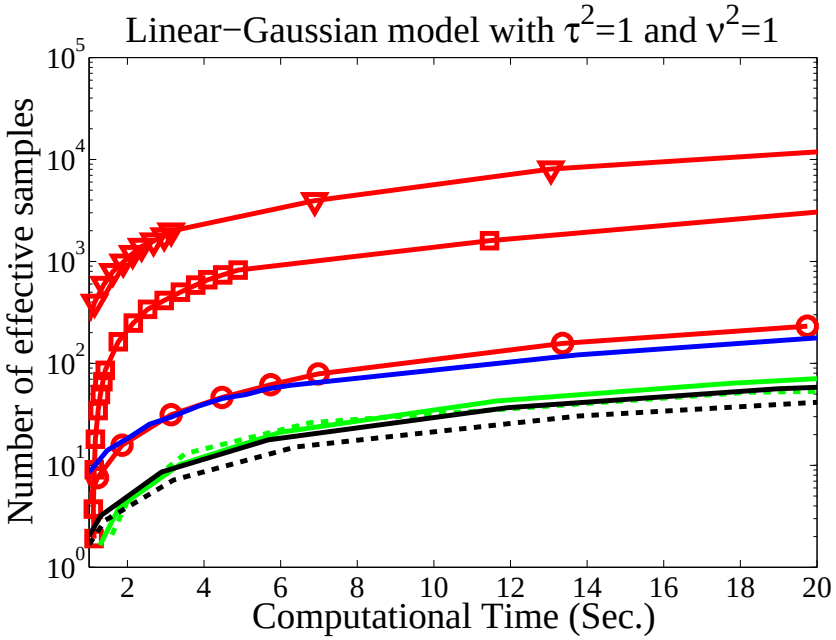


Figure 9.2: N_{eff} for all algorithms in 20 sec run. [▼] Backward information smoother**,** [■] Linear Smoother (Fearnhead), [●] Fast FFBSi, [−] Filter Smoother , [−] Two-Filter Smoother (Briers), [− −] Two-Filter Smoother (Fearnhead), [−] Forward-Backward Smoother, [− −] FFBSi.

Figure 9.2 illustrated a better view about how the effective samples size differ for each algorithm. For a short time of run (up to 3sec) N_{eff} has a non-linear growth for all the algorithms. But for times greater than 5sec this rate of growth becomes linear. As all the figures are in log scale, this linear growth is shown as a constant slope. Because of this linear growth in number of the effective samples, the linear algorithms are our best choices. This is because in these algorithms we are able to run the algorithm with large number of particles and as result we can have a better performance. Moreover, even for a small period of running, the linear smoother of Fearnhead and backward information smoother outperform all other algorithms. Our new algorithm is by far the best algorithm among all. The only limitation of our new smoother is that we should have access to a very good approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$. This is possible only when the transition state updates are linear and the additive noise is Gaussian. This will limit us to a special cases of the problems. One suggestion that has been made previously, is to use a Gaussian-mixture approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$. In the our next example in this section, we will go through a non linear problem to deal with this

problem.

Clearly, Fearnhead's linear smoother is also a very good smoother and non of other algorithms can compete with it. It is also has no limitation on the choice of artificial density and as result it does not face with problems similar to backward information smoother. Again, fast FFBSi is really not a competitor for the other two linear smoother algorithms in this special example, but the implementation of this algorithm is by far the easiest among the linear smoother algorithms proposed recently, as there is no need to introduce any new proposal density after filtering.

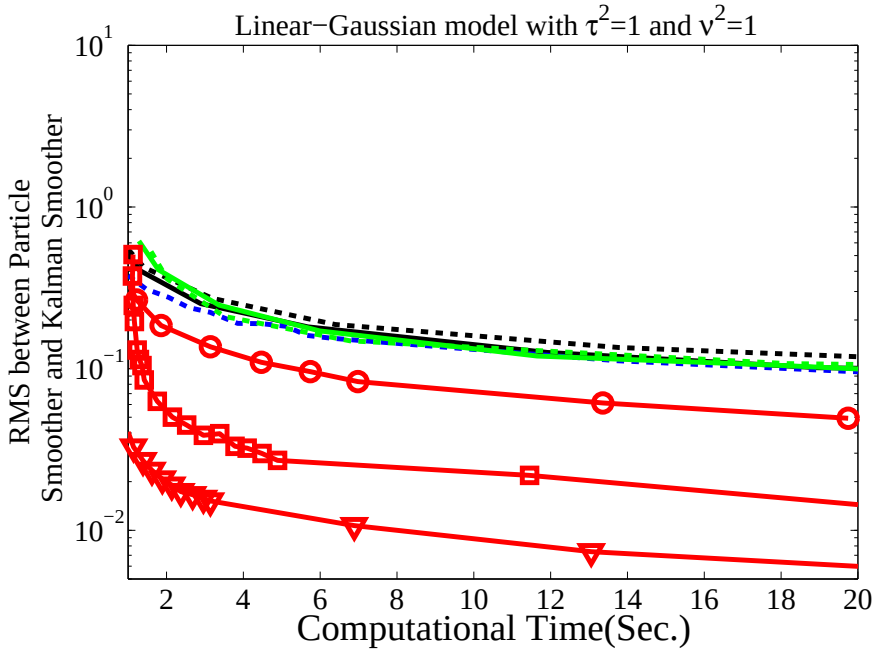


Figure 9.3: RMS between Particle smoother estimate and Kalman smoother. [▼] Backward information smoother**,** [■] Linear Smoother (Fearnhead), [●] Fast FFBSi, [–] Filter Smoother , [–] Two-Filter Smoother (Briers), [– –] Two-Filter Smoother (Fearnhead), [–] Forward-Backward Smoother, [– –] FFBSi.

In Figure 9.3 we compared all the algorithms in terms of RMS between the Kalman smoother and Particle smoother, which here we used

$$\text{RMS} = \sqrt{\frac{\sum_{t=1}^{T} \left(\hat{\mathbf{x}}_{t,d} - \mu_{t,d}\right)^2}{T}},$$

where $\mu_{t,d}$ is the true mean of the smoother obtained from Kalman smoother and $\hat{\mathbf{x}}_{t,d}$ is the random estimate form a particle smoother. This gives us a way

to compare different algorithms in a slightly different way than what is used before, as we do not use the covariance matrix of the Kalman smoother in this comparison. As RMS compare the different results in terms of euclidean distance, it is easy to see how the algorithms perform and how much they are nearer to the optimal solution (Kalman smoother estimate). Here again, for computational times less than $5sec$ we can see a non-linear improvement in the performance, which is very fast for linear smothers. But after that time and up to 20 sec of running, all the algorithms have a linear improvement in the performance. It is obvious that because of the linearity of the three new algorithms, we are able to use large number of particles and that is the reason of a better performance of these algorithms.

