UiO : Department of Mathematics
University of Oslo

Twisting targets in Sequential Monte Carlo

Martin Strøm Olsen

Master's Thesis, Spring 2022

Abstract

Sequential Monte Carlo methods are often used for inference in state space models that are nonlinear and non-Gaussian. Inference about the latent variables of a state space model can be performed in an online setting by using Sequential Monte Carlo methods. It is also possible to obtain estimates of the marginal likelihood of the observations from a state space model in an online setting. There are different ways of introducing flexibility in Sequential Monte Carlo methods. One way is by considering the target distributions where one may alter all the intermediate target distributions except the final target distribution. Altering or twisting the intermediate target distributions can be done in different ways. One possibility is to alter the transition density and the observation density of the state space model by utilising a set of functions. These functions have few requirements and they may utilise observations from the state space model. There exists a specific set of functions which utilise all the observations, which implies an offline setting. This specific set of functions can also be used to alter the transition density and the observation density of the state space model. If Sequential Monte Carlo methods now are used to estimate the likelihood it can be shown that the variance of the likelihood estimates is minimised. The functions in the specific set are in general intractable and additionally defined in an offline setting. We therefore need to approximate these functions in an offline setting, subsequently we can use these to alter the transition density and the observation density. The motivation being that we now may obtain likelihood estimates with lower variance. Having the final target distribution unaltered implies that the marginal likelihood obtained by using intermediate twisting target distributions is equal to the marginal likelihood obtained by using unaltered target distributions.

We will consider a modified setup for utilising the twisting target framework in a batch setting, that is we assume that observations become available in batches. We also consider numerical experiments to check the effect of the batch setting on the variance of the likelihood estimates.

Acknowledgements

I primarily want to thank my supervisor Geir Storvik for his guidance, feedback and patience throughout the year. I also want to thank those accompanying me at Blindern campus throughout yet another Covid-year.

Contents

CHAPTER 1

Introduction

General methodologies such as Sequential Monte Carlo (SMC) are important for statistical inference in state space models. Many SMC methods have traditionally focused on inference in an online setting while it is also possible to utilise SMC methods in an offline setting (Naesseth, Lindsten, and Schön, 2019). The marginal likelihood related to the observations of a state space model is often of particular interest when considering parameter estimation (Guarniero, Johansen, and Lee, 2017). When estimating the marginal likelihood related to a state space model, the offline setting can be utilised in order to reduce the variance of the likelihood estimates.

There exist algorithms operating in an offline setting which can minimise the variance of the likelihood estimates. The setup for the algorithms yielding minimised variance likelihood estimates is however generally intractable and must be approximated. Approximations can however be iteratively improved to further reduce the variance of the likelihood estimates (Guarniero, Johansen, and Lee, 2017; Heng et al., 2020). Low variance is important for further inference and this will be our main focus in this thesis.

1.1 Structure

This thesis is structured into three main parts which are related to background concepts, twisting target distributions and numerical experiments. For all numerical examples and experiments we use R, Python and ggplot2 (Lutz, 2014; R Core Team, 2021; Wickham, 2016). We now consider a brief outline of the structure.

Background concepts

In this part we cover some of the background concepts that we will need in the following parts. The main focus here is SMC methods and likelihood estimation.

Chapter 2 contains some aspects related to state space models and Monte Carlo integration. State space models are thoroughly covered in Cappé, Moulines, and Rydén (2005). We briefly consider a connection between state space models and Bayesian networks which is reviewed in Koller and N. Friedman (2009) and Koller and Lerner (2001). We then consider importance sampling concepts and finally the SMC algorithm.

Chapter 3 covers some inferential aspects in state space models. Specifically we consider some aspects related to the filtering recursions and particle filter algorithms. Here we consider likelihood estimation by using particle filter algorithms. We also briefly consider the motivation for low variance likelihood estimates in relation to parameter estimation and some aspects of smoothing in state space models.

Twisting target distributions

In this part we introduce twisting target distributions by considering its motivation. Further we consider twisting functions as a strategy of obtaining twisting target distributions.

Chapter 4 introduces twisting target distributions and twisting functions. We consider the motivation for twisting target distributions and how it can be used to reduce the variance of the likelihood estimates. Twisting functions as a strategy to obtain twisting target distribution is also introduced. Here we consider the iterated auxiliary particle filter presented in Guarniero, Johansen, and Lee (2017). This is an offline algorithm utilising iteratively improving twisting functions to reduce the variance of the likelihood estimates.

Chapter 5 considers a setup that aims to utilise iteratively improving twisting functions in a batch setting. This is a setup that we have not seen before. We consider grouping the timeline consisting of iterations as a batch setting. This setup is utilising many of the concepts from Guarniero, Johansen, and Lee (2017) adapted to a batch setting. We aim to utilise these iteratively improving twisting functions in twisting target distributions and consequently reduce the variance of the likelihood estimates.

Chapter 6 contains an alternative way of approximating directly optimal twisting functions. Optimal twisting functions are introduced in Guarniero, Johansen, and Lee (2017) and can theoretically yield likelihood estimates with 0 variance. The alternative way of approximating the optimal twisting functions use reformulation and conditional independence properties of the state space model. This is an approximation that we have not seen before. Further we consider one of the deterministic approximations of twisting functions from Lindsten, Helske, and Vihola (2018) utilised in combination with the approach of Guarniero, Johansen, and Lee (2017) in order to reduce computational cost in some special cases.

Numerical experiments

We consider numerical experiments in order to compare the variance of the likelihood estimates obtained by algorithms in the online, batch and offline setting. Our aim is here to compare the effect of batching on the variance of the likelihood estimates.

Chapter 7 contains a series of numerical experiments where we focus on the variance of the likelihood estimates. Our motivation is mainly to compare the methodology of twisting target distributions in a batch setting to the traditional online and offline setting. The main focus here is also variance of the likelihood estimates.

Chapter 8 is a summary of the main topics that we have considered. Specifically, we summarise what has been done in this thesis and further possibilities.

Appendix A contains extended calculations related to some of the topics. These are are included in order to give more details about some of the topics covered in the text.

Appendix B contains calculations that were omitted from the numerical examples in order to focus on the most important parts.

1.2 Notation

We follow Doucet, Freitas, and Gordon (2001) and use lowercase letters z_t for both random variables and their realisations. For notational simplicity we also use lowercase z for both scalar and vector variables. The notation $z_{s:t} = (z_s, \ldots, z_t)$ is used for a sequence where $s < t$.

When it is either ambiguous or we want to highlight which distribution the expectation or variance is with respect to, we will use the notation E_p or V_p. Throughout we use θ as a generic parameter notation for both scalar and vector parameters. Variables on the form z_t^i use subscript t to indicate iteration and superscript i is used to indicate sequence number. When either the subscript and/or superscript is not relevant in the context it will be omitted for notational simplicity.

In general we will use $p(x)$ to denote a generic probability density functions and probability mass function. For simplicity we will refer to probability mass functions as also as probability density functions and therefore use the same notation. We use other letters to indicate density functions or distributions when they have a specific meaning in different contexts. Throughout we will also use the term iteration and notation t when referring to the time index of variables.

CHAPTER 2

Monte Carlo methods and state space models

In this chapter we start by a brief overview of state space models. We also consider Monte Carlo integration and some aspects of importance sampling in the context of state space models. Importance sampling will be introduced as the foundations of SMC methods.

2.1 State space models

To introduce the concept of a state space model (SSM) we mainly follow the approach provided in Chopin and Papaspiliopoulos (2020). We will in general refer to $t = 1, \dots, T$ as iterations where it is assumed that T is the final iteration. The description provided here is a brief overview, for a comprehensive review of state space models see Cappé, Moulines, and Rydén (2005). We will denote general parameters by θ when these are unknown and of general interest. The variables ϵ_t and η_t are independent and identically distributed variables at iteration t (Chopin and Papaspiliopoulos, 2020). We can define a SSM by using three general, deterministic, functions

$$\begin{aligned}
x_1 &= S_1(\epsilon_1, \theta) \\
x_t &= S(x_{t-1}, \epsilon_t, \theta) \\
y_t &= O(x_t, \eta_t, \theta).
\end{aligned}$$

Here we will in addition assume that the model is time homogenous, that is the functions S_1, S and O are equal for every iteration t. We will follow Creal (2012) and assume that the functions may be nonlinear and that the variables in the model may be continuous or discrete. In general we refer to the sequence $x_{1:t}$ as the latent variables. Similarly, we refer to the sequence $y_{1:t}$ as the observable variables or observations. These are often referred to as stochastic processes (Cappé, Moulines, and Rydén, 2005). At each iteration, t, we can think of receiving an observation y_t and we want to utilise the information from that observation in order to make inference about the latent variable x_t. The function S_1 is the first or initial state equation, it differs from the general S because it has no dependence on any previous latent variable. The function S is often called state equation while the function O is called observation equation (Cappé, Moulines, and Rydén, 2005). It is also possible to define a SSM by using density functions for the latent variables and the observations. We here use the generic notation p, to denote a density function. We can then define a SSM with density functions, adapted from equation (4) in Naesseth, Lindsten, and Schön (2019). We then have

$$
\begin{aligned}
x_1 &\sim p(x_1|\theta) && \text{initial density} && (2.1a)\\
x_t|x_{t-1} &\sim p(x_t|x_{t-1},\theta) && \text{transition density} && (2.1b)\\
y_t|x_t &\sim p(y_t|x_t,\theta) && \text{observation density.} && (2.1c)
\end{aligned}
$$

Here, the observation y_t only depends on the latent variable x_t. Further, the latent variable x_t only depends on the previous latent variable x_{t-1} and not any variables preceding this (Naesseth, Lindsten, and Schön, 2019). The model defined in equation (2.1) is sometimes referred to as a hidden Markov model (HMM) when y_t only depends on x_t and x_t only depends on x_{t-1}. We will however mainly refer to this model as a SSM. The parameters θ will in general be omitted from the functions when they are assumed known for notational simplicity.

There are many variations within the SSM framework. In special cases such as finite SSMs or linear Gaussian SSMs, we are able to find recursive, analytical, expressions related to the latent variables. When considering the linear Gaussian case, this is referred to as Kalman filtering. We can further use these expressions to find analytical posterior density functions, such as $p(x_t|y_{1:t})$ and likelihood factors such as $p(y_t|y_{1:t-1})$. Utilising likelihood factors, we can find a recursive, analytical, expression for the likelihood. This is given by the decomposition into likelihood factors

$$p(y_{1:t}) = p(y_1)\prod_{s=2}^{t} p(y_s|y_{1:s-1}).$$

See e.g. chapter 5 of Cappé, Moulines, and Rydén (2005) for detailed reviews of inference in finite SSMs and linear Gaussian SSMs. We note that even though the two special cases of SSMs are of major importance, the general case is that SSMs can be both nonlinear and non-Gaussian. For the general case, we often need approximations based on e.g. simulation. We will now consider some properties of SSMs which we utilise when it comes to further calculations and inference. The first is related to the latent variables and the second is related to the observations. Combined, these form what we will refer to as the conditional independence (CI) properties of the SSM.

Latent variables

The Markov assumption is a CI assumption between the last latent variable x_t in the model and all preceding latent variables except x_{t-1}. In general, we will consider the case where the Markov assumption holds for all t. We can be summarise this by the following CI statements (Koller and N. Friedman, 2009, p. 201). We then have

$$(x_t \perp x_{1:t-2}|x_{t-1}).$$

This simplifies a lot of the calculations when considering inference in SSM through different methods. It is possible to have higher order Markov structures for the latent variables as discussed in Naesseth, Lindsten, and Schön (2019). In this case, the latent variable x_t may depend on an arbitrary number of previous latent variables and in the extreme all the previous latent variables $x_{1:t-1}$. Assume a situation where x_t has a higher order Markov structure and depends on the previous latent variables, $x_{t-k:t-1}$, where $k > 1$. It is possible to define the latent variable as a vector consisting of $x_{t-k+1:t}$ to get the original Markov structure back (Cappé, Moulines, and Rydén, 2005, p. 4). This flexibility can often be important as there are scenarios where the first order Markov structures might be insufficient to define a model.

Observations

Another property which also is simplifying when considering inference calculations is the CI properties of the observations. In words we could say that the latent variable x_t shields the observation y_t from all other variables. That is, when we have x_t in the conditioning set, all other variables are independent of y_t. This corresponds to the following CI statements from Cappé, Moulines, and Rydén (2005, p. 2)

$$(y_t \perp x_{1:t-1}, y_{1:t-1} | x_t).$$

In the same manner as for the latent variables, we can consider situations where it is a higher order Markov structure between the observation y_t and latent variables. In the full dependence case we have $y_t|x_{1:t}$ and $x_t|x_{1:t-1}$ and this is often called non-Markovian latent variable model Naesseth, Lindsten, and Schön (2019, p. 7). This increase in dependence will naturally complicate calculations related to the model.

We will see that the Markov structure in both the latent variables and the observations will simplify calculations considerably. We can consider the conceptual structure of a state space model in Figure 2.1.

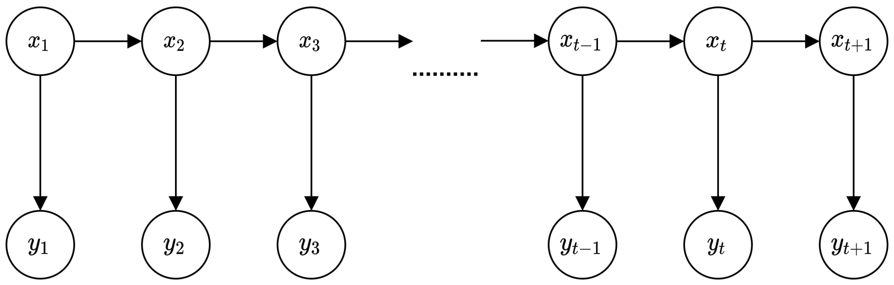


Figure 2.1: Conceptual SSM at $t + 1$.

Following chapter 6 of Koller and N. Friedman (2009), we can consider an example HMM with first order Markov structure and finite, discrete, variables. We can then view this model as a dynamic Bayesian network (DBN) which form a Bayesian network (BN) with a conditional probability distribution (CPD) for each node. We can then think of the DBN over the variables $x_{1:t}$ and $y_{1:t}$ as being conceptually stopped at a given iteration t and we view the resulting network as a traditional BN. The DBN structure lets us repeat the $x_t \to y_t$ structure for increasing t and the parent structure $\text{Pa}_{x_t} = x_{t-1}$ connects the time slices together. The observations follow a similar parent structure $\mathrm{Pa}_{y_t} = x_t$. See chapter 6 of Koller and N. Friedman (2009) for a detailed review. Because the DBN can be viewed as a BN for a given iteration t, we can utilise the CI structure present in BN when considering the model at that iteration. We would then have the following structure where the arrows represent trails between the variables. We then have

$$\begin{aligned} x_{t-1} \rightarrow & x_t \rightarrow y_t \\ y_t \leftarrow & x_t \rightarrow x_{t+1}. \end{aligned}$$

Here the first and second trail above correspond to a causal trail and common cause respectively. They are both inactive when x_t is being conditioned on, implying conditional independence between $(x_{t-1},\, y_t)$ and between $(y_t,\, x_{t+1})$. The general trails which extend these trails either way are then also inactive (Koller and N. Friedman, 2009, p. 71). This would imply that the following trails

$$\begin{aligned} a \rightarrow & x_t \rightarrow b \\ c \leftarrow & x_t \rightarrow b. \end{aligned}$$

Here a, b, c are trails on either side of the edges going in to/out from x_t. Trails between nodes in a and nodes in b are active as long as none of the variables in the trails connecting them are in the conditioning set. This is because the trails a, b, c does not contain any v-structures (Koller and N. Friedman, 2009, p. 71). We can then think of the variable x_t as blocking influence between any x_s or y_s before where $s < t$ and any x_s or y_s where $s > t$ when it is in the conditioning set. A detailed review can also be found in Koller and Lerner (2001). The BN structure can be useful when assessing how influence flows between the variables in the model.

2.2 Special distributions

In this section we briefly consider two special distributions, the target distribution and the importance sampling distribution. We start by considering the target distribution. It can be useful to use a general definition of the target distributions. We follow the notation of Naesseth, Lindsten, and Schön (2019) and define the general target distributions for $t = 1, \ldots, T$ by

$$f_t(x_{1:t}) = \frac{\tilde{f}_t(x_{1:t})}{Z_t}. \tag{2.2}$$

Here Z_t is the normalising constant for the positive and integrable function $\tilde{f}_t(x_{1:t})$ at iteration t (Naesseth, Lindsten, and Schön, 2019, pp. 4-6). We often refer to $\tilde{f}_t(x_{1:t})$ as the unnormalised target distribution. Therefore we have that the normalising constant can be written as $Z_t = \int \tilde{f}_t(x_{1:t})dx_{1:t}$. In the settings we will consider the target distributions are often closely related to the components of the SSM. In these setting we often define the unnormalised target distributions $\tilde{f}_t(x_{1:t})$ to be joint distributions over variables from the model. The resulting target distributions $f_t(x_{1:t})$ are then posterior distributions of the latent variables, x, conditioning on the observations, y. These posterior distributions will be our main focus until Chapter 4.

Because of this, we will often define the unnormalised target distribution on the form $\tilde{f}(x) = p(x, y)$ and the target distribution on the form $f(x) = p(x|y)$. The notation x and y denote some subset of latent variables and observations respectively. Until we consider twisting target distributions in Chapter 4, we will therefore refer to posteriors on the conceptual form $p(x|y)$ as the target distributions. The main specific target distributions of interest are the posterior distributions on the form $p(x_{1:t}|y_{1:t}, \theta)$. For this reason we define these specific target distributions now. The unnormalised target distribution is defined by $\tilde{f}_t(x_{1:t}) = p(x_{1:t}, y_{1:t}|\theta)$ following Naesseth, Lindsten, and Schön (2019, eq. 6). The specific target distributions then become $f_t(x_{1:t}) = p(x_{1:t}|y_{1:t}, \theta)$ for $t = 1, \dots, T$. We can now consider general reformulations of the target distributions combined with the conditional independence properties of the SSM to see a recursive structure. We then have

$$p(x_{1:t}|y_{1:t}, \theta) = \frac{p(y_t|x_t, \theta)p(x_t|x_{t-1}, \theta)}{p(y_t|y_{1:t-1}, \theta)} p(x_{1:t-1}|y_{1:t-1}, \theta).$$

In general, we omit the parameters θ when they are assumed known. The joint recursive form of the target distributions without θ is then given by

$$p(x_{1:t}|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|y_{1:t-1})} p(x_{1:t-1}|y_{1:t-1}) \tag{2.3a}$$

$$p(x_{1:t}|y_{1:t}) \propto p(y_t|x_t)p(x_t|x_{t-1})p(x_{1:t-1}|y_{1:t-1}). \tag{2.3b}$$

The fraction before the target distribution at iteration $t-1$ consists of the density functions from from the model defined in equation (2.1). We refer to the term $p(y_t|y_{1:t-1})$ as the likelihood factor at iteration t, following Chopin and Papaspiliopoulos (2020). When focusing on the posterior distribution, the likelihood factor does not depend on $x_{1:t}$ and can be considered constant. This implies that we are able to recursively express the current target distribution up to proportionality using the previous target distribution and the components from the SSM.

One important decomposition of the unnormalised target distribution $p(x_{1:t}, y_{1:t})$ is the joint distribution over all the variables expressed by the transition densities and observation densities from the model. We then continue the recursive structure in (2.3b) and use the CI properties of the SSM. Straightforward calculations lets us decompose the joint distribution $p(x_{1:t}, y_{1:t})$ into

$$p(x_{1:t}, y_{1:t}) = p(y_1|x_1)p(x_1) \prod_{s=2}^{t} p(y_s|x_s)p(x_s|x_{s-1}). \tag{2.4}$$

We denote the generic importance sampling function by g and in general this will be needed whenever simulating variables directly from the target distribution is intractable. An importance sampling function can also be useful if it is possible, but too computationally costly to simulate directly from the target distribution. We can instead simulate variables from some other importance sampling function g. Then, we can weight the variables which now are simulated from the importance sampling distribution instead of the target distribution. As we will see this is the idea in importance sampling (IS). We often use a conditional distribution where we have the observations up until the current iteration in the conditioning set, then we can denote the importance sampling distribution by $g(x_{1:t}|y_{1:t})$. One possibility when it comes to the structure of g is to define it by a recursive structure where we consider the leftmost term to simulate x_t and the rightmost term as the distribution for the already simulated $x_{1:t-1}$ (Creal, 2012, eq. 10). The importance sampling distribution is then given by

$$g(x_{1:t}|y_{1:t}) \equiv g(x_t|y_{1:t}, x_{1:t-1})g(x_{1:t-1}|y_{1:t-1}). \tag{2.5}$$

This structure is important for the efficacy of the sequential weight structures used in importance sampling-based algorithms. As we will see, the importance sampling distributions and the target distributions are important components in many of the importance sampling-based algorithms. By adjusting both of these one may increase the efficiency of many of the algorithms as is covered in section 3 of Naesseth, Lindsten, and Schön (2019).

2.3 The Monte Carlo method

We first consider the curse of dimensionality to motivate Monte Carlo as a numerical integration method. In order to have a given number of n^d evaluation points in high-dimensional integrals, one need e.g. n points for each dimension. To have the same number of evaluation points in all d dimensions we need n^d points in total. The number of total points needed then increase quickly with increasing d. This rapid growth of needed evaluation point when the dimensionality increase is often referred to as the curse of dimensionality.

We follow one of the examples in Hastie, Tibshirani, and J. Friedman (2009) in order to illustrate the curse of dimensionality. This can be illustrated by considering the ratio $v = \frac{v_s}{v_1}$ of two volumes. We think of each of the volumes containing evenly distributed points and we now want to consider the ratio v of points contained in a smaller volume v_s to the unit volume v_1. The volume $v_s = e^d$ is a d-dimensional hypercube with edge length e. We refer to this as the smaller hypercube because it will be contained in the unit hypercube. The volume v_1 is that of a unit hypercube in d dimensions. Consider a unit hypercube in d dimensions. That is, a line $[0,1]$ in $d = 1$, a square $[0,1] \times [0,1]$ in $d = 2$, a cube $[0,1] \times [0,1] \times [0,1]$ in $d = 3$ and so on. Because all sides of v_1 has edge length 1 we know that the volume of the unit hypercube is $v_1 = 1$. We consider the ratio v as a function of the edge length $e \in [0,1]$ and the dimensionality $d \in \mathbb{N}^+$. The ratio v is then given by a function

$$v = e^d.$$

To illustrate the concept we can plot v as a function of the edge length $e \in [0,1]$ and dimensions $d = 1, \ldots, 20$. We see from Figure 2.2 that with higher dimensionality d we need to increase each edge length e in the smaller hypercube in order to cover the same ratio of volume v. As an example we see that already at dimension 4 we need to have edge length $e > 0.8$ in order to cover about half of the unit hypercube volume. In contrast, when considering 20 dimensions, an edge length of 0.8 yields a hypercube that covers less than 0.05 of the unit hypercube.

Metropolis and Ulam presented an example of calculating a complex, finite, volume in dimension d located within a unit hypercube. The calculation of

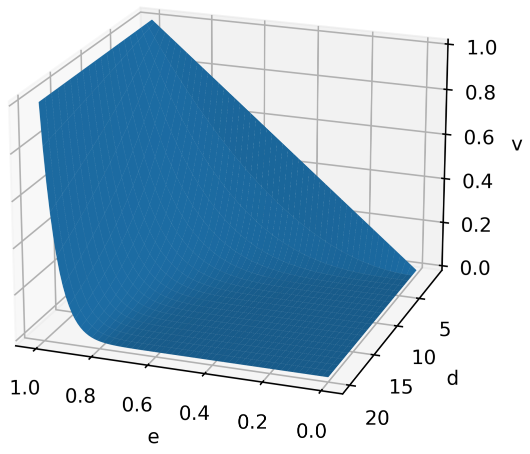


Figure 2.2: Ratio of smaller hypercube volume to unit hypercube volume. Here, v denotes the ratio of volume, e the edge length and d the dimension.

the complex volume can be done with a multiple integral where the combined integral bounds describe the complex volume. Assume that we want n evaluation points for each variable, resulting in n^d evenly distributed evaluation points. It is then suggested evaluating $M \ll n^d$ randomly sampled points and denote m as the number of points within the complex volume. One could estimate the complex volume by the fraction $\frac{m}{M}$ when M is sufficiently large due to the law of large numbers (Metropolis and Ulam, 1949, pp. 336-337). In this way, we can reduce the number of evaluation points and be able to estimate high-dimensional integrals with less impact from the curse of dimensionality because $M \ll n^d$. This suggests that one may use a sample of points as an alternative to n points in each dimension in order to estimate integrals.

Monte Carlo integration is a general method for numerical integration. We therefore present the Monte Carlo method in a general form and then briefly in the context of SSMs. Assume we are interested in calculating the expectation of the function $k(x)$ with respect to a generic density $p(x)$. We denote this expectation by $\mu_k = E_p\left[k(x)\right]$ and it is by the definition equal to the following integral

$$\mu_k = E_p\left[k(x)\right] = \int k(x)p(x)dx.$$

The expectation is a standard integral when we consider $k(x)p(x)$ as the integrand of the integral. In traditional numerical integration one often evaluates the integrand on a specified number of points. The objective is then to approximate the integral by summarising the area of different geometric shapes. Intuitively, more evaluation points produces more agile shapes that are able to follow the integrand closer and therefore increase the accuracy of the estimate. More points in each dimension quickly becomes intractable when the dimensionality is high. We briefly consider the strong law of large numbers, adapted from Givens and Hoeting (2013).

Theorem 2.3.1. *Define $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^i$ and assume that $n \to \infty$. In addition, assume that the variables x^i for $i = 1, \dots, n$ are i.i.d. from a distribution with expectation μ and $E\left[|x^i|\right] < \infty$. Then we have almost sure (a.s.) convergence,*

$$\hat{\mu} \overset{a.s.}{\to} \mu.$$

Monte Carlo integration

One often divide numerical integration into stochastic and deterministic methods. Deterministic methods often select uniform evaluation points along different dimensions and related to the curse of dimensionality, the number of total points in high-dimensional problems quickly becomes intractable. Stochastic methods simulate evaluation points from a distribution rather than uniformly along all dimensions (Gelman et al., 2013). This implies that the distribution from which we sample from also will have an impact on the effectiveness of the numerical integration. We define an estimator for the expectation denoted by $\hat{\mu}_k = \frac{1}{n}\sum_{i=1}^{n} k(x^i)$. Assume now that we have n variables x^i for $i = 1, \dots, n$ simulated i.i.d. from a generic distribution $p(x)$. We can then define the Monte Carlo method following Givens and Hoeting (2013).

Definition 2.3.1. *When $n \to \infty$ and the variables are i.i.d. from $p(x)$ we have*

$$\begin{aligned} & x^i \sim p(x) && i = 1, \dots, n \\ & \hat{\mu}_k = \frac{1}{n}\sum_{i=1}^{n} k(x^i) \to \int k(x)p(x)dx. \end{aligned}$$

The convergence of the Monte Carlo method is ensured by the strong law of large numbers from Theorem 2.3.1 (Givens and Hoeting, 2013; Robert and Casella, 2004). The Monte Carlo method in Definition 2.3.1 holds for both scalars and vectors because it holds for each variable of a vector (Chopin and Papaspiliopoulos, 2020, p. 81). In order to utilise the Monte Carlo method, we need a large i.i.d. sample of variables. If the distribution $p(x)$ however is intractable to simulate from this can be problematic. In addition, we are often interested in considering the variance of the estimator $\hat{\mu}_k$. Assuming now that $E_p[k(x)^2] < \infty$ we can then use the Monte Carlo method to estimate the variance of the estimator $\hat{\mu}_k$ as well, following Robert and Casella (2004). We then have an estimate of the variance given by

$$\begin{aligned} V\left[\hat{\mu}_k\right] = V\left[\frac{1}{n}\sum_{i=1}^{n} k(x^i)\right] &= \frac{1}{n}\int \left[\hat{\mu}_k - k(x)\right]^2 p(x)dx \\ &\approx \frac{1}{n^2}\sum_{i=1}^{n}\left[\hat{\mu}_k - k(x^i)\right]^2. \end{aligned}$$

An estimate of the integral is inserted in the last step by using the Monte Carlo method, adapted from (Robert and Casella, 2004, pp. 83-84). When the sample size $n \to \infty$, we can use the central limit theorem, see e.g. Devore and Berk (2012), to motivate the standardisation of $\hat{\mu}_k$ to a $\mathcal{N}(0, 1)$ as is also done in (Robert and Casella, 2004, p. 84). Using this standardisation, we are able to say something about the uncertainty of the Monte Carlo-estimates when the sample size n is large.

Monte Carlo integration in SSMs

In the context of SSMs we often consider situations where the target distribution is defined as the posterior distribution $p(x_{1:t}|y_{1:t})$. Assume now that we are interested in expectations with respect to this posterior distribution. We can then use the Monte Carlo-estimator in order to approximate the integral numerically in the same manner as the general case. Assume that we can simulate n sequences $x^i_{1:t}$ for $i = 1, \ldots, n$ directly from the posterior $p(x_{1:t}|y_{1:t})$. An estimate from using Monte Carlo integration is then given by

$$\begin{aligned} E_p\left[k(x_{1:t})\right] &= \int \ldots \int k(x_{1:t}) p(x_{1:t}|y_{1:t}) dx_{1:t} \\ &\approx \frac{1}{n} \sum_{i=1}^{n} k(x^i_{1:t}). \end{aligned} \tag{2.6}$$

In general it is however often difficult to simulate sequences directly from the posterior $p(x_{1:t}|y_{1:t})$. This is mainly because we generally do not have access to the distribution $p(x_{1:t}|y_{1:t})$ in closed form. Recall for example the joint distribution $p(x_{1:t}, y_{1:t})$ from equation (2.4). It is only in some special cases that we are able to evaluate $p(x_{1:t}|y_{1:t})$ in closed form. The normalising constant in this case corresponds to the marginal likelihood $p(y_{1:t})$ and in this case is generally not available in closed form. We will see that estimating the likelihood of the observations in a SSM is possible using different algorithms. Often, we are interested in estimates of the likelihood to use in further inference.

2.4 Weights

Using variables i.i.d. from a density p is a necessary condition in standard Monte Carlo integration defined in Definition 2.3.1. However, simulating directly from p might be intractable. The target distribution we consider in the SSM context is in general intractable to simulate directly from. The target distributions that we consider are often not available in closed form and in addition we can often only evaluate it up to proportionality. When simulating directly from the target distribution is intractable, alternative strategies to simulate the variables are needed. One can often distinguish between exact and approximate simulation. In exact simulation, the resulting sampling distribution is equal to the target distribution. This often requires the inverse cumulative distribution function (CDF) of the target density or a variant of rejection sampling. In approximate simulation, the sampling distribution approximates the target distribution. This approach is often based on importance sampling concepts and utilise weighting of variables (Givens and Hoeting, 2013). If the inverse CDF is not available, rejection sampling is one of the main exact simulation methods. Even though rejection sampling is an exact simulation method, we may choose importance sampling from within the approximate simulation methods instead. This is related to that the computational cost of simulating n variables with rejection sampling is not fixed. The computational cost of simulating n variables with e.g. importance sampling is on the other hand fixed. For an overview of simulation methods, see e.g. chapter 6 of Givens and Hoeting (2013).

Our focus will be on approximate simulation. Therefore we consider weighting of variables which is fundamental in importance sampling-based methods. The ratio $\frac{p(x)}{g(x)}$ is often referred to as an importance weight for x (Givens and Hoeting, 2013). We use p to denote the target density and g to denote the importance sampling function used to simulate variables. Combined, the simulated variables and the corresponding importance weights represents a weighted sample. The variables simulated from g are then combined with the importance weights in calculations. In the SSM context our target distribution is often the posterior of the latent variables $x_{1:t}$ conditioning on the observations $y_{1:t}$. The importance sampling function, g, is important when it comes to the efficiency of Monte Carlo-based methods. Because the concept of importance weights is used in all of the importance sampling-based methods, we consider it here. Note that we are considering a specific sequence $x^i_{1:t}$.

Definition 2.4.1. *Let $p(x_{1:t}|y_{1:t})$ denote the target and $g(x_{1:t}|y_{1:t})$ denote the importance sampling distribution. Then the importance weight and normalised weight are defined respectively*

$$\tilde{w}(x^i_{1:t}) = \frac{p(x^i_{1:t}|y_{1:t})}{g(x^i_{1:t}|y_{1:t})} \qquad w(x^i_{1:t}) = \frac{\tilde{w}(x^i_{1:t})}{\sum_{l=1}^n \tilde{w}(x^l_{1:t})}.$$

For notational simplicity $\tilde{w}^i_t = \tilde{w}(x^i_{1:t})$ and $w^i_t = w(x^i_{1:t})$ are often used.

It is also possible to use the unnormalised target distribution, $p(x_{1:t}, y_{1:t})$, to define the importance weights. This can be useful if we do not have access to the normalising constant (Naesseth, Lindsten, and Schön, 2019). In these cases we can only calculate the importance weights up to proportionality, but after normalisation the unknown normalising constants cancel. We follow the convention from Creal (2012) among many others and refer to the pair $(x^i_{1:t}, \tilde{w}^i_t)$ for $i = 1, \dots, n$ as the particles at iteration t. We can utilise the recursive structure from equation (2.3a) for the target and equation (2.5) for the importance sampling distribution. Then following the approach of Creal (2012) in order to rewrite the importance weights to a recursive form

$$\tilde{w}(x_{1:t}) = \frac{p(x_{1:t-1}|y_{1:t-1})\frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|y_{1:t-1})}}{g(x_{1:t-1}|y_{1:t-1})g(x_t|y_{1:t}, x_{1:t-1})} = \tilde{w}(x_{1:t-1})\frac{\frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|y_{1:t-1})}}{g(x_t|y_{1:t}, x_{1:t-1})}.$$

Note that the recursive structure of the importance weights requires us to evaluate the likelihood factor $p(y_t|y_{1:t-1})$. Generally, except for special cases, we cannot evaluate the likelihood factor exactly in the SSM context. We therefore calculate the importance weights up to proportionality because the likelihood factor will be constant when the observations $y_{1:t}$ are assumed to be fixed. This implies we can write the importance weights up to proportionality in a recursive form

$$\tilde{w}(x_{1:t}) \propto \tilde{w}(x_{1:t-1})\frac{p(y_t|x_t)p(x_t|x_{t-1})}{g(x_t|y_{1:t}, x_{1:t-1})}. \tag{2.7}$$

The recursive structure is useful because it allows for incremental updates of the importance weights. The incremental update at iteration t consists multiplying the importance weight from iteration $t-1$ with the fraction in equation (2.7). The alternative to an incremental update is to first simulate a new $x^i_{1:t}$ from g and then calculate $\tilde{w}(x^i_{1:t})$ by evaluating $p(x^i_{1:t}|y_{1:t})$ and $g(x^i_{1:t}|y_{1:t})$ for every t. This will often be computationally costly both because we need to simulate the entire $x^i_{1:t}$ for every t, but also because we need evaluate the entire expression for $\tilde{w}^i_t$ (Givens and Hoeting, 2013). The recursive structure of the importance weights is therefore an essential component in many of the sequential simulation methods we will consider. The structure in equation (2.7) makes it possible to see a recursive structure in the normalised weights akin to equation 1.6 in Doucet, Freitas, and Gordon (2001)

$$\begin{aligned}\frac{\tilde{w}(x_{1:t})}{\sum_{l=1}^n \tilde{w}(x^l_{1:t})} &\propto \frac{\tilde{w}(x_{1:t-1})}{\sum_{l=1}^n \tilde{w}(x^l_{1:t-1})}\frac{p(y_t|x_t)p(x_t|x_{t-1})}{g(x_t|y_{1:t},x_{1:t-1})}\\ w(x_{1:t}) &\propto w(x_{1:t-1})\frac{p(y_t|x_t)p(x_t|x_{t-1})}{g(x_t|y_{1:t},x_{1:t-1})}.\end{aligned} \tag{2.8}$$

Because simulating from approximations of the target distribution is especially important in Monte Carlo methods, we will summarise some of the most important methods in the following section.

2.5 Importance sampling concepts

As we will see, importance sampling is a fundamental part of many related methods. It is generally not possible to simulate directly from the target distributions that result from a SSM context. This is in part a result of the transition and observation densities that can be nonlinear and non-Gaussian. In order to use Monte Carlo integration we therefore depend on alternative simulation methods such as IS. In the following section we consider importance sampling and two closely related methods. These are sampling importance resampling (SIR) and sequential importance sampling (SIS). We focus mainly on importance sampling and consider briefly SIR and SIS. We state some relatively general assumptions related to the effectiveness of importance sampling function g adapted to our notation from (Givens and Hoeting, 2013, p. 181) and (Robert and Casella, 2004, p. 92).

Assumptions 2.5.1.

- *$supp(p) \subset supp(g)$*
- $\frac{p(x_{1:t}|y_{1:t})}{g(x_{1:t}|y_{1:t})} < \infty$ *and* $g(x_{1:t}|y_{1:t})$ *is more heavy-tailed than* $p(x_{1:t}|y_{1:t})$

Assume now that we are able to evaluate the importance weight $\tilde{w}_t$ exactly. Following Naesseth, Lindsten, and Schön (2019) we are then able to summarise IS as Algorithm 1. Note that if we are to normalise $\tilde{w}_t$ after simulating n sequence we can also use the unnormalised target distributions $p(x_{1:t}, y_{1:t})$ in the importance weights. We get the particles $(x^i_{1:t}, \tilde{w}^i_t)$ for $i = 1, \ldots, n$ as output from the algorithm. Consider then the scenario where we want to estimate the integral in equation (2.6) by using Monte Carlo integration. Because simulating directly from the target distribution is generally intractable,

Algorithm 1 Importance sampling

```
1: for $i \in (1, \dots, n)$ do
2:     $x^i_{1:t} \sim g(x_{1:t}|y_{1:t})$
3:     $\tilde{w}^i_t = \frac{p(x^i_{1:t}|y_{1:t})}{g(x^i_{1:t}|y_{1:t})}$
4: end for
```

we aim to reformulate the integral so it corresponds to an expectation with respect to the importance sampling distribution. We are able to express the expectation with respect to the importance sampling distribution by multiplying the function k inside the expectation with importance weights or normalised weights (Givens and Hoeting, 2013, pp. 180-181). Assume that the importance sampling distribution $g(x_{1:t}|y_{1:t}) > 0$ whenever $k(x_{1:t})p(x_{1:t}|y_{1:t}) > 0$ (Chopin and Papaspiliopoulos, 2020). We can then we can always reformulate the integrand in the expectation. By starting from the original integral on the left-hand side, we get

$$\mu_k = E_p\left[k(x_{1:t})\right] = \int k(x_{1:t})\frac{p(x_{1:t}|y_{1:t})}{g(x_{1:t}|y_{1:t})}g(x_{1:t}|y_{1:t})dx_{1:t} \tag{2.9a}$$

$$= E_g\left[k(x_{1:t})\frac{p(x_{1:t}|y_{1:t})}{g(x_{1:t}|y_{1:t})}\right] \tag{2.9b}$$

This equality by reformulation is often referred to as the IS fundamental identity (Robert and Casella, 2004, p. 92). We can then use the Monte Carlo method for the integral in equation (2.9a). We insert the expression for the importance weight in the integral. Then we can use n simulated sequences of variables from g combined with Monte Carlo integration to get

$$\int k(x_{1:t})\tilde{w}_t g(x_{1:t}|y_{1:t})dx_{1:t} \approx \frac{1}{n}\sum_{i=1}^{n} k(x^i_{1:t})\tilde{w}^i_t = \tilde{\mu}^{\text{IS}}_k$$

This is useful because we now only need to sample from the importance sampling function g and then weight the sampled value using the importance weight $\tilde{w}^i_t$. We will often refer to $\tilde{\mu}^{\text{IS}}_k$ as the unnormalised IS estimator.

We can follow the derivation in (Chopin and Papaspiliopoulos, 2020, p. 86) in order to see another valid reformulation of the original integral in equation (2.6) called normalised IS. Recall that we are using the target distribution $p(x_{1:t}|y_{1:t}) = \frac{p(x_{1:t}, y_{1:t})}{Z_t}$ where Z_t is the normalising constant at iteration t. In general, we are not able to evaluate the normalising constant Z_t and consequently we can only evaluate the importance weight $\tilde{w}_t$ up to proportionality. We can use the importance weight $\tilde{w}_t = \frac{p(x_{1:t}, y_{1:t})}{g(x_{1:t}|y_{1:t})}$ where the numerator is the unnormalised target distribution. We then have the following, see extended calculations in Appendix A.1,

$$E_p\left[k(x_{1:t})\right] = \frac{E_g\left[k(x_{1:t})\frac{p(x_{1:t}, y_{1:t})}{g(x_{1:t}|y_{1:t})}\right]}{E_g\left[\frac{p(x_{1:t}, y_{1:t})}{g(x_{1:t}|y_{1:t})}\right]}.$$

We see that we can approximate both the numerator and the denominator by using the Monte Carlo method. We denote the estimator for this expectation by $\hat{\mu}_k^{\text{IS}}$. We then have

$$\hat{\mu}_k^{\text{IS}} = \frac{\frac{1}{n}\sum_{i=1}^{n} k(x_{1:t}^i)\frac{p(x_{1:t}^i, y_{1:t})}{g(x_{1:t}^i|y_{1:t})}}{\frac{1}{n}\sum_{i=1}^{n} \frac{p(x_{1:t}^i, y_{1:t})}{g(x_{1:t}^i|y_{1:t})}} = \sum_{i=1}^{n} k(x_{1:t}^i)w_t^i$$

by using the definition of normalised weights, Definition 2.4.1. As this estimator is using the normalised weights, we refer to it as the normalised estimator. We now have the unnormalised estimator $\tilde{\mu}_k^{\text{IS}}$ which uses the importance weights directly. In addition we have the normalised estimator $\hat{\mu}_k^{\text{IS}}$ where it suffices to know the target distribution up to proportionality. Now we consider some properties related to the IS estimators before considering SIR and SIS.

Properties of the estimators

We now want to consider the expectation and variance of the two IS estimators. For the normalised IS estimator we need to use Taylor expansions to approximate the estimator and subsequently consider the expectation of the approximation. The estimate $\hat{\mu}_k^{\text{IS}}$ obtained with normalised IS converges a.s. to $E_p\left[k(x_{1:t})\right]$ when $n \to \infty$ (Naesseth, Lindsten, and Schön, 2019, p. 16). For the properties of the estimators we follow Givens and Hoeting (2013). When considering properties of the estimators, we use the simplified notation $w = \frac{p(x_{1:t}|y_{1:t})}{g(x_{1:t}|y_{1:t})}$ and $u = wk(x_{1:t})$. Their means are denoted by $\bar{w}$ and $\bar{u}$ respectively. It can be shown that we can approximate $\hat{\mu}_k^{\text{IS}}$ with a multivariate Taylor expansion. First we note that $\hat{\mu}_k^{\text{IS}} = \frac{\bar{u}}{\bar{w}}$, then we have the first order and second order Taylor approximations around the expansion point $\theta_0 = (\mu_k, 1)$. That is $\hat{\mu}_k^{\text{T1}}$ and $\hat{\mu}_k^{\text{T2}}$ respectively. We are then able to approximate expectation and variance of the estimator from the Taylor approximation of the estimator. First we consider the expectation of the estimators. We then have

$$\begin{aligned} E_g\left[\tilde{\mu}_k^{\text{IS}}\right] &= \mu_k \\ E_g\left[\hat{\mu}_k^{\text{IS}}\right] &\approx E_g\left[\hat{\mu}_k^{\text{T2}}\right] = \mu_k - \frac{1}{n}\text{Cov}(u,w) + \mu_k\frac{1}{n}V_g\left[w\right]. \end{aligned}$$

Then we consider the variance of the estimators,

$$\begin{aligned} V_g\left[\tilde{\mu}_k^{\text{IS}}\right] &= \frac{1}{n}V_g\left[u\right] \\ V_g\left[\hat{\mu}_k^{\text{IS}}\right] &\approx V_g\left[\hat{\mu}_k^{\text{T1}}\right] = \frac{1}{n}V_g\left[u\right] - 2\mu_k\frac{1}{n}\text{Cov}\left[u,w\right] + \frac{1}{n}\mu_k^2 V_g\left[w\right]. \end{aligned}$$

From the expectation of the normalised estimator $\hat{\mu}_k^{\text{IS}}$ we see that when $n \to \infty$, the expectation approaches the true expectation μ_k. Comparing the variances of the two estimators we see that the variance of the normalised IS estimator can be expressed by using the variance of the normalised IS estimator. That is,

$$V_g\left[\hat{\mu}_k^{\text{IS}}\right] \approx V_g\left[\tilde{\mu}_k^{\text{IS}}\right] - 2\mu_k\frac{1}{n}\text{Cov}\left[u,w\right] + \frac{1}{n}\mu_k^2 V_g\left[w\right].$$

This highlights the relationship between the estimators and that it is possible for the normalised IS estimator to have lower variance than the unnormalised IS estimator. We can compare the estimators $\hat{\mu}_k^{\text{IS}}$ and $\tilde{\mu}_k^{\text{IS}}$ when it comes to mean squared error (MSE). It is possible to show that, see Appendix A.1, that we should choose $\hat{\mu}_k^{\mathrm{IS}}$ over $\tilde{\mu}_k^{\mathrm{IS}}$ when the following holds

$$\mathrm{Corr}\left[u, w\right] > \frac{\mathrm{CV}\left[w\right]}{2\mathrm{CV}\left[u\right]}.$$

The expression is equivalent to Givens and Hoeting (2013, eq. 6.45). We can reformulate the unnormalised estimator as $\tilde{\mu}_k^{\mathrm{IS}} = \frac{1}{n}\sum_{i=1}^{n} u$ and the normalising term as $\frac{1}{n}\sum_{i=1}^{n} w$. Then we see that the normalised estimator can be written as $\hat{\mu}_k^{\mathrm{IS}} = (\frac{1}{n}\sum_{i=1}^{n} u)/(\frac{1}{n}\sum_{i=1}^{n} w)$. When the terms within the sums in the numerator and denominator have high correlation, we should use the normalised estimator (Givens and Hoeting, 2013, p. 183).

Sampling importance resampling

Resampling can be regarded as one of the key extensions within the IS framework. Resampling allows us to move from n weighted sequences $(x_{1:t}^i, \tilde{w}_t^i)$ for $i = 1, \ldots, n$ to a sample of size n_r which is unweighted. This is achieved through resampling from the original weighted sample. Assume we have simulated $x_{1:t}^i$ for $i = 1, \ldots, n$ i.i.d. from an importance sampling function g. Assume further that our target distribution is $p(x_{1:t}|y_{1:t})$ and that we have the normalised weights w_t^i for $i = 1, \ldots, n$ from Algorithm 1. We can then resample n_r sequences from $x_{1:t}^i$ for $i = 1, \ldots, n$ with replacement using w_t as a discrete probability distribution. We then then have the n_r resampled sequences $z_{1:t}^i$ for $i = 1, \ldots, n_r$ which is approximate i.i.d. from the target distribution as is discussed in Givens and Hoeting (2013). It can then be shown that when $n \to \infty$ and $\frac{n_r}{n} \to 0$ the distribution of $z_{1:t}^i$ for $i = 1, \ldots, n_r$ converge to the target distribution, see section 6.3 in Givens and Hoeting (2013) for the derivations and proof.

Because the n_r variables are approximately i.i.d. from the target distribution, we can use the Monte Carlo method from Definition 2.3.1 directly to get the estimator $\hat{\mu}_k^{\mathrm{SIR}}$. This follows because the resampled sample is now approximately i.i.d. from the target distribution. It can be shown, see Givens and Hoeting (2013, p. 183), that the expectation of $\hat{\mu}_k^{\mathrm{SIR}}$ is the same as for normalised estimator $\hat{\mu}_k^{\mathrm{IS}}$, but the variance is equal or higher. This is an incentive to use the normalised estimator $\hat{\mu}_k^{\mathrm{IS}}$ because even if we do not need the weights directly in $\hat{\mu}_k^{\mathrm{SIR}}$ we need to calculate them in order to resample. This would imply that we already have the weights available. Further, we can then use $\hat{\mu}_k^{\mathrm{IS}}$ to potentially reduce variance. An advantage with the SIR estimator is in a scenario where we have limited memory available. We only need to store the n_r resampled sequences as opposed to all the n sequences where $n_r < n$. In addition we can discard the weights after resampling and keep only the sequences as opposed to the sequences and their weights. Other arguments for using the SIR estimator include scenarios where the function k might be computationally costly to evaluate and scenarios where we need approximately i.i.d. sequences for other calculations (Skare, Bølviken, and Holden, 2003).

Sequential importance sampling

Using IS we need to evaluate the entire target distribution $p(x_{1:t}|y_{1:t})$ at every iteration t. When t increases, we simulate increasingly larger $x_{1:t}$ from g at every iteration t. We also need to evaluate increasingly more complex expressions to calculate the weights. The SIS introduce a sequential aspect to IS by utilising conditional distributions in combination with a recursive structure of both the target distribution and the importance sampling function. We can use the recursive structure of the target distribution in equation (2.3a) and the recursive importance sampling function g in equation (2.5) when available. This implies that we only need to simulate x_t from the conditional distributions in order to calculate the weights at t. This is because we can use the weights from $t-1$ and multiply in the weight update. We can start with the recursive structure of the importance weights defined in equation (2.7). We then have that

$$\tilde{w}_t^i \propto \tilde{w}_{t-1}^i u_t^i \qquad\qquad u_t^i = \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{g(x_t^i|y_{1:t}, x_{1:t-1}^i)} \tag{2.10}$$

which is adapted from (Givens and Hoeting, 2013, pp. 170-171). We denote u_t^i as the weight update function, similar to the incremental update factor in (Creal, 2012, p. 252). In general we can update the previous weight in order to get the current weight in a sequential manner by using a weight update function. The expression for the for the importance weight can also be written in a slightly different form where we only need to evaluate the target distribution and the conditional part of the importance sampling distribution,

$$\tilde{w}_t^i \propto \tilde{w}_{t-1}^i \frac{p(x_{1:t}^i|y_{1:t})}{p(x_{1:t-1}^i|y_{1:t-1})g(x_t^i|y_{1:t}, x_{1:t-1}^i)} = \tilde{w}_{t-1}^i u_t^i, \tag{2.11}$$

adapted from (Naesseth, Lindsten, and Schön, 2019, p. 17). Note that both these recursive update expressions for the importance weights also can be extended to the normalised weights in equation (2.8). Extended calculations for the weight update expression are in Appendix A.1. This implies that it is possible to update the weight for sequence i sequentially by sampling $x_t^i \sim g(x_t|y_{1:t}, x_{1:t-1}^i)$ at iteration t. We use this notation for the weight updates and restate the algorithm from (Naesseth, Lindsten, and Schön, 2019, p. 18). We assume that $\tilde{w}_0 = 1$, $p(x_{1:0}|y_{1:0}) = 1$ and $g(x_1|y_1, x_{1:0}) = g(x_1|y_1)$ for the first iteration of the algorithm. We can then state SIS as Algorithm 2 in the context of SSMs. Similar to importance sampling we get the particles $(x_{1:t}^i, \tilde{w}_t^i)$ for $i = 1, \ldots, n$

Algorithm 2 Sequential importance sampling

```
1: for $t \in (1, \ldots, T)$ do
2:     for $i \in (1, \ldots, n)$ do
3:         $x_t^i \sim g(x_t|y_{1:t}, x_{1:t-1}^i)$
4:         $\tilde{w}_t^i \propto \tilde{w}_{t-1}^i u_t^i$
5:     end for
6:     $w_t^i = \frac{\tilde{w}_t^i}{\sum_{l=1}^n \tilde{w}_t^l}$ for $i = 1, \ldots, n$
7: end for
```

as output. From line 3 in Algorithm 2 however, we are now able to sample the variables x_t for $t = 1, \ldots, T$ sequentially in contrast to regular IS. This implies that we can sample one variable, of the same dimension, at a time compared to IS where the length of $x_{1:t}$ increase with each t. We then update the importance weights with the weight update u_t^i. The weight update u_t^i is defined in equation

(2.10) and from its structure we can consider its influence on the importance weight. We see that a simulated variable x_t^i can reduce the importance weight of sequence i if it is inconsistent with the previous part of the sequence, $x_{1:t-1}^i$. It can also reduce the importance weight if it is inconsistent with the observation y_t. With multiplicative updating, a simulated x_t^i which is inconsistent with $x_{1:t-1}^i$ and y_t can reduce the importance weight for the entire sequence relative to the other sequences. When normalising the weights, the sequences which contain an inconsistent variable will have low normalised weights. When t increase, many of the n sequences will therefore have low normalised weights. This implies that a small number of sequences will have normalised weights close to 1. This phenomenon is referred to as weight degeneracy (Givens and Hoeting, 2013, p. 171).

2.6 Sequential Monte Carlo

Selecting an importance sampling distribution which is close to to the target distribution increase the efficiency of importance sampling-methods. Because we are not simulating directly from the target distribution our objective is to find the particles $(x_{1:t}^i, \tilde{w}_t^i)$ for $i = 1, \ldots, n$ at iteration t. The weights should optimally adjust for the fact that we are simulating from the importance sampling distribution $g(x_{1:t}|y_{1:t})$ and not the target distribution $p(x_{1:t}|y_{1:t})$. In SIS we saw that the sequential weight structure resulted in weight degeneracy when t increased. In this section we will consider the general SMC algorithm. We also briefly consider the choice of importance sampling function g and reduction of weight degeneracy.

Algorithm

In this section we will consider the standard SMC algorithm. We follow the setups presented in Creal (2012) and Naesseth, Lindsten, and Schön (2019). In the following presentation, we resample at every iteration t. We handle weight degeneracy by resampling the sequences according to a specific resampling scheme. The addition of a resampling step implies that sequences $x_{1:t}$ having high normalised weights w_t are duplicated while some of the sequences with low normalised weights are not resampled at all (Chopin and Papaspiliopoulos, 2020, p. 114). One possible resampling scheme is called multinomial resampling and here the sequences $x_{1:t}^i$ for $i = 1, \ldots, n$ are sampled independently with replacement using the normalised weights as a discrete probability distribution (Naesseth, Lindsten, and Schön, 2019). Here we select the importance sampling distribution with the recursive structure from equation (2.5). We then have the incremental importance sampling distributions given by $g(x_t|y_{1:t}, x_{1:t-1})$ for $t \geq 2$ and $g(x_1|y_1)$ for $t = 1$. The algorithm is an adaption of algorithm 3 in Naesseth, Lindsten, and Schön (2019). Note that we resample the entire sequence up to $t-1$ at line 7 from the discrete approximation $\hat{p}(x_{1:t-1}|y_{1:t-1})$, then the resampled set is used when sampling from the importance sampling distribution at line 8. There are several methods of resampling, but we will unless otherwise stated use multinomial resampling, see e.g. Chopin and Papaspiliopoulos (2020) for a review of different resampling schemes. We also note that the importance sampling distribution $g(x_t|y_{1:t}, x_{1:t-1})$ is selected by

Algorithm 3 Sequential Monte Carlo

```
1: for $t \in (1, \dots, T)$ do
2:     for $i \in (1, \dots, n)$ do
3:         if $t = 1$ then
4:             $x_1^i \sim g(x_1|y_1)$
5:             $\tilde{w}_1^i = \frac{p(x_1^i, y_1)}{g(x_1^i|y_1)}$
6:         else
7:             $x_{1:t-1}^i \sim \hat{p}(x_{1:t-1}|y_{1:t-1})$
8:             $x_t^i \sim g(x_t|y_{1:t}, x_{1:t-1}^i)$
9:             $\tilde{w}_t^i = \frac{p(x_{1:t}^i, y_{1:t})}{p(x_{1:t-1}^i, y_{1:t-1}) g(x_t^i|y_{1:t}, x_{1:t-1}^i)}$
10:        end if
11:    end for
12:    $w_t^i = \frac{\tilde{w}_t^i}{\sum_{l=1}^n \tilde{w}_t^l}$ for $i = 1, \dots, n$
13:    $\hat{p}(x_{1:t}|y_{1:t}) = \sum_{l=1}^n w_t^l \delta_{x_{1:t}^l}(x_{1:t})$
14: end for
```

the user and therefore it do not necessarily depend on the observations $y_{1:t}$ or all the previous simulated variables $x_{1:t-1}$.

Importance sampling distribution

The effectiveness of Algorithm 3 is dependent on the importance sampling distribution. The importance sampling distribution is important primarily in order to simulate variables that are approximately distributed according to the target distribution. Optimally, we would have an importance sampling distribution that also is easy to simulate from. We ideally want to select the importance sampling function such that the variance of the weights conditional on the previous variables at $t-1$ is 0. This can be seen as the optimal importance sampling function when the objective is to minimise the variance of the weight update u_t in equation (2.10) conditional on the previous weights through $x_{1:t-1}$ and $y_{1:t}$.

Proposition 2.6.1. *As proposition 2 in (Doucet, Godsill, and Andrieu, 2000, p. 199) we have that the importance sampling function*

$$g(x_t|y_{1:t}, x_{1:t-1}^i) = p(x_t|y_t, x_{t-1}^i)$$

is optimal w.r.t. reducing variance of the importance weight of x_t^i when conditioning on $y_{1:t}$ and $x_{1:t-1}^i$.

The proof can be found in Doucet, Godsill, and Andrieu (2000). Note that the importance sampling function g in Proposition 2.6.1 is optimal in the scenarios where we only can use variables available from the previous and current iteration t. The setting where we use only variables available at iteration s where $s \leq t$ is often referred to as an online setting (Naesseth, Lindsten, and Schön, 2019). We can also consider the importance weight $\tilde{w}_t^i$ from the SMC algorithm when using a straightforward reformulation of the optimal importance sampling function. We then have

$$\tilde{w}_t^i = p(y_t|x_{t-1}^i).$$

We note that the expressions for both the optimal $g(x_t|y_{1:t}, x^i_{1:t-1})$ and the weight update function $p(y_t|x^i_{t-1})$ in closed forms are rarely available. However, we can often utilise approximations of the optimal importance sampling distribution with a Gaussian by using local linearisation. This and related approaches are reviewed in part 2 of Doucet, Godsill, and Andrieu (2000).

Weights

The weights play a central role in the importance sampling-framework and as we saw with SIS there is a problem with weight degeneracy when t increases. Related to weight degeneracy is the variance of importance weights as t increase. It is also possible to show that with the sequential weight structure the variance of the importance weights increase with t. That is,

$$V\left[\tilde{w}_t\right] \leq V\left[\tilde{w}_{t+1}\right].$$

This is stated in Creal (2012) and proved in Kong, J. S. Liu, and Wong (1994). It also follows that because the variance of the importance weights increase, a few or one of the importance weights will be large. When normalising the importance weights, we will have a few or one normalised weights close to 1. Obviously we want to avoid or at least decrease weight degeneracy. We first want to estimate the current weight degeneracy and then we consider strategies to decrease weight degeneracy.

Estimating weight degeneracy

We can estimate weight degeneracy through the coefficient of variation (CV) which measures the standard deviation divided by the mean of an estimator (Chopin and Papaspiliopoulos, 2020, p. 93). We then start by finding an expression for the squared CV of the normalised weights. It can be shown, see e.g. Givens and Hoeting (2013), that the squared CV of the normalised weights can be expressed through an expectation

$$\mathrm{CV}^2\left[w_t\right] = E\left[(nw_t - 1)^2\right].$$

By estimating $\mathrm{CV}^2\left[w_t\right]$ we can define the effective sample size (ESS) approximately as variables with nonzero weights and then reformulate so to get an expression for ESS that we can estimate. We use the ESS to measure the efficiency of sampling. As discussed in Chopin and Papaspiliopoulos (2020) we have that $1 \leq \mathrm{ESS} \leq n$. ESS can then be expressed as:

$$\mathrm{ESS} = \frac{1}{\sum_{i=1}^{n}(w^i_t)^2}. \tag{2.12}$$

In an optimal situation we have that the importance weights $\tilde{w}^i_t = 1$ for $i = 1, \ldots, n$. This consequently implies that we have $w_t = \frac{1}{n}$ for $i = 1, \ldots, n$ for the normalised weights. In this situation we also see that $\mathrm{CV}^2\left[w_t\right] = 0$ which is the minimum point for the squared coefficient of variation.

Decreasing weight degeneracy

In order to reduce weight degeneracy there are several strategies available. Different frameworks of resampling are common strategies in order to reduce weight degeneracy, see e.g. Creal (2012) and Givens and Hoeting (2013). A popular setup is the adaptive resampling setup where we resample when ESS $< \alpha \cdot n$ where $\alpha \in [0, 1]$ (Creal, 2012). We then often refer to α as a resampling threshold.

The resampling scheme can be multinomial with replacement where the probability for selecting each sequence is given by the normalised weights w_t (Givens and Hoeting, 2013). An important aspect of resampling in order to decrease weight degeneracy is that resampling duplicates entire sequences $x_{1:t}$ so when t increases we will have few unique subsequences $x_{1:s}$ where $s < t$. This implies that our estimate of $p(x_{1:s}|y_{1:s})$ can be based on few unique sequences. As is discussed in e.g. Creal (2012) and Naesseth, Lindsten, and Schön (2019) resampling increase short-term variance by duplication, but decrease weight degeneracy at a later point.

This is based on that after resampling we continue resampling from sequences that are more likely based on the normalised weights. The multinomial resampling scheme is not the only resampling scheme nor the optimal in terms of Monte Carlo variation. Other possible resampling schemes include the residual, systematic or stratified which are discussed in chapter 9 Chopin and Papaspiliopoulos (2020). We will however often employ an adaptive resampling setup with multinomial resampling by using $\alpha = 0.5$ as a resampling threshold, that is we resample if ESS $< \alpha \cdot n$.

CHAPTER 3

Inference in state space models

The concepts in Chapter 2 based on importance sampling are general methods not limited to the SSM context. In this chapter we will consider some aspects of inference in state space models. Our main focus will here be on filtering and likelihood estimation. We will also see strategies to approximate filtering and smoothing recursions. These recursions are only available in tractable forms for two special cases: discrete models with a finite state space and linear Gaussian models, see chapter 5 of Cappé, Moulines, and Rydén (2005). In general cases we need approximations of high-dimensional integrals obtained using e.g. Monte Carlo integration. In addition, we consider particle filter (PF) algorithms and how one can obtain likelihood estimates from particle filter algorithms.

3.1 Motivation

In general, we are often interested in inference related to the posterior distribution $p(x_{1:t}|y_{1:t})$ or a marginal of this distribution (Doucet, Freitas, and Gordon, 2001). In addition, we often want unbiased estimates of the marginal likelihood denoted by $p(y_{1:t})$ at each t. Recall that we have omitted the dependence on the parameters θ when these are assumed to be known. Unbiased estimates of the likelihood $p(y_{1:t}|\theta)$ is specially important when it comes to parameter estimation as it allows us to compare different parameters θ. As discussed in Kantas et al. (2015), the marginal likelihood is generally not available in closed for general SSMs. Some of the inferential aspects in SSMs have received special names and we consider a subset of the more comprehensive list in Chopin and Papaspiliopoulos (2020).

1. Filtering: we follow Kantas et al. (2015) and often refer to $p(x_{1:t}|y_{1:t})$ as the filtering distribution. We then refer to $p(x_t|y_{1:t})$ as the marginal filtering distribution.

2. Smoothing: we will refer to $p(x_{1:T}|y_{1:T})$ as the joint smoothing distribution and $p(x_t|y_{1:T})$ as the marginal smoothing distribution.

3. Likelihood: the marginal likelihood $p(y_{1:t})$ can be estimated through what we will refer to as likelihood factors, denoted by $p(y_t|y_{1:t-1})$.

In the two special cases of finite discrete SSMs and linear Gaussian SSMs we have tractable analytical expressions which we are able to sum and integrate respectively. These special cases are covered in e.g. chapter 5 of Cappé, Moulines, and Rydén (2005). This implies that for the linear Gaussian models, we can find analytical expressions for e.g. the likelihood. In general models however, we need to simulate variables by exact or approximate simulation methods and then use Monte Carlo methods to approximate the intractable integrals.

Approximating distributions

We will generally need discrete approximations of the distributions of interest because these distributions are generally intractable. We first follow Doucet, Freitas, and Gordon (2001) and consider the case where we assume that we can simulate n sequences i.i.d. from $p(x_{1:t}|y_{1:t})$. We then have the discrete approximation

$$\hat{p}(x_{1:t}|y_{1:t}) = \sum_{i=1}^{n} \frac{1}{n} \delta_{x_{1:t}^i}(x_{1:t})$$

where $\delta_{x_{1:t}^i}(x_{1:t})$ is a Dirac measure. This is however an unlikely scenario when considering that the general SSM can be nonlinear and non-Gaussian. We can instead formulate a similar approximation by using particles $(x_{1:t}^i, \tilde{w}_t^i), i = 1, \ldots, n$ from an IS-based algorithm. Note that the sequences $x_{1:t}^i$ for $i = 1, \ldots, n$ now are simulated from the importance sampling distribution instead of directly from the target distribution $p(x_{1:t}|y_{1:t})$. We can then view the importance weights $\tilde{w}_t^i$ for $i = 1, \ldots, n$ as adjusting for the fact that we are not simulating from the target distribution (Naesseth, Lindsten, and Schön, 2019). When we use these particles in order to approximate the target distribution $p(x_{1:t}|y_{1:t})$ we need to normalise the weights according to Definition 2.4.1. We then get the normalised weights $w(x_{1:t}^i)$ for $i = 1, \ldots, n$ which we combine with the Dirac measure $\delta_{x_{1:t}^i}(x_{1:t})$ to approximate the target distribution. The target distribution can then be approximated by the discrete approximation

$$\hat{p}(x_{1:t}|y_{1:t}) = \sum_{i=1}^{n} w(x_{1:t}^i) \delta_{x_{1:t}^i}(x_{1:t}). \tag{3.1}$$

This is the traditional discrete approximation of the target distribution found in e.g. Chopin and Papaspiliopoulos (2020), Creal (2012), and Naesseth, Lindsten, and Schön (2019). The discrete approximation of the target distribution will be important when considering approximations of the filtering and smoothing recursions.

Approximating integrals

Similarly to having discrete approximations in order to estimate distributions, we can utilise the particles in order to estimate integrals. This again allows for estimating different quantities of interest within the Monte Carlo integration framework (Creal, 2012). Consider calculating the expectation of the function $k(x_{1:t})$ with respect to the target distribution $p(x_{1:t}|y_{1:t})$ as the objective of inference. Assume that the sequences $x_{1:t}^i$ for $i = 1, \ldots, n$ are simulated from the importance sampling distribution $g(x_{1:t}|y_{1:t})$ and let $w(x_{1:t}^i)$ denote the normalised weights for $x^i_{1:t}$. We then have the following setup by using Monte Carlo integration

$$E_p\left[k(x_{1:t})\right] = \int \dots \int k(x_{1:t})p(x_{1:t}|y_{1:t})dx_{1:t} \approx \sum_{i=1}^{n} k(x^i_{1:t})w(x^i_{1:t}).$$

Note that even though we are simulating from the importance sampling distribution $g(x_{1:t}|y_{1:t})$ which most likely is easier to simulate from, we are still simulating the entire sequence $x_{1:t}$ at every t (Creal, 2012). This naturally becomes computationally costly as t increase and in addition it implies that we need all observations before simulating any x_t (Doucet, Freitas, and Gordon, 2001). Similarly, we can use Monte Carlo integration to estimate the expectation of the function $k(x_{1:t}) = x_s$ where $s \leq t$. Selecting this specific function k gives an estimate of the first moment with respect to the marginal target distribution. We then have the approximation

$$E_p\left[x_s\right] = \int x_s p(x_s|y_{1:s})dx_s \approx \sum_{i=1}^{n} x^i_s w(x^i_{1:s}). \tag{3.2}$$

The particles at iteration s can be used to approximate the marginal target distribution $p(x_s|y_{1:s})$ through a weighted measure (Doucet, Freitas, and Gordon, 2001, p. 12). That is, the variables x^i_s and the normalised weights w^i_s for $i = 1, \dots, n$. The estimates resulting from the Monte Carlo integration with particles from many particle filters can be proved to be consistent. That is, for Equation (3.2) where $k(x_{1:t}) = x_s$, we would have that

$$\sum_{i=1}^{n} x^i_s w(x^i_{1:s}) \overset{a.s.}{\to} E_p\left[x_s\right]$$

as in (Creal, 2012, eq. 37) when $n \to \infty$, under some assumptions on w and the function k. Proofs are found in Chopin (2004). In addition there are asymptotic variance expressions available for some particle filters where one can show that the variance is finite and bounded in time, but generally not in the dimensionality of x (Creal, 2012).

The particles allow for inference about $x_{1:t}$ or some subset, usually x_t at iteration t. As t increase we will have an approximation of the marginal filtering distribution $p(x_t|y_{1:t})$ at every iteration by using the particles. We will refer to this as an online setting. In an online setting we are not bounded to wait until we have all observations $y_{1:T}$ in order to make inference about x_t. The necessity for real-time inference about parameters, θ, may also be a motivation for algorithms that are suited for an online setting. See e.g. section 5.2 and 6.2 in Kantas et al. (2015).

We see that even though we are able to make online inference at every iteration t, this can be computationally costly if we are simulating the entire $x_{1:t}$ again for every iteration t. We therefore want to utilise the SIS approach by simulating only the last x_t and append this to the existing sequence $x_{1:t-1}$ by using equation (2.5) as an importance sampling distribution. As we saw with the SIS approach we then needed to evaluate only the weight update functions in order to update the importance weights.

3.2 Filtering recursions

We have often used the posterior $p(x_{1:t}|y_{1:t})$ as the target distribution. We have also seen the recursive structure of this distribution which allow for sequential approximation. The marginal distribution, $p(x_t|y_{1:t})$, is the main objective of interest when it comes to filtering. We will refer to this as the marginal filtering distribution. We will see that this distribution also can be decomposed into a recursive structure. In this section we will focus on the marginal filtering recursions and consequently the distribution $p(x_t|y_{1:t})$. We will see that the filtering recursion can be used in order to update the filtering distribution in an online setting. This can be important in real-time applications where estimates are needed dynamically as the observations are available. We follow the presentation from Creal (2012) when it comes to the predict and update step below. Assume now that at iteration $t-1$ we have observations, $y_{1:t-1}$, available. Then we can formulate the predictive distribution through the following marginalisation

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}.$$

At iteration t, we assume that observation y_t is available. We can now formulate the filtering distribution at t by using the above predictive distribution which itself contains the marginal filtering distribution at $t-1$. This recursive structure of the filtering distributions allows for sequential updates. We then get by straightforward reformulations

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

Here, the CI properties of the SSM is used in the last equality. We are now able to summarise the two equations above referred to as the predict- and update step respectively by the following equations

$$\text{predict step: } p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{3.3a}$$

$$\text{update step: } p(x_t|y_{1:t}) = \frac{p(y_t|x_t)}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}p(x_t|y_{1:t-1}). \tag{3.3b}$$

We see that inserting equation (3.3a) into equation (3.3b) yields the recursion.

Approximations

In general, we want to consider the discrete approximation $\hat{p}(x_t|y_{1:t})$. We assume that the approximations at iteration $t-1$, that is $\hat{p}(x_{1:t-1}|y_{1:t-1})$ and consequently $\hat{p}(x_{t-1}|y_{1:t-1})$ are available. By using equation (3.1) we can see that the joint posterior distribution, $p(x_{1:t}|y_{1:t})$, and the marginal posterior distribution, $p(x_t|y_{1:t})$, can be approximated by mixtures of Dirac measures. Hence we need to update the set of particles from iteration $t-1$ to a new set of particles at iteration t. Conceptually, this can be done through obtaining the most recent x_t^i and updating the normalised weights so we get w_t^i for $i = 1, \ldots, n$. Once the updated particles are available, we can construct the approximations

$$\hat{p}(x_{1:t}|y_{1:t}) = \sum_{i=1}^{n} w_t^i \delta_{x_{1:t}^i}(x_{1:t}) \qquad \hat{p}(x_t|y_{1:t}) = \sum_{i=1}^{n} w_t^i \delta_{x_t^i}(x_t).$$

Here we consider approximations of both the joint and the marginal filtering distributions. By considering only x_t and the related weights w_t we are focusing on the marginal filtering distribution, see e.g. section 7.3 of Cappé, Moulines, and Rydén (2005). We now look at how we can approximate the predict and update step for the marginal filtering distribution $p(x_t|y_{1:t})$. By using these recursive approximations, we can transition from the marginal posterior distribution $\hat{p}(x_{t-1}|y_{1:t-1})$ to the marginal posterior distribution $\hat{p}(x_t|y_{1:t})$. For the predict step in equation (3.3a) we need to approximate $p(x_{t-1}|y_{1:t-1})$ by $\hat{p}(x_{t-1}|y_{1:t-1})$, while the transition density $p(x_t|x_{t-1})$ is given by the model. Assume that we have the approximation of the filtering distribution at the iteration $t-1$, that is we have the weighted measure. We start by approximating the update step

$$\begin{aligned}\hat{p}(x_t|y_{1:t-1}) &= \int p(x_t|x_{t-1})\hat{p}(x_{t-1}|y_{1:t-1})dx_{t-1} \\ &= \int p(x_t|x_{t-1})\sum_{i=1}^{n} w_{t-1}^i \delta_{x_{t-1}^i}(x_{t-1})dx_{t-1} \\ &= \sum_{i=1}^{n} w_{t-1}^i p(x_t|x_{t-1}^i).\end{aligned}$$

Now we have an approximation of the predictive distribution and can approximate the filtering distribution. We already have the observation density $p(y_t|x_t)$ from the SSM. Inserting the approximation $\hat{p}(x_t|y_{1:t-1})$ then gives

$$\begin{aligned}\hat{p}(x_t|y_{1:t}) &= \frac{p(y_t|x_t)}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}\hat{p}(x_t|y_{1:t-1}) \\ &\propto p(y_t|x_t)\sum_{i=1}^{n} w_{t-1}^i p(x_t|x_{t-1}^i).\end{aligned}$$

Now we have both the predict step and the update step on approximate forms. These can be summarised as an analogue to (3.3a) and (3.3b) respectively

$$\text{predict step: } \hat{p}(x_t|y_{1:t-1}) = \sum_{i=1}^{n} p(x_t|x_{t-1}^i)w_{t-1}^i \tag{3.4a}$$

$$\text{update step: } \hat{p}(x_t|y_{1:t}) \propto p(y_t|x_t)\sum_{i=1}^{n} p(x_t|x_{t-1}^i)w_{t-1}^i. \tag{3.4b}$$

The equations (3.4a) and (3.4b) give us an intuition of how the recursions can be approximated. Recall that the approximation $\hat{p}(x_{t-1}|y_{1:t-1})$ at iteration $t-1$ needs (x_{t-1}^i, w_{t-1}^i) for $i = 1, \ldots, n$ where w_{t-1} are the normalised weights. Consider the conceptual movement from right to left in equation (3.4b). The normalised weight w_{t-1}^i represents $\hat{p}(x_{t-1}^i|y_{1:t-1})$, then we can simulate a new variable $x_t^i \sim p(x_t|x_{t-1}^i)$. Note here that simulation of x_t^i can be done from other distributions. Once we have the simulated variable x_t^i, we can update the weight. Because we simulated from the transition density, we get that

$$w_t^i \propto w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{p(x_t^i|x_{t-1}^i)} = w_{t-1}^i p(y_t|x_t^i).$$

At this point, we have the simulated variables x_t^i for $i = 1, \dots, n$ and $w_t^i \propto w_{t-1}^i p(y_t|x_t^i)$. We want the normalised weights w_t, hence we need the common normalisation constant C in $w_t^i = C w_{t-1}^i p(y_t|x_t^i)$. The normalised weights should sum to 1, therefore we have the normalising constant and normalised weight respectively

$$C = \frac{1}{\sum_{l=1}^n w_{t-1}^l p(y_t|x_t^l)} \qquad w_t^i = \frac{w_{t-1}^i p(y_t|x_t^i)}{\sum_{l=1}^n w_{t-1}^l p(y_t|x_t^l)}$$

for the variables x_t^i where $i = 1, \dots, n$. We now have (x_t^i, w_t^i) and we can find the discrete approximation at iteration t, consequently, $\hat{p}(x_t|y_{1:t}) = \sum_{i=1}^n w_t^i \delta_{x_t^i}(x_t)$. We also note that simulating a new variable from $p(x_t|x_{t-1}^i)$ is a user-specified choice. We can e.g. simulate from the general importance sampling distribution $x_t^i \sim g(x_t|x_{1:t-1}^i, y_{1:t})$. From this, we have that

$$w_t^i \propto w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{g(x_t^i|x_{1:t-1}^i, y_{1:t})} \qquad w_t^i = \frac{w_{t-1}^i \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{g(x_t^i|x_{1:t-1}^i, y_{1:t})}}{\sum_{l=1}^n w_{t-1}^l \frac{p(y_t|x_t^l)p(x_t^l|x_{t-1}^l)}{g(x_t^l|x_{1:t-1}^l, y_{1:t})}}.$$

As in the case where we simulate from the transition density, we now have (x_t^i, w_t^i) and can approximate the target distribution at iteration t. After obtaining the updated weighted measure (x_t^i, w_t^i), for $i = 1, \dots, n$, one may resample n variables x_t with replacement using w_t as probabilities. Once resampled, the importance weights are reset, $\tilde{w}_t = \frac{1}{n}$, implying also that the normalised weights, $w_t = \frac{1}{n}$. This may be done at every iteration which is in line with the standard particle filter strategy (Givens and Hoeting, 2013). Resampling with replacement increase variance because of the replacement in the short term, but may reduce variance in the long term because we are propagating particles from more likely particles (Naesseth, Lindsten, and Schön, 2019).

As in the standard SMC algorithm, one may therefore utilise adaptive resampling based on ESS from equation (2.12) in particle filters. Using this intuition, we can outline a generic particle filter algorithm with adaptive resampling using threshold α. We focus on a specific particle and therefore omit the superscript notation, note however that the steps must be repeated for all the particles $i = 1, \dots, n$. We also define $w_0 \equiv 1$ together with $p(x_1|x_0) \equiv p(x_1)$ and $g(x_1|x_{1:0}, y_1) \equiv g(x_1|y_1)$ to simplify notation. We also use α as a resampling threshold. Compared to the standard SMC setup from Algorithm 3 we here only

Algorithm 4 Outline of generic particle filter

```
1: for $t \in (1, \dots, T)$ do
2:     $x_t \sim g(\cdot|x_{1:t-1}, y_{1:t})$
3:     $\tilde{w}_t \propto \tilde{w}_{t-1} \frac{p(y_t|x_t)p(x_t|x_{t-1})}{g(x_t|x_{1:t-1}, y_{1:t})}$
4:     calculate normalised weights $w_t$
5:     if ESS($w_t$) $< \alpha \cdot n$ then
6:         resample $x_{1:t}$ with probability $w_t$ and set $\tilde{w}_t = \frac{1}{n}$
7:     end if
8: end for
```

evaluate the incremental target $p(y_t|x_t)p(x_t|x_{t-1})$ and the chosen importance function $g(x_t|x_{1:t-1}, y_{1:t})$ at iteration t. Assume the objective is drawing inference related to the marginal filtering distribution $p(x_t|y_{1:t})$ and that the function g only depends on x_{t-1} and y_t. Then we only need to store x_{t-1} and w_{t-1} from the previous iteration. This can be useful when considering memory requirements in situations where the number of particles n and/or T is large.

3.3 Particle filters

In this section we briefly consider some aspects of three particle filter algorithms, we start by considering the particle filter algorithm that would result from the conditionally optimal importance sampling distribution. Then we consider the popular bootstrap particle filter and then finally we consider the auxiliary particle filter.

Optimal particle filter

The conditionally optimal importance sampling distribution is given in Proposition 2.6.1 and we follow the setup from Creal (2012) when considering what we refer to as the optimal particle filter. This importance sampling distribution is optimal in the sense that the variance of the weighs when conditioning on $x_{1:t-1}$ and $y_{1:t}$ is minimised. We see from the defined structure of g in equation (2.5) that we can simulate x_t from the incremental importance sampling function $g(x_t|y_{1:t}, x_{1:t-1})$ and create the new sequence $x_{1:t} = (x_{1:t-1}, x_t)$ (Creal, 2012, p. 252). Recall the sequential structure of the importance sampling distribution in equation (2.5). Combined with a recursive structure of the target distribution this allowed us to update the previous importance weight with a weight update function. Assume now that we select the optimal importance sampling distribution. From Proposition 2.6.1, we would then set our incremental importance sampling function equal to the conditional distribution

$$g(x_t|y_{1:t}, x_{1:t-1}) = p(x_t|y_t, x_{t-1}).$$

Assume now that we are able to simulate directly from $p(x_t|y_t, x_{t-1})$. Then we can consider the resulting weight update. Recall the general form for the weight update from equation (2.10). We reformulate the incremental importance sampling distribution which we have selected by straightforward calculations. When the reformulated optimal importance sampling function is inserted, we get the weight update

$$u_t = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{g(x_t|y_{1:t}, x_{1:t-1})} = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{\frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|x_{t-1})}} = p(y_t|x_{t-1}).$$

This expression is generally not available in closed form. In addition, simulating variables directly from $p(x_t|y_t, x_{t-1})$ might be intractable. We can reformulate the weight update as an integral (Doucet, Godsill, and Andrieu, 2000). We then have

$$p(y_t|x_{t-1}) = \int p(y_t|x_t)p(x_t|x_{t-1})dx_t.$$

Despite containing the observation and transition density from the SSM, the integral is in general intractable except in special cases. To summarise, the incremental importance sampling function $p(x_t|y_t, x_{t-1})$ is generally intractable to simulate directly from and the resulting weight update function $p(y_t|x_{t-1})$ is generally difficult to evaluate. Consequently, even though optimal with respect to reducing variance of the importance weights at t given $x_{1:t-1}$ and $y_{1:t}$ it can rarely be used directly in practice.

Bootstrap particle filter

We now consider the bootstrap particle filter (BPF). In this particle filter, we set the importance sampling function g equal to the transition density. We then have

$$g(x_t|x_{1:t-1}, y_{1:t}) = p(x_t|x_{t-1}).$$

Consider then the weight update that results from using the transition density as the importance sampling function. This is given by

$$u_t = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{g(x_t|x_{1:t-1}, y_{1:t})} = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(x_t|x_{t-1})} = p(y_t|x_t).$$

The importance sampling function which is used in the bootstrap particle filter implies that we ignore the observation y_t when simulating x_t. This is the reason BPF also has been referred to as using blind proposals (Creal, 2012). The weight update, u_t, then corresponds to the observation density from the SSM. The bootstrap particle filter is considerably easier to implement than the optimal particle filter because we already know the transition and observation density from the defined SSM. One would however expect that the transition density performs worse as an importance sampling function than the optimal from Proposition 2.6.1. The BPF is still a popular particle filter because of all the components being available a priori, making implementation easy in practice.

In situations with informative observations $y_{1:T}$ we see in general that particle filters that include y_t or possibly $y_{t:T}$ in the conditioning set of the importance sampling function improve performance compared to traditional BPF (Chopin and Papaspiliopoulos, 2020). Including observations $y_{t:T}$ at iteration t is however not possible in an online setting because we only have access to the observations $y_{1:t}$, the future observations are not available. We refer to an offline setting as the situation where all observations $y_{1:T}$ are available at every iteration. This also implies that the calculations cannot performed before all observations are available. We will see that utilising information from iteration s at iteration t where $s > t$ is often referred to as lookahead strategies, see e.g. Lin, Chen, and J. S. Liu (2013). At iteration t, we however have observation y_t available and this can be included in the importance sampling function. By conditioning also on y_t we have the same importance sampling function as in the optimal particle filter. We can now consider the BPF in algorithmic form. For notational simplicity we have the following assumption $p(x_1|x_0^i) = p(x_1)$ in the algorithm. We use α as the resampling threshold together with the generic function $\mathrm{ESS}(w_t)$ which calculates the effective sample size, in addition the resampling step is with replacement. The algorithm is

Algorithm 5 BPF with adaptive resampling

```
1:  for $t \in (1, \dots, T)$ do
2:      for $i \in (1, \dots, n)$ do
3:          $\hat{x}_t^i \sim p(x_t|x_{t-1}^i)$
4:          $\hat{x}_{1:t}^i = (x_{1:t-1}^i, \hat{x}_t^i)$
5:          $w(\hat{x}_{1:t}^i) \propto w(x_{1:t-1}^i)p(y_t|\hat{x}_t^i)$
6:      end for
7:      $w(\hat{x}_{1:t}^i) = \frac{w(\hat{x}_{1:t}^i)}{\sum_{l=1}^n w(\hat{x}_{1:t}^l)}$ for $i = 1, \dots, n$
8:      if $\text{ESS}(w_t) < \alpha \cdot n$ then
9:          resample $x_{1:t}^i$ from $\hat{x}_{1:t}^i$ using $w(\hat{x}_{1:t}^i)$ for $i = 1, \dots, n$
10:         $w(x_{1:t}^i) = \frac{1}{n}$ for $i = 1, \dots, n$
11:     else
12:         $x_{1:t}^i = \hat{x}_{1:t}^i$ for $i = 1, \dots, n$
13:         $w(x_{1:t}^i) = w(\hat{x}_{1:t}^i)$ for $i = 1, \dots, n$
14:     end if
15: end for
```

adapted to our notation from the bootstrap particle filter in Doucet, Freitas, and Gordon (2001).

Auxiliary particle filter

The auxiliary particle filter (APF) is another possible PF algorithm. Compared to the BPF, the APF at iteration t is utilising the observation y_t in two ways. First it is resampling the previous variables x_{t-1} based on their predictive ability by calculating $p(y_t|\tilde{x}_t)$. Here $\tilde{x}_t$ is some estimator of a likely value given x_{t-1} from the distribution $p(x_t|x_{t-1})$. Often, the conditional mean of the distribution is used (Pitt and Shephard, 1999, p. 592). Then these resampled variables, x_{t-1}, are used in order to simulate the latent variables x_t.

This is done by introducing auxiliary variables that represent the latent variables at the previous iteration of the algorithm. We will follow Pitt and Shephard (1999) in this section in order to see the motivation for introducing the auxiliary variables. We denote the auxiliary variable by $k \in (1, \dots, n)$. At iteration t, the auxiliary variable, k, is referring to one of the, previous, latent variables x_{t-1}. Assume now that we have an outlier observation, y_t, at t. The weight update $p(y_t|x_t^i)$ for $i = 1, \dots, n$ in the BPF can then yield values with high variance because of y_t. This is in part because BPF is ignoring the potentially extreme observation y_t when simulating x_t. It is proposed that considering the joint posterior distribution over $p(x_t, k|y_{1:t})$ in order to sample x_t will give more equal normalised weights (Pitt and Shephard, 1999, p. 592). We then have the joint posterior distribution and the importance sampling distribution

$$
\begin{aligned}
p(x_t, k|y_{1:t}) &\propto p(y_t|x_t)p(x_t|x_{t-1}^k)w(x_{1:t-1}^k) \\
g(x_t, k|y_{1:t}) &\propto p(y_t|\tilde{x}_t^k)g(x_t|x_{t-1}^k)w(x_{1:t-1}^k).
\end{aligned}
$$

The importance sampling function, g, is the generic importance sampling function proposed in (Pitt and Shephard, 1999). We can see the resemblence to the generic reformulation by considering the joint distribution over x_t and x_{t-1} and using CI properties. We then have

$$p(x_t, x_{t-1}|y_{1:t}) \propto p(y_t|x_t)p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1}).$$

Conceptually, we want to marginalise out x_t from the importance sampling distribution (Pitt and Shephard, 1999). This is done in order to find an expression for the conditional distribution of k given by

$$g(k|y_{1:t}) \propto p(y_t|\tilde{x}_t^k)w(x_{1:t-1}^k).$$

We then consider the two stage weighting-setup from the original definition of APF combined with the setup from Creal (2012). The first stage weight is associated with the previously simulated latent variable x_{t-1}^k for $k = 1, \ldots, n$. We consider the unnormalised weights $\tilde{\lambda}$ which we then normalise. We then have unnormalised and normalised weights respectively

$$\tilde{\lambda}_t^k = g(k|y_{1:t}) \qquad\qquad \lambda_t^k = \frac{\tilde{\lambda}_t^k}{\sum_{j=1}^n \tilde{\lambda}_t^j}$$

If we resample with respect to the discrete distribution with probabilities λ_t, then x_{t-1}^k is expected to give a high value for the likelihood $p(y_t|\tilde{x}_t^k)$. To sample x_t we now use x_{t-1}^k where k is sampled according to λ_t^k. We then sample $x_t \sim g(x_t|x_{t-1}^k)$ where we can think of the auxiliary variable k as including information from the observation y_t. Conceptually, we can then think of $g(x_t|x_{t-1}^k) = \hat{p}(x_t|x_{t-1}, y_t)$. For the weight update u_t used to update the importance weights we have the more traditional incremental target function and the implicit incremental importance sampling function

$$u_t^k = \frac{p(y_t|x_t^k)p(x_t^k|x_{t-1}^k)}{p(y_t|\tilde{x}_t^k)g(x_t^k|x_{t-1}^k)}. \tag{3.5}$$

A special case is when we have an analytical expression for the predictive likelihood $p(y_t|x_{t-1})$. We then use the reformulation $p(y_t|x_{t-1})p(x_t|x_{t-1}, y_t) = p(y_t|x_t)p(x_t|x_{t-1})$. The distribution $g(x_t|x_t^k)$ is implicitly approximating $p(x_t|x_{t-1}, y_t)$ and we can follow Creal (2012) in setting $p(y_t|\tilde{x}) = p(y_t|x_{t-1})$ and $g(x_t|x_t^k) = p(x_t|x_{t-1}, y_t)$. We insert the known expressions for the approximations in equation (3.5) and consequently we have $u_t^k = 1$. This is referred to as fully adapted in Creal (2012).