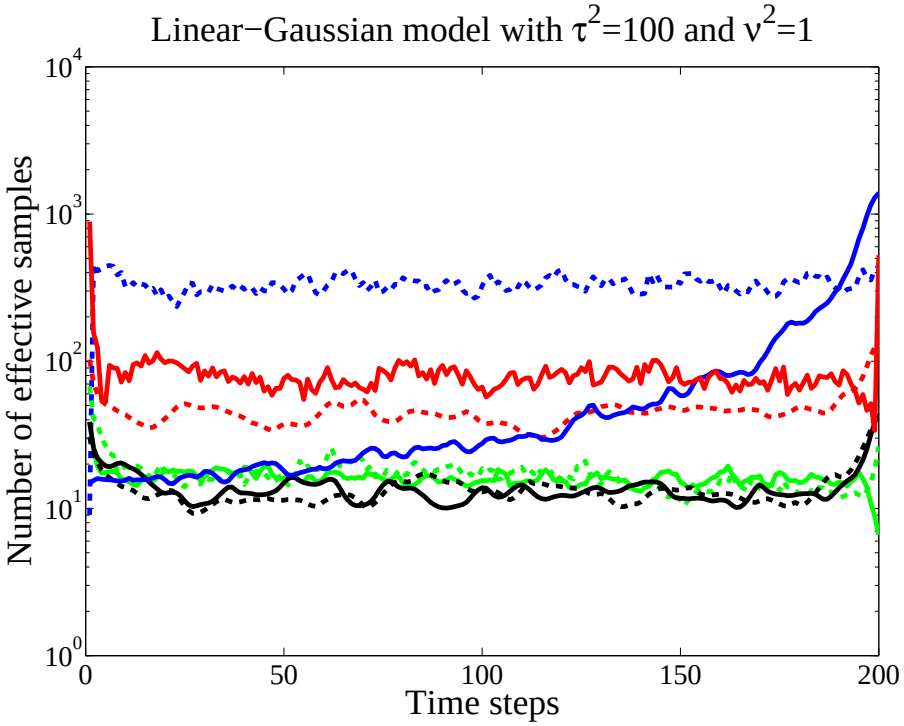


Figure 9.4: N_{eff} for fixed number of particles with $\tau^2 = 100$. [−−] Backward information Smoother (N=3000**),** [−] Linear Smoother (Fearnhead, N=3000), [−−] Fast FFBSi (N=1000), [−] Two-Filter Smoother (Fearnhead, N=300), [−−] Two-Filter Smoother (Briers, N=300), [−] Forward-Backward Smoother (N=300), [−−] FFBSi (N=300), [−] Filter Smoother (Kitagawa 1996, N=10000).

Another important aspect of the simulation results is to see how the algorithms perform when there is a high correlation between the states through

time. It is mentioned that this is still an issue for almost all of the smoothers and they perform poorly in this case. Here we can give a comparison for this case. To do so, we should change the parameter $\tau^2 = 100$. In this setting, we make the states to be high correlated through time to each other. Like the way we discussed about the results for previous case, first we compare the results for a set of fixed number of particles through all the time steps. Figure 9.4 shows the results for this comparison.

It is clear that number of the effective samples reduced by a large factor for all the algorithms. This is a big problem for particle smoother algorithms, which must be taken into account. One way to look at this problem is that when the states are highly correlated to each other through time, in non of the smoother algorithms we use this information, because our algorithms are based on the assumption that the states are not highly dependent to each other through the time. To make this problem more obvious we can use a three step of a Markov chain as the simplest example to describe it. If we assume that we are looking at time t, then times $t-1$ and $t+1$ will be its neighbors. In all the algorithms except linear smoother (Kitagawa, 1996) and Fearnhead linear smoother, we just update the weights and samples in backward direction by using the information available at time $t+1$. Even in this case we do not use the information about the correlation between times t and $t+1$. In Fearnhead linear smoother, the problem is more sever, as we condition on both information at $t-1$ and $t+1$ but wrongly by assuming independent transition between the states. Although, it is still not clear why the performance reduces this much because of having correlated states through time, but there may exist different solutions to this problem, like weighting the conditions differently, i.e. if we know that information at time t is highly correlated to $t-1$ then we can give more weight to this conditioning in some way.

Here we also put the results of the performance for different computational time up to 20 sec in this scenario. The results show that the performance is lower than the case with $\tau^2 = 1$ in all the times. Figure 9.5 and Figure 9.6 show the results for these comparisons.

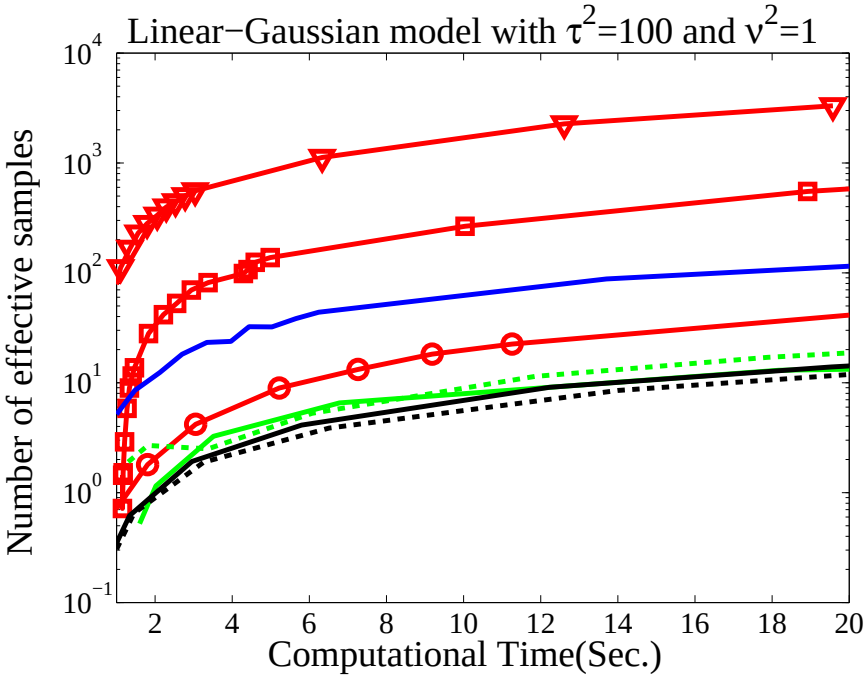


Figure 9.5: N_{eff} for all algorithms in 20sec run with $\tau^2 = 100$. [▼] Backward information smoother**,** [■] Linear Smoother (Fearnhead), [●] Fast FFBSi, [−] Filter Smoother , [−] Two-Filter Smoother (Briers), [− −] Two-Filter Smoother (Fearnhead), [−] Forward-Backward Smoother, [− −] FFBSi.

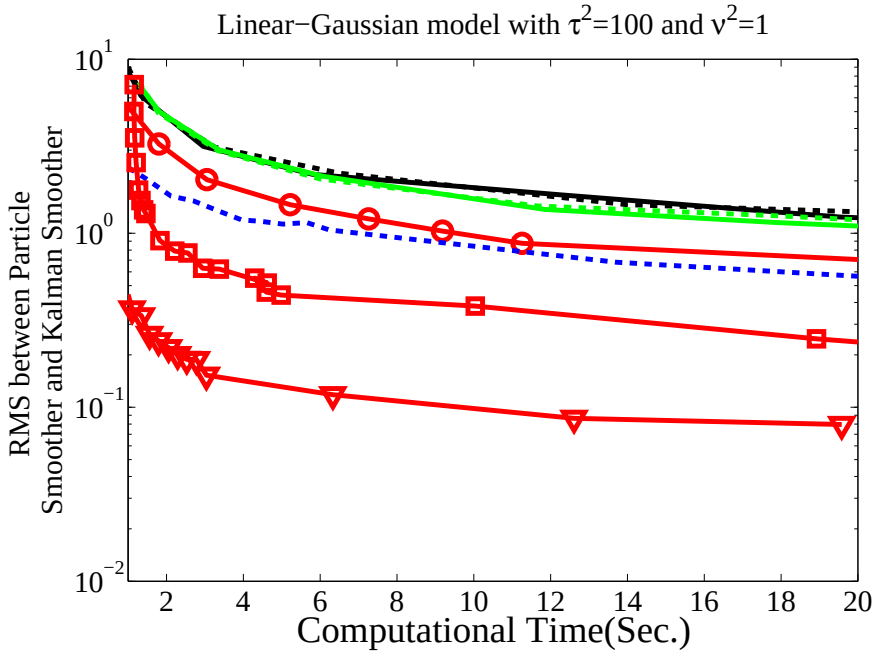


Figure 9.6: RMS between particle smoother estimate and Kalman smoother with $\tau^2 = 100$. [▾] Backward information smoother, [■] Linear Smoother (Fearnhead), [●] Fast FFBSi, [−] Filter Smoother , [−] Two-Filter Smoother (Briers), [− −] Two-Filter Smoother (Fearnhead), [−] Forward-Backward Smoother, [− −] FFBSi.