Informally one could say that the APF is resampling the previous x_{t-1} at iteration t based on the expected predictive likelihood. This resampled set of x_{t-1} is then incorporating information from the observations y_t. This can again increase the quality of the simulated set x_t which is now simulated by conditioning on more information than in the BPF. It is also possible to see APF as a standard SMC algorithm with a slightly different target distribution than $p(x_{1:t}|y_{1:t})$. See Johansen and Doucet (2008) for a further discussion. The APF can also be seen as a true lookahead strategy, this is discussed in Lin, Chen, and J. S. Liu (2013), this is because it is performing the lookahead using only available observations $y_{1:t}$ at iteration t. Conceptually, we can think of it as utilising y_t which is future relative to x_{t-1}.

3.4 Smoothing recursions

In this section we briefly consider smoothing recursions. We follow the approach of Briers, Doucet, and Maskell (2010) and consider briefly two approaches used for smoothing. We will in general be interested in the target distribution $p(x_t|y_{1:T})$, often referred to as the marginal smoothing distribution (Creal, 2012). Smoothing is fundamentally related to an offline setting because we require $y_{1:T}$ at iteration t where $T > t$. That is, we need all the observations $y_{1:T}$ at every iteration in order to consider the marginal smoothing distributions. Recall that through the filtering recursions, we have $p(x_{1:T}|y_{1:T})$ at $t = T$. By integration, one could conceptually obtain the marginal smoothing distribution for x_t by marginalising out every latent variable except x_t. This gives the high-dimensional integral

$$p(x_t|y_{1:T}) = \int \dots \int p(x_{1:T}|y_{1:T}) dx_{1:t-1} dx_{t+1:T}.$$

Similarly to the analytical expressions for the filtering density, we have that the integrals on this form are intractable. In general terms, the smoothing algorithms often rely on approximate filtering distributions for $t = 1, \dots, T$. The smoothing recursions then utilise these distributions in different ways when moving backwards $t = T, \dots, 1$. One issue related to these approaches is the resampling of particles introduced to reduce weight degeneracy in the particle filters. Recall that entire sequences $x^i_{1:t}$ for $i = 1, \dots, n$ were resampled. When $t = T$, many of $x^i_{1:T}$ for $i = 1, \dots, n$ will then be equal in $x_{1:s}$ when $s \ll T$ due to resampling. This is referred to as path degeneracy in Naesseth, Lindsten, and Schön (2019) because at the earlier parts of the sequences almost all or all sequences will share the same path.

Forward filtering-backward smoothing

The forward filtering-backward smoothing (FFBS) is one approach to smoothing. Here, we use a particle filter moving forwards. When $t = T$, we have approximate filtering distributions $\hat{p}(x_s|y_{1:s})$ for $s = 1, \dots, T$. In order to see the motivation for FFBS we start by a reformulation of the marginal smoothing recursion that result in (Kitagawa, 1987, eq. 2.4, eq. 2.5). We also note that in this setting, we have approximations to all the filtering distributions because we start the smoothing after running the particle filter to iteration T. We therefore start from the last iteration and iterate backwards, that is $t = T, \dots, 1$. The marginal smoothing distribution can then be expressed by

$$\begin{aligned} p(x_t|y_{1:T}) &= \int p(x_t|x_{t+1}, y_{1:T}) p(x_{t+1}|y_{1:T}) dx_{t+1} \\ &= \int p(x_t|x_{t+1}, y_{1:t}) p(x_{t+1}|y_{1:T}) dx_{t+1} \\ &= p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} p(x_{t+1}|y_{1:T}) dx_{t+1}. \end{aligned} \tag{3.6}$$

Recall the CI properties of the SSM. In the second equality there are no active trails between x_t and $y_{t+1:T}$ given x_{t+1}. Therefore x_t and $y_{t+1:T}$ are d-separated given x_{t+1}, implying that $(x_t \perp y_{t+1:T}|x_{t+1})$ (Koller and N. Friedman, 2009).

We can then omit $y_{t+1:T}$ from the conditioning set of $p(x_t|x_{t+1}, y_{1:T})$. Assume now that we have approximations of the predictive and filtering distributions available from the particle filter. In addition assume that because we are iterating backwards, we have an approximation of $p(x_{t+1}|y_{1:T})$ at iteration t. Then we have all the necessary terms to approximate the recursion above.

Two-filter

Another possible reformulation of the marginal smoothing distribution is called the two-filter. This is motivated by the following reformulation

$$p(x_t|y_{1:T}) = \frac{p(y_{t:T}|x_t)p(x_t|y_{1:t-1})}{p(y_{t:T}|y_{1:t-1})} \propto p(y_{t:T}|x_t)p(x_t|y_{1:t-1}).$$

The first equality follows by omitting $y_{1:t-1}$ from the conditioning set because x_t is in the conditioning set. This is another case of there being no active trails from any of the variables in $y_{1:t-1}$ to $y_{t:T}$ because of x_t. It is possible to express the term $p(y_{t:T}|x_t)$ in a recursive manner. We then have

$$\begin{aligned} p(y_{t:T}|x_t) &= \int p(y_{t:T}, x_{t+1}|x_t)dx_{t+1} \\ &= p(y_t|x_t)\int p(y_{t+1:T}|x_{t+1})p(x_{t+1}|x_t)dx_{t+1}. \end{aligned} \tag{3.7}$$

This makes it possible to calculate the smoothing distribution up to proportionality. The term $p(y_{t:T}|x_t)$ is often called the backward information filter (Briers, Doucet, and Maskell, 2010, eq. 7, eq. 8). We see that the backward information filter can be calculated recursively by iterating backwards. The last reformulation follows by moving the observation density out from the integral as it does not depend on x_{t+1}.

Approximations

We can approximate smoothing recursions in a similar way to how we can approximate filtering recursions. This allows for making inference in a similar manner as with the approximate filtering distributions. In this section we will briefly see how we can approximate the recursions for FFBS. It is also possible to derive approximations for the smoothing distributions using the two-filter approach, see e.g. Briers, Doucet, and Maskell (2010).

For the approximation of the forward-backward recursion, we follow the approach of Doucet, Godsill, and Andrieu (2000). First assume that we have a weight expression which incorporates future observations from $t+1$ up to T already at t. Denote these weights by $w^j_{t+1|T}$, we can then use these weights in combination with Dirac mixtures in the same way as when we have discrete approximations of filtering distributions. We then have the approximation $p(x_{t+1}|y_{1:T}) \approx \sum_{j=1}^n w^j_{t+1|T}\delta_{x^j_{t+1}}(x_{t+1})$ available at t. Then we note that the numerator in the recursion (3.6) can be reformulated and approximated. This then gives

$$\hat{p}(x_{t+1}|y_{1:t}) = \sum_{k=1}^n w_t^k \int p(x_{t+1}|x_t)\delta_{x_t^k}(x_t)dx_t$$

$$= \sum_{k=1}^{n} w_t^k p(x_{t+1}|x_t^k).$$

This corresponds to equation (59) in Doucet, Godsill, and Andrieu (2000). We can then set up the approximation to the recursion given equation (3.6) by using particle estimation of the different distributions. This then result in an expression equivalent to equation (60) in Doucet, Godsill, and Andrieu (2000) in our notation. We therefore have

$$\begin{aligned}\hat{p}(x_t|y_{1:T}) &= \sum_{i=1}^{n} w_t^i \delta_{x_t^i}(x_t) \sum_{j=1}^{n} w_{t+1|T}^j \int \frac{p(x_{t+1}|x_t)}{\sum_{k=1}^{n} w_t^k p(x_{t+1}|x_t^k)} \delta_{x_{t+1}^j}(x_{t+1}) dx_{t+1} \\ &= \sum_{i=1}^{n} w_t^i \left[\sum_{j=1}^{n} w_{t+1|T}^j \frac{p(x_{t+1}^j|x_t^i)}{\sum_{k=1}^{n} w_t^k p(x_{t+1}^j|x_t^k)} \right] \delta_{x_t^i}(x_t) \\ &= \sum_{i=1}^{n} w_{t|T}^i \delta_{x_t^i}(x_t).\end{aligned}$$

By using this approximation, we have all the components needed to calculate the approximate smoothing distribution at t. This allows us to iterate backwards and generate new smoothing weights for the different particles. We note that only the weights are updated with this approach. This implies that the problems related to degeneracy still remains because of the resampling steps used in the forward filtering.

We can then use this approximation in order to setup the fixed-interval smoothing algorithm described in Doucet, Godsill, and Andrieu (2000). This algorithm estimates the smoothing weights by iterating backwards from $t = T$. The initialisation in the fixed-interval smoothing algorithm starts by setting $w_{T|T}^i = w_T^i$ for $i = 1, \ldots, n$ where w_T is the normalised weights from the final iteration of the particle filter.

3.5 Likelihood estimation in particle filters

The marginal likelihood related to the observations of a SSM can be used for further inference. Both maximum likelihood and Bayesian methods in online and offline settings rely on likelihood estimates for parameter estimation. In general, we can express marginal likelihood through marginalisation $p(y_{1:t}) = \int \ldots \int p(y_{1:t}, x_{1:t}) dx_{1:t}$ as in e.g. Devore and Berk (2012). Recall that we have an expression for $p(x_{1:t}, y_{1:t})$ in equation (2.4) which consists of the transition- and observation density. This gives a multiple integral for the marginal likelihood at iteration T. We then have

$$\begin{aligned}p(y_{1:T}) &= \int \ldots \int p(x_{1:T}, y_{1:T}) dx_{1:T} \\ &= \int \ldots \int p(y_1|x_1) p(x_1) \prod_{s=2}^{T} p(y_s|x_s) p(x_s|x_{s-1}) dx_{1:T}. \qquad (3.8)\end{aligned}$$

The integral is in general high-dimensional and the integrand can be on an intractable form. This implies that we have to estimate the integral numerically by using e.g. Monte Carlo integration. For effective parameter estimation by using likelihood estimates we also want the estimates to have low variance. In an online setting, we can utilise the fact that the likelihood expression can be reformulated in a recursive manner (Kantas et al., 2015). This allows us to approximate integrals of lower dimensions and combine the approximations. This is compared to approximating high-dimensional integrals directly. We denote the marginal likelihood at iteration t by $Z_t = p(y_{1:t})$ and similarly the estimated likelihood by $\hat{Z}_t$.

As discussed in Kantas et al. (2015) we also need likelihood estimates $\hat{Z}_t$ for $t = 1, \dots, T$ when estimating parameters in an online setting. The recursive structure of the likelihood expression allows for online estimation of the marginal likelihood. By straightforward calculations we also note that at iteration $s < T$ we will have the likelihood $p(y_{1:s})$ by including likelihood factors up to and including iteration s. We then have the following expression for the marginal likelihood,

$$p(y_{1:T}) = p(y_1) \prod_{s=2}^{T} p(y_s|y_{1:s-1}). \tag{3.9}$$

In order to estimate the marginal likelihood we often utilise the decomposition in equation (3.9). We start by an expansion of the likelihood factor given by

$$p(y_s|y_{1:s-1}) = \int p(y_s|x_s) \left[\int p(x_s|x_{s-1}) p(x_{s-1}|y_{1:s-1}) dx_{s-1} \right] dx_s$$

If we are able to evaluate $p(y_s|y_{1:s-1})$ exactly, we could utilise these likelihood factors combined with equation (3.9) in order to calculate the marginal likelihood directly. As we generally cannot evaluate $p(y_s|y_{1:s-1})$ exactly, we need approximations. We denote approximations of the likelihood factor by $\hat{p}(y_s|y_{1:s-1})$. If we can obtain approximations $\hat{p}(y_1)$ and $\hat{p}(y_s|y_{1:s-1})$ for $s = 2, \dots, T$, we can combine these using equation (3.9) in order to approximate the marginal likelihood.

We now consider approximations of the likelihood factors in combination with a BPF. Assume now that the particles from iteration $s-1$ are available. We can use these to obtain a discrete approximation of the filtering distribution $p(x_{s-1}|y_{1:s-1})$. We can then approximate the likelihood factor which gives us

$$\hat{p}(y_s|y_{1:s-1}) = \sum_{i=1}^{n} w_{s-1}^i \int p(y_s|x_s) p(x_s|x_{s-1}^i) dx_s.$$

For $s = 1$ and $s \geq 2$ respectively we have approximations of the likelihood factors

$$\hat{p}(y_1) = \sum_{i=1}^{n} \frac{1}{n} p(y_s|x_s^i) \qquad \hat{p}(y_s|y_{1:s-1}) = \sum_{i=1}^{n} w_{s-1}^i p(y_s|x_s^i). \tag{3.10}$$

We can then utilise the structure in equation (3.9) when estimating the marginal likelihood using a particle filter. This gives an expression for estimating the marginal likelihood using a particle filter by inserting from equation (3.10). We then have

$$\hat{Z}_T = \hat{p}(y_1) \prod_{s=2}^{T} \hat{p}(y_s|y_{1:s-1}). \tag{3.11}$$

An important property of the likelihood estimate $\hat{Z}_t$ is that it is unbiased, see e.g. chapter 16 of Chopin and Papaspiliopoulos (2020) for a proof. In the case of a BPF we can estimate the likelihood factor at each iteration and then combine these with equation (3.11) in order to estimate the marginal likelihood. The marginal likelihood is also important when it comes to the target distribution in equation (2.2). We often have the posterior $p(x_{1:t}|y_{1:t})$ as the target distribution and the joint distribution $p(x_{1:t}, y_{1:t})$ as the unnormalised target distribution in the filtering context. This yields that the normalising constant at iteration t is given by $Z_t = \int \ldots \int p(x_{1:t}, y_{1:t}) dx_{1:t}$. We therefore see that the normalising constants are given by $Z_t = p(y_{1:t})$ for $t = 1, \ldots, T$.

A more direct approach of likelihood estimation can be seen when considering IS and SIS and is discussed in Naesseth, Lindsten, and Schön (2019). This is related to the unnormalised target distribution $\tilde{f}_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$ which we use in a filtering context. Assume now that we sample n sequences $x_{1:t}$ from an importance sampling distribution. We denote the importance sampling distribution by $g(x_{1:t})$ and construct the importance weight $\tilde{w}(x_{1:t})$ as discussed in Section 2.5. We can then see from the definition of the likelihood integral that we can estimate this by using e.g. Monte Carlo integration. We then have that

$$\int \ldots \int \tilde{w}(x_{1:t}) g(x_{1:t}) dx_{1:t} \approx \frac{1}{n} \sum_{i=1}^{n} \tilde{w}(x_{1:t}^i).$$

In the case of online parameter estimation, we need estimates of the marginal likelihood $\hat{Z}_t$ possibly at every iteration $t = 1, \ldots, T$. Utilising particle filters in an online setting makes it possible to obtain these estimates. If we however are not limited to an online setting, it is possible to obtain likelihood estimates that typically have lower variance. By moving to an offline setting we can introduce flexibility by using alternative intermediate $t < T$ target distributions in addition to selecting the importance sampling distribution. This is discussed in e.g. Lindsten, Helske, and Vihola (2018) and Naesseth, Lindsten, and Schön (2019).

As an motivation for our interest in estimating the likelihood, we here briefly consider some aspects of parameter estimation where we see the need for likelihood estimates. Parameter estimation is a large field in itself. Therefore our aim here is only to consider briefly the role of likelihood estimates within Bayesian parameter estimation. A detailed review can be found in e.g. Kantas et al. (2015). Recall that we had unknown parameters θ that can be a scalar or a vector. These were omitted for notational simplicity when assumed known. In general these are not known and we have to estimate the unknown parameters, denoted by $\hat{\theta}$. We briefly focus on some aspects of Bayesian parameter estimation in an offline and an online setting. In Bayesian parameter estimation we assign a prior denoted by $p(\theta)$ for the unknown parameter and use the posterior $p(x_{1:t}, \theta|y_{1:t})$ in online setting or $p(x_{1:T}, \theta|y_{1:T})$ in offline settings for inference about the parameter (Kantas et al., 2015).

Within offline Bayesian parameter estimation we have the marginal Metropolis-Hastings algorithm where we use SMC in order to estimate the likelihood $p(y_{1:T}|\theta)$. This results in what is often referred to as a particle version of marginal Metropolis-Hastings as discussed in section 2.4.2 of Andrieu, Doucet, and Holenstein (2010). Estimates of the likelihood $p(y_{1:T}|\theta)$ can then be used in the Metropolis-Hastings ratio within the particle version of Metropolis-Hastings setup. This is also part of the motivation in Guarniero, Johansen, and Lee (2017) for obtaining unbiased likelihood estimates with low variance.

Within online Bayesian parameter estimation the most straightforward approach consists of only simulating θ from a prior $p(\theta)$ and use $p(x_{1:t}, \theta|y_{1:t})$ as the target distribution. However simulating θ only once at the initial iteration and then resampling at later iterations implies many duplicated values of the parameter (Kantas et al., 2015). Another approach within online Bayesian parameter estimation is that of J. Liu and West (2001) where noise is added to the parameter at every iteration. Here we focus on the target distribution $p(x_{1:t}, \theta_{1:t}|y_{1:t})$ and define $p(\theta_{t+1}|\theta_t)$ as a transition density for θ in order to simulate new parameters when t increases (J. Liu and West, 2001). If the main assumption is that θ is a fixed parameter, then introducing $p(\theta_{t+1}|\theta_t)$ which introduce noise to the parameter may not be optimal. However there are situations where this dynamic behaviour of the parameter can be reasonable.

CHAPTER 4

Twisting target distributions

We have seen that selecting the importance sampling distribution can influence the efficiency of different SMC methods. We also considered the optimal importance sampling distribution at iterations $t = 1, \ldots, T$ in Proposition 2.6.1 when observations up until iteration t were available. Twisting target distributions refers to altering the intermediate, that is $t < T$, target distributions when our main objective is making inference using the final, $t = T$, target distribution. Being able to alter the target distributions at iterations $t < T$ in addition to selecting every importance sampling distribution increase the flexibility of general SMC methods. Altering, or twisting, in this context often refers to utilising future observations $y_{s:T}$ at iteration s (Naesseth, Lindsten, and Schön, 2019). Utilising future observations $y_{s:s+\Delta}$ where $\Delta > 0$ for inference about x_s is a part of the general lookahead framework reviewed in Lin, Chen, and J. S. Liu (2013). Including all the observations, $y_{1:T}$, in the conditioning set at every iteration is intuitively optimal. We are then utilising all available information at every iteration, however this requires an offline setting. Recall the general form of target distributions from equation (2.2). In the filtering context, we had the unnormalised target distribution defined as $\tilde{f}_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$ and the specific target distribution $f_t(x_{1:t}) = p(x_{1:t}|y_{1:t})$. We then denoted the target distributions directly by $p(x_{1:t}|y_{1:t})$ for clarity. We will now use $f_t(x_{1:t})$ for the target distribution and $\tilde{f}_t(x_{1:t})$ for the unnormalised target distribution. To distinguish it from the twisting target distributions, we will refer to the specific target distributions $p(x_{1:t}|y_{1:t})$ as the untwisted target distributions.

In this chapter we first consider the motivation for using twisting target distributions. We start with what we refer to as the optimal target distributions for simulating $x_{1:T}$. These are characterised by conditioning on all observations $y_{1:T}$ at every iteration. The optimal target distributions are generally intractable, but motivates how we can define twisting target distributions. We then introduce what we will refer to as twisting functions. This is a pair of functions defined for every iteration. We will see that we can use twisting functions to define twisting target distributions. We then consider some properties related to the twisting target distributions. Finally we will briefly consider the iterated auxiliary particle filter (iAPF) of Guarniero, Johansen, and Lee (2017) which is using twisting functions and a twisted model. Other approaches to using twisting target distributions is the controlled SMC from Heng et al. (2020) and the approach of Lindsten, Helske, and Vihola (2018) and Naesseth, Lindsten, and Schön (2019).

Our main strategy for utilising twisting target distributions is via the twisted model, following the approach of Guarniero, Johansen, and Lee (2017). The transition and observation density of the SSM given in equation (2.1) will be replaced with a twisted transition density and a twisted observation function in the twisted model. We sometimes refer to concepts as untwisted in the twisting target context. Such as referring to a traditional SSM as an untwisted model. The untwisted model is repeated here without parameters for completeness,

$$\begin{aligned} x_1 &\sim p(x_1) && \text{initial density} \\ x_t|x_{t-1} &\sim p(x_t|x_{t-1}) && \text{transition density} \\ y_t|x_t &\sim p(y_t|x_t) && \text{observation density.} \end{aligned}$$

4.1 Motivation

So far we have mainly focused on the filtering context. There we had $f_t(x_{1:t}) = p(x_{1:t}|y_{1:t})$ and $\tilde{f}_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$ for $t = 1, \dots, T$. The unnormalised target distributions in the filtering context follow the decomposition

$$\tilde{f}_t(x_{1:t}) = p(x_{1:t}, y_{1:t}) = p(y_1|x_1)p(x_1) \prod_{s=2}^{t} p(y_s|x_s)p(x_s|x_{s-1}).$$

At the final iteration, the target distribution is $p(x_{1:T}|y_{1:T})$ in the filtering context. Therefore we want a sample of simulated sequences $x_{1:T}$ to be as close as possible to a sample originating directly from $p(x_{1:T}|y_{1:T})$. The simulated sequences can then be used to estimate the normalising constant $Z_T = p(y_{1:T})$ and discrete approximations of $p(x_{1:T}|y_{1:T})$. For the motivation of twisting target distributions we mainly follow the discussion in Naesseth, Lindsten, and Schön (2019). We can define twisting target distributions by altering the intermediate untwisted target distributions. Hence the sequence of untwisted target distributions $p(x_{1:t}|y_{1:t})$ for $t = 1, \dots, T-1$, can be viewed as one of many possible sequences that yield $p(x_{1:T}|y_{1:T})$ at the final iteration. To see the motivation for twisting target distributions we consider the optimal intermediate target distributions. These are referred to as optimal with respect to simulating $x_{1:T}$ from the final target distribution $p(x_{1:T}|y_{1:T})$.

Optimal target distributions

We denote the optimal target distributions by $f_t^*(x_{1:t})$ for iterations $t = 1, \dots, T$. Assume we are in an offline setting and have observations $y_{1:T}$ available at every iteration. We want to utilise the observations in order to define optimal target distributions. This corresponds to a nonconstant lookahead of $\Delta = T - t$ at iteration t, compared to the online setting where $\Delta = 0$ at every iteration. Following Naesseth, Lindsten, and Schön (2019), the optimal target distributions can be defined as marginalised versions of the final posterior distribution $p(x_{1:T}|y_{1:T})$. That is,

$$f_t^*(x_{1:t}) = \int \dots \int p(x_{1:T}|y_{1:T}) dx_{t+1:T}$$

and we see that the optimal target distribution can be formulated as the posterior distribution

$$f_t^*(x_{1:t}) = p(x_{1:t}|y_{1:T}). \tag{4.1}$$

This corresponds to defining the unnormalised target distribution, $\tilde{f}_t^*(x_{1:t}) = p(x_{1:t}, y_{1:T})$, and consequently having the normalising constant $Z_t = p(y_{1:T})$. We see that the optimal target distribution have all observations in the conditioning set at every iteration. The optimal target distribution $f_t^*(x_{1:t})$ and the untwisted target distribution $p(x_{1:t}|y_{1:t})$ are equal at the final iteration. That is,

$$f_T^*(x_{1:T}) = p(x_{1:T}|y_{1:T})$$

which we will refer to as final target equivalence. From this we see that one can choose a sequence of untwisted target distributions $f_t(x_{1:t}) = p(x_{1:t}|y_{1:t})$ or a sequence of optimal target distributions $f_t^*(x_{1:t}) = p(x_{1:t}|y_{1:T})$ for the intermediate iterations if the objective is $p(x_{1:T}|y_{1:T})$. Both are equal at the final iteration, but the optimal target distributions are conditioning on all the observations at every iteration. This is in contrast to the sequence of target distributions on the form $p(x_{1:t}|y_{1:t})$ which is used in the filtering context. If we however are in an online setting, we only have access to $y_{1:t}$ at iteration t.

We now want to reformulate the optimal, intermediate, target distributions $f_t^*(x_{1:t})$ into a recursive form similar to equation (2.3a). We will use this recursive form of the optimal target distributions when defining the twisting target distributions. At iteration $t = 1$, we have

$$f_1^*(x_1) = p(y_1|x_1)p(x_1)\frac{p(y_{2:T}|x_1)}{p(y_{1:T})}$$

and at iteration $1 < t \leq T-1$, we have

$$\begin{aligned}
f_t^*(x_{1:t}) &= f_{t-1}^*(x_{1:t-1})p(x_t|x_{t-1}, y_{t:T}) \\
&= f_{t-1}^*(x_{1:t-1})p(y_t|x_t)p(x_t|x_{t-1})\frac{p(y_{t+1:T}|x_t)}{p(y_{t:T}|x_{t-1})} \\
\tilde{f}_t^*(x_{1:t}) &= \tilde{f}_{t-1}^*(x_{1:t-1})p(y_t|x_t)p(x_t|x_{t-1})\frac{p(y_{t+1:T}|x_t)}{p(y_{t:T}|x_{t-1})}.
\end{aligned} \tag{4.2}$$

Because all optimal target distributions have the same normalisation constants the unnormalised optimal target distributions follow the same recursive structure. This is given in equation (4.2). We will refer to $p(y_{t+1:T}|x_t)$ as a predictive likelihood term at iteration t. A more direct motivation for the recursive structure when we know the form of the optimal target distribution $p(x_{1:t}|y_{1:T})$ can be seen by

$$\begin{aligned}
p(x_{1:t}|y_{1:T}) &= p(x_{1:t-1}|y_{1:T})p(x_t|x_{1:t-1}, y_{1:T}) \\
&= p(x_{1:t-1}|y_{1:T})p(x_t|x_{t-1}, y_{t:T})
\end{aligned}$$

which we see follow a recursive structure by utilising CI assumptions about the model.

Optimal importance sampling distributions

As in the general IS framework, we can select the importance sampling distribution. The optimal, trivial, case is when the importance sampling distribution is equal to the optimal target distribution. Following Naesseth, Lindsten, and Schön (2019), we find the optimal importance sampling distributions for the optimal target distributions. This can be done by using the same sort of calculations as for Proposition 2.6.1. We denote this importance sampling distribution by

$$g_t^*(x_t|x_{1:t-1}, y_{1:t}) = p(x_t|x_{t-1}, y_{t:T}). \tag{4.3}$$

Note that this importance sampling distribution uses all future observations at iteration t. These are not conditionally independent of x_t given x_{t-1} because we have no variables in the conditioning set that blocks influence from observations that are future relative to x_t. We can reformulate equation (4.3) to recognise terms from the optimal intermediate target distribution. We then have

$$\begin{aligned} p(x_t|x_{t-1}, y_{t:T}) &= \frac{p(y_t, y_{t+1:T}, x_t, x_{t-1})}{p(y_{t:T}, x_{t-1})} \\ &= \frac{p(y_{t+1:T}|x_t)p(y_t|x_t)p(x_t|x_{t-1})}{p(y_{t:T}|x_{t-1})}. \end{aligned} \tag{4.4}$$

Optimal weight update

We now consider the weight update that follows from using the optimal importance sampling distributions in equation (4.3). Simulating from the optimal importance sampling distribution while targeting the optimal target distribution yields that the simulated sequence, $x_{1:T}$, form a perfect sample from $p(x_{1:T}|y_{1:T})$ which is the final untwisted target distribution (Naesseth, Lindsten, and Schön, 2019, p. 48). Consider now the weight update function from equation (2.10). At iteration t, we can insert the optimal intermediate target distribution and the optimal importance distribution. We then have that

$$\begin{aligned} u_t &= \frac{p(y_t|x_t)p(x_t|x_{t-1})\frac{p(y_{t+1:T}|x_t)}{p(y_{t:T}|x_{t-1})}}{g_t^*(x_t|x_{1:t-1}, y_{1:t})} \\ &= \frac{p(y_t|x_t)p(x_t|x_{t-1})\frac{p(y_{t+1:T}|x_t)}{p(y_{t:T}|x_{t-1})}}{\frac{p(y_{t+1:T}|x_t)p(y_t|x_t)p(x_t|x_{t-1})}{p(y_{t:T}|x_{t-1})}} = 1. \end{aligned}$$

The weight update functions become 1 for all iterations, this implies that all the samples will have normalised weights of $\frac{1}{n}$. We generally cannot sample directly from $g_t^*(x_t|x_{1:t-1}, y_{1:t})$. In addition, we generally cannot evaluate the predictive likelihood terms in the optimal target distribution. If we however can do this, it would correspond to sampling from the optimal target distribution.

Target distribution in factorised form

We follow the same assumption as in Naesseth, Lindsten, and Schön (2019, p. 48) about being able to factorise the final, unnormalised, target distribution into multiplicative factors. This is not restricted to twisting target distributions, we have seen this in equation (2.4). The factors are equivalent to what Creal (2012) refers to as incremental target distributions.

Definition 4.1.1. *For $t = 1$ and $t = 2, \dots, T$ we define the factors respectively as*

$$\phi_1(x_1) \equiv p(y_1|x_1)p(x_1)$$
$$\phi_t(x_t, x_{t-1}) \equiv p(y_t|x_t)p(x_t|x_{t-1}).$$

The dependence on y_t are omitted because the observations are assumed fixed.

Following Naesseth, Lindsten, and Schön (2019) we can express the unnormalised target distribution by using factors at iteration $t = T$. Consider the case where we defined the unnormalised target distribution to be $\tilde{f}_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$. The normalisation constant then corresponded to the marginal likelihood at iteration t, that is, $Z_t = p(y_{1:t})$. This is the setup we considered in the filtering context. For the final, unnormalised, target distribution this gives the expression

$$\begin{aligned}\tilde{f}_T(x_{1:T}) = p(x_{1:T}, y_{1:T}) &= p(y_1|x_1)p(x_1)\prod_{s=2}^{T} p(y_s|x_s)p(x_s|x_{s-1}) \\ &= \phi_1(x_1)\prod_{s=2}^{T}\phi_s(x_s, x_{s-1}).\end{aligned}$$

4.2 Twisting target distributions

In this section we first consider what we will refer to as twisting functions. Then we consider how twisting functions can be used to define twisting target distributions. Finally, we consider some properties of the resulting twisting target distributions.

Twisting functions

The twisting functions at iteration t is a pair of functions. This pair consist of a lookahead function denoted by $\psi_t(x_t)$ and a normalising function denoted by $\tilde{\psi}_t(x_t)$. At iteration t we then refer to the twisting functions as $(\psi_t(x_t), \tilde{\psi}_t(x_t))$. We will in general use the simplified notation $\psi_t(x_t) = \psi_t$ and $\tilde{\psi}_t(x_t) = \tilde{\psi}_t$, the twisting functions can then be denoted by $(\psi_t, \tilde{\psi}_t)$ at iteration t. In addition, we use the notation $(\psi, \tilde{\psi})$ when referring to twisting functions in general. We start by defining some general assumptions for the twisting functions, following Ala-Luhtala et al. (2016) and Guarniero, Johansen, and Lee (2017).

Assumptions 4.2.1. *We have the following assumptions about the lookahead functions*

- *ψ_t for $t = 1, \dots, T$ are continuous and real-valued*
- *$0 < \psi_t < \infty$ for $t = 1, \dots, T$*

and $\tilde{\psi}_T \equiv 1$ for the final normalising function.

These assumptions implies that we can define ψ with a lot of flexibility. We now consider the normalising function $\tilde{\psi}$ which can be defined by using the lookahead function ψ. The definition of the normalising function is motivated by normalising the altered transition density $\psi_{t+1}(x_{t+1})p(x_{t+1}|x_t)$. We will see that the normalised version of the altered transition density becomes what we will refer to as the twisted transition density. Motivated by this, we can define the normalising function, $\tilde{\psi}_t(x_t)$, at iteration $0 < t \leq T$ and the normalising constant $\tilde{\psi}_0$ at iteration $t = 0$.

Definition 4.2.1. *Given the lookahead functions $\psi_t(x_t)$, for $t = 1, \dots, T$, we define the normalising functions*

$$\tilde{\psi}_t(x_t) = \int \psi_{t+1}(x_{t+1})p(x_{t+1}|x_t)dx_{t+1}$$

for $t = 1, \dots, T-1$. For $t = 0$ we define the normalising constant

$$\tilde{\psi}_0 = \int \psi_1(x_1)p(x_1)dx_1.$$

We see that selecting a lookahead function $\psi_{t+1}(x_{t+1})$ conjugate to the transition density $p(x_{t+1}|x_t)$ will simplify calculations for the normalising function $\tilde{\psi}_t(x_t)$. This is because the integral often will be tractable and we can evaluate it analytically. The general twisting functions $(\psi, \tilde{\psi})$ have been defined under Assumptions 4.2.1 and the definition of the normalising function. Because the assumptions for the lookahead functions are relatively flexible, there are many valid functions that can be used as lookahead functions. The normalising functions then follow from the lookahead functions and result in valid twisting functions. One example is defining the lookahead functions by $\psi_t(x_t) = 1$ for $t = 1, \dots, T$ and all x_t. These lookahead functions are valid and hence the resulting twisting functions are valid. It further simplifies discussion to consider all the lookahead functions, ψ, as a sequence.

Definition 4.2.2. *Denoting $\psi_t(x_t) = \psi_t$, we have the sequence of lookahead functions*

$$\Psi_{1:T} \equiv \psi_1, \dots, \psi_T.$$

Defining twisting target distributions

One possible approach to twisting target distributions is given in Lindsten, Helske, and Vihola (2018) and Naesseth, Lindsten, and Schön (2019). With this approach we also have that $\tilde{\psi}_0 \equiv 1$. The intermediate twisting target distributions can be defined in any way we want as long as final target equivalence holds. We are defining the structure of the twisting target distributions motivated by the structure of the optimal target distributions in equation (4.2). We define the unnormalised twisting target distributions recursively and denote these by $\tilde{f}_t^{\psi}(x_{1:t})$ at iteration t. Adapted to the HMM setting and our notation from Naesseth, Lindsten, and Schön (2019, eq. 62), the twisting target distribution is then defined as

$$\tilde{f}_t^{\psi}(x_{1:t}) = \tilde{f}_{t-1}^{\psi}(x_{1:t-1})\phi_t(x_t, x_{t-1})\frac{\tilde{\psi}_t(x_t)}{\tilde{\psi}_{t-1}(x_{t-1})}. \tag{4.5}$$

The focus of this approach is the recursive structure and multiplication with the ratio of normalising functions. This approach has also been referred to as a multiplicative adjustment in Lindsten, Helske, and Vihola (2018). We can use any importance sampling distribution with this approach as discussed in Naesseth, Lindsten, and Schön (2019). In order to use this approach, we therefore only need to be able to evaluate the normalising functions. We can view this unnormalised twisting target distribution as an approximation to the unnormalised optimal target distribution in equation (4.2). Based on this we see that by setting $\tilde{\psi}_t(x_t) = p(y_{t+1:T}|x_t)$, then equation (4.5) is equal to equation (4.2). We also see that defining $\tilde{\psi}_t(x_t) = C$ for all t and all x_t, then the twisting target distribution in equation (4.5) is equal to the untwisted target distribution $p(x_{1:t}, y_{1:t})$.

Another approach to twisting target distributions is to combine twisting functions with a twisted model. This is the approach described in Guarniero, Johansen, and Lee (2017) where the twisted model is defined by twisted transition densities and twisted observation functions. We define this as the twisted model.

Definition 4.2.3. *The twisted model defined in equations (5) and (6) in Guarniero, Johansen, and Lee (2017) in our notation is defined as*

$$p_1^{\psi}(y_1|x_1) = \frac{p(y_1|x_1)\tilde{\psi}_1(x_1)\tilde{\psi}_0}{\psi_1(x_1)} \qquad p_1^{\psi}(x_1) = \frac{p(x_1)\psi_1(x_1)}{\tilde{\psi}_0}$$

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

$$p_t^{\psi}(y_t|x_t) = \frac{p(y_t|x_t)\tilde{\psi}_t(x_t)}{\psi_t(x_t)} \qquad p_t^{\psi}(x_t|x_{t-1}) = \frac{p(x_t|x_{t-1})\psi_t(x_t)}{\tilde{\psi}_{t-1}(x_{t-1})}.$$

Note that we need to have the twisting functions $(\psi, \tilde{\psi})$ before we can use the twisted model. In addition, we see that selecting $\psi_t(x_t)$ conjugate to $p(x_t|x_{t-1})$ is a significant advantage when wanting to simulate from $p_t^{\psi}(x_t|x_{t-1})$. We will mainly focus on the twisted model when defining twisting target distributions. It will however be useful to compare it to the recursively defined twisting target distribution in equation (4.5).

Properties of twisting target distributions

Now we consider some properties of the twisting target distributions that are defined by twisting functions. First we want to check that final target equivalence holds. We consider the recursively defined twisting target distribution in equation (4.5) and the unnormalised twisting target distribution that results from the twisted model. Recall that the final twisting target distributions should be equal to the final, untwisted and unnormalised, target distribution denoted by $p(x_{1:T}, y_{1:T})$. If we assume that $p(x_{1:T}, y_{1:T})$ can be seen as a product of factors then so should the unnormalised twisting target distribution at $t = T$ (Naesseth, Lindsten, and Schön, 2019, pp. 48-49). That is, $\tilde{f}_T^{\psi}(x_{1:T}) = p(x_{1:T}, y_{1:T})$. Consider first the recursively defined unnormalised twisting target distribution. Recall that that in this approach it is assumed that $\tilde{\psi}_0 \equiv 1$. We expand the expression from equation (4.5). We see that $\tilde{\psi}_t(x_t)$ for $t = 1, \ldots, T-1$ cancels and then we insert the assumptions to get the product of factors,

$$\begin{aligned}\tilde{f}_T^{\psi}(x_{1:T}) &= \phi_1(x_1)\frac{\tilde{\psi}_1(x_1)}{\tilde{\psi}_0}\phi_2(x_2,x_1)\frac{\tilde{\psi}_2(x_2)}{\tilde{\psi}_1(x_1)}\dots\phi_T(x_T,x_{T-1})\frac{\tilde{\psi}_T(x_T)}{\tilde{\psi}_{T-1}(x_{T-1})}\\ &= \phi_1(x_1)\frac{1}{\tilde{\psi}_0}\left[\prod_{t=2}^{T-1}\phi_t(x_t,x_{t-1})\right]\phi_T(x_T,x_{T-1})\tilde{\psi}_T(x_T)\\ &= p(y_1|x_1)p(x_1)\prod_{t=2}^{T}p(y_t|x_t)p(x_t|x_{t-1}) = p(x_{1:T},y_{1:T}).\end{aligned}$$

We can also consider the unnormalised twisting target distribution at $t = T$ from the twisted model in Definition 4.2.3. We then have

$$\begin{aligned}&p_1^{\psi}(y_1|x_1)p_1^{\psi}(x_1)\dots p_T^{\psi}(y_T|x_T)p_T^{\psi}(x_T|x_{T-1})\\ =&\frac{p(y_1|x_1)\tilde{\psi}_1(x_1)\tilde{\psi}_0}{\psi_1(x_1)}\frac{p(x_1)\psi_1(x_1)}{\tilde{\psi}_0}\dots\frac{p(y_T|x_T)\tilde{\psi}_T(x_T)}{\psi_T(x_T)}\frac{p(x_T|x_{T-1})\psi_T(x_T)}{\tilde{\psi}_{T-1}(x_{T-1})}\\ =&p(y_1|x_1)p(x_1)\prod_{t=2}^{T}p(y_t|x_t)p(x_t|x_{t-1}) = p(x_{1:T},y_{1:T}).\end{aligned}$$

This reformulation and cancellation for the final, unnormalised, distributions corresponds to part of the calculations in the proof of proposition 1 from Guarniero, Johansen, and Lee (2017). In both the approaches, we therefore see that at the final iteration, $t = T$, we have that the unnormalised twisting target distribution is equal to the unnormalised untwisted target distribution. That is, we have final target equivalence.

The recursively defined, unnormalised, twisting target distribution in equation (4.5) is closely related to the twisted model from Definition 4.2.3. We now want to consider these twisting target distributions at some iteration $t < T$ where they are not required to be equal to the unnormalised, untwisted, target distribution. We then have from the twisted model

$$\begin{aligned}&p_1^{\psi}(y_1|x_1)p_1^{\psi}(x_1)\prod_{s=2}^{t}p_s^{\psi}(y_s|x_s)p_s^{\psi}(x_s|x_{s-1})\\ =&p(y_1|x_1)p(x_1)\left[\prod_{s=2}^{t}p(y_s|x_s)p(x_s|x_{s-1})\right]\tilde{\psi}_t(x_t)\\ =&p(x_{1:t},y_{1:t})\tilde{\psi}_t(x_t).\end{aligned}$$

All the lookahead functions $\psi_s(x_s)$ for $s = 1,\dots,t$ cancels. In addition, the normalising constant $\tilde{\psi}_0$ cancels in the first twisted transition density and the first twisted observation function. Now we consider the recursively defined, unnormalised, twisting target distribution from equation (4.5). Here, we expand the recursion and use that $\tilde{\psi}_0 \equiv 1$. We then have that

$$\begin{aligned}\tilde{f}_t^{\psi}(x_{1:t}) &= \tilde{f}_{t-1}^{\psi}(x_{1:t-1})\phi_t(x_t,x_{t-1})\frac{\tilde{\psi}_t(x_t)}{\tilde{\psi}_{t-1}(x_{t-1})}\\ &= \left[\tilde{f}_{t-2}^{\psi}(x_{1:t-2})\phi_{t-1}(x_{t-1},x_{t-2})\frac{\tilde{\psi}_{t-1}(x_{t-1})}{\tilde{\psi}_{t-2}(x_{t-2})}\right]\phi_t(x_t,x_{t-1})\frac{\tilde{\psi}_t(x_t)}{\tilde{\psi}_{t-1}(x_{t-1})}\end{aligned}$$

$$\begin{aligned}
&= \left[\phi_1(x_1)\prod_{s=2}^{t-1}\phi_s(x_s, x_{s-1})\right]\phi_t(x_t, x_{t-1})\tilde{\psi}_t(x_t) \\
&= p(x_{1:t}, y_{1:t})\tilde{\psi}_t(x_t).
\end{aligned}$$

The recursively defined, unnormalised, twisting target distribution in equation (4.5) combined with $\tilde{\psi}_0 \equiv 1$ is equal to the unnormalised twisting target distribution from the twisted model at iteration t. This is because the lookahead functions $\psi_t(x_t)$ used in the twisted model are cancelling at every iteration and $\tilde{\psi}_0 \equiv 1$ in the recursively defined twisting target distribution.

We have seen that, when using twisting functions, we can twist the intermediate, untwisted, target distributions and have final target equivalence. The final target equivalence is useful because it also ensures that the likelihood when using twisting target distributions is equal to the likelihood for untwisted target distributions at $t = T$. Recall that we can define the marginal likelihood of the SSM by marginalising out the latent variables from the joint distribution over the latent variables and the observations. We follow Guarniero, Johansen, and Lee (2017) and refer to the likelihood integral with a joint distribution from twisting target distributions as Z^{ψ}, we then have that

$$\begin{aligned}
Z^{\psi} &= \int \dots \int \tilde{f}_T^{\psi}(x_{1:T})dx_{1:T} \\
&= \int \dots \int p(x_{1:T}, y_{1:T})dx_{1:T} = Z.
\end{aligned}$$

This implies that the maeginal likelihood at $t = T$ from using twisting target distributions is equal to the marginal likelihood from using untwisted target distributions. As long as final target equivalence holds, we can use the twisting target distributions to estimate the marginal likelihood. Conceptually, we can therefore alter the intermediate target distributions of our SSM of interest by using twisting functions as long as final target equivalence holds. The likelihood integral obtained from the twisted model will then be equal to the likelihood integral from the original model.

4.3 Optimal twisting functions

So far we have considered an optimal situation where it is assumed that we can simulate from the optimal importance sampling distribution in equation (4.3) and in addition evaluate the optimal target distribution. In general, we are not able to simulate from the optimal importance sampling distribution and unable to evaluate the optimal target distributions. In order to evaluate the optimal target distributions, we ideally would have closed form expressions for the predictive likelihood terms. We have considered general twisting functions and seen that there are many valid twisting functions within Assumptions 4.2.1. We now want to utilise the recursive structure in the optimal target distributions from equation (4.2) to motivate the configuration of twisting target distributions. This is following the same approach as in section 3 of Naesseth, Lindsten, and Schön (2019). We have recursively defined twisting target distributions in equation (4.5) and we now utilise that the twisting target distributions are defined with the same structure as the optimal target distributions.

We also note that the predictive likelihood term from the optimal target distribution is related to the backward information filter in equation (3.7). This can be seen with a straightforward reformulation

$$p(y_{t:T}|x_t) = p(y_t|x_t)p(y_{t+1:T}|x_t).$$

If the backward information filter and the transition density are conjugate, we see that simulation from the optimal importance sampling distribution in equation (4.4) can be tractable. From the recursively defined twisting target distributions in equation (4.5), we see that setting $\tilde{\psi}_t(x_t) = p(y_{t+1:T}|x_t)$ if possible means that the twisting target distributions are equal the optimal target distributions. We generally do not have access to the predictive likelihood terms or the backward information filter. We start with a reformulation of the predictive likelihood term $p(y_{t+1:T}|x_t)$ to see if we can use a similar structure for the normalising functions $\tilde{\psi}_t(x_t)$. This gives

$$\begin{aligned}
p(y_{t+1:T}|x_t) &= \int p(y_{t+1:T}, x_{t+1}|x_t)dx_{t+1} \\
&= \int p(y_{t+1}|x_{t+1})p(y_{t+2:T}|x_{t+1})p(x_{t+1}|x_t)dx_{t+1} \\
&= \int p(y_{t+1:T}|x_{t+1})p(x_{t+1}|x_t)dx_{t+1}.
\end{aligned}$$

From the second equality we see that the predictive likelihood terms follow a backward recursive structure. From the last equality we see that predictive likelihood term equals the expectation of the backward information filter at iteration $t + 1$ with respect to the transition density $p(x_{t+1}|x_t)$.

We are now interested in finding a similar recursive structure for the normalising functions that we so far have defined in general terms. The structure of the optimal target distributions is known and we can use this to find a recursive structure for the normalising functions. The optimal target distribution at t can always be defined as a marginalised version of the optimal target distribution at $t + 1$ because they are all marginal distributions of the final untwisted target distribution Naesseth, Lindsten, and Schön (2019, eq. 63). We then have the unnormalised optimal target distribution in recursive form,

$$\tilde{f}_t^*(x_{1:t}) = \int \tilde{f}_{t+1}^*(x_{1:t+1})dx_{t+1}. \tag{4.6}$$

We start with the unnormalised optimal target distribution at iteration t and $t+1$ in equation (4.6). We substitute these with the unnormalised twisting target distributions from equation (4.5). Recall that the twisting target distributions are defined using general normalising functions. The motivation is now to find a recursive structure for the normalising functions by utilising that we know the recursive structure of the optimal target distributions. This recursive structure is defined by the recursive structure from the unnormalised, optimal, target distributions. Inserting the unnormalised twisting target distributions that we defined in equation (4.5) into equation (4.6) then gives us

$$\tilde{f}_t^{\psi}(x_{1:t}) = \int \tilde{f}_t^{\psi}(x_{1:t})\phi_{t+1}(x_{t+1}, x_t)\frac{\tilde{\psi}_{t+1}(x_{t+1})}{\tilde{\psi}_t(x_t)}dx_{t+1}$$

$$= \frac{\tilde{f}_t^{\psi}(x_{1:t})}{\tilde{\psi}_t(x_t)} \int \phi_{t+1}(x_{t+1}, x_t)\tilde{\psi}_{t+1}(x_{t+1})dx_{t+1}.$$

Reformulating the expression above gives us the recursive structure for the normalising function $\tilde{\psi}_t(x_t)$. At iteration t we can express $\tilde{\psi}_t(x_t)$ recursively by using $\tilde{\psi}_{t+1}(x_{t+1})$. This can be further expanded and expressed with the transition and observation density,

$$\tilde{\psi}_t(x_t) = \int \phi_{t+1}(x_{t+1}, x_t)\tilde{\psi}_{t+1}(x_{t+1})dx_{t+1} \tag{4.7a}$$

$$= p(y_{t+1}|x_{t+1}) \int \tilde{\psi}_{t+1}(x_{t+1})p(x_{t+1}|x_t)dx_{t+1}. \tag{4.7b}$$

These normalising functions follow a recursive structure which was defined by the unnormalised optimal target distributions. Equation (4.7a) correspond to equation (64) in Naesseth, Lindsten, and Schön (2019) and we follow their convention and refer to these normalising functions as optimal with the notation $\tilde{\psi}_t^*(x_t)$. Based on this we can define the recursive structure for the optimal normalising functions.

Definition 4.3.1. *For* $t = 1, \ldots, T-1$

$$\tilde{\psi}_t^*(x_t) = \int \phi_{t+1}(x_{t+1}, x_t)\tilde{\psi}_{t+1}^*(x_{t+1})dx_{t+1}$$

and for $t = T$

$$\tilde{\psi}_T^*(x_T) \equiv 1$$

Closed form expressions for $\tilde{\psi}_t^*(x_t)$ are generally not be available as the integral is often intractable. The recursive definition also allows us to consider the motivation for the optimal normalising functions from a different aspect. We begin by the assumption $\tilde{\psi}_T^*(x_T) \equiv 1$ and insert this into Definition 4.3.1. We assume that we are able to evaluate the integrals that follow,

$$\tilde{\psi}_{T-1}^*(x_{T-1}) = \int \phi_T(x_T, x_{T-1})\tilde{\psi}_T^*(x_T)dx_T$$
$$= \int p(y_T|x_T)p(x_T|x_{T-1})dx_T = p(y_T|x_{T-1}).$$

Which we again can insert into the expression for $\tilde{\psi}_{T-2}(x_{T-2})$ and so on. Continuing this backward recursion with $t = T, \ldots, 1$ implies the general form

$$\tilde{\psi}_t^*(x_t) = \int p(y_{t+1:T}|x_{t+1})p(x_{t+1}|x_t)dx_{t+1} = p(y_{t+1:T}|x_t) \tag{4.8}$$

which we recall as the predictive likelihood from the optimal target distributions. Recall the general definition of $\tilde{\psi}_t(x_t)$, given by Definition 4.2.1 as

$$\tilde{\psi}_t(x_t) = \int \psi_{t+1}(x_{t+1})p(x_{t+1}|x_t)dx_{t+1}.$$

This implies the form for the optimal lookahead function $\psi_t^*(x_t)$ by setting the integrals equal to each other, $\tilde{\psi}_t^*(x_t) = \tilde{\psi}_t(x_t)$. This implies

$$\psi_t^*(x_t) = p(y_{t:T}|x_t). \tag{4.9}$$

Another motivation for the sequence of optimal lookahead functions, denoted by $\Psi^*_{1:T}$, can be seen in proposition 2 of Guarniero, Johansen, and Lee (2017). Using these optimal lookahead functions with a twisted model results in a likelihood estimate $\hat{Z}^{\psi}$ being equal to Z with Pr. = 1 (Guarniero, Johansen, and Lee, 2017). We now consider this sequence of lookahead functions in our notation.

Definition 4.3.2. *The optimal lookahead functions from proposition 2 in Guarniero, Johansen, and Lee (2017) are for* $t = 1, \dots, T-1$

$$\psi^*_t(x_t) = p(y_t|x_t) \int \dots \int \prod_{s=t+1}^{T} p(y_s|x_s) \prod_{s=t+1}^{T} p(x_s|x_{s-1}) dx_{t+1:T}$$

and for $t = T$

$$\psi^*_T(x_T) = p(y_T|x_T).$$

The optimal normalising constant $\tilde{\psi}^*_0$ has a special interpretation when this sequence of optimal lookahead functions is used. To see this, we consider the standard definition of the normalising constant, $\tilde{\psi}_0$, and follow the proof of proposition 2 from Guarniero, Johansen, and Lee (2017) in our notation. From the definition of the normalising constant $\tilde{\psi}_0$, we can insert the optimal lookahead function at $t = 1$. We then have

$$\begin{aligned}\tilde{\psi}^*_0 &= \int \psi^*_1(x_1)p(x_1)dx_1 \\ &= \int p(y_1|x_1) \left[\int \dots \int \prod_{s=2}^{T} p(y_s|x_s) \prod_{s=2}^{T} p(x_s|x_{s-1}) dx_{2:T} \right] p(x_1)dx_1 \\ &= \int \dots \int p(x_{1:T}, y_{1:T}) dx_{1:T} = Z.\end{aligned}$$

We see that the last integral is equal to the likelihood given in equation (3.8). We also consider an alternative recursive definition of the optimal lookahead function $\psi^*_t(x_t)$, which will be useful. It is possible to show that for $t = 1, \dots, T$ we have

$$\psi^*_t(x_t) = p(y_t|x_t)\tilde{\psi}^*_t(x_t). \tag{4.10}$$

See Appendix A.2 for calculations. This is also stated in the proof of proposition 2 in Guarniero, Johansen, and Lee (2017). We can now summarise the optimal twisting functions and their structure,

$$\begin{aligned}\psi^*_t(x_t) &= p(y_{t:T}|x_t) \\ \tilde{\psi}^*_t(x_t) &= \int \psi^*_{t+1}(x_{t+1})p(x_{t+1}|x_t)dx_{t+1} \\ &= \int p(y_{t+1:T}|x_{t+1})p(x_{t+1}|x_t)dx_{t+1} = p(y_{t+1:T}|x_t).\end{aligned}$$

This is the same setup as in Guarniero, Johansen, and Lee (2017). Because we can alter the sequence of intermediate, $t < T$, target distributions as long as the final twisting target distribution $t = T$ is equal to the distribution of interest there are several alterations possible. We have now considered general twisting functions $(\psi, \tilde{\psi})$ and the optimal twisting functions with respect to minimising variance of the likelihood estimates $(\psi^*, \tilde{\psi}^*)$.

4.4 Twisting target distributions in the particle filter framework

Utilising twisting target distributions in a PF context is straightforward when we have defined the twisting functions and a twisted model. The twisting functions can hypothetically, and in some special cases, be the optimal twisting functions. These are however generally not available in closed form and we use approximations of the optimal twisting functions which are denoted by $(\bar{\psi}, \tilde{\bar{\psi}})$. When discussing twisting functions in particle filters in general, we will however use the generic notation $(\psi, \tilde{\psi})$ for notational simplicity. In this section we mainly follow the approach of Guarniero, Johansen, and Lee (2017) by using a twisted model which defines a sequence of twisted transition densities and twisted observation functions. When used in a BPF we often refer to this setup as a twisted particle filter.

When using the twisted model in a particle filter framework we set the importance sampling function equal to the twisted transition density. That is, we simulate variables from $p_1^{\psi}(x_1)$ for $t = 1$ and from $p_t^{\psi}(x_t|x_{t-1})$ for $t \geq 2$. Assume that we can easily simulate variables from the transition densities $p(x_1)$ and $p(x_t|x_{t-1})$. To easily simulate from the twisted transition densities, we need $\psi_1(x_1)$ to be conjugate to $p(x_1)$ and $\psi_t(x_t)$ to be conjugate to $p(x_t|x_{t-1})$. Also recall the optimal importance sampling distributions from equation (4.3). For $t > 1$ and $t = 1$ we see that the twisted transition densities defined in Definition 4.2.3 can be viewed as approximating the optimal importance sampling distributions. That is

$$p_1^{\psi}(x_1) \approx g_1^*(x_1|y_1) \qquad\qquad p_t^{\psi}(x_t|x_{t-1}) \approx g_t^*(x_t|x_{1:t-1}, y_{1:t}).$$

The transition densities $p(x_t|x_{t-1})$ are given in both the twisted transition densities and the optimal importance sampling distributions. This also motivates the approximation of the optimal twisting functions $(\psi_t^*, \tilde{\psi}_t^*)$ because when inserted into the twisted transition densities, we are approximating the optimal importance sampling distribution. In this manner when ψ is close to ψ^*, the twisted transition density in this case is close to the optimal importance sampling distribution. When we use the twisted transition density as the importance sampling function, we have that the weight update functions become

$$u_1 = p_1^{\psi}(y_1|x_1) \qquad\qquad u_t = p_t^{\psi}(y_t|x_t).$$

This follows from the fact that we are using the twisted transition density as the importance sampling function.

For completeness, we include a slightly modified version of algorithm 5 from Guarniero, Johansen, and Lee (2017) in our notation. We refer to this as a twisted BPF with n particles. The notation t_{start} and t_{stop} is used to denote the first and last iteration in the algorithm. In an offline setting we set $t_{\text{start}} = 1$ and $t_{\text{stop}} = T$. We will see in Chapter 5 that this algorithm can be used also for subsets of the iterations. To use it for a subset of the iterations, we only need the simulated variables x_{t-1}^i and importance weights $\tilde{w}_{t-1}^i$ for $i = 1, \ldots, n$ if $t = t_{\text{start}} > 1$. We use β as a resampling threshold and calculate the ESS using $\tilde{w}_{t-1}^i$ for $i = 1, \ldots, n$ which is denoted by $\text{ESS}(\tilde{w}_{t-1})$ at line 7.

Algorithm 6 Twisted BPF

```
for $t \in (t_{\text{start}}, \dots, t_{\text{stop}})$ do
    for $i \in (1, \dots, n)$ do
        if $t = 1$ then
            $x_1^i \sim p_1^\psi(x_1)$
            $\tilde{w}_1^i = p_1^\psi(y_1|x_1^i)$
        else
            if $\text{ESS}(\tilde{w}_{t-1}) < \beta \cdot n$ then
                $x_t^i \sim \sum_{i=1}^n w_{t-1}^i p_t^\psi(x_t|x_{t-1}^i)$
                $\tilde{w}_t^i = p_t^\psi(y_t|x_t^i)$
            else
                $x_t^i \sim p_t^\psi(x_t|x_{t-1}^i)$
                $\tilde{w}_t^i = \tilde{w}_{t-1}^i p_t^\psi(y_t|x_t^i)$
            end if
        end if
    end for
    $w_t^i = \frac{\tilde{w}_t^i}{\sum_{l=1}^n \tilde{w}_t^l}$ for $i = 1, \dots, n$
end for
```

Likelihood estimation

To estimate the marginal likelihood using Algorithm 6 with n particles we follow Guarniero, Johansen, and Lee (2017). When we resample at every iteration or using adaptive resampling we estimate the marginal likelihood by

$$\hat{Z}_T^\psi = \prod_{s=1}^T \left[\frac{1}{n} \sum_{i=1}^n p_s^\psi(y_s|x_s^i) \right] \tag{4.11a}$$

$$\hat{Z}_T^\psi = \left[\frac{1}{n} \sum_{i=1}^n \tilde{w}_T^i \right] \prod_{\text{ESS}} \left[\frac{1}{n} \sum_{i=1}^n \tilde{w}_t^i \right] \tag{4.11b}$$

respectively. When using adaptive resampling, the condition ESS in the product denotes whether $\text{ESS}(\tilde{w}_t)$ indicated resampling in line 7 at iteration t. Both methods provide unbiased estimates of the marginal likelihood (Guarniero, Johansen, and Lee, 2017).

Now we can consider the sequence $\Psi_{1:T}^*$ of optimal lookahead functions yielding that the recursive structure from equation (4.10) holds exactly. Assume that we are resampling at every iteration, then we use equation (4.11a). Assume additionally that the optimal twisting functions are available, we have the following estimate for the marginal likelihood

$$\begin{aligned}
\hat{Z}_T^\psi &= \left[\frac{1}{n} \sum_{i=1}^n \frac{p(y_1|x_1^i)\tilde{\psi}_1^*(x_1^i)\tilde{\psi}_0^*}{\psi_1^*(x_1^i)} \right] \prod_{s=2}^T \left[\frac{1}{n} \sum_{i=1}^n \frac{p(y_s|x_s^i)\tilde{\psi}_s^*(x_s^i)}{\psi_s^*(x_s^i)} \right] \\
&= \left[\frac{1}{n} \sum_{i=1}^n \frac{p(y_1|x_1^i)\tilde{\psi}_1^*(x_1^i)\tilde{\psi}_0^*}{p(y_1|x_1^i)\tilde{\psi}_1^*(x_1^i)} \right] \prod_{s=2}^T \left[\frac{1}{n} \sum_{i=1}^n \frac{p(y_s|x_s^i)\tilde{\psi}_s^*(x_s^i)}{p(y_s|x_s^i)\tilde{\psi}_s^*(x_s^i)} \right] \\
&= \tilde{\psi}_0^*
\end{aligned}$$

By inserting the recursive structure $\psi_t^*(x_t) = p(y_t|x_t)\tilde{\psi}_t^*(x_t)$ we see that the contribution to the marginal likelihood from all iterations $t \geq 2$ sum to 1. In addition, the denominator of the contribution from the first iteration cancels everything except for $\tilde{\psi}_0^*$ because of the recursive structure that holds exactly when we have the optimal twisting functions. We then have $\hat{Z}^\psi = \tilde{\psi}_0^* = Z$ with Pr. $= 1$ which is the result provided in proposition 2 of Guarniero, Johansen, and Lee (2017).

If we are using twisting functions $(\psi, \tilde{\psi})$ that are not the optimal twisting functions, the recursive structure generally does not hold exactly. As a result of this we do not get the perfect cancellation in the likelihood estimates that we do with the optimal twisting functions. This can also be a motivation to consider twisting functions where the recursive structure holds for as many of the iterations $t = 1, \ldots, T$ as possible, but not necessarily all.

4.5 Approximation of the optimal twisting functions

As discussed in Naesseth, Lindsten, and Schön (2019), the optimal normalising functions in Definition 4.3.1 are rarely available in closed form. Because of this we usually need to approximate the sequence of optimal twisting functions. The twisting functions can also be approximated with a constant lookahead, that is defining the lookahead function with a constant C future observations relative to the current iteration. The lookahead functions will in this case have the conceptual form $p(y_{t:t+C}|x_t)$ at iteration t, see e.g. Ala-Luhtala et al. (2016). Other approaches for approximating twisting functions include numerical optimisation in Guarniero, Johansen, and Lee (2017), Heng et al. (2020), and Naesseth, Lindsten, and Schön (2019). In addition, they can be approximated by expectation propagation, belief propagation and Laplace approximations as discussed in Lindsten, Helske, and Vihola (2018).

When approximating the lookahead functions, we will follow the same approach as in Guarniero, Johansen, and Lee (2017) and Naesseth, Lindsten, and Schön (2019) by utilising the recursive structure from equation (4.10). A local case where the recursive structure Definition 4.3.1 only is true for one iteration ahead is also considered in Naesseth, Lindsten, and Schön (2019). This coincides with the fully adapted case which is also discussed in e.g. Creal (2012) and Johansen and Doucet (2008). We focus more on the general case where we assume that the recursive structure holds until $t = T$. In general, we will use the notation $(\bar{\psi}, \tilde{\bar{\psi}})$ when we are referring to approximations of twisting functions.

Defining the approximate twisting functions

We now want to define approximations of the optimal twisting functions which we denote by $(\bar{\psi}_t, \tilde{\bar{\psi}}_t)$ at iteration t and refer to as approximate twisting functions. We can utilise the recursive structure in Definition 4.3.1 which was derived based on the optimal target distributions. We define the approximate twisting functions to approximately follow the same recursive structure as the optimal twisting functions $(\psi_t^*, \tilde{\psi}_t^*)$ at iteration t. We start with the definition of the approximate normalising function $\tilde{\bar{\psi}}_t(x_t)$ which is an analogue definition to

$\tilde{\psi}_t(x_t)$ in Definition 4.2.1. We then have

$$\tilde{\bar{\psi}}_t(x_t) = \int \bar{\psi}_{t+1}(x_{t+1})p(x_{t+1}|x_t)dx_{t+1} \qquad \text{for } t = 1, \ldots, T-1 \tag{4.12}$$

$$\tilde{\bar{\psi}}_T(x_T) \equiv 1 \tag{4.13}$$

which corresponds to Naesseth, Lindsten, and Schön (2019, eq. 74) in our notation. We know that the optimal normalising functions follow the recursive structure in Definition 4.3.1. The general idea for deriving the approximations $(\bar{\psi}, \tilde{\bar{\psi}})$ is to use this recursive structure of the optimal normalising functions. To define a similar recursive structure in the approximate twisting functions, we set the integrals $\tilde{\bar{\psi}}_t(x_t) = \tilde{\psi}^*_t(x_t)$ for $t = T-1, \ldots, 1$. To find the approximate lookahead function $\bar{\psi}_{t+1}(x_{t+1})$, we then see what expression in the optimal integral $\tilde{\psi}^*_t(x_t)$ the approximate lookahead function must estimate. We utilise the optimal recursive structure and the defined recursive structure combined. We start at $t = T$ and follow the backward recursive structure in order to find the approximations, see Appendix A.2 for the start of the recursion. To find the approximation $\bar{\psi}_t(x_t)$ we have that

$$\begin{aligned}
\tilde{\psi}^*_{t-1}(x_{t-1}) &= \int \phi_t(x_t, x_{t-1})\tilde{\psi}^*_t(x_t)dx_t \\
&= \int p(y_t|x_t)\tilde{\psi}^*_t(x_t)p(x_t|x_{t-1})dx_t \\
\tilde{\bar{\psi}}_{t-1}(x_{t-1}) &= \int \bar{\psi}_t(x_t)p(x_t|x_{t-1})dx_t.
\end{aligned}$$

This implies that the approximations $\bar{\psi}_t(x_t)$ can be formulated as

$$\begin{aligned}
\bar{\psi}_t(x_t) &= p(y_t|x_t)\tilde{\psi}^*_t(x_t) \\
&\approx p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t).
\end{aligned}$$

We see that the approximate twisting functions $(\bar{\psi}_t, \tilde{\bar{\psi}}_t)$ then approximate the recursive structure of the optimal twisting functions $(\psi^*_t, \tilde{\psi}^*_t)$ in equation (4.10). We provide a definition of the approximate lookahead function, $\bar{\psi}_t(x_t)$, similar to equation (75) in Naesseth, Lindsten, and Schön (2019) and the recursive estimation in proposition 4 of Guarniero, Johansen, and Lee (2017).

Definition 4.5.1. *The approximate twisting functions are defined by*

$$\begin{aligned}
\bar{\psi}_t(x_t) &\approx p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t) \qquad && t = 1, \ldots, T-1 \\
\bar{\psi}_T(x_T) &\approx p(y_T|x_T).
\end{aligned}$$

Ideally we would like equality in Definition 4.5.1, but in general this will be intractable because we would need $\tilde{\bar{\psi}}_t(x_t)$ to be conjugate to the observation density in order for $\bar{\psi}_t(x_t)$ to be used in the next approximate normalising function $\tilde{\bar{\psi}}_{t-1}(x_{t-1})$ and so on. By following the approach of Guarniero, Johansen, and Lee (2017) we therefore find the closest approximation $\bar{\psi}_t(x_t)$ to $p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$ where the approximate lookahead function is conjugate to $p(x_t|x_{t-1})$. Then we can find a closed form expression for the next integral

$\tilde{\bar{\psi}}_{t-1}(x_{t-1})$ in the backward recursive approximation. We see from this definition that if we are able to calculate the approximations recursively and we have the approximation $\bar{\psi}_{t+1}(x_{t+1})$ we can use equation (4.12) in order to calculate $\tilde{\bar{\psi}}_t(x_t)$. In this sense Definition 4.5.1 and the approximate normalising function in equation (4.12), constitutes an analogue definition to Definition 4.3.1. We see this by

$$\begin{aligned}\tilde{\bar{\psi}}_t(x_t) &= \int \bar{\psi}_{t+1}(x_{t+1})p(x_{t+1}|x_t)dx_{t+1}\\ &\approx \int p(y_{t+1}|x_{t+1})\tilde{\bar{\psi}}_{t+1}(x_{t+1})p(x_{t+1}|x_t)dx_{t+1}\\ &= \int \phi_{t+1}(x_{t+1},x_t)\tilde{\bar{\psi}}_{t+1}(x_{t+1})dx_{t+1}.\end{aligned}$$

Which we see approximates the same recursive structure as the optimal $\tilde{\psi}^*_t(x_t)$.

4.6 Calculating the approximations

There are several possibilities for calculating the approximate twisting functions $(\bar{\psi}, \tilde{\bar{\psi}})$. Some of these are reviewed in e.g. Ala-Luhtala et al. (2016), Guarniero, Johansen, and Lee (2017), Heng et al. (2020), and Lindsten, Helske, and Vihola (2018). We mainly focus on parametric approximation following Guarniero, Johansen, and Lee (2017) in this section, but briefly consider some aspects of the nonparametric approximation.

Nonparametric approximation

Defining the approximation $\bar{\psi}_t$ for $t = 1, \dots, T$ through nonparametric approximation is also possible and there are different methods for doing this, see e.g. chapter 6.6 in Hastie, Tibshirani, and J. Friedman (2009) or chapter 10.2 in Givens and Hoeting (2013). Evaluation of a nonparametric approximation $\bar{\psi}$ is however generally computationally costly compared to a parametric approximation. An intuitive advantage with using a nonparametric approximation is that we have a more flexible approximation compared to the parametric approximation. We do not need to specify a parametric class $\bar{\Psi}$ in advance that restrict the form of $\bar{\psi}$.

Parametric approximation

The parametric approximation defines a class $\bar{\Psi}$ of functions on a specific parametric form and we denote the parametric approximation by $\bar{\psi}_t(x_t, \theta_t) \in \bar{\Psi}$. Here θ_t denotes the parameters used in the parametric form for the parametric approximation at iteration t. The parameters θ_t define the parametric approximation and we focus on estimating these. To motivate how we can estimate the parameters, recall the approximate recursive structure Definition 4.5.1 of $\bar{\psi}_t$, where we have substituted in the parametric form

$$\begin{aligned}\bar{\psi}_t(x_t, \theta_t) &\approx p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t, \theta_{t+1}) \qquad (4.14)\\ &= p(y_t|x_t)\int \bar{\psi}_{t+1}(x_{t+1}, \theta_{t+1})p(x_{t+1}|x_t)dx_{t+1}.\end{aligned}$$

Following the backward recursive structure, at iteration t we have all the terms in $\tilde{\bar{\psi}}_t(x_t, \theta_{t+1})$ available. One choice of class, $\bar{\Psi}$, suggested in Guarniero, Johansen, and Lee (2017) is such that the parametric approximations $\bar{\psi}_t(x_t, \theta_t)$ are conjugate to the transition density. This ensures that the integrand also has a parametric form (Gelman et al., 2013). Because of this we can find a closed form expression for the integral $\tilde{\bar{\psi}}_t(x_t, \theta_{t+1})$. Combining this with the observation density, we can evaluate all the term on the right-hand side of equation (4.14). We often refer to the right hand-side as the target and denote this with

$$\dot{\psi}_t(x_t) = p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t, \theta_{t+1}).$$

Selecting a conjugate form for $\bar{\psi}_t(x_t, \theta_t)$ for $t = 1, \ldots, T$ is useful because we then can evaluate $\dot{\psi}_t(x_t)$ in closed form. We note that the conjugacy between approximate lookahead function and the transition density is especially useful due to the analytical evaluation of the integral. When we are at iteration t of the backwards recursive structure, the parameters θ_{t+1} are available and we now consider how θ_t can be estimated. We define a distance function, denoted by D, to measure the distance between $\bar{\psi}_t(x_t, \theta_t)$ and $\dot{\psi}_t(x_t)$. We then minimise D with respect to θ_t in order to find the parameters that minimises the distance. The parameters that minimise the distance are denoted by $\bar{\theta}_t$, we then have

$$\bar{\theta}_t = \text{argmin}_{\theta_t} D(\theta_t). \tag{4.15}$$

We denote the distance function using only θ_t as a parameter for notational simplicity. Note however that x_t^i for $i = 1, \ldots, n$ and θ_{t+1} if $t < T$ are required to evaluate the distance function.

Distance functions

There are several possibilities when it comes to choice of distance function D. The distance function denoted by D_{iAPF} is given in section 5.1 of Guarniero, Johansen, and Lee (2017). With our notation and setup, this is given by

$$D_{\text{iAPF}} = \sum_{i=1}^{n} \left[\bar{\psi}_t(x_t^i, \theta_t) - \lambda_t \dot{\psi}_t(x_t^i)\right]^2 + r_t(x_t^{1:n}, \theta_t, \lambda_t). \tag{4.16}$$

The function r_t is a regularisation function which can be included in order to avoid trivial solutions $\bar{\theta}_t$ when minimising the distance function D. A trivial solution in this context is e.g. $\bar{\theta}_t$ which contains a variance parameter that yields $\bar{\psi}_t(x_t, \bar{\theta}_t)$ so diffuse that it is close to 0 at every x_t obtained from the particles. Another possible distance function is given in Naesseth, Lindsten, and Schön (2019, eq. 78) where the normalised weights are used to define a weighted distance.

During our numerical simulations, primarily when x_t were scalars, we saw that D_{iAPF} when used with $r_t(x_t^{1:n}, \theta_t, \lambda_t) = 0$ for $t = 1, \ldots, T$ could be sensitive to initial values in the numerical minimisation. This is discussed further in Section 4.9. Related to this sensitivity we used a slightly altered version of the distance function D_{iAPF}. We denote this by D_{iAPF2} and it is given by

$$D_{\text{iAPF2}} = \sum_{i=1}^{n} \left[\lambda_t \bar{\psi}_t(x_t^i, \theta_t) - \dot{\psi}_t(x_t^i)\right]^2. \tag{4.17}$$

In the case of scalar variables, heuristics implies that this distance function, in our specific numerical optimisation framework, tended to be numerically more stable. This is mainly compared to the distance function D_{iAPF}.

We now present a recursive algorithm that can be used to calculate parametric approximations $(\bar{\psi}, \tilde{\bar{\psi}})$. The algorithm is adapted from algorithm 3 and equation (15) in Guarniero, Johansen, and Lee (2017). However this algorithm may run backwards from an arbitrary t_{stop} to an arbitrary $t_{\text{start}} < t_{\text{stop}}$. In an offline setting we set $t_{\text{stop}} = T$ and $t_{\text{start}} = 1$ to calculate all the twisting functions. The algorithm is presented for parametric approximations and we assume that $\dot{\psi}_t(x_t)$ can be calculated within the distance function D.

Algorithm 7 Recursive estimation of twisting functions

```
1: for $t \in (t_{\text{stop}}, \dots, t_{\text{start}})$ do
2:     if $t = t_{\text{stop}}$ then
3:         $\tilde{\bar{\psi}}_t(x_t) = 1$
4:     else
5:         $\tilde{\bar{\psi}}_t(x_t) = \int \bar{\psi}_{t+1}(x_{t+1}, \bar{\theta}_{t+1}) p(x_{t+1}|x_t) dx_{t+1}$
6:     end if
7:     $\bar{\theta}_t = \text{argmin}_{\theta_t} D(\theta_t)$
8: end for
```

Numerical minimisation

After selecting a class $\bar{\Psi}$ and a distance function D, we need to find the parameters $\bar{\theta}_t$. When $\bar{\Psi}$ and D are selected, the problem in equation (4.15) is a general numerical optimisation problem. The θ_t is normally a vector of parameters defining the approximate lookahead function. In the case of defining $\bar{\psi}$ as a mixture of distributions or a multivariate distribution it consists of the parameters for several mixture components or dimensions respectively. Our main approach to numerically minimise (4.15) has been based on Newton's method which is conceptually defined in Givens and Hoeting (2013, eq. 2.33). This is given here in our notation

$$\theta_t^{j+1} = \theta_t^j - (H(\theta_t^j))^{-1} \nabla D(\theta_t^j).$$

The term $\nabla D(\theta_t^j)$ denotes the gradient of the distance function while H denotes the Hessian matrix at the numerical optimisation iteration j. We use Newton-like methods for the numerical minimisation, see e.g. chapter 2 of Givens and Hoeting (2013). We do not go into details about the numerical optimisation, but some general expressions that can be used for Newton-like optimisation methods can be found in Appendix A.2. See e.g. Givens and Hoeting (2013) and Spall (2003) for general approaches to numerical optimisation problems.

Initial values in the minimisation

Assume that we use a parametric approximation $\bar{\psi}_t(x_t, \theta_t)$ consisting of a single Gaussian. For the numerical optimisation we need initial values for the parameters θ_t and λ_t if we use the distance function D_{iAPF}. We want the expectation parameter of $\bar{\psi}_t(x_t, \theta_t)$ to be close to the mode of $\dot{\psi}_t(x_t)$. We therefore extract a subset of x_t^i for $i = 1, \dots, n$ that has the highest values of $\dot{\psi}_t(x_t)$ and use the mean of these values as the initial value for μ_t.

For σ_t^2 we use a SIR approach with the importance weight set to $\tilde{w}_t = \frac{\dot{\psi}_t(x_t)}{w_t}$ where w_t is the normalised weights for x_t. In practice, we often resampled x_t according to $w_t \propto \tilde{w}_t = \dot{\psi}_t(x_t)$ to avoid storing the normalised weights. We then calculate the sample variance. Then we add a constant C and use the result as the initial value. We usually set $C = 1$ in the numerical experiments. The motivation for adding the constant C to the initial value of the variance is related to starting the optimisation from a more uninformative initial distribution.

As an initial value for λ_t we solve $\frac{\partial}{\partial \lambda_t} D_{\text{iAPF}} = 0$, then insert the initial values of μ_t and σ_t^2. In the case of using the distance function D_{iAPF}, this implies that

$$\lambda_t = \frac{\sum_{i=1}^n \bar{\psi}_t(x_t^i, \theta_t)\dot{\psi}_t(x_t^i)}{\sum_{i=1}^n \dot{\psi}_t(x_t^i)^2} \tag{4.18}$$

can be used as an initial value for λ_t.

4.7 Lookahead setting

We have so far seen two main approaches to approximating the optimal twisting functions, parametric and nonparametric. Common for these are that all the observations $y_{1:T}$ are required due to the backwards recursive structure in Algorithm 7. The optimal lookahead functions are similar in character to the general lookahead strategies in Lin, Chen, and J. S. Liu (2013). It is therefore natural to consider if one can approximate the optimal lookahead function ψ^* with a general constant lookahead strategy. Recall the constant lookahead setting, that is, we assume that we have the observations $y_{1:t+C}$ available at iteration t where C is a positive constant. In contrast to the offline setting discussed in e.g. Guarniero, Johansen, and Lee (2017), Heng et al. (2020), and Naesseth, Lindsten, and Schön (2019), we are here restricted to a constant lookahead of C observations relative to iteration t. This constant lookahead setting is part of a more general framework often referred to as lookahead strategies (Lin, Chen, and J. S. Liu, 2013).

Unless the lookahead is $C \geq T - t$ for $t = 1, \dots, T$, we are not in the offline setting. If one use a constant lookahead of c iterations, Guarniero, Johansen, and Lee (2017) argues that ψ becomes constant lookahead functions and therefore not optimal lookahead functions. If we use a constant lookahead of C iterations, we have that the recursive structure from the optimal lookahead functions $\psi_t^*(x_t) = p(y_t|x_t)\tilde{\psi}_t^*(x_t)$ may not hold. Here, we denote the constant lookahead functions by $\psi_t^c(x_t) = p(y_{t:t+C}|x_t)$ at iteration t. We start from the right-hand side of the recursive structure. Substituting $\psi_{t+1}^c(x_{t+1})$ into the definition of the normalising function in equation (4.2.1) gives

$$\begin{aligned} p(y_t|x_t)\tilde{\psi}_t^c(x_t) &= p(y_t|x_t)\int \psi_{t+1}^c(x_{t+1})p(x_{t+1}|x_t)dx_{t+1} \\ &= p(y_t|x_t)\int p(y_{t+1:t+C+1}|x_{t+1})p(x_{t+1}|x_t)dx_{t+1} \\ &= p(y_{t:t+C+1}|x_t). \end{aligned}$$

We see that this do not correspond to the left hand side $\psi_t^c(x_t) = p(y_{t:t+C}|x_t)$. With a constant lookahead of C iterations, we therefore have that the recursive structure that follows from the optimal twisting functions may not hold.

The constant lookahead setting, with a constant lookahead of C future observations, corresponds to the delayed-sample method (Guarniero, Johansen, and Lee, 2017). From section 4.2 of Lin, Chen, and J. S. Liu (2013) we have that the delayed-sample method uses an importance sampling distribution that in our notation correspond to $p(x_t|x_{1:t-1}, y_{1:t+C}) = p(x_t|x_{t-1}, y_{t:t+C})$. Assume now that we use a constant lookahead function $\psi_t^c(x_t)$. We can check that by using this constant lookahead function in the twisted transition density we have

$$p_t^{\psi}(x_t|x_{t-1}) = p(x_t|x_{t-1}, y_{t:t+C}).$$

In the constant lookahead setting with C iterations of lookahead at all iterations, the importance sampling distribution in the twisted model is equal to the importance sampling distribution in the delayed-sample method. This is seen by the following at iteration $t \geq 2$ and with constant lookahead of $C \leq T - t$. We then have

$$\begin{aligned} p_t^{\psi}(x_t|x_{t-1}) &= \frac{p(x_t|x_{t-1})p(y_{t:t+C}|x_t)}{p(y_{t:t+C}|x_{t-1})} \\ &= \frac{p(x_t, x_{t-1})p(y_{t:t+C}|x_t, x_{t-1})}{p(y_{t:t+C}, x_{t-1})} \\ &= p(x_t|x_{t-1}, y_{t:t+C}). \end{aligned}$$

For iteration $t = 1$, we first consider the normalising constant $\tilde{\psi}_0^c$. In the constant lookahead setting this becomes

$$\tilde{\psi}_0^c = \int \psi_1^c(x_1)p(x_1)dx_1 = p(y_{1:1+C}).$$

We then insert $\tilde{\psi}_0^c$ into $p_1^{\psi}(x_1)$ from the twisted model

$$p_1^{\psi}(x_1) = \frac{p(x_1)\psi_1^c(x_1)}{\tilde{\psi}_0^c} = \frac{p(y_{1:1+C}, x_1)}{p(y_{1:1+C})} = p(x_1|y_{1:1+C}).$$

From this we see that when using lookahead functions with a constant lookahead of C iterations the twisted transition densities correspond to the importance sampling function from the delayed-sample method. The fact that the recursive structure from the optimal lookahead functions does not hold is also part of the motivation for utilising twisting target distributions in the batch setting as we consider in Chapter 5.

4.8 Iterated auxiliary particle filter

Here we briefly review the iAPF from algorithm 4 in Guarniero, Johansen, and Lee (2017). This is using an iteratively improving approximation of $\bar{\psi}$ with two stages: run a particle filter with approximate twisting functions in a twisted model to obtain a set of particles, then improve the lookahead functions $\bar{\psi}$ by using the obtained particles. A stopping criterion for the iteratively improving approximations and a dynamic number of particles is used. The number of particles at iteration i of the algorithm, n^i, is determined by the last k estimates of the likelihood $\hat{Z}^{\psi}$ from the particle filter. This is done by checking if the sequence of likelihood estimates are monotonically increasing which is done in the generic function mi. The algorithm starts with an initial n^0 number of particles, but this may increase during the run of the algorithm. In order to measure the stability of the likelihood estimates, the condition $\frac{\text{sd}(\hat{z}^k)}{\text{mean}(\hat{z}^k)} < \tau$ is checked with τ being a specified threshold. The variable $\hat{z}^k$ is the sequence of the last k likelihood estimates. We also assume that the first set of n^0 sequences $(x^1_{1:T}, \dots, x^{n^0}_{1:T})$ are obtained from a BPF. This corresponds to the case where we use a constant function $\bar{\psi} = 1$.

Here, we use i to denote the iteration of the algorithm and we use ψ to indicate all the approximate lookahead functions. We denote the approximation $\bar{\psi}$ by ψ in order to simplify notation in the algorithm. This implies that ψ^i was obtained using particles from the ith iteration. The function $mi(\hat{z}^k)$ checks if

Algorithm 8 iAPF

```
1:  $i = 1$, $\psi^i = 1$, continue $= True$
2:  while continue do
3:      $\hat{z}^i = Z^{\psi^i}_{n^i}$ from Algorithm 6 with $\psi^i$
4:      if $[i > k]$ and $\left[\frac{\text{sd}(\hat{z}^k)}{\text{mean}(\hat{z}^k)} < \tau\right]$ then
5:          $\hat{z}^i = Z^{\psi^i}_{n^i}$ from Algorithm 6 with $\psi^i$
6:          return $\hat{z}^i$
7:      else
8:          $\psi^{i+1}$ from Algorithm 7
9:          if $\left[n^{i-k} == n^i\right]$ and $\left[\neg mi(\hat{z}^k)\right]$ then
10:             $n^{i+1} = 2 \cdot n^i$
11:         else
12:             $n^{i+1} = n^i$
13:         end if
14:         $i = i + 1$
15:     end if
16: end while
```

the sequence of the last k likelihood estimates, $\hat{z}^k$, is monotonically increasing and returns $True$ if it does and $False$ if not. The algorithm corresponds to algorithm 4 from Guarniero, Johansen, and Lee (2017) in our notation. In Algorithm 8 the recursive estimation of the twisting functions in Algorithm 7 is always started from $t = T$ and stopped at $t = 1$ because we are in an offline setting.

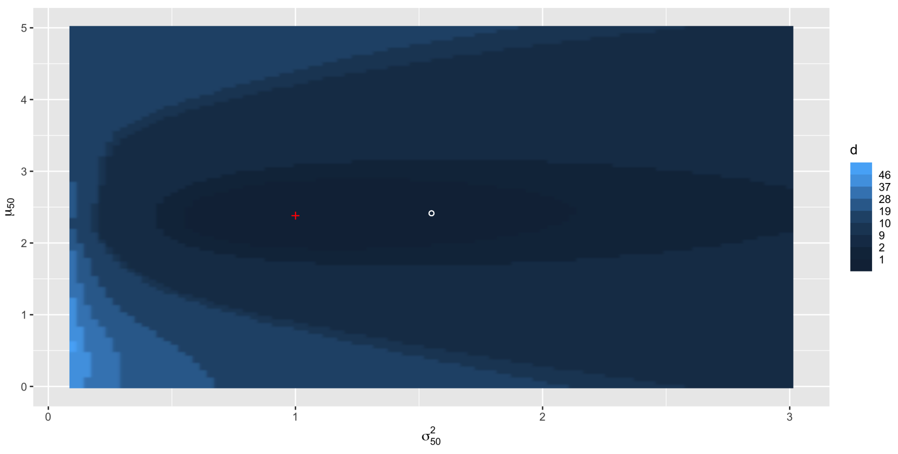
4.9 Example: instability in estimation

In our experience, estimating the twisting functions recursively in a backward manner might be unstable in 1D. The first iteration of recursive estimation in Algorithm 7 was sensitive to the initial values used in the numerical minimisation. To illustrate this with an example, we use model 1 from Chapter 7.

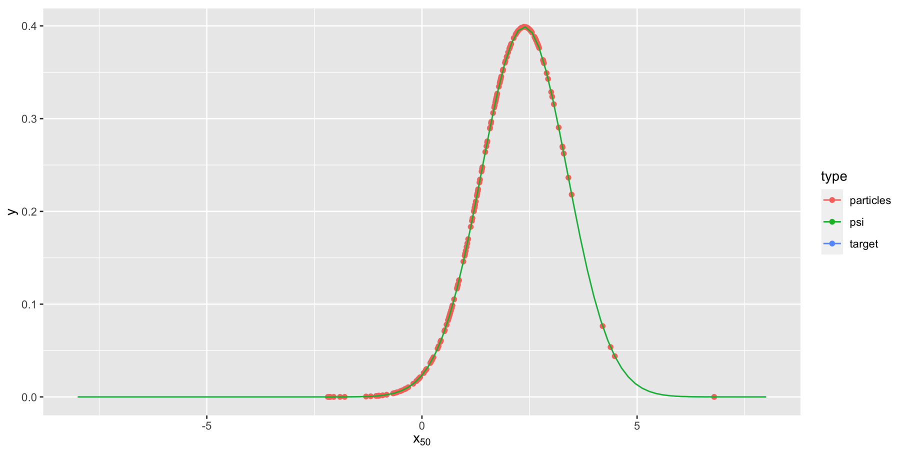
We simulate $T = 50$ latent variables and observations, then we obtain a set of $n = 500$ particles from a BPF using an adaptive resampling with a threshold of 0.5. We now consider the first two iterations of the backwards recursive estimation in Algorithm 7 when using the distance function D_{iAPF}. We highlight the instability by plotting the distance function as a surface over a set of evenly spaced μ-values and σ^2-values. The surface of the distance function is plotted before the numerical minimisation starts. This is to highlight the effect of the initial value selected for λ_t on the minimisation problem. We start with $\lambda_t = 1$ as an initial value. We then illustrate how the surface plot changes when we use the same set of evenly spaced for μ_t and σ_t^2, but with the initial value of λ_t set equal to the initial value from equation (4.18).