In Figure 9.5 and Figure 9.6 we can see the same trend as the case with normal correlation between states. The only difference is that the number of the effective samples is reduced for all the algorithms and in the same manner RMS is higher for all of them. Backward information filter has by far the best performance. This suggests us that there is a way to find a low complex approximation of $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$, then we can approximate the smoothing with high accuracy.

But generally, a linear-Gaussian model cannot give us a clear view of the performance of different algorithms as we already applied optimal proposal densities to all algorithms. In the next section we will compare Fearnhead's linear smoother, fast FFBSi and backward information smoother with each other when the states space model is not linear to see which one can approximate the smoother in a better and easier way.

9.2 Tracking a robot arm with planar two-link manipulator

So far, we talked mostly about linear-Gaussian state-space model. The power of sequential Monte Carlo methods are to solve nonlinear problems. To compare new linear smoother algorithms which we discussed in previous chapters, here we try to give a different example to be able to compare these methods in a more general case that we do not have access to optimal propagation and resampling probability densities.

The model here is a robot arm with planar two-link manipulator. Figure 9.7 shows a simplified model of such a robot arm where l_1 and l_2 are the lengths of the links and θ_1 and θ_2 are their respective angles.

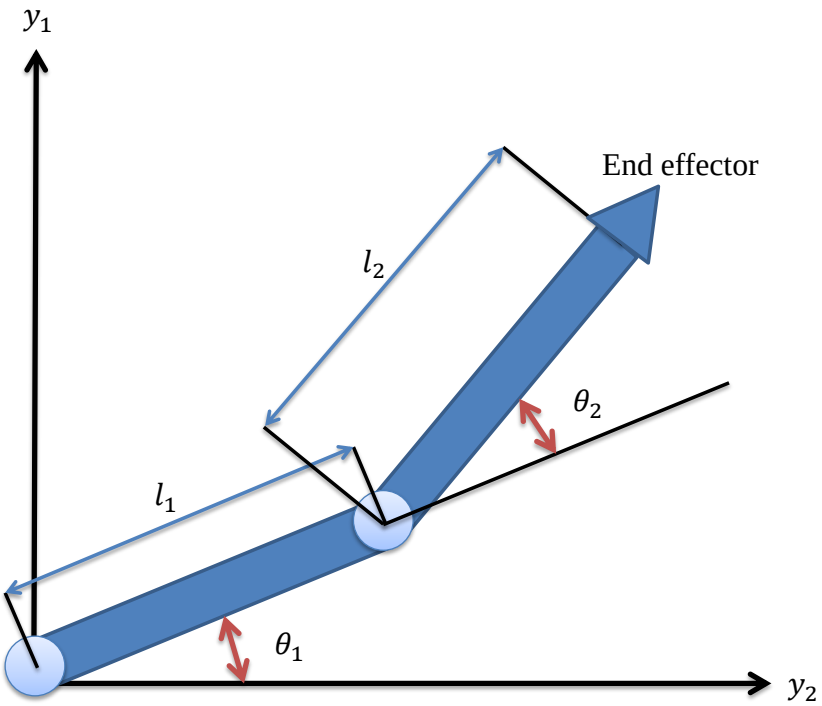


Figure 9.7: Simplified model of a robot arm with planar two-link manipulator.

We can show that the position of the end effector can be formulated as

$$\begin{aligned} y_1 &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ y_2 &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2), \end{aligned}$$

where y_1 and y_2 are the position of the end effector in Cartesian coordinates $(\mathcal{Y}_1, \mathcal{Y}_2)$ and $l_1 = 0.8$, $l_2 = 0.2$, $\theta_1 \in \left[0.3, 1.2\right]$ and $\theta_2 \in \left[\frac{\pi}{2}, \frac{3\pi}{2}\right]$. Here there are two different ways to look at the problem. One is called forward kinematics which is a way to map from (θ_1, θ_2) to (y_1, y_2). The other one, which is a more interesting and useful mapping is to map from (y_1, y_2) to (θ_1, θ_2) and called inverse kinematics, because we mostly like to give a robot our desired positions to for example drilling or painting and we expect the robot to be able to manage how it can change the angles correctly.

An inverse kinematics can be formulated as a state space model. This will give us the opportunity to implement the particle smoother algorithms for this problem. Let state vector $\mathbf{x}$ be $\mathbf{x} = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^{\mathrm{T}}$ and the measurement vector $\mathbf{y}$ be $\mathbf{y} = \begin{bmatrix}y_1 & y_2\end{bmatrix}^{\mathrm{T}}$. The state-space model of the inverse kinematics problem can now be written as

$$\begin{aligned}\mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{w}_{t-1} \\ \mathbf{y}_t &= \begin{pmatrix} l_1 \cos(\theta_{1,t}) + l_2 \cos(\theta_{1,t} + \theta_{2,t}) \\ l_1 \sin(\theta_{1,t}) + l_2 \sin(\theta_{1,t} + \theta_{2,t}) \end{pmatrix} + \mathbf{v}_t.\end{aligned}$$

Further we assume the state equation to follow a random walk model perturbed by white Gaussian noise $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, where

$$\mathbf{Q} = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.1 \end{bmatrix},$$

and measurement equation is nonlinear with measurement noise $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, where

$$\mathbf{R} = \begin{bmatrix} 0.005 & 0 \\ 0 & 0.005 \end{bmatrix}.$$

For this example, first we compare the results of particle filter and particle smoother for each algorithm to see if the smoother algorithms can give a better estimate of the angles than the particle filter. We will run all the three algorithms for 630 time steps and for roughly 25 sec, which is equal to choose $N = 12500$, 7500 and 500 for backward information smoother, Freanhead's linear smoother and fast FFBSi respectively. Before talking about the results, it is worthy to show how the robot will perform with the assumptions we made about the angles and noises. Figure 9.8 shows how the position and angles change in the whole steps.