We start the backward recursive estimation at $t = 50$ where we have plotted the particles and the target $\dot{\psi}_t(x_t)$ in figure 4.1b. In figure 4.1a, the surface of D_{iAPF} has been plotted over a set of evenly spaced μ-values and σ^2-values. The white circle in figure 4.1a indicates the initial values for μ_{50} and σ_{50}^2. The red cross indicates the minimum values of μ_{50} and σ_{50}^2 found by using `optim` in R with the quasi-Newton method BFGS. The minimisation use the analytical gradients. The parameters $(\mu_{50}, \sigma_{50}^2)$ found by the numerical minimisation is used in the approximation $\bar{\psi}_{50}$. We see from figure 4.1b that this seems reasonable as the approximation is overlapping with the target $\dot{\psi}_{50}$. We then proceed to the second iteration of the recursive estimation, that is $t = 49$, while using an initial value of $\lambda_t = 1$.

From figure 4.2a we see that the local minimum point found, indicated by the red cross, is far from the μ_{49} and σ_{49}^2 that are reasonable for the target. The target $\dot{\psi}_{49}$ is plotted in blue in figure 4.2b. From figure 4.2b, a value of $\mu_{49} \approx 2$ can be a reasonable approximation. The minimum point found by the numerical minimisation however indicate $\mu_{49} \approx -15$ which is unreasonable. When $t = 49$ we see from figure 4.2b that the approximation $\bar{\psi}_{49}$, indicated by the green line, is far from the target $\dot{\psi}_{49}$ indicated by the blue line. In this case, the approximation, $\bar{\psi}_{49}(x_{49}, \theta_{49})$, has a low value of D_{iAPF}, but only because all $\bar{\psi}_{49}(x_{49}, \theta_{49})$ at those x values are close to 0. If we however use the initial value for λ_{49} from equation (4.18) we get another initial surface for the distance function, D_{iAPF}. From figure 4.3 we see the initial surface of D_{iAPF} when using equation (4.18) to calculate the initial λ_{49}. Even though the initial values for μ and σ^2 are the same the initial surface is quite different. We also note that if the parameters μ_{49} and σ_{49}^2 obtained with initial value $\lambda_{49} = 1$ is selected, the following estimation of the parameters at $t < 49$ is more difficult. This is due to the backward recursive structure in Algorithm 7. Further, approximations of twisting functions far from the optimal twisting functions will cause the following run of the twisted particle filter to obtain suboptimal particles. This will again affect the next recursive approximation of $(\bar{\psi}, \tilde{\bar{\psi}})$.

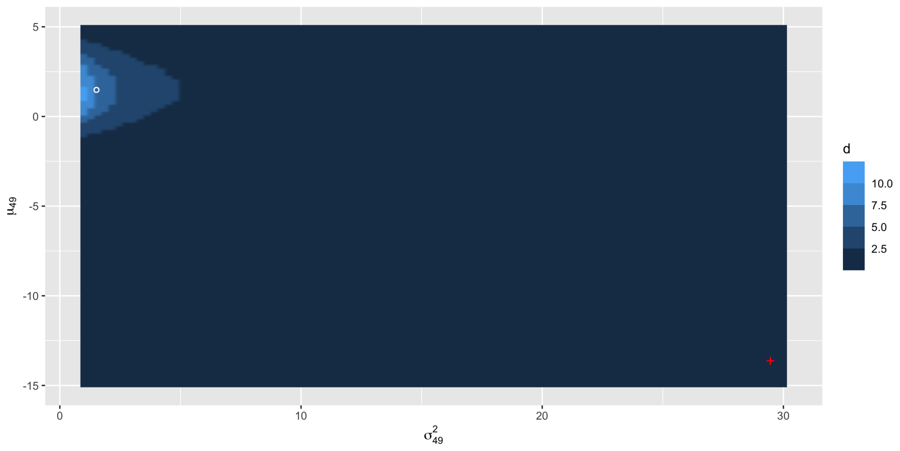


(a) Surface of distance function over a grid of μ_{50} and σ^2_{50}.

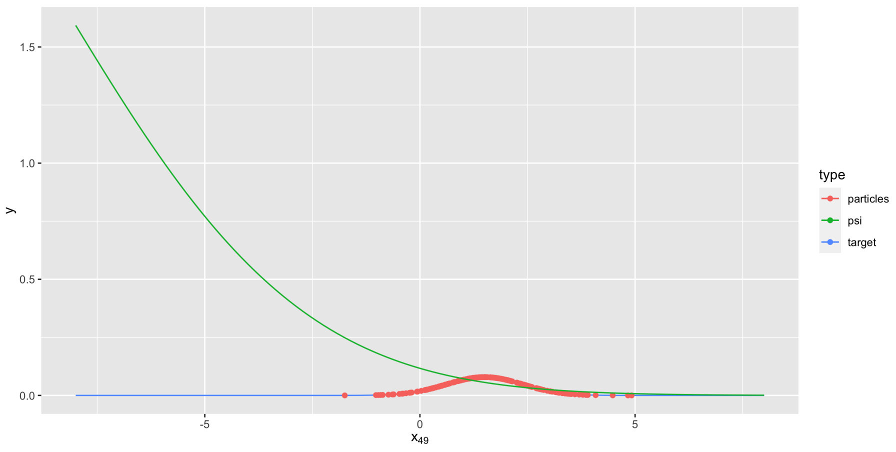


(b) Particles as red circles, approximation $\bar{\psi}_{50}$ as green line and target $\dot{\psi}_{50}$ as blue line.

Figure 4.1: First iteration of the recursive estimation using initial value $\lambda_{50} = 1$.



(a) Surface of distance function over a grid of μ_{49} and σ^2_{49}.



(b) Particles as red circles, approximation $\bar{\psi}_{49}$ as green line and target $\dot{\psi}_{49}$ as blue line.

Figure 4.2: Second iteration of the recursive estimation using initial value $\lambda_{49} = 1$.

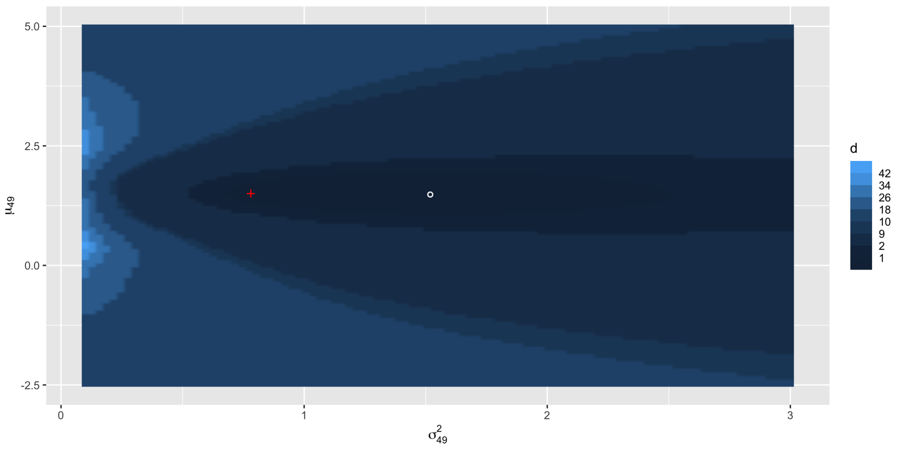


Figure 4.3: Second iteration of the recursive estimation using the alternative initial value for λ_{49}.

CHAPTER 5

Twisting targets in a batch setting

In this chapter we consider a different setup for utilising twisting target distributions which we have not seen before. We will use the term batch setting when dividing all the iterations $1, \ldots, T$ of the timeline into subsets of iterations. We think of observations becoming available in batches corresponding to these iterations. In the extremum, a batch setting with one batch is equal to an offline setting. Fundamentally, we expect estimates obtained in an offline setting to have lower numerical variance than estimates obtained in an online setting. In the context of estimating likelihood, we expect lower variance for likelihood estimates obtained in an offline setting than in an online setting. This is related to the additional information available. We also saw that if were able to utilise all future observations both in simulating latent variables and when evaluating the target distribution, this would result in all incremental weights being equal to 1.

In an offline setting it is possible to utilise twisting target distributions obtained with twisting functions in order to reduce the variance of the likelihood estimates. Algorithms such as the iterated auxiliary particle filter of Guarniero, Johansen, and Lee (2017) and Controlled SMC of Heng et al. (2020) are utilising this general strategy combined with numerical optimisation. The obtained likelihood estimates often have significantly lower variance compared to likelihood estimates from a BPF using more particles.

We consider a batch algorithm that is utilising twisting functions to reduce the variance of likelihood estimates. The main idea is to utilise iteratively improved twisting functions within a batch setting. As we have seen, there are infinitely many valid twisting functions $(\psi, \tilde{\psi})$. Among the approximate twisting functions there are e.g. the parametric forms from within predefined classes and the nonparametric forms. In the batch algorithm, we will see that we can use any form of twisting functions in a similar manner to that we can use any form of twisting functions in the offline setting. This gives us much of the same flexibility as in the offline setting. We also want to have iterative improvement of the twisting functions within the batches. This is an adaption of the offline iterative improvement that was used in the iAPF of Guarniero, Johansen, and Lee (2017). We then consider likelihood expressions in the batch setting, estimation of the likelihood and the connection between the batch setting and the twisting functions in an offline setting. We will in the following chapter use the generic notation $(\psi, \tilde{\psi})$ for the general, offline, twisting functions.

5.1 Motivation

In this section we consider the motivation and structure of a batch setting used in the batch algorithm. Conceptually, we define a batch as a subset of all iterations. We think of the observations at these iterations as becoming available in batches, then we simulate the latent variables within the batch. We denote the batches by $e = 1, \dots, E$ and the last iteration in the last batch by T. We can consider two different approaches when it comes to structuring the iterations into batches.

One way to think of a batch is as a fixed size subset of L observations that becomes available, then these L observations are utilised in the current batch. Conceptually, we then think of these observations as becoming available at the start of the respective batch and when the batch ends the observations are discarded. Since our first iteration is $t = 1$, the first batch consists of the iterations $t = 1, \dots, L$. The next batch then consists of the iterations $t = L + 1, \dots, 2L$ and so on. We refer to this as the batch setting and it is illustrated in Figure 5.1. In Figure 5.1 the iteration numbers are indicated over

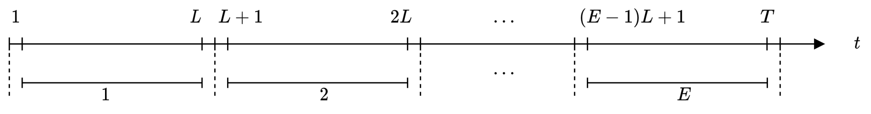


Figure 5.1: Conceptual structure of the batch setting.

the timeline, the dashed lines represent the boundaries of the batches and the batch numbers are indicated below the timeline indicating what observations are available in the current batch.

The other way is to think of incrementally increasing subsets of iterations to define incrementally larger batches. In this setting, we think of a fixed size subset of L observations that becomes available. The subset of observations is added to the previously stored observations to form a new batch. We refer to this as the incremental batch setting and it is illustrated in Figure 5.2. In

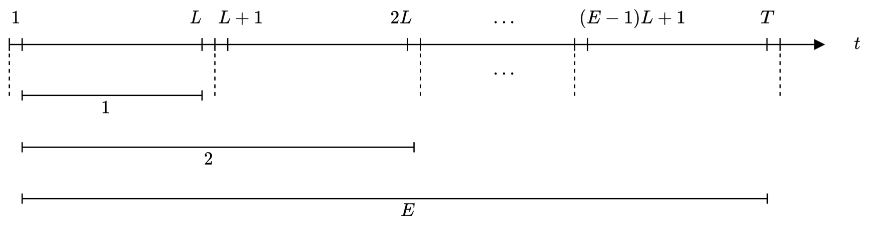


Figure 5.2: Conceptual structure of the incremental batch setting.

this way, batch e contains the observations from batch $e - 1$ as a subset. The batches are incrementally increasing by L observations as shown in Figure 5.2.

First, we consider the case of moving from an offline setting to a batch setting, then we consider the case of extending the batch setting into an incremental batch setting. The incremental batch setting is equivalent to considering the last iteration of the incremental batch as the final iteration in an offline setting.

This implies that in each incrementally larger batch, we will have another offline setting as in Chapter 4 with the final iteration defined as $T = eL$. Therefore, our focus will be on the batch setting in Figure 5.1. Note that the number of iterations defining each batch in the batch setting does not necessarily need to be fixed. In the batch setting we will however assume that each batch consists of L iterations for notational simplicity.

Consider the first batch that contains the iterations $1, \dots, L$. We can think of having the observations $y_{1:L}$ available at time L. We then think of having the current observation y_1 and future observations $y_{2:L}$ available at iteration 1 in an algorithm. This is related to the discussion of lookahead and delay strategies in Lin, Chen, and J. S. Liu (2013) and we follow their notion of conceptually thinking that we have future observations available at the current iteration. This implies that already at iteration $t = 1$, we have $y_{1:L}$ available and can use these observations e.g. when simulating x_1. In terms of delay strategies, this would correspond to delaying simulation of x_1 until iteration $t = L$ where we have $y_{1:L}$ available. The idea is the same as in the offline setting where we consider that at iteration $t = 1, \dots, T$, we have all observations $y_{1:T}$ available.

The main motivation for a batch algorithm is that at iteration $t = 1$ of the algorithm we have a batch of observations, $y_{1:L}$, available and we want to be able to start a particle filter. That is, we want to start simulating latent variables so that we can estimate both the final target distribution in the batch, $p(x_{1:L}|y_{1:L})$ and the marginal likelihood $p(y_{1:L})$. This is in contrast to an offline setting where we wait until $t = T$ to start the particle filter. When we are at iteration $t = L$, that is the end of batch 1, we also want an estimate of the marginal likelihood $p(y_{1:L})$. Then we assume that batch 2 consisting of observations $y_{L+1:2L}$ becomes available. At the end of batch 2 we want an estimate of the marginal likelihood $p(y_{1:2L})$ and so on. The first motivation for the batch algorithm is the fact that we do not have to wait until we have all the observations $y_{1:T}$ in order to start the calculations. Secondly, we can have estimates of the marginal likelihood at the end of each batch which may be useful for e.g. parameter estimation. Thirdly, the batch algorithm can also have constant memory requirements over time compared to in an offline setting when the objective is to estimate the marginal likelihood.

The second motivation can be related to online Bayesian parameter estimation. We may think of of the approach where we considered the distribution $p(x_{1:t}, \theta_{1:t}|y_{1:t})$ and assumed a transition density $p(\theta_{t+1}|\theta_t)$ discussed in J. Liu and West (2001). If we assume the parameter θ is fixed or very slowly changing we can be motivated to only simulating this at the start of a batch. This is in contrast to simulating θ at every iteration which might be somewhat unreasonable if we assume θ is fixed or very slowly changing. Ideally, we would never need to resample, at least very rarely, which also would help when considering reducing degeneracy of θ within each batch. In practice one may think of this as a trade-off between using an offline setting where one could use particle MCMC methods as discussed in Andrieu, Doucet, and Holenstein (2010) and an online setting where one could use the approach of J. Liu and West (2001).

We assume in our presentation that all the batches have equal length, for notational simplicity, and therefore that the last iteration in the last batch, E, corresponds to the final iteration $T = EL$. If one were to consider the comparable offline setting, this would consists of the iterations $1, \dots, T$. At the end of the batches we also have simulated sequences $x_{1:L}, x_{1:2L}, \dots, x_{1:T}$

along with the corresponding weights available. This also makes it possible to estimate the distributions $p(x_{1:L}|y_{1:L}), p(x_{1:2L}|y_{1:2L}), \ldots, p(x_{1:T}|y_{1:T})$. If we estimate the marginal likelihood at the end of each batch we have marginal likelihood estimates on the form $p(y_{1:L}), p(y_{1:2L}), \ldots, p(y_{1:T})$. We use the notation Z_t to denote the marginal likelihood up to and including iteration t. This is only available analytically in special cases such as in a linear Gaussian model when using a Kalman filter. For the likelihood estimates, we will use $\hat{Z}_t$ to denote the likelihood estimate up to and including iteration t. The BPF estimates the marginal likelihood online, so we would expect the estimates $\hat{Z}_t$ to be consistent estimates for every t. Moving from the offline setting to a batch setting implies that we will be able to utilise less information at every iteration. Therefore we also expect that the likelihood estimates in an offline setting will have lower variance than in a batch setting.

When moving from an offline setting to a batch setting we can conceptually think of a modified twisted model and twisting functions within each batch. We will see that this idea correspond to a special sequence $\Psi^s_{1:T}$ of functions which we refer to as the batch sequence. We will also see that the batch sequence have some additional structural requirements compared to the traditional sequence $\Psi_{1:T}$.

5.2 The batch sequence

Our focus in this section is $\Psi^s_{1:T}$, the sequence which we refer to as the batch sequence. For notational simplicity when we discuss iterations in the batch setting we will think of a generic batch $e > 1$. We denote the last iteration of batch $e - 1$ by $q = (e - 1)L$. Consequently $q + 1$ corresponds to the first iteration of batch e. We then define the batch sequence by

$$\Psi^s_{1:T} = \psi_1(x_1), \ldots, \psi_q(x_q), \psi_{q+1}(x_{q+1}, x_q), \psi_{q+2}(x_{q+2}), \ldots, \psi_T(x_T)$$

and we see that at iterations $q+1$ the functions ψ_{q+1} depend on both the current and the previous latent variables, x_{q+1} and x_q respectively. We note here that the lookahead functions defined in Naesseth, Lindsten, and Schön (2019) at iteration t depends on $x_{1:t}$. This implies that we might can see the functions $\psi_{q+1}(x_{q+1}, x_q)$ as a special case of these functions. We use another function or combination of functions to define every function ψ_t in the batch sequence. As long as the functions we use to define every ψ_t are selected with some requirements, our aim is that the batch sequence also follows Assumptions 4.2.1 and therefore is a valid sequence. We denote the functions that we use to define each ψ_t in the batch sequence by $(\psi^b, \tilde{\psi}^b)$ and refer to these as batch twisting functions. We then consider how the batch sequence $\Psi^s_{1:T}$ can be used in combination with the twisted model to define what we will refer to as batch twisted models. Finally we focus on what we will refer to as batch optimal twisting functions, denoted by $(\psi^{b*}, \tilde{\psi}^{b*})$ which is just a specific choice of batch twisting functions.

The motivation is now to define a batch sequence $\Psi^s_{1:T}$ that fulfils Assumptions 4.2.1. This again ensures that a twisting target distributions using this sequence and a twisted model has final target equivalence with the joint distribution $p(x_{1:T}, y_{1:T})$. This is seen in proposition 1 of Guarniero, Johansen, and Lee (2017). As we have seen, finding a general sequence $\Psi_{1:T}$

that fulfils these requirements is not necessarily difficult. There are infinitely many valid sequences, one of them being the sequence that results from setting all the functions in the sequence equal to a constant C. So far we have only specified that every function ψ_t in the batch sequence $\Psi^s_{1:T}$ should be defined by another function or combination of functions which we denoted by $(\psi^b, \tilde{\psi}^b)$. We now consider the batch twisting functions before returning to how we can use these to define every ψ_t in the batch sequence.

Batch twisting functions

Our aim is to define every function in the sequence $\Psi^s_{1:T}$. That is, we want to define ψ_t for $t = 1, \ldots, T$ and the only restrictions we have for each ψ_t are those in Assumptions 4.2.1. To define each ψ_t in the batch sequence we use either one function or a combination of functions from another set of functions which we denote by $(\psi^b, \tilde{\psi}^b)$. We refer to the set of functions used to define every ψ_t as batch twisting functions. For now the batch twisting functions are simply functions used to define each ψ_t in the batch sequence. Because the batch twisting functions are just a set of functions, we also select ψ^b_t so Assumptions 4.2.1 is fulfilled. From these assumptions, we still have a lot of flexibility when it comes to defining the batch twisting functions. In addition, we denote the batch normalising functions for the batch lookahead function $\psi^b_{q+1}(x_{q+1})$ by $\tilde{\psi}^b_{0e}(x_q)$. We then have for batch $e = 1$ and $e \geq 2$ respectively

$$\tilde{\psi}^b_{01} = \int \psi^b_1(x_1)p(x_1)dx_1 \tag{5.1a}$$

$$\tilde{\psi}^b_{0e}(x_q) = \int \psi^b_{q+1}(x_{q+1})p(x_{q+1}|x_q)dx_{q+1}. \tag{5.1b}$$

Here, we have denoted the normalising function by using $0e$ even though it can be viewed as a traditional normalising function. The motivation for this is based on its placement in the batch sequence $\Psi^s_{1:T}$ where we will see that it is included at iteration $q + 1$ instead of at iteration q. We have now have defined the batch twisting functions and further we want to use these functions to define every ψ_t in the batch sequence $\Psi^s_{1:T}$. Note that even though we have used the notation $(\psi^b, \tilde{\psi}^b)$, these functions are so far only used in order to define each function ψ_t in the batch sequence.

Defining the batch sequence

We now want to use the set of batch twisting functions which we denote by $(\psi^b, \tilde{\psi}^b)$ to define each of the ψ_t in the sequence $\Psi^s_{1:T}$. Recall that we defined iteration q as the last iteration of batch $e-1$ such that iteration $q+1$ corresponds to the first iteration of the following batch $e \geq 2$. We now define the function ψ_t for $t = 1, \ldots, T$ which gives

$$\psi_t(x_t) = \psi^b_t(x_t) \qquad \text{if } t \neq q+1 \tag{5.2a}$$

$$\psi_t(x_t, x_{t-1}) = \frac{\psi^b_t(x_t)}{\tilde{\psi}^b_{0e}(x_{t-1})} \qquad \text{if } t = q+1. \tag{5.2b}$$

When each of these functions are defined we have all the functions in the batch sequence $\Psi^s_{1:T}$. Our goal is that the batch sequence also fulfils the original assumptions defined in Guarniero, Johansen, and Lee (2017) when we have defined each ψ_t by using batch twisting functions. As we have seen, this ensures final target equivalence when the twisting functions obtained from the batch sequence is used in a twisted model. Note that so far, the iterations $q+1$ are only iterations of the batch sequence where we have defined ψ_t in a different way. The batch sequence therefore has much of the flexibility of a general sequence $\Psi_{1:T}$. This is because at iterations $t \neq q+1$ we simply have $\psi_t(x_t) = \psi_t^b(x_t)$. The function ψ_t at iterations $q+1$ also depends on x_q which corresponds to the latent variable of the last iteration of the previous batch.

The batch sequence $\Psi_{1:T}^s$ is a valid sequence because of the assumptions made about the batch lookahead functions ψ^b which again defines every ψ_t in the batch sequence. To define the batch sequence we first define ψ^b and then ψ are defined based on ψ^b. We can think of this as inserting ψ^b into equation (5.2a) and (5.2b) which defines every ψ_t. When this is done, we have the sequence $\Psi_{1:T}^s$ and we can use this in a twisted model. The twisted model can then be used in a twisted particle filter. This also mean we can estimate the likelihood in the same way as in Guarniero, Johansen, and Lee (2017).

We now want to consider the normalising functions associated with the batch sequence. The normalising function $\tilde{\psi}_t(x_t)$ is defined as the expectation of $\psi_{t+1}(x_{t+1})$ with respect to the transition density $p(x_{t+1}|x_t)$. Recall that we denoted iteration q as the last iteration in batch $e-1$ for batch $e \geq 2$ and therefore that $q+1$ corresponds to the first iteration in batch e. Therefore, if we think of iteration q as the last iteration in batch $e = 1, \dots, E-1$ we have that

$$\begin{aligned}\tilde{\psi}_q(x_q) &= \int \psi_{q+1}(x_{q+1}, x_q) p(x_{q+1}|x_q) dx_{q+1} \\ &= \int \frac{\psi_{q+1}^b(x_{q+1})}{\tilde{\psi}_{0e}(x_q)} p(x_{q+1}|x_q) dx_{q+1} \\ &= \int \frac{\psi_{q+1}^b(x_{q+1})}{\int \psi_{q+1}^b(x_{q+1}) p(x_{q+1}|x_q) dx_{q+1}} p(x_{q+1}|x_q) dx_{q+1} = 1. \end{aligned} \tag{5.3}$$

Note that the second equality follows from substituting $\psi_{q+1}(x_{q+1}, x_q)$ with equation (5.2b). We then have that the normalising functions $\tilde{\psi}_t(x_t)$ which are defined by the batch sequence are given by the following

$$\tilde{\psi}_t(x_t) = \begin{cases} \int \psi_{t+1}^b(x_{t+1}) p(x_{t+1}|x_t) dx_{t+1} = \tilde{\psi}_t^b(x_t) & \text{if } t \neq q \\ 1 & \text{if } t = q. \end{cases}$$

Note that the last normalisation function associated with the sequence $\Psi_{1:T}^s$ is $\tilde{\psi}_T(x_T) \equiv 1$ because $\tilde{\psi}_T^b(x_T) \equiv 1$. To summarise we now see that the functions ψ_t at iterations t that correspond to the first iteration of batches $e \geq 2$ are defined differently from the remaining functions. For the iterations that correspond to the first of batches, we have $\psi_t(x_t, x_{t-1})$ which we saw in equation (5.2b). For the remaining functions we have that $\psi_t(x_t) = \psi_t^b(x_t)$. As a consequence of this, we see that the normalising functions $\tilde{\psi}_t(x_t)$ at iterations t that corresponds to the last of batches $e \geq 1$ are equal to 1 while at all the other iterations we have $\tilde{\psi}_t(x_t) = \tilde{\psi}_t^b(x_t)$.

In proposition 1 of Guarniero, Johansen, and Lee (2017), it is stated that we need the final, unnormalised, twisting target distribution to be equal to the joint distribution $p(x_{1:T}, y_{1:T})$ when the final target distribution is the posterior $p(x_{1:T}|y_{1:T})$. As a result of this, the likelihood integral from the twisted model is equal to the likelihood integral from the SSM used to define the twisted model. As long as the sequence $\Psi^s_{1:T}$ is valid, we will have final target equivalence at the final iteration $t = T$ when using a twisted model and this sequence. So far, the batch sequence is just a sequence of functions ψ_t with a special way of defining the functions at iterations that correspond to the first iteration of each batch $e \geq 2$. Consider now the case of selecting batch lookahead functions $\psi^b_t(x_t) = 1$ for every x_t and every t. All the functions ψ_t in $\Psi^s_{1:T}$ would then be equal to 1 and consequently, all the normalising functions $\tilde{\psi}$ would be equal to 1. This is a valid choice of batch lookahead functions ψ^b and the resulting functions ψ_t which all are 1 are also valid within Assumptions 4.2.1. The resulting batch sequence will however not be very useful with respect to reducing variance of the likelihood estimates. This is because the twisted transition density and twisted observation functions then will be equal to the transition and observation density respectively.

In the next section we consider how we can define what we will refer to as batch twisted models by inserting the specially defined batch sequence into a twisted model. The batch twisted models are subsets of the sequence of twisted transition densities and observation functions from equation (5) and (6) in Guarniero, Johansen, and Lee (2017) after $\Psi^s_{1:T}$ is inserted. This is done in the same manner as we think of splitting the iterations into subsets and referring to these as batches. We start by inserting the batch sequence into one twisted model. Then, we split the sequence of resulting twisted transition densities and observation functions into subsets corresponding to the iterations that defines batches. We then think of each batch as having a batch twisted model which is defined over the iterations of the batch. This implies that we can always return to thinking of one sequence $\Psi^s_{1:T}$ and one twisted model. In Figure 5.3

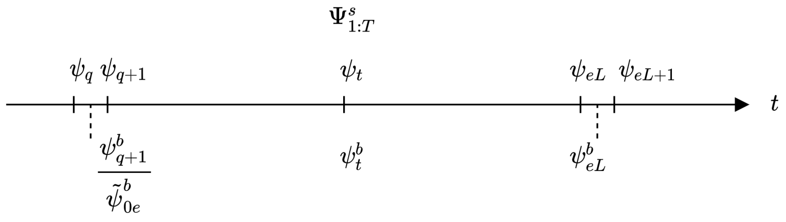


Figure 5.3: The batch sequence at iterations corresponding to batch $e \geq 2$.

we consider a subset of the batch sequence at iterations corresponding to a batch $e \geq 2$. The boundaries of the batch are denoted by the dashed lines. Over the timeline we have the functions ψ_t and below we have its definition from equations (5.2a) and (5.2b). See Appendix A.3 for an overview of $\Psi^s_{1:T}$.

5.3 Batch twisted model

The batch sequence $\Psi^s_{1:T}$ is a sequence of general functions ψ_t that can be used directly with the twisted model as long as Assumptions 4.2.1 holds for the functions ψ^b. This implies that the unnormalised twisting target distributions that result from the twisted model have final target equivalence with the joint distribution $p(x_{1:T}, y_{1:T})$. In this section we will consider what we refer to as batch twisted models by inserting the batch sequence into a twisted model. We split the sequence of twisted transition densities and twisted observation functions into subsets of iterations defined by the batches.

Our motivation for thinking of subsets of the twisted model as batch twisted models is based on how we defined each of the functions ψ_t in the batch sequence by using the functions ψ^b. Up until now, we have been thinking of the functions ψ^b only as functions that define every ψ_t. We want to use the framework defined in Guarniero, Johansen, and Lee (2017) within each batch by now viewing the functions ψ^b as batch lookahead functions. Our aim is to have target equivalence with $p(x_{1:t}, y_{1:t})$ also at the iterations t that correspond to the end of batches when using batch twisted models. Note that this can trivially be achieved by selecting each $\psi_t^b = 1$ for all t. Then every twisted transition density and twisted observation function in a twisted model would correspond to the transition density and observation density of the model that defined the twisted model. The unnormalised twisting target distributions would then correspond to $p(x_{1:t}, y_{1:t})$ at every iteration, including at the iterations that correspond to the end of each batch. However, in that case we are not utilising any future information.

To define the batch twisted models, we now insert the batch sequence $\Psi^s_{1:T}$ into a twisted model for iterations $t = 1, \ldots, T$. We start with iterations $t = 1, \ldots, L$ which correspond to the iterations in batch 1 and ψ_t corresponding to $\Psi^s_{1:L}$ is inserted into the twisted model. Note that we omit dependence on x in these expressions to simplify notation. This then gives

$$\begin{aligned}
&p_1^{\psi}(y_1|x_1)p_1^{\psi}(x_1)\ldots p_L^{\psi}(y_L|x_L)p_L^{\psi}(x_L|x_{L-1}) && \text{(5.4a)}\\
=&\frac{p(y_1|x_1)\tilde{\psi}_1^b\tilde{\psi}_{01}^b}{\psi_1^b}\,\frac{p(x_1)\psi_1^b}{\tilde{\psi}_{01}^b}\cdots\frac{p(y_L|x_L)\cdot 1}{\psi_L^b}\,\frac{p(x_L|x_{L-1})\psi_L^b}{\tilde{\psi}_{L-1}^b} &&\\
=&p(y_1|x_1)p(x_1)\prod_{s=2}^{L}p(y_s|x_s)p(x_s|x_{s-1}). && \text{(5.4b)}
\end{aligned}$$

After inserting the functions ψ_t into the twisted model, the batch twisting functions $(\psi^b, \tilde{\psi}^b)$ remain in the first equality. When considering the unnormalised twisting target distribution up to the end of batch 1 all the batch twisting functions cancels and the joint distribution $p(x_{1:L}, y_{1:L})$ results in the second equality. Recall that the last normalising function is $\tilde{\psi}_L(x_L) = 1$ from equation (5.3). We now want to consider the unnormalised twisting target distribution at the end of batch e. Recall that this corresponds to iteration eL, we then have that

$$p_1^{\psi}(y_1|x_1)p_1^{\psi}(x_1)\prod_{s=2}^{eL}p_s^{\psi}(y_s|x_s)p_s^{\psi}(x_s|x_{s-1}) \tag{5.5}$$

follows from the twisted transition density and twisted observation functions. For notational simplicity, we denote the last iteration of batch $e-1$ by $q = (e-1)L$ and therefore iteration $q+1$ is the first of batch e. We then have that equation

(5.5) can be reformulated into

$$\left[p(y_1|x_1)p(x_1)\prod_{s=2}^{L}p(y_s|x_s)p(x_s|x_{s-1})\right]$$

$$\vdots$$

$$\left[\frac{p(y_{q+1}|x_{q+1})\tilde{\psi}^b_{q+1}\tilde{\psi}^b_{0e}}{\psi^b_{q+1}}\frac{p(x_{q+1}|x_q)\psi^b_{q+1}}{\tilde{\psi}^b_{0e}}\cdots\frac{p(y_{eL}|x_{eL})\cdot 1}{\psi^b_{eL}}\frac{p(x_{eL}|x_{eL-1})\psi^b_{eL}}{\tilde{\psi}^b_{eL-1}}\right].$$

This is a result of inserting $\Psi^s_{1:eL}$ into the twisted model. For iterations up to iteration L equations (5.4a) and (5.4b) are used in the first pair of square brackets. We insert ψ_t into batches $2,\dots,e-1$ in the same way as for the first batch and show the expression only for iterations corresponding to batch e. We see the resulting expression after inserting ψ_t from $\Psi^s_{q+1:eL}$ in the lower pair of square brackets. The batch twisting functions $(\psi^b,\tilde{\psi}^b)$ cancels in the batches $2,\dots,e$ equivalently as in the first batch and we have

$$\begin{aligned}&\left[p(y_1|x_1)p(x_1)\prod_{s=2}^{L}p(y_s|x_s)p(x_s|x_{s-1})\right]\\ &\vdots\\ &\left[p(y_{q+1}|x_{q+1})p(x_{q+1}|x_q)\prod_{s=q+2}^{eL}p(y_s|x_s)p(x_s|x_{s-1})\right]\\ =&p(y_1|x_1)p(x_1)\prod_{s=2}^{eL}p(y_s|x_s)p(x_s|x_{s-1}).\end{aligned}$$

This implies that at the end of batch e, the unnormalised twisting target distribution is $p(x_{1:eL},y_{1:eL})$. Recall that we denoted the last batch by E and that the last iteration of this batch was denoted by $T=EL$. By considering the last iteration of the last batch, we would therefore have that the final unnormalised twisting target distribution would be equal to the joint distribution $p(x_{1:T},y_{1:T})$. Recall that it is only for notational simplicity that we think of all batches as consisting of L iterations. We can define each batch to contain any number of iterations and consequently, the batch twisted models will consist of a varying number of twisted transition densities and twisted observation functions.

We have now considered $\Psi^s_{1:T}$ to be one sequence of functions ψ_t that fulfils target equivalence at iterations eL for $e=1,\dots,E$ when inserted into the twisted model. Now we define what we will refer to as batch twisted models. Conceptually, we consider each batch to have its own twisted model and twisting functions. These twisting functions are provided by the batch sequence $\Psi^s_{1:T}$ because of how we defined the functions ψ_t in this sequence. Recall that we defined the functions ψ^b under the general Assumptions 4.2.1. We can then think of $(\psi^b,\tilde{\psi}^b)$ as the batch twisting functions in combination with a batch twisted model. This motivates that at the end of a batch, that is iteration eL, we have target equivalence and therefore the marginal likelihood from the twisted model at the end of a batch should correspond to the marginal likelihood of the SSM at that iteration. In a filtering context we can have $p(x_{1:t}, y_{1:t})$ as unnormalised target distributions at iterations $t = 1, \dots, T$. Similarly, we consider a batch algorithm as having unnormalised twisting target distributions $\tilde{f}^b_t(x_{1:t})$ which by target equivalence are equal to $p(x_{1:t}, y_{1:t})$ at iterations $t = eL$ for $e = 1, \dots, E$. Also in a similar way, we consider offline algorithms using twisting functions as having target equivalence with $p(x_{1:t}, y_{1:t})$ only at the final iteration, $t = T$.

We now consider the sequence of twisted transition densities and twisted observations functions corresponding to the iterations contained in the batch as the batch twisted model when $\Psi^s_{1:T}$ is inserted into the twisted model. Similarly, we consider the $(\psi^b_t, \tilde{\psi}^b_t)$ as the twisting functions for iterations t in the batch combined with the normalising function in equation (5.1b) for batch $e > 1$ or the normalising constant in equation (5.1a) for the first batch. As in proposition 1 of Guarniero, Johansen, and Lee (2017) we see that final target equivalence is necessary for the marginal likelihood of the SSM to be equal to that of the twisted model at the final iteration. In addition to having the same likelihood integral at the final iteration, we want to have the same likelihood integral at the end of each batch. We saw from the definition of $\Psi^s_{1:T}$ that when used in a twisted model, we had target equivalence at iterations eL where $e = 1, \dots, E$ because of how it was defined. We consider the structure of the batch twisted models by considering the first batch twisted model and the batch twisted model for batch e to highlight the structure and how they are connected.

Batch twisted models

We now consider $\Psi^s_{1:T}$ inserted into a twisted model, this gives a sequence of T twisted transition densities and twisted observation functions. Here we consider a subset of the twisted transition densities and twisted observation functions. The subset consists of the twisted transition densities and twisted observation functions corresponding to the iterations of batch e. Recall that we denoted the last iteration of batch $e-1$ by $q = (e-1)L$. Hence we have that the iterations $t = q+1, \dots, eL$ are included in batch e. The initial twisted transition density and twisted observation function for batch $e = 1$ are

$$p^\psi_1(y_1|x_1) = \frac{p(y_1|x_1)\tilde{\psi}^b_1(x_1)\tilde{\psi}^b_{01}}{\psi^b_1(x_1)} \qquad p^\psi_1(x_1) = \frac{p(x_1)\psi^b_1(x_1)}{\tilde{\psi}^b_{01}}$$

and for batch $e \geq 2$ these are given by

$$p^\psi_{q+1}(y_{q+1}|x_{q+1}, x_q) = \frac{p(y_{q+1}|x_{q+1})\tilde{\psi}^b_{q+1}(x_{q+1})\tilde{\psi}^b_{0e}(x_q)}{\psi^b_{q+1}(x_{q+1})}$$
$$p^\psi_{q+1}(x_{q+1}|x_q) = \frac{p(x_{q+1}|x_q)\psi^b_{q+1}(x_{L+1})}{\tilde{\psi}^b_{0e}(x_q)}$$

and for the remaining iterations in a batch, we have

$$p^\psi_t(y_t|x_t) = \frac{p(y_t|x_t)\tilde{\psi}^b_t(x_t)}{\psi^b_t(x_t)} \qquad p^\psi_t(x_t|x_{t-1}) = \frac{p(x_t|x_{t-1})\psi^b_t(x_t)}{\tilde{\psi}^b_{t-1}(x_{t-1})}$$
$$p^\psi_t(y_{eL}|x_{eL}) = \frac{p(y_{eL}|x_{eL}) \cdot 1}{\psi^b_{eL}(x_{eL})} \qquad p^\psi_{eL}(x_{eL}|x_{eL-1}) = \frac{p(x_{eL}|x_{eL-1})\psi^b_{eL}(x_{eL})}{\tilde{\psi}^b_{eL-1}(x_{eL-1})}$$

Recall that $\tilde{\psi}_{eL}(x_{eL}) = 1$ from equation (5.3) in the last twisted observation function in batch e. When $e \geq 2$, the term $\tilde{\psi}^b_{0e}(x_q)$ is the normalising function for the first twisted transition density in the batch. When $e = 1$, the term $\tilde{\psi}^b_{01}$ is the normalising constant. These are defined in equation (5.1b) and (5.1a) respectively. This follow from the special definition of the functions in the sequence $\Psi^s_{1:T}$ at iterations $q+1$. Within batch e consisting of iterations $q+1, \dots, eL$, we consider the batch twisting functions. We have the lookahead functions for batch e which we denote by Ψ^b_e to emphasise that this is a sequence of batch e lookahead functions. We can define this sequence as

$$\Psi^b_e = \psi^b_{q+1}, \psi^b_{q+2}, \dots, \psi^b_{eL-1}, \psi^b_{eL} \tag{5.6}$$

and the corresponding normalising functions for batch e are denoted by

$$\tilde{\psi}^b_{0e}, \tilde{\psi}^b_{q+1}, \dots, \tilde{\psi}^b_{eL-1}, \tilde{\psi}^b_{eL}. \tag{5.7}$$

Note that $\tilde{\psi}^b_{eL}$ do not appear in batch e because of how $\psi_{eL+1}(x_{eL+1})$ is defined in the batch sequence $\Psi^s_{1:T}$. We see however that the last twisted observation function in batch e has 1 in the numerator resulting from $\tilde{\psi}_{eL}$. For this reason we set $\tilde{\psi}^b_{eL} \equiv 1$.

Batch twisted model target equivalence

We only inserted the functions from $\Psi^s_{1:T}$ into a twisted model to define the batch twisted models. This implies that we always can return to the view of one sequence $\Psi^s_{1:T}$ and one twisted model. By conceptually considering batch twisted models we see that we have target equivalence with the untwisted target distribution $p(x_{1:t}, y_{1:t})$ at iterations t that correspond to the last of each batch. Conceptually, we think of the unnormalised twisting target distribution $\tilde{f}^b_t(x_{1:t})$ in the batch setting as being composed by the twisted transition densities and twisted observation functions from batch twisted models up to iteration t. Consider observations $y_{(e-1)L+1:eL}$ becoming available at the start of batch e and then we consider the twisting target distribution at iteration eL. We can think of this as extending the unnormalised twisting target distribution. At the end of batch e, we then consider the unnormalised twisting target distribution as being

$$\begin{aligned}\tilde{f}^b_{eL}(x_{1:eL}) &= p^\psi_1(y_1|x_1)p^\psi_1(x_1)\prod_{s=2}^{eL} p^\psi_s(y_s|x_s)p^\psi_s(x_s|x_{s-1}) \\ &= p(y_1|x_1)p(x_1)\prod_{s=2}^{eL} p(y_s|x_s)p(x_s|x_{s-1}) = p(x_{1:eL}, y_{1:eL})\end{aligned}$$

which we will refer to as batch target equivalence. We can think of this as utilising twisting target distributions for the iterations that are intermediate within the batch. That is, we use twisting target distributions at the intermediate iterations $(e-1)L+1, \dots, eL-1$ and then at iteration eL we have batch target equivalence with the untwisted target distribution $p(x_{1:eL}, y_{1:eL})$. Note that this follows from composing the twisting target distributions by the batch twisted models which again is equivalent to inserting $\Psi^s_{1:T}$ into the twisted model.

Batch target equivalence is achieved at the end of batch e by extending an unnormalised target distribution which have batch target equivalence with $p(x_{1:(e-1)L}, y_{1:(e-1)L})$ by the next batch e. This consists of the intermediate iterations $(e-1)L+1, \dots, eL$ and at the end of batch e we have batch target equivalence with $p(x_{1:eL}, y_{1:eL})$. We denote the last iteration of batch $e-1$ by q for notational simplicity. Then $p(x_{1:q}, y_{1:q})$ is the unnormalised twisting target distribution at the end of batch $e-1$ and

$$p(x_{q+1:eL}, y_{q+1:eL} | x_{1:q}, y_{1:q})$$

is the extension to the unnormalised target distribution by batch e. This can be reformulated to

$$p(x_{q+1:eL}, y_{q+1:eL} | x_q) = \prod_{s=q+1}^{eL} p(y_s|x_s)p(x_s|x_{s-1})$$

due to the CI properties of the model. This results in the unnormalised and untwisted target distribution $p(x_{1:eL}, y_{1:eL})$ at the end of batch e. We can compose the extension $\tilde{f}^b_{eL}(x_{q+1:eL})$ from the twisted transition densities and observation functions in batch e. When extending the target distribution $p(x_{1:q}, y_{1:q})$ at the end of batch $e-1$, we see that it has to be equal to the untwisted extension $p(x_{q+1:eL}, y_{q+1:eL}|x_q)$ in order to have batch target equivalence at the end of batch e. This motivates the idea that we are using twisting target distributions for the intermediate iterations in each batch.

5.4 Optimal batch twisting functions

We have considered the batch sequence $\Psi^s_{1:T}$ of functions ψ defined by another set of functions, ψ^b, which we refer to as batch twisting functions. In principle, we can define the ψ^b however we want as long as the resulting functions, ψ, they define are valid. We have seen that a valid choice is to define $\psi^b_t = 1$ for all t. This results in that $\psi_t = 1$ for all t, hence all functions in the batch sequence are equal to one. The twisted model then corresponds to a SSM because all twisted transition densities correspond to the transition densities and all the twisted observation functions correspond to the observation functions. Our motivation for using twisting target distributions is to reduce the variance of likelihood estimates. The sequence $\Psi^*_{1:T}$ from Guarniero, Johansen, and Lee (2017) is optimal with respect to minimising the variance of the likelihood estimates when combined with a twisted model in an offline setting.