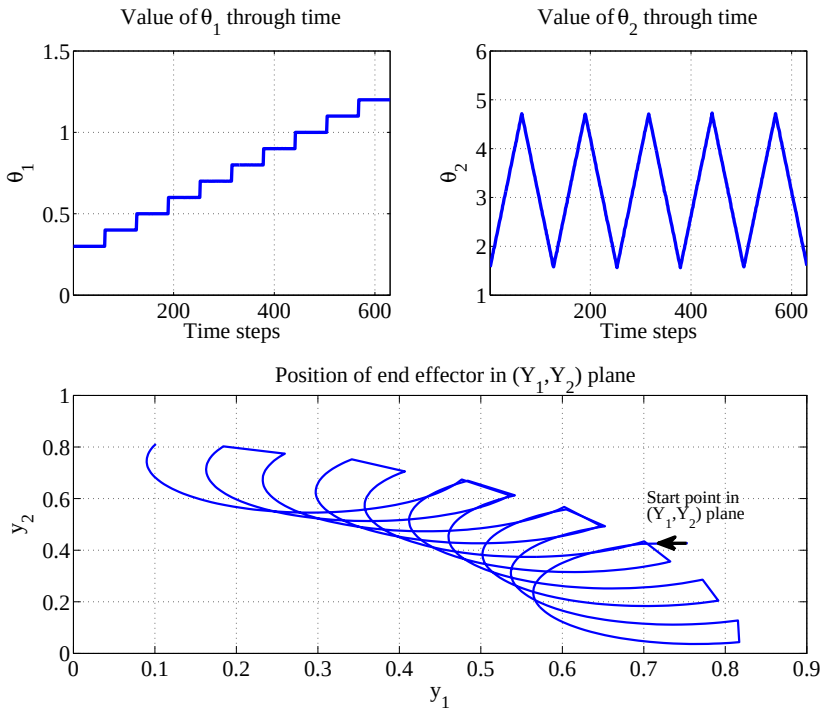


Figure 9.8: The transition of θ_1 through time [top-left], θ_2 through time [top-right] and (y_1, y_2) [bottom].

It is desirable for us to approximate (θ_1, θ_2) accurately for a given position. To do this we first implemented a particle filter that can estimate the filtering with good accuracy. The propagation and resampling density which we used are

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t) \propto f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$$

and

$$\beta_t^{(i)} \propto w_{t-1}^{(i)} g(\mathbf{y}_t \mid \mathbf{x}_{t-1}^{(i)}),$$

respectively. This is clear that we did not use the measurement at time t in propagation density, but in this way we can keep the algorithms as simple as possible.

For fast FFBSi we do not need any proposal density. The only parameters which are important to choose are ρ and R_{max}, which here we used $\rho = 1$ and $R_{max} = N/2$ like what we used for linear-Gaussian model.

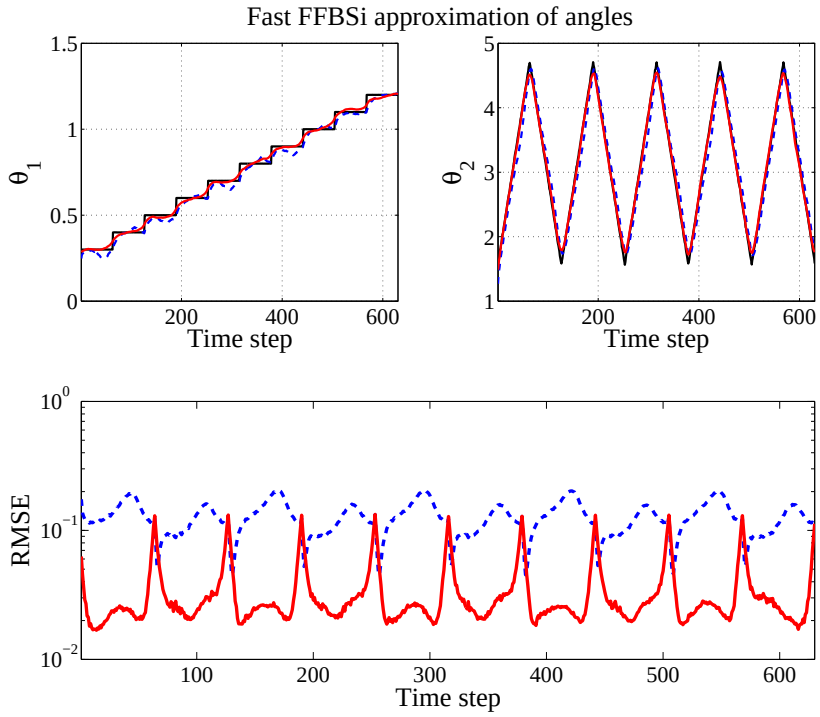


Figure 9.9: [−] True trajectory, [− −] filtered and [−] smoothed approximation of angles (top) and RMSE (bottom) for fast FFBSi algorithm.

The results in Figure 9.9 show that even by choosing $R_{max} = N/2$, which is not an optimal choice for this algorithm and excluding the information from the observation at time t in forward filtering, fast FFBSi can give a very good approximation of the smoother in terms of roor mean square error (RMSE) as the error is less than filtering and also is can be seen visually in the approximation if the angles in the figure.

As for backward information filter we need to approximate $\tilde{q}(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_t)\tilde{\beta}_t^{(j)}$ in a way that the samples and the weights in backward information filter can approximate the smoother accurately. As we do not have access to optimal densities because of the nonlinearity in the observation, we need to approximate them with some densities that drawing samples from them is easy. To be fair in our comparisons we exclude the information from the observation at time t here as well in the propagation density. Here we choose

$$\begin{aligned} \tilde{q}(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}, \mathbf{y}_t) &\propto p(\mathbf{x}_t \mid \tilde{\mathbf{x}}_{t+1}^{(j)}) \\ \tilde{\beta}_t^{(j)} &\propto p(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{y}_{1:T}), \end{aligned}$$

where $\tilde{\mathbf{x}}_{t+1}^{(j)}$ and their according weights are approximation of the smoothing. The results for implementation of this algorithm are shown in Figure 9.10.

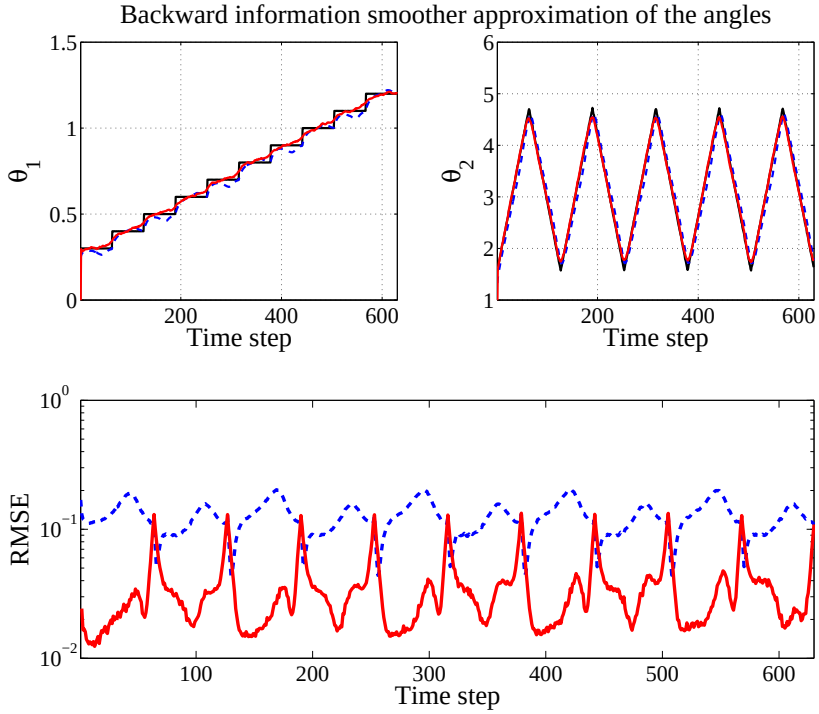


Figure 9.10: [—] True trajectory, [– –] filtered and [—] smoothed approximation of angles (top) and RMSE (bottom) for backward information smoother.

Results show that backward information smoother is also a powerful tool for approximating the smoother density. But as we know, the bottleneck of this algorithm is in approximating $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ with a good Gaussian mixture. To do this approximation in this special example, first we should observe the filtering and see if it is Gaussian or not. Because the state evolution is linear and Gaussian, if the filtering approximation is also Gaussian then we can approximate $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ with just one Gaussian density and as result the complexity of the algorithm remains linear. Clearly, it is kind of state of the art problem to approximate $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$, but when we can do it in a simple way, the algorithm can outperform other algorithms and has less problems in implementation. To clarify how we did the adjustments, here we have shown five kernel density estimation of the filtering $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ at time steps $40, 45, 50, 55$ and 60 for θ_2 (Figure 9.11).

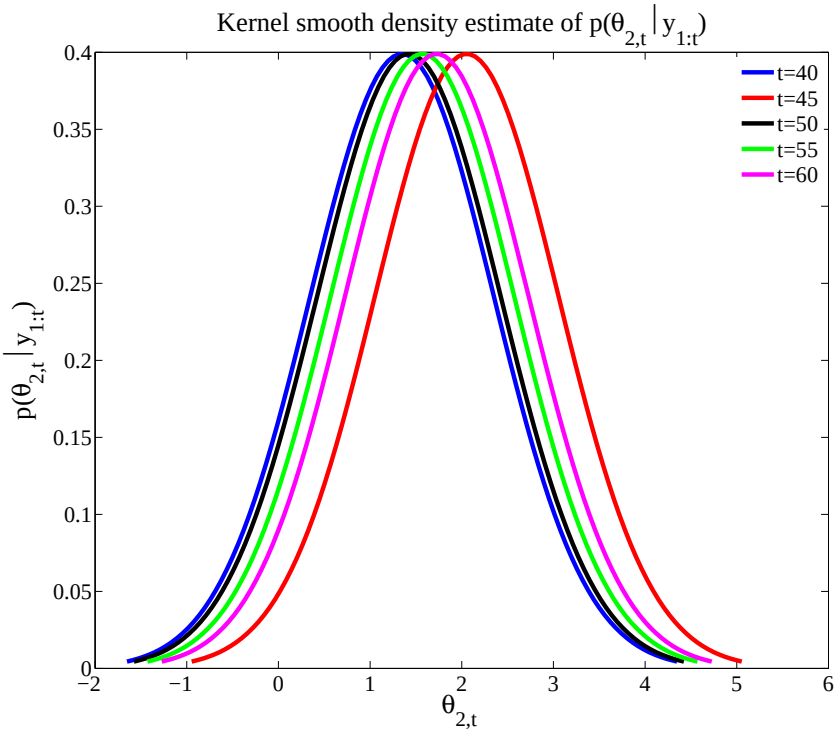


Figure 9.11: Kernel smooth density estimate of $p(\theta_{2,t} \mid \mathbf{y}_{1:t})$.

It is obvious that $p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ can be approximated with just one Gaussian density. The results for θ_1 are not shown here but they follow the same properties.

We implemented Feanhead's linear smoother for this problem as our last linear smoother algorithm. Here we should approximate $\bar{q}(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)})\bar{\beta}_t^{(i,j)}$as our proposal density. Like the other two algorithms, we are not going to use the observation at time t and will try to choose propagation density regardless of that information. As result, we will have

$$\begin{aligned} \bar{q}(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_t, \tilde{\mathbf{x}}_{t+1}^{(j)}) &\propto f(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}) f(\tilde{\mathbf{x}}_{t+1}^{(j)} \mid \mathbf{x}_t) \\ \bar{\beta}_t^{(i,j)} &\propto \beta_{t,\text{filter}}^{(i)} \beta_{t,\text{information}}^{(j)}, \end{aligned}$$

where $\tilde{\mathbf{x}}_{t+1}^{(j)}$ and $\beta_{t,\text{information}}^{(j)}$ are from backward information filter with artificial density equal to a uniform distribution. This choice of artificial may case degeneracy in different problems, but for this model it worked well. The result of this implementation is shown in Figure 9.12.

Figure 9.12 shows the performance of Fearnhead's linear smoother. RMSE is almost the same as the other two smoothers that we used here, which means

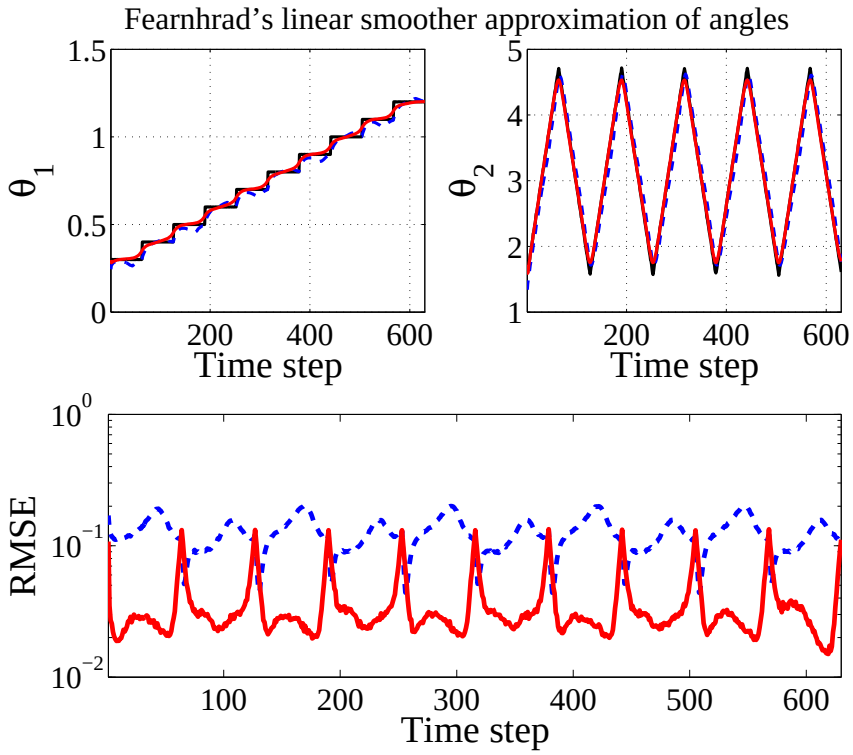


Figure 9.12: [−] True trajectory, [− −] filtered and [−] smoothed approximation of angles (top) and RMSE (bottom) for Fearnhead's linear smoother.

the propagations, resampling and artificial densities which we chose are well defined. To have a better understanding of how proposal density and other parameters can be designed for a specific problem, one can look at Fearnhead et al. (2010, section 5), which gives a complete solution for a famous problem in statistics.

Finally, as we have the results for these three linear smoother algorithms we are going to compare them with each other. First we compare them in terms of RMSE and then we compare them in terms of the accuracy of the position which they could approximate in comparison with true trajectory. Of course both of these comparisons will tell us the same results but the second one is a more visual presentation of how good are these algorithms for a non-linear problem. Figure 9.13 shows the comparison for all three algorithms in terms of RMSE.