If we consider the iterations corresponding to batch e we have a batch twisted model and we have the batch lookahead functions in Ψ^b_e which is defined in equation (5.6). The corresponding batch e normalising functions are given in equation (5.7). For batch e, we now have a setup of batch lookahead functions, normalising functions and twisted model akin to the setup we considered in an offline setting in Chapter 4. Motivated by this similarity we now want to consider if there are optimal batch twisting functions with respect to minimising the variance of the likelihood estimates. Then we consider the task of approximating the optimal batch twisting functions.

Defining the optimal batch twisting functions

The recursive structure of the optimal, offline, twisting functions in equation (4.10) is an important property of these twisting functions. This structure is also important in the approximation of the optimal twisting functions as seen in e.g. Guarniero, Johansen, and Lee (2017), Heng et al. (2020), and Naesseth, Lindsten, and Schön (2019). The recursive structure however implies that we need $\psi^*_{t+1}(x_{t+1})$ to calculate $\psi^*_t(x_t)$ in a backwards recursive manner. The backwards recursive structure is also used in order to calculate the approximate twisting functions which are estimating the optimal twisting functions. In a batch setting we do not have access to all the observations $y_{1:T}$ at every iteration. Because only observations from within the batch are available, we cannot utilise the full backwards recursive structure to calculate the twisting functions. If we consider batch e, only observations $y_{(e-1)L+1:eL}$ are available in that batch. We can always store all previous the observations, $y_{1:(e-1)L}$, and calculate all the the twisting functions from iteration 1 at every iteration. This however implies increasing memory requirements and increasing computational cost.

We denote the first iteration of batch e by $q+1$ and the last by eL for notational simplicity. Conceptually, we consider the optimal target distributions in the batch where $t \geq q+1$ as

$$f^{b*}_t(x_{1:t}) = p(x_{1:t}|y_{1:eL}).$$

Consider now the unnormalised optimal target distributions in the batch. As in the offline setting these can also be defined at iteration t by marginalising out x_{t+1}. We therefore have that for the unnormalised optimal target distributions in the batch,

$$\tilde{f}^{b*}_t(x_{1:t}) = \int \tilde{f}^{b*}_{t+1}(x_{1:t+1})dx_{t+1}.$$

This is akin to the offline setting in equation (4.6). Assume now that we recursively define the unnormalised batch twisting target distributions

$$\tilde{f}^b_{t+1}(x_{1:t+1}) = \tilde{f}^b_t(x_{1:t})\phi_{t+1}(x_{t+1}, x_t)\frac{\tilde{\psi}^b_{t+1}(x_{t+1})}{\tilde{\psi}^b_t(x_t)}$$

We use this recursively defined batch twisting target distribution motivated by the unnormalised optimal target distribution to define the optimal batch twisting functions which we denote by $(\psi^{b*}, \tilde{\psi}^{b*})$. See Appendix A.3 for the calculations related to the optimal batch twisting functions. Our goal is to approximate the optimal twisting functions and use these approximations to reduce variance of the likelihood estimates. We therefore want to preserve the recursive structure of the batch twisting functions as far as possible when approximating the optimal batch twisting functions. Moving from an offline setting to a batch setting, we already know that the full offline backwards recursive structure is unachievable. In order to utilise the backwards recursive structure in a batch setting, we define the twisting functions within the batches. As we are utilising future information in the optimal batch twisting functions, the strategy here is also related to the general framework of lookahead strategies from Lin, Chen, and J. S. Liu (2013).

For all batches we have that eL is the last iteration of batch e and we have denoted $q = (e-1)L$ as the last iteration of batch $e-1$. We can now define the optimal batch twisting functions within each batch. We start by the optimal batch lookahead functions ψ^{b*}.

Definition 5.4.1. *For batch e containing iterations $t = q+1, \dots, eL$ we define*

$$\psi_t^{b*}(x_t) = p(y_{t:eL}|x_t)$$

as the optimal batch lookahead functions for $q+1 \le t \le eL$.

Once the optimal batch lookahead functions, ψ^{b*}, are defined, we can consider the optimal batch normalising functions $\tilde{\psi}^{b*}$. These follow from Definition 4.2.1 and we have that

$$\begin{aligned}
&\psi_{q+1}^{b*}, \dots, \psi_{eL}^{b*} \\
&\tilde{\psi}_{0e}^{b*}, \tilde{\psi}_{q+1}^{b*}, \dots, \tilde{\psi}_{eL-1}^{b*}
\end{aligned}$$

we also here use $\tilde{\psi}_{eL}^{b*} \equiv 1$ as in the general case. Note that $\tilde{\psi}_{eL}^{b}$ is not needed to find the optimal batch twisting functions as we saw in Appendix A.3. This follows from how we defined the sequence $\Psi_{1:T}^{s}$.

Batch recursive structure

We now want to consider if the recursive structure akin to the recursive structure from the offline optimal twisting functions holds for the optimal batch twisting functions. That is, if $\psi_t^{b*}(x_t) = p(y_t|x_t)\tilde{\psi}_t^{b*}(x_t)$ for all iterations t within a batch. When $t < eL$, the right hand-side can be expanded

$$\begin{aligned}
p(y_t|x_t)\tilde{\psi}_t^{b*}(x_t) &= p(y_t|x_t)\int \psi_{t+1}^{b*}(x_{t+1})p(x_{t+1}|x_t)dx_{t+1} \\
&= p(y_t|x_t)\int p(y_{t+1:eL}|x_{t+1})p(x_{t+1}|x_t)dx_{t+1} \\
&= p(y_t|x_t)p(y_{t+1:eL}|x_t) \\
&= p(y_{t:eL}|x_t).
\end{aligned}$$

This is equal to the left-hand side by Definition 5.4.1. When $t = eL$ we have that $\tilde{\psi}_t^{b*}(x_t) \equiv 1$. The right-hand side is then given by

$$p(y_t|x_t)\tilde{\psi}_t^{b*}(x_t) = p(y_t|x_t)$$

which is equal to the left hand-side by Definition 5.4.1. Combined, this gives

$$\psi_t^{b*}(x_t) = p(y_t|x_t)\tilde{\psi}_t^{b*}(x_t). \tag{5.8}$$

Now we can utilise this recursive structure that holds within the batch in order to approximate the optimal batch lookahead functions ψ^{b*}. We use Algorithm 7 with appropriate t_{start} and t_{stop} to obtain approximate batch twisting functions denoted by $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$. We can then use these with a batch twisted model which again can be used in a twisted particle filter.

5.5 Likelihood

By using the batch sequence $\Psi^s_{1:T}$ with a twisted model we saw that we could conceptually split the the sequence of twisted transition densities and observation functions into what we referred to as batch twisted models. This setup yielded target equivalence with the joint distributions $p(x_{1:eL}, y_{1:eL})$ at iterations eL for $e = 1, \dots, E$. We now utilise the same idea as in proposition 1 of Guarniero, Johansen, and Lee (2017) in order to see that the marginal likelihood obtained by using a twisting target distributions from batch twisted models up to the the end of batch e is consistent with the marginal likelihood $p(y_{1:eL})$. We then have that

$$\int \dots \int \tilde{f}^b_{eL}(x_{1:eL}) dx_{1:eL} = \int \dots \int p(x_{1:eL}, y_{1:eL}) dx_{1:eL} = p(y_{1:eL}).$$

Because $\tilde{f}^b_{eL}(x_{1:eL}) = p(x_{1:eL}, y_{1:eL})$ at the iterations corresponding to the end of a batch. This includes the full marginal likelihood $p(y_{1:T})$ as iteration T is the last iteration of batch E. So far, we can view the batch sequence as a general sequence $\Psi_{1:T}$ with some additional requirements at the iterations that corresponds with the first iterations of batch e where $e \geq 2$. With these additional requirements, we can still select the functions ψ^b that define each of the $\psi \in \Psi^s_{1:T}$. This implies that there are still infinitely many valid $\Psi^s_{1:T}$ sequences. Selecting $\psi^b_t = 1$ for every t is a valid choice because then $\psi_t = 1$ for every t. This specific choice results in every twisted transition density and twisted observation function becoming equal to the transition density and observation density from the SSM respectively.

To highlight the conceptual batch structure in the marginal likelihood, we decompose the marginal likelihood into conditional likelihood factors. This is an analogue to the likelihood factors from equation (3.9) that we considered in an online setting. Consider the case of having batches $e = 1, \dots, E$ with iteration eL being the last in batch e. We can then use the standard decomposition of the marginal likelihood $p(y_{1:T})$ given by

$$p(y_{1:T}) = p(y_{1:L}) \prod_{e=2}^{E} p(y_{(e-1)L+1:eL} | y_{1:(e-1)L}). \tag{5.9}$$

We then refer to $p(y_{(e-1)L+1:eL} | y_{1:(e-1)L})$ as a batch likelihood factor. This represents the addition to the marginal likelihood from batch e.

Likelihood estimation

We want the likelihood estimate at the end of a batch to be consistent with the estimated marginal likelihood at the corresponding iteration. This way we can view the likelihood estimate at the end of each batch as an estimate of the marginal likelihood at the corresponding iteration. This is due to target equivalence at the end of the batch. With particle filters in an online setting, we can view the likelihood estimate $\hat{Z}_t$ at each iteration as an estimate of the marginal likelihood at the corresponding iteration. The difference is again in target equivalence, online particle filters with an untwisted target distribution $p(x_{1:t}|y_{1:t})$ has at every iteration t the normalising constant given by $p(y_{1:t})$. In a batch setting, we rely on target equivalence at iteration t in order to have $\tilde{f}^b_t(x_{1:t}) = p(x_{1:t}, y_{1:t})$. This equality will generally not be true unless t corresponds to the last iteration of a batch. Consequently the likelihood estimate at an iteration that is not the last of a batch is not consistent with the corresponding marginal likelihood. This is again because we have target equivalence only at the end of the batches.

Conceptually, we can think of estimating one batch likelihood factor in each batch and then update the marginal likelihood estimate with the estimated batch likelihood factor. This is why we focused on the extension of the unnormalised target distribution $p(x_{1:q}, y_{1:q})$ by the term $\tilde{f}^b_{eL}(x_{q+1:eL}) = p(x_{q+1:eL}, y_{q+1:eL}|x_q)$ where $q+1$ is the first iteration in batch $e \geq 2$ in order to get the next unnormalised target distribution. This implies that

$$\tilde{f}^b_{eL}(x_{1:eL}) = \tilde{f}^b_q(x_{1:q})\tilde{f}^b_{eL}(x_{q+1:eL}) = p(x_{1:eL}, y_{1:eL}).$$

When estimating the marginal likelihood in practice, we focus on the likelihood addition per iteration. We can then think of these additions combined as a batch likelihood factor. In the twisted BPF we are simulating variables from the twisted transition density. We consider the situation of using n particles in the estimates. An estimate of the marginal likelihood following equation (2) in Guarniero, Johansen, and Lee (2017) can be expressed by the twisted observation functions when resampling is assumed to occur at every iteration. We simplify notation by denoting the last iteration of batch $E-1$ by $q = (E-1)L$. The first iteration in the last batch is then denoted by $q+1$ and the last iteration in the last batch is denoted by T. This gives the following expression for the full marginal likelihood estimate

$$\begin{aligned}\hat{Z}^\psi_T = &\left[\frac{1}{n}\sum_{i=1}^n p^\psi_1(y_1|x^i_1)\right]\prod_{s=2}^L\left[\frac{1}{n}\sum_{i=1}^n p^\psi_s(y_s|x^i_s)\right]\\ &\left[\frac{1}{n}\sum_{i=1}^n p^\psi_{L+1}(y_{L+1}|x^i_{L+1}, x^i_L)\right]\prod_{s=L+2}^{2L}\left[\frac{1}{n}\sum_{i=1}^n p^\psi_s(y_s|x^i_s)\right]\\ &\vdots\\ &\left[\frac{1}{n}\sum_{i=1}^n p^\psi_{q+1}(y_{q+1}|x^i_{q+1}, x^i_q)\right]\prod_{s=q+2}^{T}\left[\frac{1}{n}\sum_{i=1}^n p^\psi_s(y_s|x^i_s)\right].\end{aligned}$$

Conceptually, we can think of each line as representing an estimate of the batch likelihood factor for batch e denoted by $\hat{Z}^\psi_{e|e-1}$. This implies the following structure of the estimated likelihood

$$\hat{Z}^\psi_T = \hat{Z}^\psi_1 \prod_{e=2}^E \hat{Z}^\psi_{e|e-1}. \tag{5.10}$$

Likelihood estimates at the end of a batch e should correspond to the estimated marginal likelihood up until the corresponding iteration eL because of target equivalence. Hence we should have that $\hat{Z}^\psi_{eL} \approx \hat{Z}_{eL}$. Further, by considering the likelihood estimate at the last iteration of the last batch, this should correspond to estimating $p(y_{1:T})$. We have considered the likelihood addition per iteration. It is however only at the iterations that correspond to the last of a batch that the estimated likelihood from the batch twisting target distribution is consistent with the likelihood from the untwisted model. This is again due to target equivalence at the end of batches.

Likelihood with batch recursive structure

We now consider likelihood estimation in the case where the batch recursive structure $\psi_t^{b*}(x_t) = p(y_t|x_t)\tilde{\psi}_t^{b*}(x_t)$ from equation (5.8) holds exactly. We can now insert the batch recursive structure into the twisted observation functions in the batch twisted models. We have for the first iterations in batch 1 and for batch $e \geq 2$ with first iteration $q+1$ on the first line and for the remaining iterations on the second line that

$$\begin{aligned} p_1^{\psi}(y_1|x_1) &= \tilde{\psi}_{01}^{b*} & \qquad p_{q+1}^{\psi}(y_{q+1}|x_{q+1}, x_q) &= \tilde{\psi}_{0e}^{b*}(x_q) \\ p_t^{\psi}(y_t|x_t) &= 1 & \qquad \text{for } t &\neq q+1 \end{aligned}$$

In this case, we see that the likelihood estimate at the end of batch e with corresponding iteration eL becomes

$$\hat{Z}_{eL}^{\psi} = \tilde{\psi}_{01}^{b*} \left[\frac{1}{n} \sum_{i=1}^{n} \tilde{\psi}_{02}^{b*}(x_L) \right] \dots \left[\frac{1}{n} \sum_{i=1}^{n} \tilde{\psi}_{0e}^{b*}(x_q) \right] \tag{5.11}$$

and the remaining terms become 1. We see from equation (5.11) that the marginal likelihood estimate at the end of batch e rely on simulated variables. This is the case also when the batch recursive structure holds exactly and this is where we see the effect of not having all observations $y_{1:T}$ available. If the recursive structure holds exactly in an offline setting we would have $\hat{Z}_T^{\psi} = \tilde{\psi}_0^*$. If we use only one batch, we have that the batch setting corresponds to the offline setting and therefore we would have $\hat{Z}_T^{\psi} = \tilde{\psi}_{01}^{b*} = \tilde{\psi}_0^*$. In general, the batch recursive structure do not hold exactly and in addition we are not able to evaluate the terms $\tilde{\psi}_{0e}^{b*}(x_q)$ exactly. For these reasons we expect that the variance of the likelihood estimates in a batch setting will be higher than the equivalent situation in an offline setting. In general we see that the normalising function $\tilde{\psi}_{0e}^{b*}(x_q)$ in batch e depend on the simulated variables of the last iteration of batch $e-1$. Recall that the final target distribution of batch $e-1$ was $p(x_{1:q}|y_{1:q})$.

The optimal situation

For batch 1 we see that if the batch recursive structure holds exactly, we get that the contribution from the first iteration of likelihood estimation is $\tilde{\psi}_{01}^{b*}$. Consider now the the same expansion that was used when estimating the likelihood factors. Now we have a twisted observation function, so this expansion will not correspond to the likelihood factor of the untwisted model, but we consider the expansion

$$\begin{aligned} &\int\int p(y_{q+1}, x_{q+1}, x_q|y_{1:q})dx_q dx_{q+1} \\ =&\int\int p_{q+1}^{\psi}(y_{q+1}|x_{q+1}, x_q)p_{q+1}^{\psi}(x_{q+1}|x_q)p(x_q|y_{1:q})dx_q dx_{q+1} \end{aligned} \tag{5.12}$$

where we see the twisted transition density and the twisted observation function for iteration $q+1$. In addition we recognise the marginal filtering distribution $p(x_q|y_{1:q})$. Recall that iteration q is the last iteration of batch $e-1$ and the final target distribution in batch $e-1$ was $p(x_{1:q}|y_{1:q})$. Assume now that we can approximate the filtering distribution with a weighted measure by using the particles from iteration q. We insert the approximation and have the following

$$\begin{aligned}&\int\int p^{\psi}_{q+1}(y_{q+1}|x_{q+1},x_q)p^{\psi}_{q+1}(x_{q+1}|x_q)\sum_{i=1}^{n} w^i_q \delta_{x^i_q}(x_q)dx_q dx_{q+1}\\ =&\sum_{i=1}^{n} w^i_q p^{\psi}_{q+1}(y_{q+1}|x^i_{q+1},x^i_q).\end{aligned}$$

Assume additionally that we have the filtering density $p(x_q|y_{1:q})$ available in closed form. Because of the batch recursive structure, we have $p^{\psi}_{q+1}(y_{q+1}|x_{q+1},x_q) = \tilde{\psi}^{b*}_{0e}(x_q)$ and recall the interpretation from the optimal normalising function $\tilde{\psi}^{b*}_{0e}(x_q) = p(y_{q+1:eL}|x_q)$ from equation (5.1b). Assume that we can evaluate the normalising function $\tilde{\psi}^{b*}_{0e}(x_q)$ and the integral, then we have

$$\begin{aligned}&\int\int p(y_{q+1:eL}|x_q)p^{\psi}_{q+1}(x_{q+1}|x_q)p(x_q|y_{1:q})dx_q dx_{q+1}\\ =&\int\int p(y_{q+1:eL},x_q,x_{q+1}|y_{1:q})dx_q dx_{q+1} = p(y_{q+1:eL}|y_{1:q}).\end{aligned}$$

The first equality follows from the reformulation

$$p(y_{q+1:eL},x_q,x_{q+1}|y_{1:q}) = p(y_{q+1:eL}|x_q)p^{\psi}_{q+1}(x_{q+1}|x_q)p(x_q|y_{1:q}).$$

5.6 Connection to the offline setting

Our main focus is how we can move from a batch setting to an offline setting and vice versa. The direction from a batch setting to an offline setting is requiring us to increase the lookahead. The other direction effectively requires us to decrease the lookahead. Decreasing lookahead can be regarded as ignoring a subset of future information.

From batch to offline setting

Moving from a batch setting to an offline setting requires all the observations. That is, the lookahead is increased to include all future observations at all iterations. Conceptually, moving from the batch setting to the offline setting can be done by using only one batch which contains all the observations $y_{1:T}$. This implies that the sequence of batch lookahead functions from the first batch will be the same as the offline sequence of lookahead functions, that is $\Psi^b_1 = \Psi_{1:T}$. We then have that $\psi_t = \psi^b_t$ for every t and the batch twisting functions $(\psi^b, \tilde{\psi}^b)$ correspond to the offline twisting functions. In the same manner, we would have only have one batch twisted model being equal to the twisted model. The normalising constant from the first iteration of the first batch would then be equal to the normalising constant in the offline setting, that is $\tilde{\psi}^b_{01} = \tilde{\psi}_0$.

From offline to batch setting

When we are in the batch setting we do not have access to all the observations $y_{1:T}$ at every iteration. From proposition 2 in Guarniero, Johansen, and Lee (2017) we know that the optimal sequence of lookahead functions requires all the observations $y_{1:T}$. This implies that if we are in a batch setting, we cannot approximate the optimal sequence of lookahead functions because we are missing observations outside the current batch. Therefore, to approximate the optimal twisting functions $(\psi^*, \tilde{\psi}^*)$ we need to be in the offline setting.

We have in the previous sections seen that we only use $(\psi^b, \tilde{\psi}^b)$ to define all the functions ψ_t in the batch sequence $\Psi^s_{1:T}$. To move from an offline setting to a batch setting, we can use the batch sequence. If the functions ψ^b that define the functions ψ in the sequence do not depend on observations at all, then the batch sequence simply makes sure that there is target equivalence at the iterations that correspond to the last iteration of a batch. If ψ^b do depend on observations, for example by approximating the optimal batch lookahead functions ψ^{b*}, we can only use observations from within the current batch. In an offline setting however, we have $y_{1:T}$ available at every iteration. This implies that all possible twisting functions we can calculate in a batch setting, we can calculate in an offline setting. This can be done by using only subsets of observations corresponding to the subsets available in each batch when defining the ψ^b in an offline setting. In light of general lookahead strategies from Lin, Chen, and J. S. Liu (2013), using batch twisting functions in an offline setting is suboptimal. This is because using batch twisting functions in an offline setting is equivalent to ignoring future information in every batch except the last batch.

If the objective is to minimise variance, one would not select the batch sequence in an offline setting. This is because the sequence $\Psi^s_{1:T}$ is not equal to the optimal sequence $\Psi^*_{1:T}$ from Definition 4.3.2. We can also look at the recursive structure of the twisting functions resulting from the batch sequence. Assume now that we have the optimal the batch twisting functions $(\psi^{b*}, \tilde{\psi}^{b*})$. We know that the recursive structure holds within the batch as we saw in equation (5.8). Assume that we select the optimal batch twisting functions when defining every ψ_t of the sequence $\Psi^s_{1:T}$. Recall that for the offline optimal twisting functions we have that the recursive structure $\psi^*_t(x_t) = p(y_t|x_t)\tilde{\psi}^*_t(x_t)$ holds for every $t = 1, \dots, T$. We now consider the functions ψ from the sequence $\Psi^s_{1:T}$ and check if the resulting twisting functions follow the recursive structure. The sequence $\Psi^s_{1:T}$ contains functions ψ_t that are equal to the functions ψ^b_t for all iterations except those corresponding to the first of a batch. Therefore we consider the recursive structure for the first iteration of a batch e which we denote by $q+1$. We then have the recursive structure

$$\psi_{q+1}(x_{q+1}, x_q) = p(y_{q+1}|x_{q+1})\tilde{\psi}_{q+1}(x_{q+1}).$$

The left-hand side is equal to

$$\frac{\psi^{b*}_{q+1}(x_{q+1})}{\tilde{\psi}^{b*}_{0e}(x_q)} = \frac{p(y_{q+1:eL}|x_{q+1})}{p(y_{q+1:eL}|x_q)}$$

and the right-hand side is equal to

$$p(y_{q+1}|x_{q+1})\tilde{\psi}^{b*}_{q+1}(x_{q+1}) = p(y_{q+1:eL}|x_{q+1}).$$

We then see that the recursive structure does not hold for iteration $q+1$ when using $\Psi^s_{1:T}$ even when we assume that we have the optimal batch twisting functions $(\psi^{b*}, \tilde{\psi}^{b*})$. From an offline viewpoint, the batch sequence stops influence in the backwards recursive structure from future observations. We consider the recursive structure of iteration q which is corresponding to the last iteration of batch $e-1$, this is

$$\psi_q(x_q) = p(y_q|x_q)\tilde{\psi}_q(x_q) = p(y_q|x_q).$$

Here we see that because of the special definition of $\psi_{q+1}(x_{q+1}, x_q)$ we stop the influence coming from iteration $q+1$ to iteration q when considered from an offline viewpoint. This is why we have $\psi_q(x_q) = p(y_q|x_q)$ when iteration q is the last of a batch with the sequence $\Psi^s_{1:T}$. When we use the optimal lookahead functions from $\Psi^*_{1:T}$, we have that $\psi^*_q(x_q) = p(y_{q:T}|x_q)$. The optimal offline lookahead function contains information from all the future observations $y_{q:T}$. The optimal batch lookahead function only contain information from the current observation y_q.

5.7 Outline of the batch algorithm

In this section we consider an outline of the batch algorithm with iteratively improving batch twisting functions. In addition, we will briefly discuss some aspects of using what we will refer to as alternative twisting functions from Chapter 6 and parametric approximations of twisting functions in the batch algorithm. The optimal batch twisting functions can be approximated in the same ways as the optimal twisting functions in the offline setting. We therefore denote the estimation of batch twisting functions by a generic estimate-ψ function. We denote approximate batch twisting functions by $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$ in the algorithm. For each batch e, we assume that we have observations $y_{(e-1)L+1:eL}$ available at the start of the batch. We want to approximate the optimal $(\psi^{b*}, \tilde{\psi}^{b*})$ and we use the observations available in the batch in order to obtain approximations $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$. These batch twisting functions are then used to define the batch twisted model for the current batch and then this batch twisted model is used in a particle filter. We assume that there are E batches and that each batch consists of L iterations.

In the algorithm outline we denote the first and last iteration of batch e by k and q respectively. We denote a generic particle filter by PF in the algorithm and a particle filter using a twisted model by TT-PF. Once we have the batch twisting functions, TT-PF will just be a particle filter utilising the batch twisted model. We denote the resampling thresholds by α in PF and β in TT-PF. The n particles obtained from TT-PF we refer to as $(x^i_{k:q}, w^i_{k:q})$ for $i = 1, \dots, n$. These are used to estimate the likelihood and to iteratively improve the approximations of the batch twisting functions in the while-loop. We denote the m particles from PF by $(\dot{x}^j_{k:q}, \dot{w}^j_{k:q})$ for $j = 1, \dots, m$. These particles are used in the first approximation of the batch twisting functions, in the following approximations within the batch we use particles from the twisted particle filter to iteratively improve $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$.

The PF use the transition and observation density of the SSM to obtain the initial particles. The improve parameter may be based on the stability of the likelihood estimates such as in Guarniero, Johansen, and Lee (2017) or it may be a fixed number of repeats, it is always set to true when a new batch of observations become available. In addition, it is assumed that PF and TT-PF uses the batch number e in order to select the appropriate transition density and twisted transition density respectively. For the batch likelihood estimates from equation (5.10) we define $\hat{Z}_0^{\psi} \equiv 1$ and $\hat{Z}_{1|0}^{\psi} \equiv \hat{Z}_1^{\psi}$ for notational simplicity. The final likelihood estimate is then $\hat{Z}_E^{\psi} = \hat{p}(y_{1:T})$ and the intermediate batch likelihood estimates are $\hat{Z}_e^{\psi} = \hat{p}(y_{1:eL})$. The estimation of $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$ in the

Algorithm 9 Batch algorithm outline

```
 1: for $e \in (1, \dots, E)$ do
 2:     $k = (e-1)L + 1$
 3:     $q = \min(eL, T)$
 4:     $x_{k:q}, w_{k:q} = \dot{x}_{k:q}, \dot{w}_{k:q} = \mathrm{PF}(\dot{x}_{k-1}, \dot{w}_{k-1}, e, m, \alpha)$
 5:     while improve do
 6:         $(\bar{\psi}^b, \tilde{\bar{\psi}}^b) = \text{estimate-}\psi(x_{k:q}, w_{k:q})$
 7:         $x_{k:q}, w_{k:q} = \text{TT-PF}(\bar{\psi}^b, \tilde{\bar{\psi}}^b, x_{k-1}, w_{k-1}, e, n, \beta)$
 8:     end while
 9:     $\hat{Z}_e^{\psi} = \hat{Z}_{e-1}^{\psi} \hat{Z}_{e|e-1}^{\psi}$
10: end for
```

generic function estimate-ψ can be done using the backwards recursive algorithm Algorithm 7. For the TT-PF, we can use Algorithm 6. In order to use TT-PF we need the batch twisted model which again requires the batch twisting functions. The batch twisting functions are assumed to be returned by the generic function denoted by estimate-ψ. The algorithm also includes iteratively improvement of $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$ in the improve step. This is similar to the iAPF. We use the particles from a TT-PF to estimate the optimal batch twisting functions. Then these approximate batch twisting functions are used in TT-PF to generate new particles. Hence we can get iteratively improving batch twisting functions.

Batch algorithm with alternative twisting functions

We can set up a possible version of Algorithm 9 that uses what we will refer to as the alternative approximation of twisting functions. We consider this alternative approximation in Chapter 6, but this can be used in the same way we can use nonparametric or parametric approximations of $(\psi^b, \tilde{\psi}^b)$. The main alteration necessary when using the alternative approximation of twisting function is that we need to estimate smoothing weights within each batch. These estimated smoothing weights are used in the generic estimate-ψ function. The particles from PF will be used to estimate the smoothing weights. When using the alternative approximation, we have that the approximation $\bar{\psi}^b$ and the transition density generally are not conjugate. This implies that the integral $\tilde{\bar{\psi}}^b$ in general is not available in closed form and has to be approximated numerically. It also implies that the twisted transition density is not available in closed form and we simulate from it by using an importance sampling-based method.

Intuitively, we expect the approximation of the batch twisting functions $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$ to be more accurate when the batches contain more iterations. That is, L increases and we are including more future information through the future observations $y_{t:eL}$ at iteration t. If we have that $L = T$, we include all future observations. This corresponds to using one batch which again corresponds to the offline setting. For the batch twisted models we use the structure defined in Section 5.3. Further, we use the batch twisted models in a twisted particle filter such as Algorithm 6. We note that there are many computationally expensive components when using the alternative approximations of the twisting functions. Because of this, we only have numerical experiments with relatively few particles when using this alternative approximation of the twisting functions.

Batch algorithm with parametric twisting functions

We can approximate the optimal batch lookahead function ψ^{b*} by using a parametric approximation. Even though more restrictive than nonparametric approximations, we select a parametric approximation of the optimal batch lookahead function. We also select a parametric form such that the approximations are conjugate to the transition density of the SSM. The advantages of conjugacy between the approximation $\bar{\psi}$ and the transition density in the offline setting also holds for the batch setting.

Using a parametric form for the approximate batch twisting functions $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$ includes selecting a class that defines the parametric form of the approximate batch lookahead functions $\bar{\psi}^b$. Compared to nonparametric approaches of approximating the batch twisting functions, we are sacrificing flexibility for reduced computational cost. Because of the conjugacy, we have a closed form expressions for the normalising function $\tilde{\bar{\psi}}^b$ and the twisted transition density. This represent a significant reduction in computational cost compared to the alternative approximation and nonparametric approximation of batch twisting functions. Because our main focus is the batch structure we follow section 3 of Guarniero, Johansen, and Lee (2017) and use a Gaussian approximation of the optimal lookahead function. A proposed structure of the approximation in Guarniero, Johansen, and Lee (2017, eq. 12) is given by a mixture of Gaussians. We follow this general strategy when selecting the approximate batch twisting functions $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$. We define approximations $\bar{\psi}^b$ of the optimal batch lookahead function on the following form

$$\bar{\psi}^b_t(x_t) = \mathcal{N}(x_t; \mu_t, \Sigma_t).$$

We use this approximate batch lookahead function for both one dimensional and multidimensional approximations. In the case of multidimensional approximations we use a diagonal covariance matrix Σ_t in the approximate batch lookahead functions. We can approximate the batch twisting functions $(\psi^b, \tilde{\psi}^b)$ by choosing a parametric form and then use Algorithm 7 for each batch $e = 1, \ldots, E$. We are then approximating the batch twisting functions within each batch by using the batch recursive structure. Once we have the approximations $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$ we can use the the batch twisted model and the batch twisting functions in a twisted particle filter.

In the twisted particle filter, we set the parameters $t_{\text{start}} = k$ and $t_{\text{stop}} = q$ for a batch e. At iteration q we have obtained n batch particles, denoted by $(x^i_{k:q}, w^i_{k:q})$ for $i = 1, \ldots, n$. With the while-loop we utilise the batch particles in order to estimate the batch twisting functions again. This can be repeated as long as the condition improve is true. The newest $(\bar{\psi}^b, \tilde{\bar{\psi}}^b)$ is then used to find new batch twisted particles $(x^i_{k:q}, w^i_{k:q})$ for $i = 1, \dots, n$. The iteratively improving batch twisting functions approximations are similar to the iteratively improving twisting function approximations in iAPF. This is motivated by minimisation of the variance of the likelihood ratio in the asymptotic case as shown in proposition 3 of Guarniero, Johansen, and Lee (2017). The improve condition is not specified in the algorithm and it can be on the form

$$\frac{\mathrm{sd}(\hat{Z}^{\psi}_e)}{\mathrm{mean}(\hat{Z}^{\psi}_e)} < \tau$$

for batch e. This is the same improve condition as in Guarniero, Johansen, and Lee (2017), but here it is used in each batch. We can use this condition with the estimated marginal likelihood at the end of each batch. We would then use a stability parameter such as s as in Algorithm 8 in order to calculate the condition for the last s estimates of the marginal likelihood. Note that by using a stochastic stopping criterion we would need to run the twisted particle filter once more with the latest batch twisting functions after the while-loop as is done in the iAPF. Then we would use the obtained particles to estimate the marginal likelihood at the end of the batch. A simpler condition for the iterative improvement is to iterate a fixed number of times in the while-loop which we will refer to as repeats. This parameter refers to the number of improvement repetitions, that is, the number of times to improve the batch twisting functions after the first necessary calculation of the batch twisting functions.

CHAPTER 6

Alternative twisting functions

In this chapter, we consider first a reformulation of the optimal lookahead function and an approximation of this reformulation. We have previously not seen this approximation in use in twisting target algorithms. We will refer to this alternative formulation of the optimal lookahead function as the alternative lookahead function and we denote it by $\psi_t'(x_t)$ for $t = 1, \ldots, T$. This alternative lookahead function is based on a direct reformulation of the optimal lookahead function $\psi_t^*(x_t)$ for $t = 1, \ldots, T$. We also use the simplified notation ψ' and ψ^* when we discuss the lookahead functions in general. As is also the case for ψ^*, we generally do not have analytical expressions for the alternative lookahead functions ψ' available.

We generally denote approximations of lookahead functions by $\bar{\psi}$ and we usually approximate the optimal lookahead functions, that is $\bar{\psi} \approx \psi^*$. As the alternative lookahead function is a direct reformulation of the optimal lookahead function, we will also consider approximations of the alternative lookahead function $\bar{\psi}' \approx \psi'$. The motivation for approximating ψ' is the connection to the optimal ψ^* and by utilising $\bar{\psi}'$ our aim is to reduce variance of the likelihood estimates.

In this chapter, we also consider a Laplace-based approximation of the observation density which can be used in order to approximate the optimal lookahead function. This approximation can in some special cases can reduce the computational cost when approximating the optimal twisting functions $(\psi^*, \tilde{\psi}^*)$. We will refer to this approximation as the Laplace approximation to distinguish it from the twisting functions based on the alternative lookahead functions.

6.1 Motivation

In general, the function $\psi_t(x_t)$ can be defined under Assumptions 4.2.1 and the normalising function $\tilde{\psi}_{t-1}(x_{t-1})$ are defined in Definition 4.2.1 as an expectation of $\psi_t(x_t)$ with respect to the transition density $p(x_t|x_{t-1})$. This is the main motivation for approximating $\psi_t(x_t)$ with a parametric form that is conjugate to the transition density in Guarniero, Johansen, and Lee (2017). We also know that the optimal lookahead function can be defined conceptually as $\psi_t^*(x_t) = p(y_{t:T}|x_t)$ and for that reason our objective is to approximate this function for $t = 1, \ldots, T$. When considering the approximate twisting functions, denoted by $(\bar{\psi}_t(x_t), \tilde{\bar{\psi}}_t(x_t))$ for $t = 1, \ldots, T$ we refer to these as $(\bar{\psi}, \tilde{\bar{\psi}})$ in simplified notation when we discuss all the approximate twisting functions.

Regardless of using a parametric or a nonparametric approximation, the objective is approximations $\bar{\psi}_t(x_t) \approx p(y_{t:T}|x_t)$ for $t = 1, \ldots, T$. The main advantage of using parametric approximations is that there are often a small number of parameters to estimate. Additionally, we have the approximate normalising function $\tilde{\bar{\psi}}$ in closed form if the approximation $\bar{\psi}$ is conjugate to the transition density. The disadvantage of using a parametric $\bar{\psi}$ is that we have to select a specific class, $\bar{\Psi}$, for the parametric form in advance which may be restrictive.

If we have the optimal lookahead functions we can obtain the optimal twisting functions denote by $(\psi^*, \tilde{\psi}^*)$. We can use these twisting functions in the iAPF with a twisted model in a twisted particle filter. From the twisted particle filter we can estimate the likelihood. Another approach to twisting functions is given in Ala-Luhtala et al. (2016). Here it is proposed to approximate the optimal lookahead function, $\psi^*_t(x_t)$, at iteration t with a constant c observations of lookahead. This can be written as $\psi^*_t(x_t) \approx p(y_{t:t+c}|x_t)$. The function $p(y_{t:t+c}|x_t)$ can be approximated by an exponential form when considering nonlinear Gaussian SSMs. Local linearisation can then be used in order to estimate the parameters of the exponential form (Ala-Luhtala et al., 2016, p. 4881). In Guarniero, Johansen, and Lee (2017), two main types of approximations for the optimal lookahead functions are considered. These are parametric and nonparametric approximation of the optimal lookahead functions ψ^*. The parametric approximations may be too restrictive in some situations as we are required to determine a class $\bar{\Psi}$ for the parametric approximate lookahead functions a priori. Fully nonparametric approximations are often flexible, but implies high computational cost as is discussed in section 6 of Guarniero, Johansen, and Lee (2017).

With the alternative lookahead functions in the following section, our aim is to utilise the conditional independence structure of the SSM combined with the conceptual form of the optimal lookahead function, $\psi^*_t(x_t) = p(y_{t:T}|x_t)$. The motivation for the alternative twisting functions is that the alternative lookahead function ψ' can be approximated without the need for numerical optimisation as is the case for parametric approximation of ψ^* as we saw in Chapter 4.

Our motivation for the alternative lookahead function is based on the trade-off between selecting the parametric or nonparametric approximation of the lookahead functions. By not requiring a parametric class of functions to be selected in advance, our aim is to gain flexibility in the approximations. Conceptually, all the components needed in the approximate alternative lookahead functions are provided by the SSM. In that sense, we can view the approximate alternative lookahead function as a trade-off between the parametric and nonparametric approximation. Our motivation for the Laplace-based approximation is based on reducing computational cost compared to utilising numerical optimisation when approximating the lookahead functions.

6.2 Alternative twisting functions

The alternative twisting functions are based on a reformulation of the optimal lookahead function which is given by

$$\psi_t^*(x_t) = p(y_{t:T}|x_t) = p(y_{t:T}|x_t, y_{1:t-1}).$$

Here we are adding $y_{1:t-1}$ back into the conditioning set at the second equality. In the same way that $y_{1:t-1}$ can be excluded from the conditioning set it can also be included. This is possible because the variable x_t blocks influence between $y_{t:T}$ and $y_{1:t-1}$. We can utilise the expression with the modified conditioning set to further reformulate the expression for the optimal lookahead function,

$$\begin{aligned}\psi_1^*(x_1) &= p(y_{1:T}|x_1) = \frac{p(x_1|y_{1:T})p(y_{1:T})}{p(x_1)}\\ \psi_t^*(x_t) &= p(y_{t:T}|x_t, y_{1:t-1}) = \frac{p(x_t|y_{1:T})p(y_{t:T}|y_{1:t-1})}{p(x_t|y_{1:t-1})}.\end{aligned}$$

We now recognise the marginal smoothing distribution as the first term in the numerator. In the denominator we see the, marginal, predictive distribution from equation (3.3a). The conditional likelihood term, $p(y_{t:T}|y_{1:t-1})$, in the numerator does not depend on x_t and will be fixed when we regard the observations as fixed. When viewing the optimal lookahead function as a function of x_t it can also be considered constant. Expanding the predictive distribution in the denominator then gives us

$$\begin{aligned}\psi_1^*(x_1) &= \frac{p(x_1|y_{1:T})p(y_{1:T})}{p(x_1)}\\ \psi_t^*(x_t) &= \frac{p(x_t|y_{1:T})p(y_{t:T}|y_{1:t-1})}{\int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}}.\end{aligned}$$

We are now able to approximate all the terms in the expression by well-known algorithms. In this section, we will follow the setup of Guarniero, Johansen, and Lee (2017) by combining twisting functions with a twisted model. Our goal in the following sections is therefore to approximate a sequence of functions that we will denote by $\Psi'_{1:T}$. Using this sequence of what we will refer to as alternative lookahead functions, we can calculate what we will refer to as the alternative twisting functions. This can be denoted by $(\psi'_t(x_t), \tilde{\psi}'_t(x_t))$ for $t = 1, \ldots, T$. We will use the following, simplified notation denoted by $(\psi', \tilde{\psi}')$ when we are referring to the alternative twisting functions for all the iterations. Once we have the alternative twisting functions, we can utilise a twisted model which we again can use in a twisted particle filter.

We now define the sequence $\Psi'_{1:T}$ of alternative lookahead functions $\psi'_t(x_t)$ for $t = 1, \ldots, T$. The definition of the alternative lookahead functions is based on the reformulation in equation (6.1) of the optimal lookahead function $\psi_t^*(x_t)$.

Definition 6.2.1. *The alternative lookahead functions are defined by*

$$\psi'_t(x_t) = \begin{cases} \frac{p(x_1|y_{1:T})p(y_{1:T})}{p(x_1)} & \text{if } t = 1\\ \frac{p(x_t|y_{1:T})p(y_{t:T}|y_{1:t-1})}{p(x_t|y_{1:t-1})} & \text{if } t > 1.\end{cases}$$

Note that the terms on the form $p(y_{t:T}|y_{1:t-1})$ will cancel if we use the alternative lookahead functions ψ' in a twisted model. Note also that if we had been able to evaluate $\psi'_t(x_t)$ exactly at each iteration t we would have $\psi'_t(x_t) = \psi^*_t(x_t)$ for $t = 1, \dots, T$. Also note that to evaluate $\psi'_t(x_t)$ at iteration t would require all observations $y_{1:T}$. Hence we see from Definition 6.2.1 that an offline setting is required if we are to use ψ' directly. Because $\psi'_t(x_t)$ is a reformulation of $\psi^*_t(x_t) = p(y_{t:T}|x_t)$ which requires an offline setting, the offline setting is also expected for $\psi'_t(x_t)$. The definition of ψ' also allows us to define a sequence similar to Definition 4.2.2.

Definition 6.2.2. *Denoting $\psi'_t(x_t) = \psi'_t$. We have a sequence of alternative lookahead functions, denoted by*

$$\Psi'_{1:T} \equiv (\psi'_1, \dots, \psi'_T)\,.$$

Having defined the lookahead functions, ψ', we can now consider the normalising functions. When we have the lookahead functions ψ' we can define the normalising functions. Following the structure from Definition 4.2.1 we have that the normalising functions and the normalising constant based on the alternative lookahead functions can be denoted by $\tilde{\psi}'$.

Definition 6.2.3. *For the normalising functions, we have:*

$$\tilde{\psi}'_t(x_t) = \begin{cases} \int \psi'_{t+1}(x_{t+1}) p(x_{t+1}|x_t) dx_{t+1} & \textit{if } t < T \\ 1 & \textit{if } t = T. \end{cases}$$

For the normalising constant, we have:

$$\tilde{\psi}'_0 = \int \psi'_1(x_1) p(x_1) dx_1. \tag{6.2}$$

In combination with Definition 6.2.1, the alternative twisting functions $(\psi', \tilde{\psi}')$ are now defined and therefore can be used with the twisted model. There are generally many valid sequences $\Psi_{1:T} = (\psi_1, \dots, \psi_T)$ which satisfy final target equivalence which we saw in Section 4.2. If we have an analytical expression for $\psi'_t(x_t)$ that we can evaluate exactly, we know that it is equal to $\psi^*_t(x_t)$ because $\psi'_t(x_t)$ is a reformulation of the optimal lookahead function. We also know that using approximations to the optimal twisting functions $(\psi^*, \tilde{\psi}^*)$ in combination with a twisted model can yield likelihood estimates with low variance (Guarniero, Johansen, and Lee, 2017). If we can find an accurate approximation of the alternative twisting functions, $(\psi', \tilde{\psi}')$, this approximation may also be close to the optimal twisting functions. This again imply that we can obtain low variance likelihood estimates from using approximations to the alternative twisting functions.

Assume now that we use a parametric approximation $\bar{\psi}$. Unless $\bar{\psi} \in \Psi^*$, that is the parametric approximation is in the class of optimal lookahead functions, $\bar{\psi}$ cannot be equal to the optimal lookahead function. By selecting a class $\bar{\Psi}$ of parametric approximations that are conjugate to the transition density of the SSM we are restricting the flexibility of $\bar{\psi}$. This implies that we might sacrifice closeness to the optimal lookahead function for being able to evaluate the integral $\tilde{\bar{\psi}}$ exactly. A possible motivation for the alternative lookahead function $\psi'_t(x_t)$ is that we are approximating the optimal lookahead function

ψ^* directly. By not using a parametric $\bar{\psi}$ conjugate to the transition density we are in general giving up exact evaluation of the integral $\tilde{\bar{}}$ and we have to approximate $\tilde{\bar{\psi}}$. It can be feasible to give up exact evaluation of the integral $\tilde{\bar{\psi}}'$ if the approximation $\bar{\psi}'$ is good.