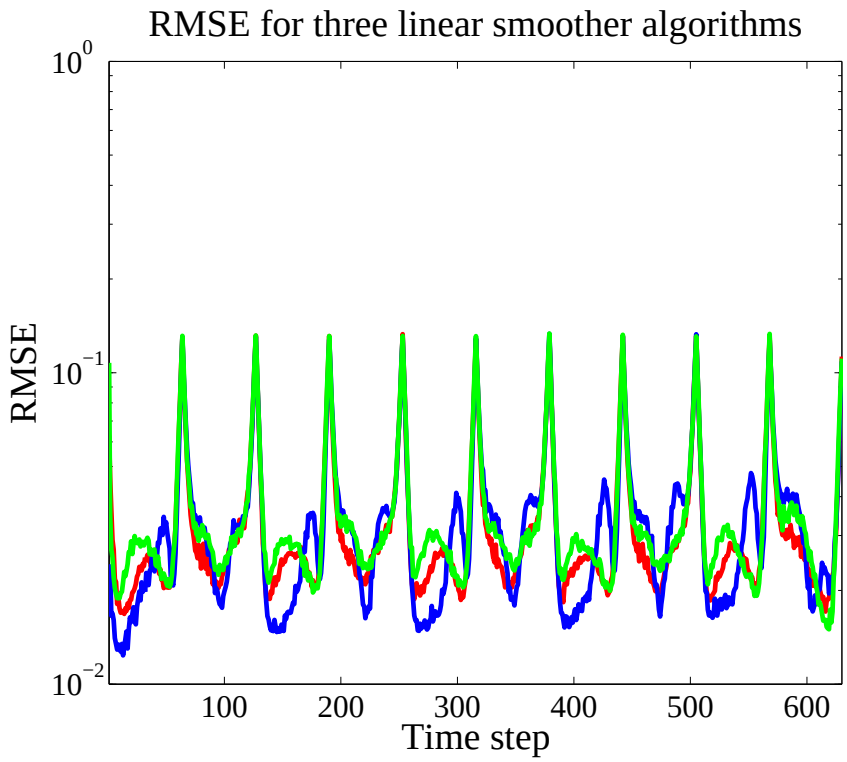


Figure 9.13: RMSE for [—] fast FFBSi, [—] backward information smoother and [—] Fearnhead's linear smoother.

Clearly, fast FFBSi, the backward information smoother and Fearnhead's linear smoother are all giving a good approximation to the smoothing distribution in terms of RMSE. The simplest algorithm to implement among these three algorithms is clearly fast FFBSi. The only drawback of this algorithm is that it suffers from time varying run over each realization. The backward information smoother algorithm is also performing very good, but it should be noticed that if we were not able to approximate the artificial density with just one Gaussian distribution, then we could not run the algorithm with linear complexity in terms of number of the particles.

Aside form many parameters to set before running Fearnhead's linear smoother, this algorithm is a reliable extension of two-filter smoother which has linear computational cost $O(N)$. Overall, it is really hard to choose an algorithm as the best in this comparison and it is highly dependent on the problems that one is interested to approximate the smoothing distribution with these algorithms. Parameters like simplicity, speed of convergence and design variables are important factors in choosing the best algorithm for each

problem. As our last comparison, it is interesting to see how these algorithms are able to approximate the position (y_1, y_2) correctly. Figure 9.14 shows how these three algorithms approximate true trajectory in $(\mathcal{Y}_1, \mathcal{Y}_2)$ plane.

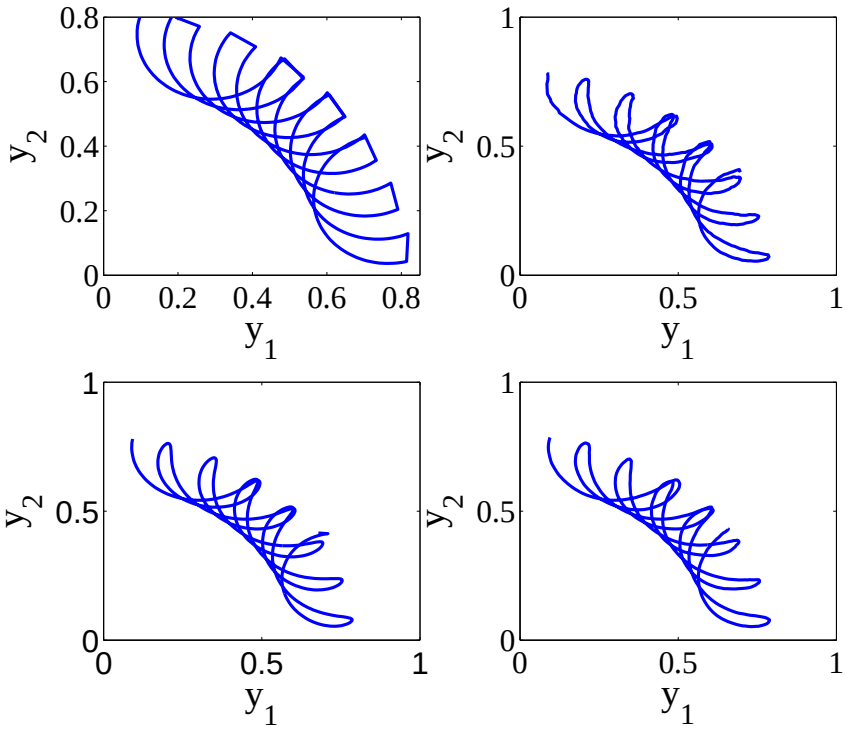


Figure 9.14: True trajectory (top-left) and approximation of the position for backward information smoother (top-right), Feanhead's linear smoother (bottom-left) and fast FFBSi (bottom-right).

Chapter 10

Discussion

In previous chapters we tried to give a complete and comprehensive explanation of how each smoother algorithm works and how the state of the art has changed between 1996 and 2011. As it is shown in the literature, the quadratic complexity of smoother algorithms was an issue in the cases which we needed to run the algorithms with large number of the particles until recently researchers introduced two different algorithms but linear in complexity (in terms of number of the particles). As these two algorithms using completely different approaches to target the smoothing distribution it is interesting to understand these two methods in depth. We were also able to introduce and investigate the ideas given in (Fearnhead et al., 2010) and suggest a new smoothing algorithm. Here in discussion chapter we will go through the strengths and weaknesses of each of these three algorithms i.e. Fearnhead's linear smoother, backward information smoother and fast FF-BSi once more and also discuss about open problems and possible updates to all these algorithms.

Fearnhead et al. (2010) gave a new way for smoothing algorithm similar to what we can see in (Briers et al., 2010) as two-filter smoother but with linear complexity in number of the particles. Clearly, the linear complexity of the algorithm in terms of number of the particles is an advantage that could not achieve before. The main idea of this algorithm is given in (Briers et al., 2005), but not so well written for a Markov chain Monte Carlo problem. The idea behind this algorithm is to use samples from the forward filter and the backward information filter to form a linear algorithm for smoothing. Besides the advantage of a complete linear algorithm (in terms of number of the particles) for any choice of N, there are a few drawbacks about this algorithm, which we explored in previous chapters. The first and most obvious problem

is many of proposal densities you need to design as well as the artificial density that algorithm is highly sensitive to these choices. It has been shown in (Fearnhead et al., 2010) that for an optimal smoother we should use the joint smoothing samples at time $t-1$ and $t+1$, but it will not be feasible to sample from the joint smoothing with linear complexity $O(N)$ due to the problem for approximating $p(\mathbf{x}_{t-1} \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:t})$. On the other hand, if we are able to find some wise choices for proposals and artificial density, the results outperform other algorithms but it is state of the art how to choose them and in sophisticated models it seems infeasible to be able to find such choices.

Backward information smoother is an extension to backward information filter with a fixed choice of artificial density. The idea for this adjustment is given in (Fearnhead et al., 2010), but further explorations have not done. In this thesis we explored such an special choice of artificial density i.e. $\gamma_t(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$ and designed a smoother algorithm, which can perform better than Fearnhead's linear smoother in the case that we have access to optimal proposal densities in a linear-Gaussian model. The only issue with this algorithm is how to design the artificial density in a general case. One way would be to approximate it with a Gaussian mixture as $\gamma_t(\mathbf{x}_t) = \sum_{k=1}^{K} \mathcal{N}\left(\mathbf{x}_t; \hat{\mathbf{x}}^{(k)}, \hat{\mathbf{P}}^{(k)}\right)$ where K is total number of Gaussian mixtures we used for approximation of $\gamma_t(\mathbf{x}_t)$. The problem here is that if we choose $K = N$ then the algorithm will be similar to FFBSi with quadratic complexity in number of the particles. But there are many cases that we can approximate this density with a very few number of Gaussian mixtures like in our non-linear example, which we can do it with just one Gaussian density. In other cases the complexity of finding a proper Gaussian mixture will grow quadratically when the dimensionality of the state-space goes higher. But there are new methods which are coming into the research area with linear complexity regarding dimension of the states for Gaussian mixture approximation.

Finally, we had fast FFBSi which has a totally different concept than the two other smoother algorithms and is based on rejection sampling. The simplicity of this algorithm is at the center of the attention as there is no need for any new design variables more than those in a particle filter. The only drawback of this algorithm is that there is no time bound for running this algorithm. It means that there are situations that this algorithm will have many unsuccessful rejection sampling iterations and all the samples will be rejected at any iteration. A very simple modification is given in (Lindsten, 2011) which is used in simulations, but this modification does not help to use large number of particles. To overcome this problem, we can use other modifications which are related to the probability of acceptance

of the particles. Giving a precise method as a wise stopping rule is part of the future work in this project. These kinds of modifications are useful and the reason is that although it is claimed that the computational complexity of the fast FFBSi will grow linearly as $N \to \infty$, in most cases we have $N = 5000$ to $N = 50000$ particles and these choices of N were not applicable before for this algorithm. This is a very crucial problem in this algorithm and by solving this problem it becomes a very powerful smoother because of simplicity and strong convergence results. In conclusion, all three new algorithms has their own advantages and disadvantages and which one to choose is depend on the problem we want to solve. But the easiest and most reliable one to implement in most of the problems would be fast FFBSi with the modifications given in this thesis and also (Lindsten, 2011). It is because this algorithm is fast and also has less number of design variables. It should be noticed that whenever we are able to design proposal densities in other two algorithms with high accuracy, then they both perform better and faster than fast FFBSi, but they are highly dependent on different choices of their design variables like artificial density and Gaussian mixture approximations.

Appendix A

Statistical efficiency of SIR-based auxiliary variable particle filter

In this part we are going to explain the reason of using SIR-based auxiliary variable particle filter instead of sequential importance sampling/resampling (SIR). This can be done by comparing the variance of the weights for both methods.

To measure the statistical efficiency of these two methods, we will minimize $\mathbb{E}[\tilde{w}_t]$, which $\tilde{w}$ denotes the normalized weights. To be more clear in reviewing the efficiency of these two methods, we start with some definitions and assumptions. Firstly, we define that $\tilde{w}_t^{(i)} = 1/N$ for both methods. Pitt and Shepherd Pitt and Shephard (1999) suggested that to understand the efficiency of the SIR method and be able to compare it with auxiliary particle filter (APF), it is useful to think of the weights of the SIR method as $\tilde{w}_t = \frac{p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t})}$, and then look into the variance of these weights, which is $\mathbb{E}\left[(\tilde{w}_t)^2\right] = \mathbb{E}\left[\left(\frac{p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t})}\right)^2\right]$. To calculate this, we should notice that here in SIR method, the function of the expectation is $p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t})$. Thus we have

$$\begin{aligned}\mathbb{E}\left[(\tilde{w}_t)^2\right] &= \mathbb{E}\left[\left(\frac{p(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1} \mid \mathbf{y}_{1:t})}\right)^2\right] \\ &= \int \left(\frac{p(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1} \mid \mathbf{y}_{1:t})}\right)^2 p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{t+1}. \qquad \text{(A.1)}\end{aligned}$$

Here it is easier to first calculate $p(\mathbf{y}_{t+1} \mid \mathbf{y}_{1:t})$, which is not dependent on

the function of expectation. Therefore for denominator we have

$$\begin{aligned}
p(\mathbf{y}_{t+1} \mid \mathbf{y}_{1:t})^2 &= \left(\int p(\mathbf{y}_{t+1}, \mathbf{x}_{t+1} \mid y_{1:t}) d\mathbf{x}_{t+1} \right)^2 \\
&= \left(\int p(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1}, \mathbf{y}_{1:t}) p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{t+1} \right)^2 \\
&= \left(\int g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1}) \left[\int p(\mathbf{x}_{t+1}, \mathbf{x}_t \mid \mathbf{y}_{1:t}) d\mathbf{x}_t \right] d\mathbf{x}_{t+1} \right)^2 \\
&= \left(\int g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1}) \times \right. \\
&\quad \left. \left[\int p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{y}_{1:t}) p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) d\mathbf{x}_t \right] d\mathbf{x}_{t+1} \right)^2 \\
&= \left(\int g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1}) \times \right. \\
&\quad \left. \left[f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_t^{(i)}) \right] d\mathbf{x}_{t+1} \right)^2 \\
&= \left(\frac{1}{N} \sum_{i=1}^{N} \int g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1}) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)}) d\mathbf{x}_{t+1} \right)^2, \quad \text{(A.2)}
\end{aligned}$$

and for numerator

$$\begin{aligned}
&\int g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})^2 p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{t+1} = \\
&= \int g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})^2 \left[f(\mathbf{x}_{t+1} \mid \mathbf{x}_t) \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_t^{(i)}) \right] d\mathbf{x}_{t+1} \\
&= \frac{1}{N} \int g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})^2 p(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)}) d\mathbf{x}_{t+1}. \quad \text{(A.3)}
\end{aligned}$$

Further if we assume

$$p_{(i)} = \int \left\{ \frac{g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)})} \right\}^2 f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)}) d\mathbf{x}_{t+1}, \quad \text{(A.4)}$$

and

$$p_{(i)}^* = \int \left\{ \frac{g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)})} \right\} f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)}) d\mathbf{x}_{t+1}, \quad \text{(A.5)}$$

then it is straight forward to show that

$$\mathbb{E}\left[(\tilde{w}_t)^2\right] = \frac{N\sum_{i=1}^{N}\lambda_{(i)}^2 p_{(i)}}{\left(\sum_{i=1}^{M}\lambda_{(i)}p_{(i)}^*\right)^2}. \tag{A.6}$$