6.3 Approximation of the alternative twisting functions

In order to approximate ψ^*, we will start by using the reformulation ψ' and approximate this instead of approximating ψ^* directly. Conceptually we can consider increasing the accuracy of the approximations for the three components in the ψ'-functions. Recall that we have the smoothing distribution $p(x_t|y_{1:T})$ and the likelihood term $p(y_{t:T}|y_{1:t-1})$ in the numerator. We also have the predictive distribution $p(x_t|y_{1:t-1})$ in the denominator.

To define an approximation of ψ', denoted by $\bar{\psi}'$, we need to approximate the components in Definition 6.2.1. We will consider how to approximate the smoothing distribution, the likelihood term and the predictive distribution below. Informally, we can think of improving the accuracy of the three estimators by increasing the number of particles used. We can then think of increasing the number of particles as an alternative to iteratively improving the lookahead functions. This is however a simplification because some of the motivation for the iteratively improving the approximations in Guarniero, Johansen, and Lee (2017) is to utilise the particles obtained from a twisted particle filter to approximate the lookahead functions. Then the approximate lookahead functions are used to obtain variables which in themselves should closer to perfect samples from $p(x_{1:T}|y_{1:T})$. This is in contrast to estimating the three components in ψ' one time with a large number of particles. We however expect that the approximations $\bar{\psi}'$ obtained by using a large number of particles to provide more variance reduction in the likelihood estimates than approximations obtained by using fewer particles.

Smoothing distribution

For the smoothing distribution $p(x_t|y_{1:T})$, we can in an offline setting use any smoothing algorithm to approximate the smoothing weights. Methods such as the forward filtering backward smoothing as discussed in e.g. Douc et al. (2011) may be used to obtain smoothing weights though at a relatively high computational cost. There are also other smoothing algorithms available that have lower computational cost, see e.g. Fearnhead, Wyncoll, and Tawn (2010).

We are mainly interested in approximating the marginal smoothing distributions for the different iterations $t = k, \dots, q$ where $q > k$. In an offline setting we would have $k = 1$ and $q = T$, but we can think of k and q as the first and last iteration of a batch respectively. In that sense, we are interested in the marginal distributions $p(x_t|y_{k:q})$ where $k \leq t \leq q$. To approximate the smoothing weights, we use the fixed-interval smoothing algorithm from Doucet, Godsill, and Andrieu (2000). For that reason we assume that when starting the estimation of the smoothing weights, we have available $(x_t^j, \tilde{w}_t^j)$ for $j = 1, \dots, m$ particles for every $t = k, \dots, q$. That is, we have m particles to approximate the marginal filtering distribution at every iteration t within the batch. Here we think of the offline setting as using only one batch with $k = 1$ and $q = T$.

In practice we ran a BPF and stored matrices of the required sets of particles. Then we ran the fixed-interval smoothing algorithm for $t = q, \dots, k$. In the outline of Algorithm 9 we ran the fixed-interval smoothing algorithm within the generic estimate-ψ function.

As mentioned in Doucet, Godsill, and Andrieu (2000), the fixed-interval smoothing algorithm uses the same variables as the approximated marginal filtering distributions as support and changes only the weights when iterating backwards. Because of this, the support of the approximate smoothing distribution still consists of discrete points. We however need continuous versions for the discrete approximations of the smoothing distribution. This is because we in general need to evaluate $\bar{\psi}_t'(x_t)$ for a x_t different than that of the discrete support of the smoothing weights.

A natural idea for a solving this is by employing nonparametric density estimation. One common approach is kernel density estimation. Consider first estimation of the generic density function p. In general we want a kernel K to be decreasing around x such that sample values x_t^j that are far away from x will influence the density estimate of $p(x)$ less than those that are closer to x. We denote the kernel density estimator by $\hat{p}(x)$. This is then given by

$$\hat{p}(x) = \frac{1}{h} \sum_{j=1}^{m} w_t^j K\left(\frac{x - x_t^j}{h}\right). \tag{6.3}$$

Here h is the bandwidth which conceptually can be viewed as a scale parameter. K indicates a typically symmetric kernel, e.g. a Gaussian. The parameter w_t^j is usually set $w_t^j = \frac{1}{n}$ for $j = 1, \dots, m$ in traditional kernel density estimation (Givens and Hoeting, 2013, pp. 325-328). In addition, the variables x_t^j for $j = 1, \dots, m$ are assumed to be simulated directly from $p(x)$ in traditional kernel density estimation (Hastie, Tibshirani, and J. Friedman, 2009, p. 208). From standard kernel density estimation it is also established that the value of the bandwidth h represents a bias-variance trade-off for the mean integrated squared error (MISE), see e.g. Givens and Hoeting (2013, pp. 329-331). When considering x_t coming from an algorithm, instead of being simulated directly from $p(x)$, we cannot use the traditional kernel density estimation setup. Adjustments to the kernel density estimation are required. These are done in order to use the particles resulting from an importance sampling-based algorithm in the kernel density estimation framework.

We want to estimate $p(x_t|y_{1:t})$ using the variables x_t^j for $j = 1, \dots, m$ by using kernel density estimation. This is done in Crisan and Miguez (2014) where a relationship between the bandwidth h and the number of variables m from the PF is given as a boundary. In addition, a similar approximation as equation (6.3) is given, when using x_t obtained with a particle filter. Let d_x denote the dimensionality of x, we then have the following relations adapted from Crisan and Miguez (2014) to our notation

$$h \geq m^{-\frac{1}{2(d_x+1)}} \qquad \hat{p}(x|y_{1:t}) \equiv \frac{1}{hm} \sum_{j=1}^{m} K\left(\frac{x - x_t^j}{h}\right).$$

Based on the same intuition as in Crisan and Miguez (2014), we would like to approximate the smoothing distribution by using symmetric kernels around the particles combined, but combined with the smoothing weights. We denote the density kernel approximation of the marginal smoothing distribution by $\hat{f}_s(x_t)$.

When using a kernel density estimate, the bandwidth, h, of the kernel function is important in terms of the bias-variance tradeoff for the estimator. In general, this could be adaptive in the sense that it varies with each observation j used in the kernel density estimate for each t or fixed in the sense that it is constant for each t. This gives two possible options for selecting the bandwidth h in the approximation.

Fixed bandwidth: We use constant h for each t, Crisan and Miguez (2014) provides a lower bound $h \geq m^{-\frac{1}{2(d_x+1)}}$ where d_x denotes the dimensionality of x_t.

Adaptive bandwidth: We have a separate bandwidth, denoted by h^j, for each observation j used in the kernel density estimate. This allows the bandwidth to adapt to each observation j (Brewer, 2000).

The fixed bandwidth case for h differs from approximations to the optimal h in standard kernel density estimation which is discussed in Givens and Hoeting (2013). The distinction from standard kernel density estimation is that the variables are not i.i.d. when obtained from particle filters and not simulated directly from the density we are trying to estimate hence we have only have what Crisan and Miguez (2014) refers to as "approximate samples". That is weighted samples from the distribution. This is in contrast to standard kernel density estimation as discussed in e.g. chapter 9 of Givens and Hoeting (2013) or chapter 6.6 of Hastie, Tibshirani, and J. Friedman (2009). Continuing, we select the fixed bandwidth-approach, following Crisan and Miguez (2014). Assume now we have m particles consisting of the variables and smoothing weights resulting from the fixed-interval smoothing. We denote the smoothing weights by $\dot{w}s_t^j$ for particle j at iteration t. Further, K denotes a symmetric kernel function and h denotes the fixed bandwidth. We can then consider an approximation of the smoothing density denoted which we denote by $\hat{f}_s(x_t)$, we then have

$$\hat{f}_s(x_t) = \frac{1}{h} \sum_{j=1}^{m} \dot{w}s_t^j K \left(\frac{x_t - x_t^j}{h} \right). \tag{6.4}$$

Naturally, we see that this approach can be computationally demanding. First we need to estimate m smoothing weights and then in order to evaluate the functions we have to evaluate the kernel function m times. In general kernel density estimation we have computational complexity in the order of $\mathcal{O}(m)$ see e.g. chapter 6 of Hastie, Tibshirani, and J. Friedman (2009). Evaluation of $\hat{f}_s(x_t)$ for one x_t is computationally demanding. Typically we have x_t^i for $i = 1, \ldots, n$ where n is the number of particles used in the twisted particle filter.

Likelihood term

For the likelihood term, $p(y_{t:T}|y_{1:t-1})$, we see can reformulate it in the following way

$$p(y_{t:T}|y_{1:t-1}) = \frac{p(y_{1:T})}{p(y_{1:t-1})} = \frac{p(y_1)\prod_{s=2}^{T} p(y_s|y_{1:s-1})}{p(y_1)\prod_{s=2}^{t-1} p(y_s|y_{1:s-1})} = \prod_{s=t}^{T} p(y_s|y_{1:s-1})$$

for $t > 1$. The likelihood factors $p(y_s|y_{1:s-1})$ can be estimated with a standard PF as we saw in equation (3.11). We see that all the likelihood terms include all future observations $y_{t:T}$. A straightforward approach for approximating all the likelihood factors $p(y_s|y_{1:s-1})$ used to estimate the likelihood term is to run a particle filter and store the estimated likelihood factors. Hence we need the approximations $\hat{p}(y_1)$ and $\hat{p}(y_s|y_{1:s-1})$ for $s = 2, \ldots, T$ in order to estimate the likelihood term.

Predictive distribution

We saw an approximation of the marginal predictive distribution $p(x_t|y_{1:t-1})$ when considering the filtering recursions. Recall the approximation in equation (3.4a), repeated here for completeness

$$\hat{p}(x_t|y_{1:t-1}) = \sum_{i=1}^{n} w_{t-1}^{i} p(x_t|x_{t-1}^{i})$$

The weights used in this estimator is also generally available after running a PF up until iteration $t-1$. In addition the transition density of the SSM is assumed known and it will generally ensure that we can evaluate $\hat{p}(x_t|y_{1:t-1})$.

Alternative lookahead function

Having the necessary components for the estimators, we can now combine these into approximate alternative lookahead functions. We denote the approximation of the alternative lookahead function $\psi_t'(x_t)$ by $\bar{\psi}_t'(x_t)$. We then have that

$$\bar{\psi}_t'(x_t) = \frac{\hat{f}_s(x_t)\hat{p}(y_{t:T}|y_{1:t-1})}{\sum_{j=1}^{m} w_{t-1}^{j} p(x_t|x_{t-1}^{j})} \tag{6.5}$$

$$\bar{\psi}_t'(x_t) \propto \frac{\hat{f}_s(x_t)}{\sum_{j=1}^{m} w_{t-1}^{j} p(x_t|x_{t-1}^{j})} \tag{6.6}$$

can be used as approximate alternative lookahead functions. In practice we can use the approximation in equation (6.6) as the term $\hat{p}(y_{t:T}|y_{1:t-1})$ is a constant that will cancel out when considering the final target distribution. The advantage of the alternative lookahead function $\bar{\psi}'$ is that we know how to estimate the three components. In addition, the accuracy of each of the components can to some extent be increased with increased computational cost. That is, by increasing computational time, we can generally increase the accuracy for the three components. Another possible advantage is that we do not have to select a class $\bar{\Psi}$ for the parametric approximations of the lookahead functions a priori. This is only a possible advantage of ψ' compared to a parametric approximation of a lookahead function, denoted by $\bar{\psi}$. Using a parametric approximation of the lookahead function that is conjugate to the transition density lets us evaluate the integral $\tilde{\bar{\psi}}$ analytically. This often reduces computational cost significantly and in addition it often lets us find a closed form expression for the twisted transition density in a twisted model. This will in turn also reduce computational cost, we now consider briefly a way to simulate approximately from the twisted transition density.

Simulating from the twisted transition density

Because the approximation $\bar{\psi}'_t(x_t)$ and the transition density are not conjugate, we generally need alternative methods for simulating from the twisted transition density. In order to simulate from the twisted transition density, we use a version of nested IS adapted from algorithm 5 in Naesseth, Lindsten, and Schön (2019). Here, we denote the unnormalised target distribution by $\tilde{f}_t(x_t)$. We set this equal to the unnormalised twisted transition density as follows

$$\tilde{f}_t(x_t) = p(x_t|x^i_{t-1})\bar{\psi}'_t(x_t).$$

Hence we have the the normalised target distribution then is $f_t(x_t) = p^{\psi}_t(x_t|x_{t-1})$. In the algorithm we then use the transition density $p(x_t|x_{t-1})$ of the SSM as the importance sampling function. We define the special case $p(x_1|x^i_0) \equiv p(x_1)$ and at line 5 we sample one value for x^i_t based on the normalised weights. Note that we first sample $k = 1, \dots, s$ values $\bar{x}^k_t$ for each x^i_{t-1}, then these s values are weighted and one of $\bar{x}_t$ is sampled as x^i_t. We here note that when s increase, this will also be a computationally costly algorithm.

Algorithm 10 Approximate simulation from twisted transition density

```
1: for $i \in (1, \dots, n)$ do
2:     $\bar{x}^k_t \sim p(x_t|x^i_{t-1})$ $k = 1, \dots, s$
3:     $\tilde{w}^k_t = \frac{\tilde{f}_t(\bar{x}^k_t)}{p(\bar{x}^k_t|x^i_{t-1})}$
4:     $w^k_t = \frac{\tilde{w}^k_t}{\sum_{l=1}^{s} \tilde{w}^l_t}$
5:     $x^i_t \sim \sum_{k=1}^{s} w^k_t \delta_{\bar{x}^k_t}(x_t)$
6: end for
```

Normalising functions

Recall the normalising functions, $\tilde{\psi}_t(x_t)$, defined by the integral in Definition 4.2.1. In order to obtain the approximate alternative twisting functions $(\bar{\psi}', \tilde{\bar{\psi}}')$ we also need approximations for the normalising functions. The integral from which these are defined is in general intractable, unless the selected approximation $\bar{\psi}$ is conjugate to the transition density. For this reason we need to approximate the approximate alternative normalising functions. Note that we also have to approximate the alternative lookahead function $\psi'_{t+1}(x_{t+1})$ in the integrand by $\bar{\psi}'_{t+1}(x_{t+1})$, hence we are not directly approximating the integral $\tilde{\psi}'_t(x_t)$. Starting with the normalising functions for $0 < t < T$, we can use a variant of the Monte Carlo method after approximating the integrand in order to estimate the integral. We then have

$$\bar{\tilde{\psi}}'_t(x_t^i) = \int \bar{\psi}'_{t+1}(x_{t+1}) p(x_{t+1}|x_t^i) dx_{t+1}$$

$$\bar{\tilde{\psi}}'_t(x_t^i) \approx \frac{1}{s} \sum_{k=1}^{s} \bar{\psi}'_{t+1}(x_{t+1}^k) \qquad x_{t+1}^k \sim p(x_{t+1}|x_t^i).$$

Note here that for each particle $i = 1, \dots, n$ at iteration t we sample s variables from $p(x_{t+1}|x_t^i)$ and are therefore estimating one integral per variable x_t^i. Ideally, we also would like $s \to \infty$ to get more accurate estimates of the integral. This implies a computational cost in the order of $\mathcal{O}(sn)$ which quickly can be intractable. We also need to approximate the normalising constant $\bar{\tilde{\psi}}'_0$ from equation 6.2. This can be estimated in the same manner as for the normalising functions

$$\bar{\tilde{\psi}}'_0 \approx \frac{1}{s} \sum_{k=1}^{s} \bar{\psi}'_1(x_1^k) \qquad x_1^k \sim p_1(x_1).$$

Alternatively we can also utilise that for the optimal ψ^*-functions, we know that the normalising constant $\tilde{\psi}_0^*$ should be equal to the marginal likelihood. In an offline setting we already have an estimate of the marginal likelihood denoted by $\hat{p}(y_{1:T})$ from the PF which we use to calculate the likelihood terms and the particles used in the estimation of the smoothing weights. In an offline setting we may use this as an alternative to Monte Carlo integration for the normalising constant. That is, we can set $\bar{\tilde{\psi}}'_0 = \hat{p}(y_{1:T})$ by using the likelihood factors from the particle filter. Another aspect of the normalising constant $\bar{\tilde{\psi}}'_0$ is that for the sequence optimal $\Psi_{1:T}^*$ in Guarniero, Johansen, and Lee (2017), all the estimated likelihood factors in the likelihood estimation except the first is equal to 1. This will obviously not be the case when we are using approximate lookahead functions $\bar{\psi}$. It still could motivate an increase in the Monte Carlo sample size s when estimating the normalising constant $\bar{\tilde{\psi}}'_0$.

The approximate alternative twisting functions which we denote by $(\bar{\psi}', \bar{\tilde{\psi}}')$ can also be used in the batch algorithm from Chapter 5. To use the alternative lookahead function in a batch sequence we define $\psi^b = \psi'$ and then insert this into the batch sequence $\Psi_{1:T}^s$. We can then view the alternative lookahead function ψ' adapted to the batch setting as a reformulation of the optimal batch lookahead function ψ^{b*} for each batch. We can calculate approximate alternative lookahead functions $\bar{\psi}'$ for each batch which estimates the optimal batch lookahead function. Then we can use the approximate alternative lookahead functions in combination with batch twisted models and a twisted particle filter. This is done in some of the numerical experiments of Chapter 7 both for an offline setting and a batch setting. However, we note that the computational cost of using these alternative lookahead functions is high in practice. The main reason for this is the evaluation of $\bar{\psi}'_t(x_t)$ from equation (6.5). Recall that evaluating $\bar{\psi}'_t(x_t)$ includes evaluating the kernel density-based approximation $\hat{f}_s(x_t)$. We need to evaluate $\bar{\psi}'_t(x_t)$ when simulating approximately from the twisted transition density and when estimating the integral $\bar{\tilde{\psi}}'$.

Regarding our initial motivation for this alternative lookahead, we have seen that we have more flexibility than the parametric approximation of the lookahead function because we do not have to select a class of parametric forms. On the other side, the computational cost is significantly higher. Both when evaluating $\bar{\psi}'$ because of the kernel density estimation and when estimating the integral $\tilde{\bar{\psi}}$. We also see that even if we have $\psi'_{t+1}(x_{t+1})$ available in closed form we would have to approximate the integral $\tilde{\psi}'_t(x_t)$ numerically unless $p(x_{t+1}|x_t)$ is conjugate to $\psi'_{t+1}(x_{t+1})$. The high computational cost related to the approximate alternative lookahead function makes it intractable to use in practice which is why we mainly focus on parametric approximations.

6.4 Laplace-based approximations

The numerical estimation of the approximate lookahead functions $\bar{\psi}$ is a computationally expensive part of the iAPF. It can be the costliest part of the algorithm when taking into account that it is run once for each improvement of $\bar{\psi}$. In this section, we consider another approach to estimating the approximate lookahead functions. This is based on the section about Laplace approximations in Lindsten, Helske, and Vihola (2018). We follow their notation by denoting $\tilde{p}(y_t|x_t)$ as a Gaussian approximation of the observation density obtained by the Laplace method. A somewhat restrictive assumption for this specific setup is that the latent process we are considering is a Gaussian Markov random field (GMRF). This implies that x_t is only dependent on its neighbours (Lindsten, Helske, and Vihola, 2018, p. 6). Another, somewhat less restrictive, assumption is that the observation density, $p(y_t|x_t)$, is at least twice differentiable. In order to estimate the Gaussian approximation, $\tilde{p}(y_t|x_t)$, we use the Laplace method on $\log p(y_t|x_t)$ around its mode. This corresponds to the initial setup for Gaussian approximations in e.g. Rue, Martino, and Chopin (2009). We then denote the point x_0 as the solution to $\frac{d}{dx}\log p(y_t|x) = 0$. We also denote $f(x_t) = \log p(y_t|x_t)$ for notational simplicity. We start by defining the Gaussian approximation $\log \tilde{p}(y_t|x_t)$ by the following

$$\log \tilde{p}(y_t|x_t) = \sum_{j=0}^{2} \frac{1}{j!} f^{(j)}(x_0)(x_t - x_0)^j.$$

This is a Taylor expansion around the point x_0 of order 2 where $f^{(j)}(x_0)$ denotes the jth derivative at x_0 (Givens and Hoeting, 2013, p. 2). The approximation $\bar{\psi}_t(x_t)$ of the lookahead function is then selected directly as the Gaussian approximation which is proportional to $\tilde{p}(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$ for $t < t_{\text{stop}}$. When $t = t_{\text{stop}}$ it is set equal to the Gaussian approximation.

This is possible because the approximate lookahead functions, denoted by $\bar{\psi}$, are conjugate to the transition densities, making the integral $\tilde{\bar{\psi}}$ tractable. This again follows from the assumption that the latent process is a GMRF. In addition the Gaussian approximation, $\tilde{p}(y_t|x_t)$, is conjugate to the normalising function $\tilde{\bar{\psi}}_t(x_t)$. We also use the assumption that $\tilde{\bar{\psi}}_T(x_T) \equiv 1$ which follows from Assumptions 4.2.1. We can calculate $\bar{\psi}_t(x_t)$ analytically which means there is no need for the computationally expensive numerical optimisation. In addition, we do not use particles in the calculations of $\bar{\psi}$. This implies that we do not have to run the initial BPF in order to obtain particles used in the estimation of $\bar{\psi}$. We can estimate $\bar{\psi}$ by running Algorithm 11 once with

Algorithm 11 Laplace approximation of ψ

```
1: for $t \in (t_{\text{stop}}, \dots, t_{\text{start}})$ do
2:     if $t = t_{\text{stop}}$ then
3:         $\bar{\psi}_t(x_t) = \tilde{p}(y_t|x_t)$
4:     else
5:         $\tilde{\bar{\psi}}_t(x_t) = \int \bar{\psi}_{t+1}(x_{t+1}) p(x_{t+1}|x_t) dx_{t+1}$
6:         $\bar{\psi}_t(x_t) \propto \tilde{p}(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$
7:     end if
8: end for
```

$t_{\text{stop}} = T$ and $t_{\text{start}} = 1$ in an offline setting. Then we can use the approximate twisting functions $(\bar{\psi}, \tilde{\bar{\psi}})$ in combination with a twisted model in a twisted particle filter. From the twisted particle filter we obtain a likelihood estimate. Note that the offline setting is still implied because of the backward recursive structure. One disadvantage of this approach is the fact that we are losing the iteratively improving $\bar{\psi}$ property of the iAPF. One way to keep the iteratively improving $\bar{\psi}$ is to use the Laplace approximation only for the first iteration and then numerical optimisation for the improvements. In that scenario however, we would still have relatively similar computational cost as the standard setup. This is because we are only reducing the number of numerical estimations of $\bar{\psi}$ by one. We will consider a numerical example in Section 6.5.

General setup

After we have the Gaussian approximation, denoted by $\tilde{p}(y_t|x_t) = \mathcal{N}(x_t, \tilde{\mu}_t, \tilde{\sigma}_t^2)$, we want to find the sequence $\bar{\Psi}_{1:T}$ of approximate lookahead functions. We will use $\tilde{\bar{\psi}}_T(x_T) \equiv 1$ and the recursive approximation structure from proposition 4 in Guarniero, Johansen, and Lee (2017). We then have that $\bar{\psi}_T(x_T) = \tilde{p}(y_T|x_T)$. The remaining lookahead functions are approximated by using the recursive structure of the optimal twisting functions. We consider the transition density on a generic form $p(x_t|x_{t-1}) = \mathcal{N}(x_t; \mu_x, \sigma_x^2)$. Here, the expectation parameter is on the form $\mu_x = \alpha_0 + \alpha x_{t-1}$. All the approximate lookahead functions are on the general form $\mathcal{N}(x_t; \mu_t, \sigma_t^2)$. We then start by

$$\bar{\psi}_T(x_T) = \tilde{p}(y_T|x_T) = \mathcal{N}(x_T; \mu_T, \sigma_T^2).$$

Assume now that we have the approximate lookahead function $\bar{\psi}_t(x_t)$ and want to calculate the normalising function $\tilde{\bar{\psi}}_{t-1}(x_{t-1})$. We reformulate the integrand by straightforward calculations to simplify the integral which then can be solved analytically. The normalising function then becomes

$$\begin{aligned}
\tilde{\bar{\psi}}_{t-1}(x_{t-1}) &= \int \bar{\psi}_t(x_t) p(x_t|x_{t-1}) dx_t \\
&= \int \mathcal{N}(\mu_x; \mu_t, \sigma_t^2 + \sigma_x^2) \mathcal{N}\left(x_t; \frac{\frac{\mu_t}{\sigma_t^2} + \frac{\mu_x}{\sigma_x^2}}{\frac{1}{\sigma_t^2} + \frac{1}{\sigma_x^2}}, \frac{1}{\frac{1}{\sigma_t^2} + \frac{1}{\sigma_x^2}}\right) dx_t \\
&= \mathcal{N}(\mu_x; \mu_t, \sigma_t^2 + \sigma_x^2).
\end{aligned}$$

We can now use the recursive structure from the optimal twisting functions in order to approximate the lookahead function at iteration $t-1$. We then have by the same type of straightforward calculations that

$$\tilde{p}(y_{t-1}|x_{t-1})\tilde{\bar{\psi}}_{t-1}(x_{t-1}) \propto \mathcal{N}\left(x_{t-1}; \frac{\frac{\tilde{\mu}_{t-1}}{\tilde{\sigma}^2_{t-1}} + \frac{\alpha\mu_t - \alpha\alpha_0}{\sigma_x^2+\sigma_t^2}}{\frac{1}{\tilde{\sigma}^2_{t-1}} + \frac{\alpha^2}{\sigma_x^2+\sigma_t^2}}, \frac{1}{\frac{1}{\tilde{\sigma}^2_{t-1}} + \frac{\alpha^2}{\sigma_x^2+\sigma_t^2}}\right).$$

By using the recursive structure in Definition 4.5.1, we approximate $\bar{\psi}_{t-1}(x_{t-1})$ directly by the right-hand side. We then have the approximate lookahead function at iteration $t-1$ on parametric form given by

$$\bar{\psi}_{t-1}(x_{t-1}) = \mathcal{N}\left(x_{t-1}; \frac{\frac{\tilde{\mu}_{t-1}}{\tilde{\sigma}^2_{t-1}} + \frac{\alpha\mu_t - \alpha\alpha_0}{\sigma_x^2+\sigma_t^2}}{\frac{1}{\tilde{\sigma}^2_{t-1}} + \frac{\alpha^2}{\sigma_x^2+\sigma_t^2}}, \frac{1}{\frac{1}{\tilde{\sigma}^2_{t-1}} + \frac{\alpha^2}{\sigma_x^2+\sigma_t^2}}\right).$$

We can now calculate the entire sequence $\bar{\Psi}_{1:T}$ analytically, without numerical optimisation. For this type of models, this represents significantly lower computational cost than the corresponding parametric approximation found by numerical optimisation. Having obtained a sequence of approximations, we now have the twisting functions $(\bar{\psi}, \tilde{\bar{\psi}})$. The twisting functions can be utilised in a twisted model in the same way any valid sequence of twisting functions can be used in a twisted model. This includes parametric and nonparametric approximation of the lookahead functions and the alternative lookahead function.

Setup for Poisson observations

We will consider the general class of models given by the structure of model 4 in Chapter 7 and see how Laplace approximation of $\bar{\psi}$ can be utilised. The setup in Lindsten, Helske, and Vihola (2018) also use the Gaussian approximations $\tilde{p}(y_t|x_t)$ to estimate the importance sampling distributions and weights. We instead follow the setup from Guarniero, Johansen, and Lee (2017) by using the twisted transition density as the importance sampling distribution. That is, we will be using the Gaussian approximation $\tilde{p}(y_t|x_t)$ only for the approximate twisting functions $(\bar{\psi}, \tilde{\bar{\psi}})$. The general class of models having the same structure as model 4 is defined by

$$p(x_1) = \mathcal{N}\left(x_1; \alpha_0, \frac{\sigma_x^2}{(1-\alpha)^2}\right) \qquad p(y_t|x_t) = \frac{(e^{x_t})^{y_t}}{y_t!}e^{-e^{x_t}}$$
$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; \alpha_0 + \alpha x_{t-1}, \sigma_x^2).$$

We see that the observation density is more than twice differentiable and we start by finding $\tilde{p}(y_t|x_t)$. We find the mode $x_0 = \log y_t$ and the Taylor expansion around x_0. We then have that

$$\begin{aligned}\log \tilde{p}(y_t|x_t) &= \frac{1}{0!}f^{(0)}(x_0) + \frac{1}{1!}f^{(1)}(x_0)(x_t - x_0)^1 + \frac{1}{2!}f^{(2)}(x_0)(x_t - x_0)^2 \\ &= \log p(y_t|x_0) - \frac{e^{x_0}}{2}(x_t - x_0)^2 \\ \tilde{p}(y_t|x_t) &= p(y_t|x_0)e^{-\frac{e^{x_0}}{2}(x_t - x_0)^2}.\end{aligned}$$

We are considering the Gaussian approximation, $\tilde{p}(y_t|x_t)$, to be a function of x_t. Here and y_t is a fixed observation and therefore the term $p(y_t|x_0)$ is constant. We recognise the Gaussian density up to proportionality on the right-hand side,

$$\tilde{p}(y_t|x_t) \propto \exp -\frac{1}{2\frac{1}{e^{x_0}}}(x_t - x_0)^2.$$

The parametric form for the approximation can therefore be expressed as $\tilde{p}(y_t|x_t) = \mathcal{N}(x_t; \tilde{\mu}_t, \tilde{\sigma}_t^2)$ where $\tilde{\mu}_t = x_0$ and $\tilde{\sigma}_t^2 = \frac{1}{e^{x_0}}$. With this setup we can use Algorithm 11 in order to calculate approximate lookahead functions which we then can use to define approximate twisting functions.

Due to the GMRF assumptions about the transition density and the Gaussian approximation of the observation density, the transition density is conjugate to the approximate lookahead function $\bar{\psi}$. In addition the integral $\tilde{\bar{\psi}}$ is tractable and is available in closed form. This reduces the computational cost as no numerical optimisation is needed to calculate the approximate lookahead functions $\bar{\psi}$. We see that if the GMRF assumption holds, we can reduce computational cost by using the Laplace approximations. However, in order to have iteratively improving estimates of the lookahead functions we still need numerical optimisation.

6.5 Example: Laplace approximations

We consider a specific example in order to compare the Laplace-based approach to approximating the lookahead functions with the traditional setup of iAPF with Gaussian lookahead functions and numerical optimisation. We include a BPF with more particles as a reference. In this example, we will use model 4 from Chapter 7 given by

$$p(x_1) = \mathcal{N}\left(x_1; 0.9, \frac{0.2^2}{(1-0.7)^2}\right) \qquad p(y_t|x_t) = \frac{(e^{x_t})^{y_t}}{y_t!}e^{-e^{x_t}}$$
$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; 0.9 + 0.7x_{t-1}, 0.2^2).$$

The iAPF uses a single Gaussian as the approximate lookahead function, that is $\bar{\psi}_t(x_t) = \mathcal{N}(x_t; \mu_t, \sigma_t^2)$. With the Laplace-based approach, we first calculate the approximate lookahead functions $\bar{\psi}$ by using Algorithm 11, then we use the Laplace-based twisting functions in combination with the twisted model. We use the setup of the iAPF with a twisted model, but we do not include any improvement steps or numerical optimisation. We denote the Laplace-based iAPF with iAPFL and we used the following setup for the example

Algorithm	Particles	Parameters
iAPF	$n_0 = 200$	$k = 3$, $\tau = 1$, $\alpha = \beta = 0.5$
iAPFL	$n = 200$	$\alpha = \beta = 0.5$
BPF	$n = 1000$	$\alpha = 0.5$

We compare all the likelihood estimates with a likelihood estimate denoted by $\hat{Z}$ from a BPF with $n = 500000$ particles. We then calculate the likelihood ratios $\frac{\hat{Z}_{\text{iAPF}}}{\hat{Z}}$, $\frac{\hat{Z}_{\text{iAPFL}}}{\hat{Z}}$ and $\frac{\hat{Z}_{\text{BPF}}}{\hat{Z}}$. Running the algorithms BPF, iAPF and the iAPF with the Laplace-based approximate lookahead functions are repeated 100 times. The likelihood estimates from each of the runs for all three algorithms are then stored. We also want to compare the effective sample size of the three algorithms. Because the algorithms use a different number of particles we consider a fraction with the ESS at the given iteration in the numerator and the number of particles in the denominator. We denote this by $\mathrm{EF}_t = \frac{\mathrm{ESS}_t}{n_t}$. For the iAPF, the fraction EF_t is calculated for the last run of the twisted particle filter. We also used the distance function D_{iAPF2} from equation (4.17) in the iAPF.

Data

We start by simulating latent variables and observations from the model for $T = 50$ iterations.

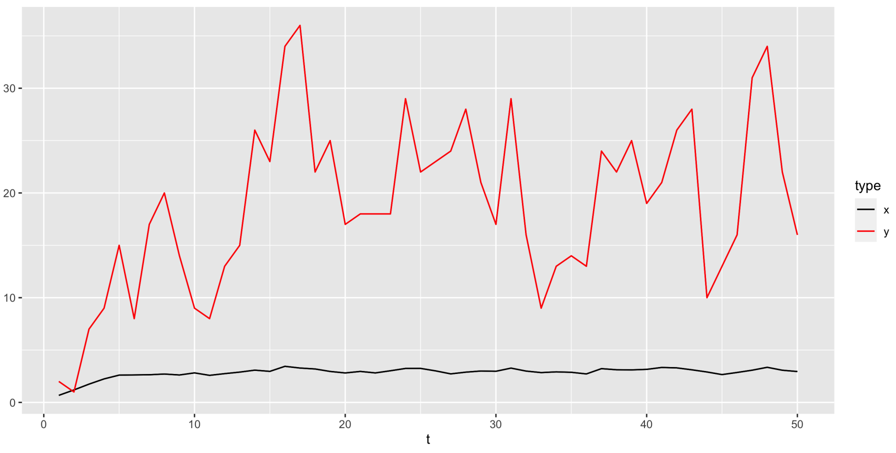


Figure 6.1: Simulated latent variables $x_{1:T}$ and observations $y_{1:T}$ for $T = 50$.

Results

The iAPF with numerical optimisation had a considerably higher computational cost than the BPF and the iAPF with the Laplace-based approximate lookaehad functions. The iAPF with numerical optimisation yielded likelihood estimates with low variance by using only $n_0 = 200$ particles. The iAPF with Laplace-based approximations had considerably lower computational cost than the iAPF with numerical optimisation. We see from Figure 6.2 that the iAPF with Laplace-based approximations achieved lower variance for the likelihood estimates compared to the BPF with five times as many particles. The likelihood ratios are plotted in Figure 6.2 with the red circle indicating the mean of the estimates. We also consider the effective sample size, measured by the mean EF_t from 100 repetitions. We see from Figure 6.3 that the mean EF_t is consistently higher for the iAPF. Recall that the ESS is measured in the last run of the twisted particle filter in this case. When taking this into account we see that the iAPFL has a relatively high effective sample size given that it has no iteratively improving of the approximate lookahead functions $\bar{\psi}$. We also consider the numerical variance of the calculated likelihood ratios and the estimated squared bias.

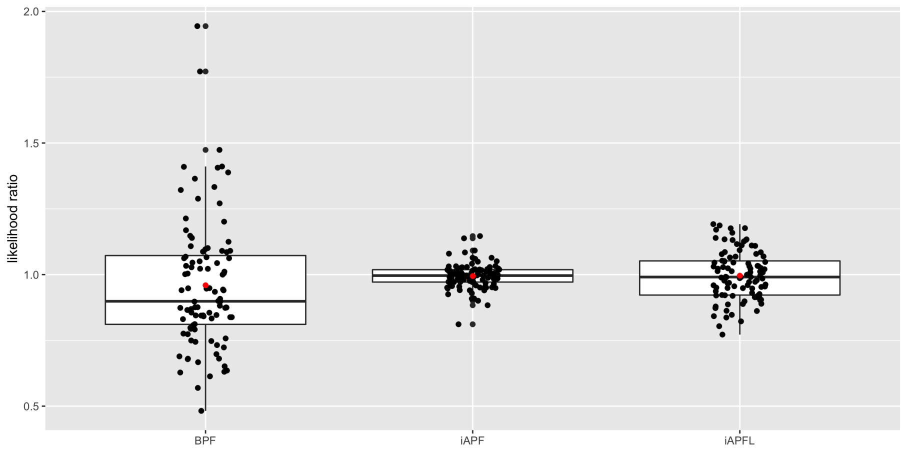


Figure 6.2: Box plots of estimated likelihood from 100 runs relative to estimated likelihood from a BPF using 500000 particles. The red circles represent the sample means.

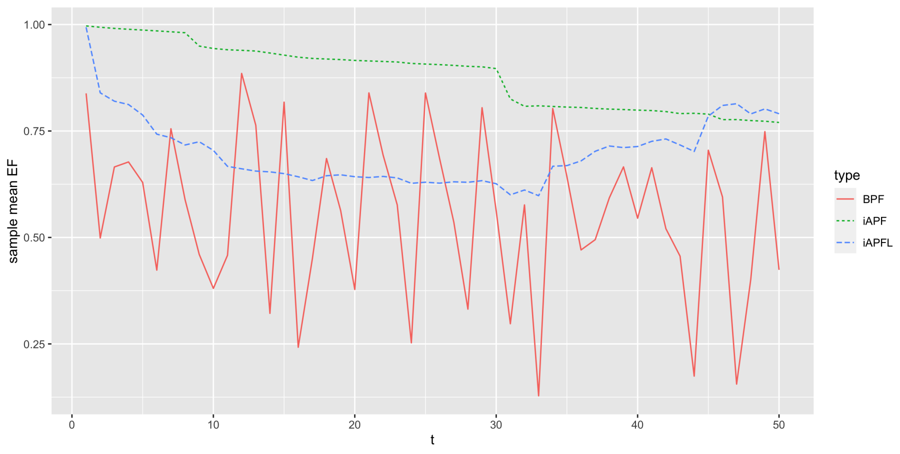


Figure 6.3: Sample mean of EF_t over 100 runs of the three algorithms.

	BPF	iAPF	iAPFL
$V\left[\hat{Z}/Z\right]$	0.0597	0.0023	0.0087
$(E\left[\hat{Z}/Z\right]-1)^2$	0.0016	3.2730e-05	1.7999e-05

CHAPTER 7

Numerical experiments

In this section we consider numerical experiments where we will focus on some of the concepts we have considered. We start by defining a set of models which we mainly use in the numerical experiments of this chapter. For each of the experiments we will provide an overview of the setup for the different algorithms.

Model 1

For the first model, we define a linear Gaussian SSM in 1D similar to the model of section 7 in Kantas et al. (2015). Because of the linear Gaussian structure, we can use a Kalman filter in order to calculate the likelihood which will be useful for comparisons with the estimated likelihood. The model structure is given by

$$\begin{aligned} x_t &= \rho_0 + \rho x_{t-1} + \sigma w_t & w_t &\sim \mathcal{N}(0, 1^2) \\ y_t &= x_t + \tau v_t & v_t &\sim \mathcal{N}(0, 1^2). \end{aligned}$$

With $\rho_0 = 0.2$, $\rho = 0.75$, $\sigma = 1$, $\tau = 1$ and $x_0 = 0$.

Model 2

For the second model, we will consider the univariate stochastic volatility model (USV) described in Guarniero, Johansen, and Lee (2017).

$$\begin{aligned} p(x_1) &= \mathcal{N}\left(x_1; \alpha_0, \frac{\sigma^2}{(1-\alpha)^2}\right) \\ p(x_t|x_{t-1}) &= \mathcal{N}(x_t; \alpha_0 + \alpha x_{t-1}, \sigma^2) \\ p(y_t|x_t) &= \mathcal{N}(y_t; 0, \beta^2 e^{x_t}) \end{aligned}$$

We define model 2 by selecting the parameters $\sigma = 0.2$, $\alpha_0 = 0$, $\alpha = 0.8$ and $\beta = 0.2$.

Model 3

For the third model, we use the multivariate linear Gaussian with dimensionality d from section 5.2 in Guarniero, Johansen, and Lee (2017). This is defined by

$$p(x_1) = \mathcal{N}(x_1; m, \Sigma)$$

$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; Ax_{t-1}, B)$$
$$p(y_t|x_t) = \mathcal{N}(y_t; Cx_t, D).$$

We use the same setup as in Guarniero, Johansen, and Lee (2017, p. 1641) where I_d denotes the $d \times d$ identity matrix. Further, the parameters have the values

- $B = C = D = \Sigma = I_d$
- $A_{ij} = \alpha^{|i-j|+1} \; i, j \in 1 \ldots, d$
- $m = \mathbf{0}$

We consider the case where y and x have the same dimensionality d. Here, $\mathbf{0}$ denotes a vector with dimension d where the components all are equal to 0.

Model 4

We here consider a model that includes discrete observations. This is based on the USV. However, each of the observations has a Poisson distribution instead of a nonlinear Gaussian. The model is then given by

$$p(x_1) = \mathcal{N}\left(x_1; \alpha_0, \frac{\sigma^2}{(1-\alpha)^2}\right)$$
$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; \alpha_0 + \alpha x_{t-1}, \sigma^2)$$
$$p(y_t|x_t) = \frac{(e^{x_t})^{y_t}}{y_t!} e^{-e^{x_t}}.$$

We denote the following as model 4 when $\alpha_0 = 0$, $\alpha = 0.75$ and $\sigma = 0.5$.

Mean squared error

We estimate the MSE from the estimate $\hat{Z}$ of the true Z as

$$\text{MSE}\left[\hat{Z}\right] = \text{V}\left[\hat{Z}\right] + \left(\text{E}\left[\hat{Z}\right] - Z\right)^2$$

adapted from Hastie, Tibshirani, and J. Friedman (2009, p. 24). Here, Z denotes the likelihood which generally is unknown. When we are considering linear Gaussian models, we use a Kalman filter in order to calculate the value of Z analytically. When considering other models than the linear Gaussian, we use the likelihood estimate from a BPF with a large number of particles as an accurate estimate for reference. In that case we also denote the likelihood estimate by Z. We use $\hat{Z}$ to denote the estimated likelihood. We will mainly be interested in the MSE of the likelihood ratio denoted by $\frac{\hat{Z}}{Z}$. This implies that we are often considering the MSE of the fraction

$$\text{MSE}\left[\frac{\hat{Z}}{Z}\right] = \text{V}\left[\frac{\hat{Z}}{Z}\right] + \left(\text{E}\left[\frac{\hat{Z}}{Z}\right] - 1\right)^2$$

We use the package from Luethi et al. (2021) to obtain the likelihood using a Kalman filter.

Notation

We will mainly be interested in some specific model configurations in these numerical experiments. We use the following abbreviations for the different setups to simplify notation in the discussion

BPF: The bootstrap particle filter as presented in Algorithm 5.

BP: The batch algorithm with parametric approximations of twisting functions (BP) presented in Algorithm 9.

BA: The batch algorithm with the alternative lookahead functions (BA) presented in Algorithm 9. We here refer to the reformulation of the optimal lookahead function. This is denoted by ψ' and the approximation discussed in Section 6.3 is denoted by $\bar{\psi}'$.

iAPF: The iAPF algorithm from Algorithm 8.

Note that the BP and BA are batch algorithms, but when used in offline settings this is equivalent to using only one batch.

7.1 Experiment 1: offline setting

We first consider an experiment in the offline setting where we use model 1. Because of the linear Gaussian structure of the model, we calculate the likelihood exactly using a Kalman filter. Recall that the when using the batch algorithms BA and BP in an offline setting, we are using only one batch. For the algorithm BA, we are using the alternative lookahead function ψ' which is approximated by $\bar{\psi}'$ discussed in Section 6.3. We compare the BPF, BP, BA and the iAPF for estimating the likelihood, denoted by $\hat{Z}$. We denote the likelihood calculated using a Kalman filter by Z and when we have estimates of the likelihood from all the algorithms, we consider the numerical variance of the likelihood ratio $\frac{\hat{Z}}{Z}$. In general we use the adaptive resampling threshold α in BPF and iAPF. We also use resampling threshold α for the initial particle filter used in BP and resampling threshold β in the twisted particle filter. For BA we use α for the internal particle filter which obtains particles for estimating the smoothing weights and approximating the predictive distributions. We then use resampling threshold β for the twisted particle filter in BA. The main focus in this numerical experiment is the effect of using twisting target distributions when considering numerical variance of the likelihood estimates.

Setup

We now present the configuration of the four algorithms in this experiment.

Algorithm	Particles	Parameters
BA	$n = 200$, $m = 400$, $s = 200$	$L = T$, $\alpha = \beta = 0.5$
BP	$m = n = 100$	repeats $= 3$, $L = T$, $\alpha = \beta = 0.5$
BPF	$n = 400$	$\alpha = 0.5$
iAPF	$n_0 = 100$	$k = 5$, $\tau = 1$, $\alpha = 0.5$

For BP we use the parametric form $\bar{\psi}_t^b(x_t) = \mathcal{N}(x_t, \mu_t, \sigma_t^2)$ when approximating the lookahead functions and for iAPF we use $\bar{\psi}_t(x_t) = \mathcal{N}(x_t, \mu_t, \sigma_t^2)$. We use the distance function D_{iAPF2} from equation (4.17) in this experiment. For the configuration of BA we present a more detailed overview.