The calculations for SIR-based auxiliary variable particle filter is as follows. Here we should notice that the function of expectation is $p(\mathbf{x}_{t+1}, i \mid \mathbf{y}_{1:t+1})$, thus the expectation will be over two variables, an integration over $\mathbf{x}_{t+1}$ and a summation over i. The expectation function can be defined similar to [section 3 of Pitt and Shephard (1999)] as

$$p(\mathbf{x}_{t+1}, i \mid \mathbf{y}_{1:t+1}) \propto p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)}) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)})\tilde{w}_t^{(i)}, \tag{A.7}$$

where $\mu_{t+1}^{(i)}$ is the mean, the mode, a draw, or some other likely value that has a relation with $f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)})$. So to calculate the variance of weights in this case we should calculate

$$\frac{\mathbb{E}_{x,i}\left[\left(\tilde{w}_t^{(i)}\right)^2\right]}{\left(\mathbb{E}_{x,i}\left[\tilde{w}_t^{(i)}\right]\right)^2},$$

where the denominator is a normalizing factor and $\tilde{w}_t^{(i)} = \frac{g(\mathbf{y}_{t+1}|\mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1}|\mu_{t+1}^{(i)})}$. Therefore, for denominator we have

$$\begin{aligned}\left(\mathbb{E}_{\mathbf{x},i}\left[\frac{g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)})}\right]\right)^2 &= \left(\sum_{i=1}^{N}\int \frac{g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)})}\times \right. \\ &\quad \left. p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)}) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)})\tilde{w}_t^{(i)} d\mathbf{x}_{t+1}\right)^2 \\ &= \left(\sum_{i=1}^{N}\lambda_{(i)}p_{(i)}^*\right)^2, \end{aligned} \tag{A.8}$$

and for numerator

$$\begin{aligned}\mathbb{E}_{\mathbf{x},i}\left[\left(\frac{g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})}{p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)})}\right)^2\right] &= \sum_{i=1}^{N}\int \frac{g(\mathbf{y}_{t+1} \mid \mathbf{x}_{t+1})^2}{p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)})^2}\times \\ &\quad p(\mathbf{y}_{t+1} \mid \mu_{t+1}^{(i)}) f(\mathbf{x}_{t+1} \mid \mathbf{x}_t^{(i)})\tilde{w}_t^{(i)} d\mathbf{x}_{t+1} \\ &= \sum_{i=1}^{N}\lambda_{(i)}p_{(i)}, \end{aligned} \tag{A.9}$$

so

$$\frac{\mathbb{E}_{\mathbf{x},i}\left[\left(\tilde{w}_t^{(i)}\right)^2\right]}{\left(\mathbb{E}_{\mathbf{x},i}\left[\tilde{w}_t^{(i)}\right]\right)^2} = \frac{\sum_{i=1}^N \lambda_{(i)} p_{(i)}}{\left(\sum_{i=1}^N \lambda_{(i)} p^*_{(i)}\right)^2}, \tag{A.10}$$

which $p_{(i)}$ and $p^*_{(i)}$ are the same as before and $\lambda_{(i)} = p(\mathbf{y}_{t+1} \mid \mu^{(i)}_{t+1})\tilde{w}^{(i)}_t$. As result, we can say an efficiency gain is existed if

$$\sum_{i=1}^N \lambda_{(i)} p_{(i)} < N \sum_{i=1}^N \lambda^2_{(i)} p_{(i)}. \tag{A.11}$$

If $p_{(i)}$ does not vary over i, then the auxiliary variable particle filter will be more efficient as $\sum_{i=1}^N \lambda_{(i)} \left(\frac{1}{N}\right) = \left(\frac{1}{N}\right) \leq \sum_{i=1}^N \lambda^2_{(i)}$. More likely is that $p_{(i)}$ depends on i, but only mildly, as $f(\mathbf{x}_{t+1} \mid \mathbf{x}^{(i)}_t)$ will be typically quite tightly peaked (much more tightly than $p(\mathbf{x}_{t+1} \mid \mathbf{y}_{1:t})$) compared to the conditional likelihood[1].

[1]This paragraph is directly taken from (Pitt and Shephard, 1999)

Appendix B

General refactorization lemma

Derivation of optimal propagation and resampling densities for a linear-Gaussian example can be simplified by introducing the following Gaussian factorization lemma (GRL).

Lemma B.0.1. *Given the L-dimensional vector $\mathbf{x}$, the M-dimensional vector $\mathbf{y}$, appropriately sized nonsingular covariance matrices $\mathbf{S}$ and $\mathbf{P}$, and the $M \times L$ matrix , the product function*

$$\eta(\mathbf{y}, \mathbf{x}) = \mathcal{N}(\mathbf{y}; \mathbf{F}\mathbf{x}, \mathbf{S})\mathcal{N}(\mathbf{x}; \mu, \mathbf{P})$$

can be refactored as

$$\eta(\mathbf{y}, \mathbf{x}) = \mathcal{N}(\mathbf{y}; \omega, \Omega)\mathcal{N}(\mathbf{x}; \lambda, \Lambda),$$

where the means and covariancees in the resulting product densities are

$$\begin{aligned} \omega &= \mathbf{F}\mu \\ \Omega &= \mathbf{S} + \mathbf{F}\mathbf{P}\mathbf{F}^{\mathrm{T}} \\ \lambda &= (\mathbf{I} - \mathbf{H}\mathbf{F})\mu + \mathbf{H}\mathbf{y} \\ \Lambda &= (\mathbf{I} - \mathbf{H}\mathbf{F})\mathbf{P}, \end{aligned}$$

with the supporting variable

$$\mathbf{H} = \mathbf{P}\mathbf{F}^{\mathrm{T}}\Omega^{-1}.$$

The parameters can also be expressed in information terms, where the infromation matrix is the inverse of the covariance matrix and the "infromation

vector" is the mean vector premultiplied by the infomation matrix. In this format, the density parameters are

$$\begin{aligned}
\Lambda^{-1}\lambda &= \mathbf{P}^{-1}\mu + \mathbf{F}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{y} \\
\Lambda^{-1} &= \mathbf{P}^{-1} + \mathbf{F}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{F} \\
\Omega^{-1}\omega &= \mathbf{D}\mathbf{P}^{-1}\mu \\
\Omega^{-1} &= (\mathbf{I} - \mathbf{D}\mathbf{F}^{\mathrm{T}})\mathbf{S}^{-1},
\end{aligned}$$

with the supporting variable

$$\mathbf{D} = \mathbf{S}^{-1}\mathbf{F}(\Lambda^{-1})^{-1}.$$

This is the main lemma that is used in the context to derive the formulas for optimal densities when the model is a linear-Gaussian model.

Bibliography

Anderson, B. and Moore, J. (1979). Optimal filtering. *Prentice-Hall Information and System Sciences Series, Englewood Cliffs: Prentice-Hall, 1979*, 1.

Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188.

Briers, M., Doucet, A., and Maskell, S. (2010). Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89.

Briers, M., Doucet, A., and Singh, S. (2005). Sequential auxiliary particle belief propagation. In *Information Fusion, 2005 8th International Conference on*, volume 1, pages 8–pp. IEEE.

Chopin, N. (2004). Central limit theorem for sequential monte carlo methods and its application to bayesian inference. *The Annals of Statistics*, 32(6):2385–2411.

Douc, R., Garivier, A., Moulines, E., and Olsson, J. (2010). Sequential monte carlo smoothing for general state space hidden markov models. *Submitted to Annals of Applied Probability*, pages 21–73.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208.

Dubarry, C. and Douc, R. (2011). Improving particle approximations of the joint smoothing distribution with linear computational cost. In *Statistical Signal Processing Workshop (SSP), 2011 IEEE*, pages 209–212. IEEE.

Fearnhead, P., Wyncoll, D., and Tawn, J. (2010). A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447.

Gardiner, C. (1985). *Stochastic methods*. Springer.

Godsill, S., Doucet, A., and West, M. (2004). Monte carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168.

Kalman, R. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kitagawa, G. (1996). Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, pages 1–25.

Lindsten, F. (2011). Rao-blackwellised particle methods for inference and identification.

Pitt, M. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, pages 590–599.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345.

Shumway, R. and Stoffer, D. (1982). An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis*, 3(4):253–264.

Taplin, R. (1998). The reverse kalman filter. *Communications in Statistics-Theory and Methods*, 27(10):2547–2558.