- Batch length: the last iteration in the first batch in this experiment is set to $L = T$ which implies that the algorithm use one batch.
- Internal particle filter: a bootstrap particle filter obtains is used to obtain particles which then is used to estimate smoothing weights and approximate the predictive distribution.
- Smoothing: the algorithm uses a fixed-interval smoothing algorithm to obtain the estimated smoothing weights.
- Twisting functions: we use the approximation $\bar{\psi}_t'(x_t)$ discussed in Equation (6.5). The symmetric kernel, K, in this experiment is set to a $\mathcal{N}(0, 1^2)$ and the bandwidth h was set to the lower bound $h = m^{-0.25}$ defined by Crisan and Miguez (2014).
- Importance sampling distribution: here we use Algorithm 10.

Data

We simulate latent variables $x_{1:T}$ and observations $y_{1:T}$ with $T = 30$ from model 1. These are shown in Figure 7.1.

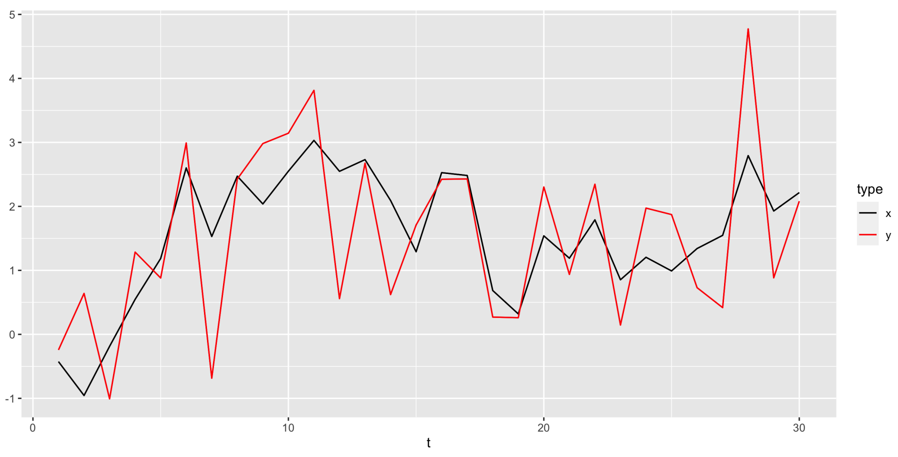


Figure 7.1: Simulated latent variables $x_{1:T}$ and observations $y_{1:T}$ from model 1, $T = 30$.

Results

We estimate the likelihood with the four algorithms 100 times and calculate the empirical variance of the likelihood ratios $\hat{Z}/Z$ where $\hat{Z}$ denotes the estimated likelihood from an algorithm. The exact likelihood Z is obtained using a Kalman filter. The likelihood ratios are shown in the box plots in Figure 7.2 and the mean of the estimates are shown in red.

The likelihood estimates from the BPF have the largest variance as expected. When considering the BP and iAPF, these have approximately the same variance of the likelihood estimates because the batch setting with one batch corresponds to the offline setting. Note that this is an artificially low variance because $p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$ is proportional to a Gaussian density. With this in mind, we see that we are using one Gaussian density, $\bar{\psi}_t(x_t)$, to approximate another Gaussian density, $p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$. Therefore, when we are in the offline setting combined with linear Gaussian transition and observation densities we expect the iAPF and BP with Gaussian approximate lookahead functions $\bar{\psi}$ to perform well. We also note that the number of particles are not directly comparable across algorithms. Both iAPF and BP are utilising iteratively improving $\bar{\psi}$ estimates which is also difficult to factor in when comparing the algorithms. We also see that the BA has lower variance of the likelihood estimates than the BPF, but at a significantly higher computational cost compared to all the other algorithms. We also consider the estimated MSE by calculating the sample

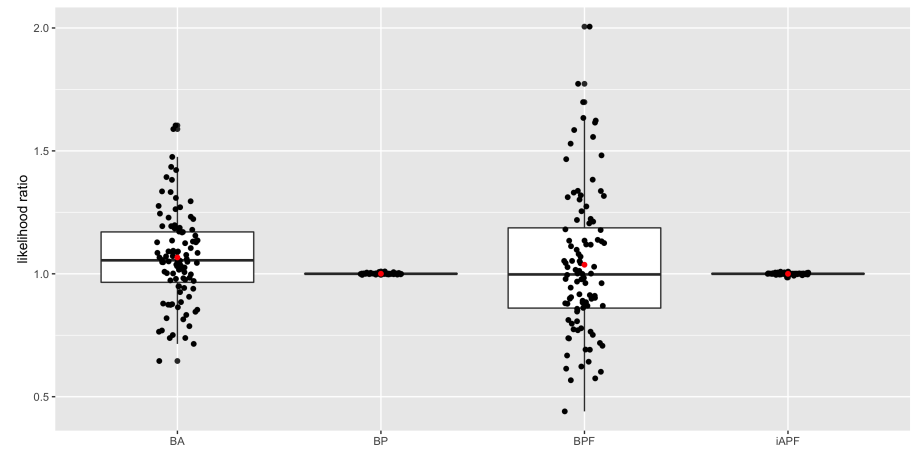


Figure 7.2: Box plots of likelihood ratios from 100 runs relative to the likelihood calculated by a Kalman filter, the red circles represent the sample means. The algorithm BPF is in an online setting, the remaining algorithms are in offline settings.

variance and an estimate of the squared bias for the likelihood ratios.

	BA	BP	BPF	iAPF
$V\left[\hat{Z}/Z\right]$	0.0344	3.9239e-06	0.0855	7.4168e-06
$(E\left[\hat{Z}/Z\right]-1)^2$	0.0044	1.1234e-07	0.0014	5.8726e-09

7.2 Experiment 2: batch setting

We use the same base configuration as in experiment 1, but with some modifications. We are interested in the variance of the likelihood estimates when moving from an offline setting to a batch setting. Because we are now in a batch setting, we cannot use the iAPF and will therefore mainly focus on BA and BP which are batch algorithms. We include the likelihood estimates obtained with a BPF using a higher number of particles as a reference. Moving from an offline setting implies that we can start the calculations before receiving all observations $y_{1:T}$. This can again be beneficial in scenarios where different estimates are needed before iteration T. This is in contrast to the iAPF which is an offline algorithm, requiring all observations $y_{1:T}$ before starting the calculations.

Setup

In this setting we use batch size of $L = 5$ which implies that we have 6 batches in the batch algorithms. The BA use the alternative lookahead functions approximated by $\bar{\psi}'$. We now present the configuration for the algorithms used in this experiment.

Algorithm	Particles	Parameters
BA	$n = 200$, $m = 200$, $s = 400$	$L = 5$, $\alpha = \beta = 0.5$
BP	$m = n = 200$	repeats $= 3$, $L = 5$, $\alpha = \beta = 0.5$
BPF	$n = 400$	$\alpha = 0.5$

The BP is using the parametric approximation $\bar{\psi}_t^b(x_t) = \mathcal{N}(x_t, \mu_t, \sigma_t^2)$ and iteratively improving approximations of $\bar{\psi}_t^b(x_t)$. We also used the distance function D_{iAPF2} from equation (4.17) for the numerical optimisation in BP.

Results

We run the three algorithms 100 times. The likelihood ratios with the likelihood Z calculated with a Kalman filter in the denominator are compared in the box plots. The box plots of the likelihood ratios are shown in Figure 7.3 where the red circles indicate the sample mean.

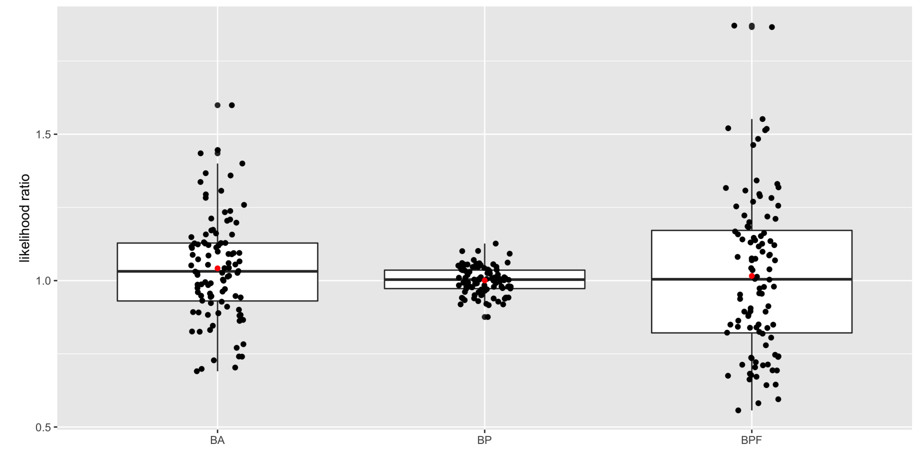


Figure 7.3: Box plots of likelihood ratios from 100 runs relative to the likelihood calculated by a Kalman filter, the red circles represent the sample means. The algorithms BA and BP are in a batch setting while the BPF is in an online setting.

We then consider the sample variance of the likelihood ratios.

	BA	BP	BPF
$V\left[\hat{Z}/Z\right]$	0.0304	0.0022	0.0736
$(E\left[\hat{Z}/Z\right]-1)^2$	0.0018	2.2253e-07	0.0003

Recall that we in experiment 1 use $L = T$, that is an offline setting. In this experiment however we are using $L = 5$, indicating that each batch contains 5 iterations. Consequently, both BA and BP are operating in a batch setting in this experiment. In Figure 7.3 we see that the variance of the likelihood estimates obtained by the BPF is highest while the BP has considerably lower variance of the obtained likelihood estimates. We here included BA mostly for reference even though it also possible to use in this experiment. In terms of computational cost the BA exceeds both BP and BPF by far, so in a practical application these are not comparable. Note that we now see the effect of being in a batch setting when considering BP. This algorithm is still utilising a Gaussian for $\bar{\psi}_t^b(x_t)$ while the target $p(y_t|x_t)\tilde{\tilde{\psi}}_t(x_t)$ is also a Gaussian for model 1. This implies that the variability in Figure 7.3 for BP can be seen as a consequence of being in a batch setting compared to an offline setting.

7.3 Experiment 3: univariate stochastic volatility

In this experiment we want to consider the USV model which corresponds to model 2. This model has a nonlinear Gaussian observation density, so we cannot use the Kalman filter to calculate the likelihood. We therefore use a BPF with $n = 500000$ particles as the reference estimate which we here denote by Z. This will then serve as an accurate estimate of the likelihood which we can compare the algorithms to. The main focus is comparing the three algorithms to each other which in we still do with this reference estimate.

We compare the batch algorithm with parametric approximations, denoted by BP, the offline iAPF and a standard BPF in the online setting. Our motivation for using the BP or the iAPF compared to the BPF is the interest in low variance likelihood estimates. We therefore compare the different likelihood estimates with the reference estimate Z. The likelihood ratios are then given by $\frac{\hat{Z}_{\text{BP}}}{Z}$, $\frac{\hat{Z}_{\text{BPF}}}{Z}$ and $\frac{\hat{Z}_{\text{iAPF}}}{Z}$ for BP, BPF and iAPF respectively. We then calculate the sample variance of these likelihood ratios.

Setup

We summarise the setup of the algorithms in this experiment.

Algorithm	Particles	Parameters
BP	$m = n = 200$	repeats = 5, $L = 10$, $\alpha = \beta = 0.5$
BPF	$n = 1000$	$\alpha = 0.5$
iAPF	$n_0 = 200$	$k = 5$, $\tau = 0.5$, $\alpha = 0.5$

For BP we use the approximation $\bar{\psi}_t^b(x_t) = \mathcal{N}(x_t; \mu_t, \sigma_t^2)$ and for iAPF we use $\bar{\psi}_t(x_t) = \mathcal{N}(x_t; \mu_t, \sigma_t^2)$. We also used the distance function D_{iAPF2} from equation (4.17) for the numerical optimisation in BP and iAPF.

Data

We simulated a sequence of latent variables $x_{1:T}$ and observations $y_{1:T}$ of length $T = 50$ from model 2. These are shown in Figure 7.4.

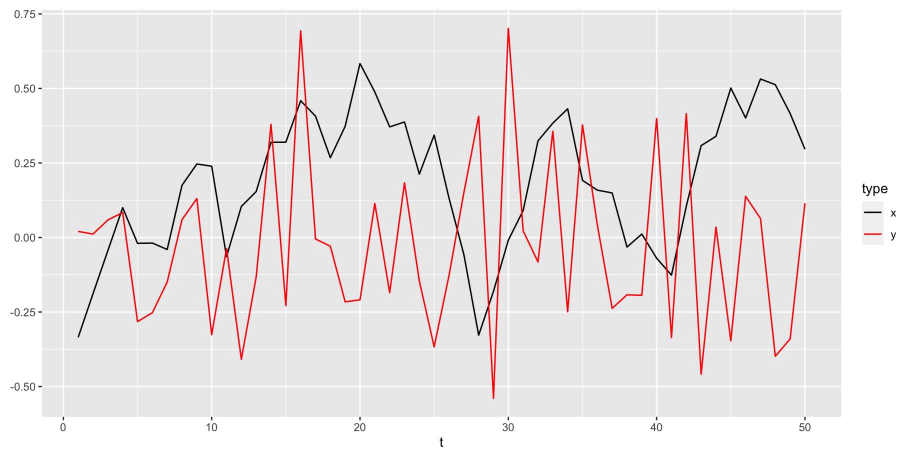


Figure 7.4: Simulated latent variables $x_{1:T}$ and observations $y_{1:T}$ from model 2, $T = 50$.

Results

We run the three algorithms and estimate the likelihood by using the algorithms 500 times. We are now comparing three algorithms in three different setting. The iAPF is in an offline setting. The batch algorithm with parametric approximations of the lookahead functions BP is in a batch setting with $L = 10$ implying that the timeline is split into 5 batches. The BPF is in an online setting, but also utilising five times as many particles as the other two algorithms. From Figure 7.5 we see the likelihood ratios and the sample mean of these denoted by a red circle. We also see that BP and iAPF have considerably lower variance of the likelihood estimates compared to the BPF utilising five times as many particles in the estimations. We consider the variance and the estimated squared bias of the likelihood ratios for the three algorithms.

	BP	BPF	iAPF
$V\left[\hat{Z}/Z\right]$	0.0037	0.0165	0.0004
$(E\left[\hat{Z}/Z\right] - 1)^2$	8.5919e-05	5.6265e-05	4.4549e-05

We see that the variance of the likelihood estimates obtained by the iAPF in an offline setting is still around 1/10 of the variance of the likelihood estimates obtain by the BP. This highlights the additional decrease in variance for the likelihood estimates obtained in an offline setting compared to the batch setting. For the squared bias estimate, the three algorithms are quite similar and all are close to 0.

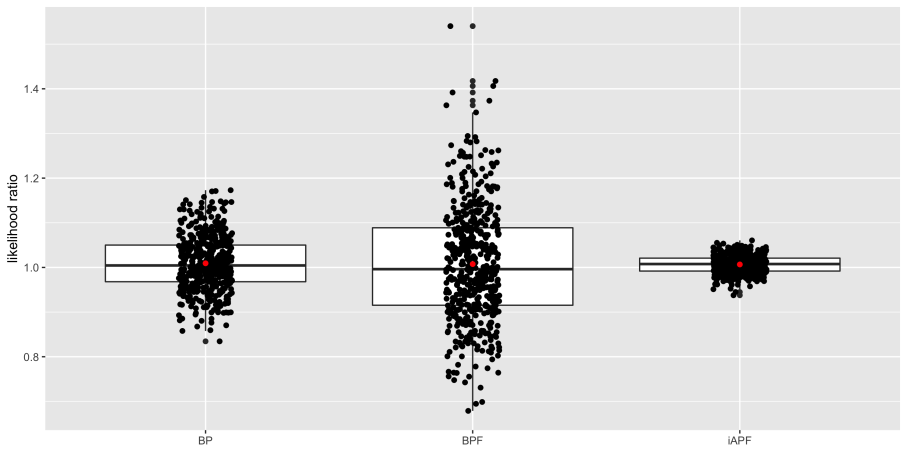


Figure 7.5: Box plots of likelihood ratios from 500 runs relative to the likelihood calculated by a BPF using 500 000 particles, the red circles represent the sample means. The algorithm BP is in a batch setting, BPF is in an online setting and iAPF is in an offline setting.

7.4 Experiment 4: multivariate setting

In this experiment we consider model 3. This model is a multivariate linear Gaussian model. We consider the case where $\alpha = 0.42$ which is the same value as in Guarniero, Johansen, and Lee (2017). In this experiment we consider $d = 4$, that is both the latent variables and the observations are vectors. Here we compare a BPF, iAPF and a BP. Our main interest is how the BP is in higher dimensions compared to the iAPF. We also run a BPF with more particles for comparison.

Again we expect the iAPF to have the lowest variance of the likelihood estimates. In this experiment we are able to calculate the likelihood using a Kalman filter as we are considering a linear Gaussian model. We denote the likelihood from the Kalman filter by Z and again consider the likelihood ratios $\frac{\hat{Z}_{\text{BPF}}}{Z}$, $\frac{\hat{Z}_{\text{iAPF}}}{Z}$ and $\frac{\hat{Z}_{\text{BP}}}{Z}$.

Setup

Algorithm	Particles	Parameters
BP	$m = n = 500$	repeats $= 5$, $L = 6$, $\alpha = \beta = 0.5$
BPF	$n = 5000$	$\alpha = 0.5$
iAPF	$n_0 = 500$	$k = 4$, $\tau = 0.5$, $\alpha = 0.5$

The BP use $\bar{\psi}^b_t(x_t) = \mathcal{N}(x_t, \mu_t, \Sigma_t)$ as approximations of the lookahead functions while iAPF use $\bar{\psi}_t(x_t) = \mathcal{N}(x_t, \mu_t, \Sigma_t)$. Note that Σ_t for $t = 1, \ldots, T$ are diagonal matrices. We also used the distance function D_{iAPF2} from equation (4.17) for BP and iAPF.

We simulate 30 latent variables and observations from model 3 defined in Chapter 7.

Results

Then we ran the three algorithms 200 times to obtain 200 likelihood estimates from each algorithm. We also want to consider the variance of the likelihood

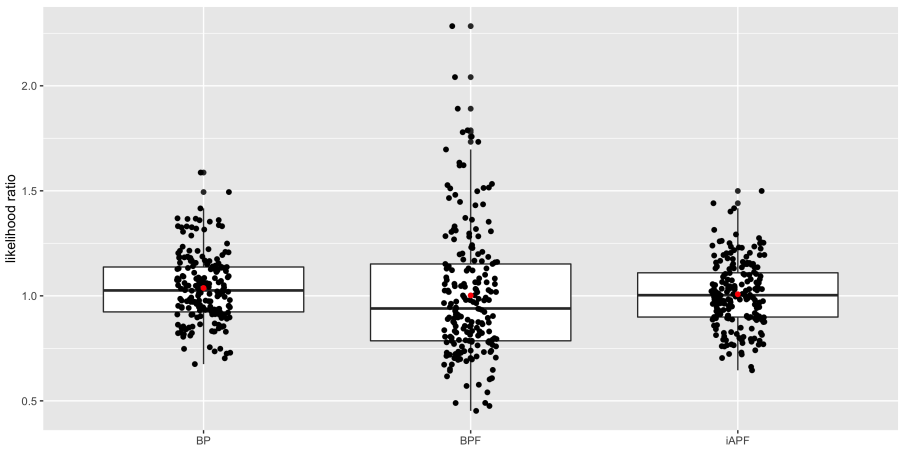


Figure 7.6: Box plots of likelihood ratios from 200 runs relative to the likelihood calculated by a Kalman filter, the red circles represent the sample means. The algorithm BP is in a batch setting, BPF is in an online setting and iAPF is in an offline setting.

ratios in this experiment.

	BP	BPF	iAPF
$V\left[\hat{Z}/Z\right]$	0.0259	0.0936	0.0245
$(E\left[\hat{Z}/Z\right]-1)^2$	0.0014	4.1386e-06	5.6387e-05

In this experiment with the given setup, we see from Figure 7.6 that the BP with batch size $L = 6$ slightly higher variance than the iAPF. Both the BP and the iAPF however have considerably lower variance of the likelihood estimates compared to the BPF which is using 10 times as many particles. We also note that with $L = 6$, the BP is using less future information than the iAPF.

7.5 Experiment 5: discrete observations

We now consider an experiment by using model 4 from Chapter 7 which includes discrete observations. We are therefore not able to calculate the likelihood by using a Kalman filter. In this case, we use a BPF with $n = 500000$ particles as a reference and denote the likelihood estimate by Z. The likelihood estimates obtained from the algorithms are then compared to this estimate. Our focus is again to compare the batch algorithm with parametric approximation of the lookahead function, denoted by BP which is in a batch setting, the iAPF in an offline setting and BPF as a reference in an online setting.

Setup

Algorithm	Particles	Parameters
BP	$m = n = 1000$	repeats $= 6$, $L = 10$, $\alpha = \beta = 0.5$
BPF	$n = 5000$	$\alpha = 0.5$
iAPF	$n_0 = 1000$	$k = 6$, $\tau = 0.5$, $\alpha = 0.5$

We use the approximations $\bar{\psi}_t^b(x_t) = \mathcal{N}(x_t; \mu_t, \sigma_t^2)$ for the lookahead functions in BP and $\bar{\psi}_t(x_t) = \mathcal{N}(x_t; \mu_t, \sigma_t^2)$ in iAPF. We also used the distance function D_{iAPF2} from equation (4.17) for BP and iAPF.

Data

We simulate 50 latent variables and observations from model 4. We then ran the three algorithms 100 times to obtain 100 likelihood estimates from each algorithm.

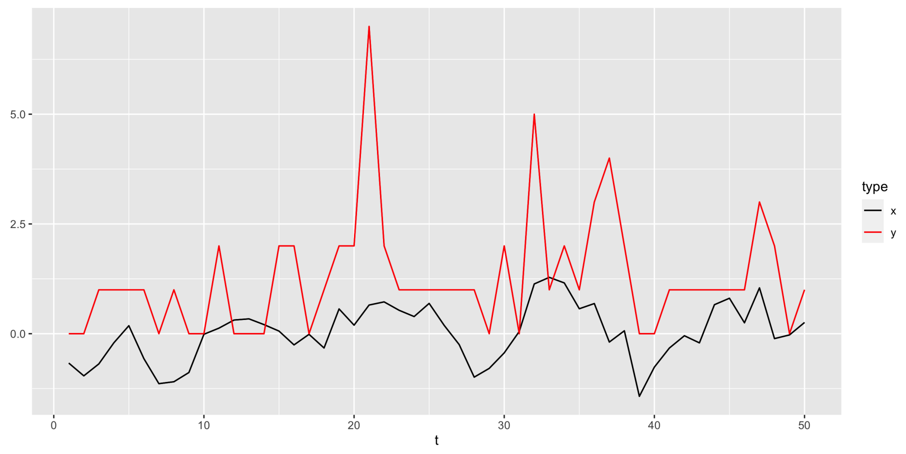


Figure 7.7: Simulated latent variables $x_{1:T}$ and observations $y_{1:T}$ from model 4, $T = 50$.

Results

We compared the likelihood ratios $\frac{\hat{Z}_{\text{BP}}}{Z}$, $\frac{\hat{Z}_{\text{BPF}}}{Z}$ and $\frac{\hat{Z}_{\text{iAPF}}}{Z}$. We still see from Figure 7.8 that the BP in a batch setting and the iAPF in an offline setting have lower variance of the likelihood estimates than the BPF in an online setting. However in this experiment with discrete observations $y_{1:T}$, both BP and iAPF have clearly higher variance than in the linear Gaussian model from experiment 1. This is also to be expected because we are still using an approximation for the lookahead function on the form $\bar{\psi}_t(x_t) = \mathcal{N}(x_t; \mu_t, \sigma_t^2)$, but now the target function, $p(y_t|x_t)\tilde{\tilde{\psi}}_t(x_t)$, is not Gaussian because of the discrete observations.

We then consider the variance and estimated squared bias of the estimated likelihood ratios from the three algorithms in this numerical experiment.

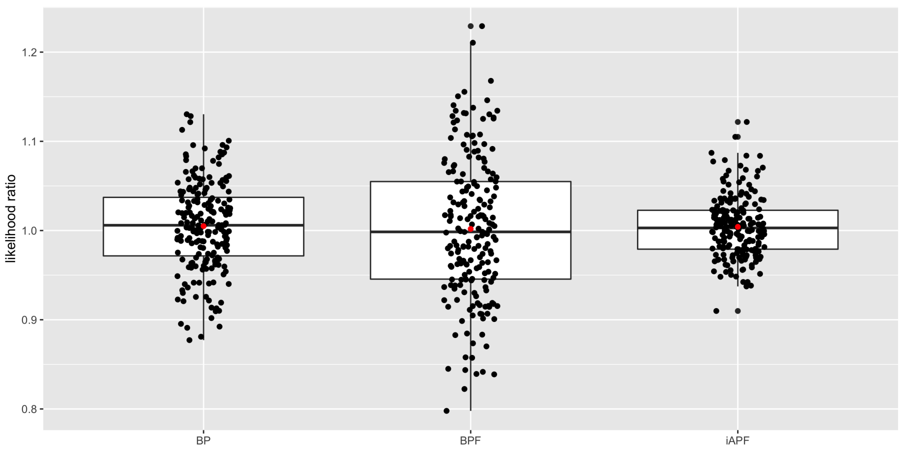


Figure 7.8: Box plots of likelihood ratios from 200 runs relative to the likelihood calculated by a BPF using 500 000 particles, the red circles represent the sample means. The algorithm BP is in a batch setting, BPF is in an online setting and iAPF is in an offline setting.

	BP	BPF	iAPF
$V\left[\hat{Z}/Z\right]$	0.0025	0.0063	0.0012
$(E\left[\hat{Z}/Z\right]-1)^2$	2.4530e-05	3.2908e-06	1.7103e-05

We see that the estimated variance of the likelihood estimates is lowest for the iAPF as expected. For the BP it is about two times higher. The estimated variance for BPF is about 2.5 times higher than the estimated variance of the likelihood estimates obtained from BP. The estimated squared bias is similar for all three algorithms. The BP is however using $L = 10$, that is, a batch size of 10 while iAPF is in an offline setting. The BPF is included mostly as a reference and we note here that it is using five times as many particles as the other algorithms.

7.6 Experiment 6: multivariate setting with higher dimensionality

In this experiment we consider model 3 from Chapter 7 with dimensionality $d = 8$ and the same $\alpha = 0.42$ as in Guarniero, Johansen, and Lee (2017).

Setup

Algorithm	Particles	Parameters
BP	$m = n = 1000$	repeats = 8, $L = 10$, $\alpha = \beta = 0.5$
BPF	$n = 5000$	$\alpha = 0.5$
iAPF	$n_0 = 1000$	$k = 7$, $\tau = 1$, $\alpha = 0.5$

The BP use $\bar{\psi}^b_t(x_t) = \mathcal{N}(x_t, \mu_t, \Sigma_t)$ as approximations of the lookahead functions while iAPF use $\bar{\psi}_t(x_t) = \mathcal{N}(x_t, \mu_t, \Sigma_t)$. Note that Σ_t for $t = 1, \ldots, T$ are diagonal matrices. We also used the distance function D_{iAPF2} from equation (4.17) for BP and iAPF. We simulate 30 latent variables and observations from model 3 defined in Chapter 7.

Results

We then consider the likelihood ratios for the three algorithms in Figure 7.9.

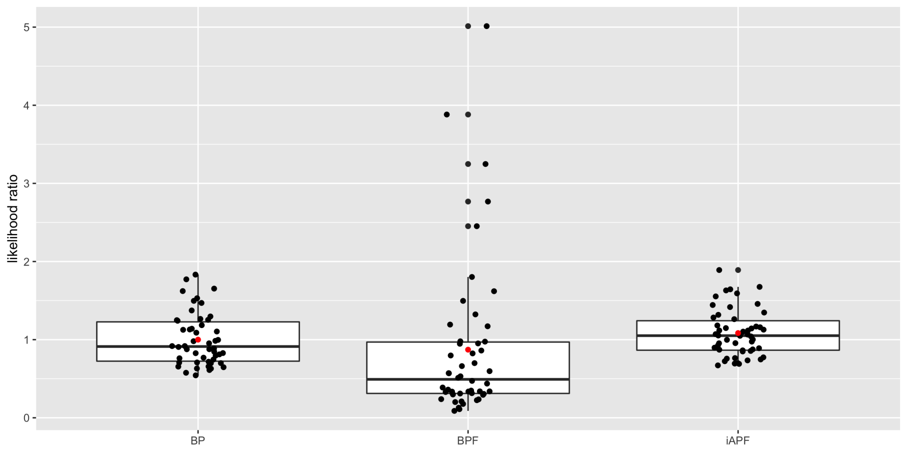


Figure 7.9: Box plots of likelihood ratios from 50 runs relative to the likelihood calculated by a Kalman filter, the red circles represent the sample means. The algorithm BP is in a batch setting, BPF is in an online setting and iAPF is in an offline setting.

We can then consider the variance and the estimated squared bias of the likelihood ratios from the three algorithms.

	BP	BPF	iAPF
$V\left[\hat{Z}/Z\right]$	0.1112	1.0119	0.0899
$(E\left[\hat{Z}/Z\right]-1)^2$	4.6485e-09	0.0165	0.0071

Again we see from Figure 7.9 that the iAPF has the lowest variance of the likelihood estimates compared to the BP and BPF. We however see that the variance of the likelihood estimates obtained by the BP is only slightly higher in this case. From Figure 7.9 we also see that the number of particles for BPF is likely too low considering the dimensionality ($d = 8$) of the data in this experiment. We also note that we are using a higher stability lag parameter $k = 7$ than Guarniero, Johansen, and Lee (2017) and a higher threshold $\tau = 1$. See Algorithm 8 for more details.

7.7 Experiment 7: effective sample size

Recall that if the optimal offline sequence $\Psi^*_{1:T}$ is used with a twisted model all the weight updates would be equal to 1 which again implies that all weights would be equal. In this scenario we would have that the effective sample size

$\mathrm{ESS}_t = n$ for $t = 1, \ldots, T$ and therefore we would never resample. In this numerical experiment we will focus on ESS in the batch setting with BP, in the offline setting with iAPF and in the online setting with BPF as a reference algorithm.

We will consider ESS and resampling with these algorithms both in the optimal scenario of a linear Gaussian model which is corresponding to model 1. Recall that in a linear Gaussian model with Gaussian approximations for the lookahead functions, we have that $p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$ is also Gaussian. We will also consider the scenario with discrete observations which is corresponding to model 4, which is no longer an optimal situation. In the iAPF, we report the estimates of ESS from the last iteration of the algorithm. We expect these estimates to be the best of all the algorithmic iterations of the iAPF algorithm. This is because of the iteratively improving estimates of $\bar{\psi}_t(x_t)$. Because the iAPF can increase the number of particles during one run of the algorithm, we will consider the quantity which we will refer to as the effective sample size fraction (EF) $\mathrm{EF}_t = \frac{\mathrm{ESS}_t}{n}$. This fraction will have values between $\frac{1}{n}$ and 1 where 1 indicates that the effective sample size is equal to the number of particles.

Setup

We consider a common setup for the three algorithms in this experiment. The same setup is used in both the models we consider.

Algorithm	Particles	Parameters
BP	$m = n = 100$	repeats = 3, $L = 10$, $\alpha = \beta = 0.5$
BPF	$n = 500$	$\alpha = 0.5$
iAPF	$n_0 = 100$	$k = 3$, $\tau = 0.5$, $\alpha = 0.5$

For approximations of the lookahead functions we will use $\bar{\psi}_t^b(x_t) = \mathcal{N}(x_t, \mu_t, \sigma_t^2)$ for the BP and $\bar{\psi}_t(x_t) = \mathcal{N}(x_t, \mu_t, \sigma_t^2)$ for the iAPF.

First we consider some aspects that are common for both the models. Our focus will be on estimating the expectation for EF_t estimated by the sample mean. We use a nonparametric bootstrap approach, see e.g. section 7.11 Hastie, Tibshirani, and J. Friedman (2009). The number of bootstrap samples is denoted by $B = 1000$ and we use $z_{0.05}$ as critical values in order to approximate a 90% confidence interval. We calculated an approximate confidence interval for each fraction EF_t for $t = 1, \ldots, T$. Our initial sample of $\mathrm{EF}_t^1, \ldots, \mathrm{EF}_t^{500}$ for every t was used as the initial sample for each of the bootstrap confidence intervals. In Figure 7.10 and Figure 7.12 below we have denoted the bootstrap confidence intervals as error bars for the different fractions EF_t.

The different lines in Figure 7.10 and Figure 7.12 represent the sample mean EF_t for $t = 1, \ldots, T$ for the three algorithms. Common for both models considered in this experiment is that we expect the iAPF to have the highest EF and therefore the lowest number of resampling counts. We expect this specially in the linear Gaussian case as we are approximating a linear Gaussian target function $p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$ with another Gaussian function $\bar{\psi}_t(x_t)$. We include histograms of the resampling counts in Figure 7.11 and Figure 7.13 for the three algorithms over the 500 runs. The main interest of this experiment is the BP algorithm. Also here we expect a trade-off in performance since we are no longer using all observations $y_{1:T}$ in a batch setting compared to an offline setting.

Linear Gaussian model

We simulated latent variables and observations from model 1 with length $T = 50$. We also used a batch length $L = 10$ for the batch version BP. This is to so we can consider the effect of going from the offline setting of iAPF to the batch setting of BP. We also used the distance function D_{iAPF2} from equation (4.17) for BP and iAPF. We consider the EF_t at the different t in Figure 7.10 and the number of resampling counts in Figure 7.11. We see from Figure 7.10 that

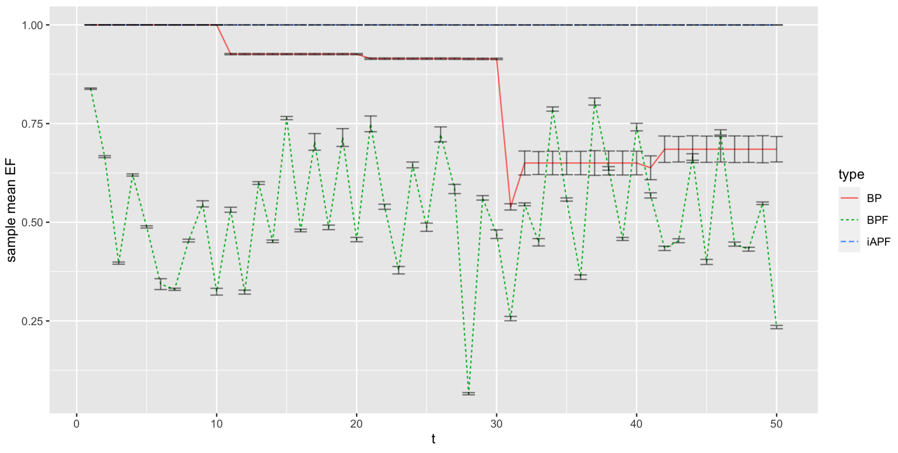


Figure 7.10: Sample mean EF_t for algorithms BP, BPF and iAPF over 500 runs using model 1. The error bars indicate estimated 90% confidence intervals.

EF_t is consistently high for the iAPF. This is again to be expected because we are here in an offline setting and we have an approximation $\bar{\psi}_t(x_t)$ on the same parametric form as $p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$ which we referred to as an optimal scenario. Considering BP in Figure 7.10, we see the effect of using batches with $L = 10$ compared to an offline setting. This is clear at iterations eL, that is $t = 10, 20, 30, 40$ as we often see a decrease in EF_t after these iterations. From Figure 7.11 we see the same as in Figure 7.10 when it comes to the iAPF, that is we do not resample in this optimal scenario. We also see in Figure 7.11 that the BP is often not resampling at all and in some runs it is resampling 1 time. This is likely resampling related to the iterations at the start of batch 4, that is iterations $t = 31, \ldots, 40$ as we see an increase in $\bar{\text{EF}}_t$ around iterations $t = 31, \ldots, 35$. We also see a slight increase in $\bar{\text{EF}}_t$ at the start of batch 5. The estimated confidence intervals are also wider around these iterations which might indicate that the EF_t in most of the runs were just above the resampling threshold. In some of the runs however, EF_t were below the threshold and resampling were performed.

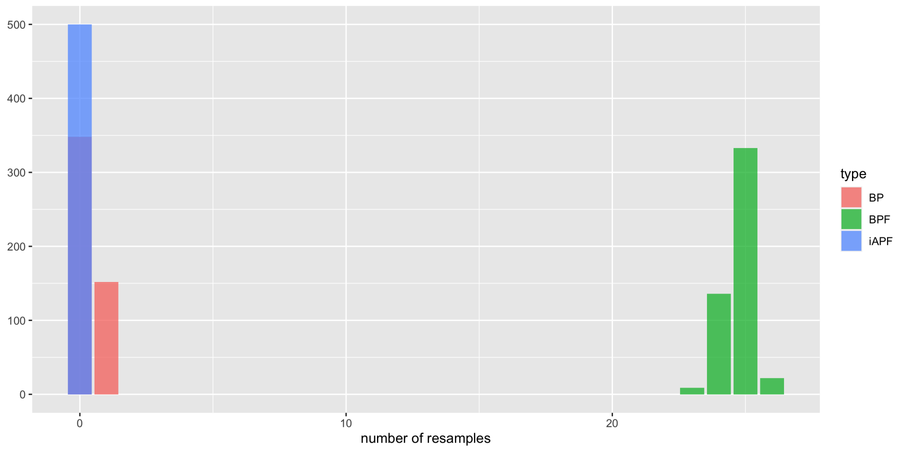


Figure 7.11: Resampling counts for algorithms BP, BPF and iAPF over 500 runs using model 1.

Model with discrete observations

When we are not using a linear Gaussian model we expect iAPF and BP to resample more often because we use the same approximations $\bar{\psi}_t(x_t)$ for $t = 1, \ldots, T$. We also used the distance function D_{iAPF2} from equation (4.17) for BP and iAPF. Now we have that the target $\dot{\psi}_t(x_t) = p(y_t|x_t)\tilde{\bar{\psi}}_t(x_t)$ are products of discrete Poisson probability mass functions and $\tilde{\bar{\psi}}_t(x_t)$. We now consider the EF_t for different t in Figure 7.12 and the resampling counts in Figure 7.13. As with the linear Gaussian model, we see from Figure 7.12 that

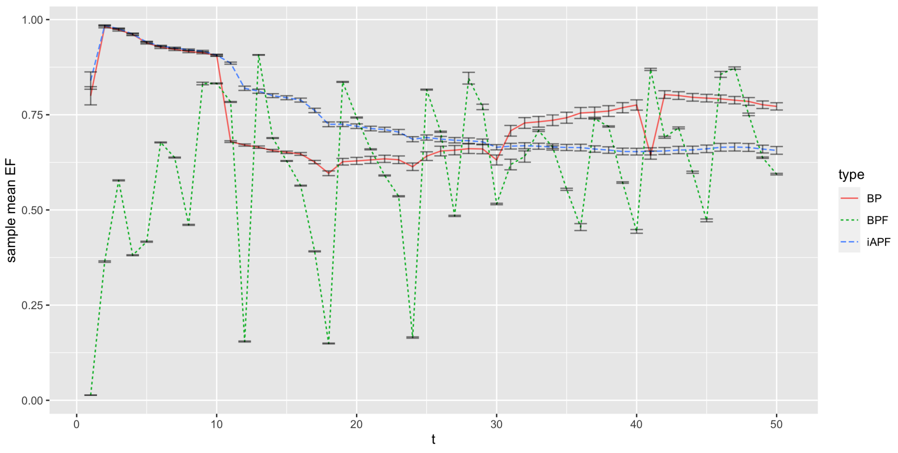


Figure 7.12: Sample mean EF_t for algorithms BP, BPF and iAPF over 500 runs using model 4. The error bars indicate estimated 90% confidence intervals.

the $\bar{\text{EF}}_t$ falls at the start of a new batch. We also note that $\bar{\text{EF}}_1$ for the BPF is very low. This is due to observation y_1 being an outlier (not shown). Compared to the situation in Figure 7.10 we see a bigger decrease in $\bar{\text{EF}}_t$ in this case. We also see from Figure 7.12 that the $\bar{\mathrm{EF}}_t$ obtained from the iAPF has a gradual decrease with relatively narrow estimated confidence intervals. The BP is resampling more often as we see in Figure 7.13 which also helps explain the increase in $\bar{\mathrm{EF}}_t$ in the last two batches. Also from Figure 7.12

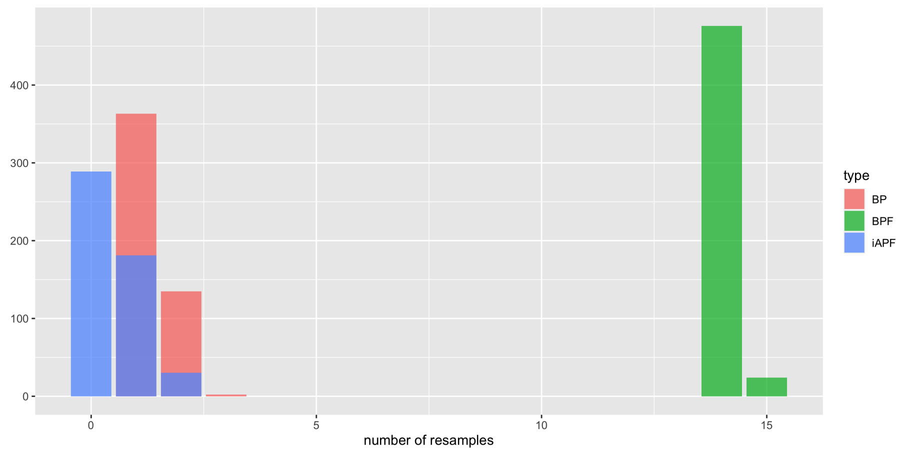


Figure 7.13: Resampling counts for algorithms BP, BPF and iAPF over 500 runs using model 4.

we recognise the decrease in $\bar{\mathrm{EF}}_t$ in the first iteration of a new batch followed by an increase. This is again likely due to resampling. We can think of the iterations at the end of a batch utilising fewer and fewer future observations when simulating x_t. Simulation of the last latent variables in a batch is only utilising the current y_t. Recall from Section 5.3 that the next batch is utilising these simulated variables in the initial twisted transition density and twisted observation function. This may also help explain the abrupt decrease in $\bar{\mathrm{EF}}_t$ at the start of batches. Comparing the linear Gaussian model with the experiment with the discrete observations we also note the suboptimal approximations of $\bar{\psi}_t(x_t)$ in the iAPF and the BP. This is seen as a steady decrease of the sample mean $\bar{\mathrm{EF}}_t$ with t compared to the optimal case with the linear Gaussian model.

7.8 Experiment 8: batch likelihood

We now want to consider the estimate of the marginal likelihood obtained at the end of each batch. We use model 1 and model 2 in this experiment. Model 1 is selected in order to calculate the likelihood exactly by using the Kalman filter. We also consider model 2 because model 1 represents the optimal situation that we saw in experiment 1 and 2.

Linear Gaussian model

We start by considering model 1 where the likelihood obtained from the Kalman filter is denoted by Z. Our main interest is to consider the variance of the estimated marginal likelihood at the end of each batch. The Kalman filter is used to calculate the marginal likelihood up until the iteration that corresponds to the last of each batch. The likelihood estimates $\hat{Z}$ at the end of each batch are then compared to the likelihood Z obtained from running a Kalman filter up until the corresponding iteration.

Setup

Recall that the number of iterations within each batch is L. We therefore expect that the algorithm with the highest number of iterations within each batch should have lower variance for the likelihood estimates. We consider two slightly different setups of the same experiment. Two setups of the algorithm BP are defined. The first setup is denoted by BP_{10} and here we use $L = 10$. The second setup is denoted by BP_{20} and here we use $L = 20$. We use the same number of particles, improvement steps and resampling thresholds.

Algorithm	Particles	Parameters
BP_{10}	$m = n = 100$	repeats $= 3$, $L = 10$, $\alpha = \beta = 0.5$
BP_{20}	$m = n = 100$	repeats $= 3$, $L = 20$, $\alpha = \beta = 0.5$

We use the approximations $\bar{\psi}_t^b(x_t) = \mathcal{N}(x_t; \mu_t, \sigma_t^2)$ for the lookahead functions within the two variants of the BP algorithm. We also used the distance function D_{iAPF2} from equation (4.17) for BP.

Data

We simulate $x_{1:80}$ latent variables and $y_{1:80}$ observations from model 1. These are shown in Figure 7.14.

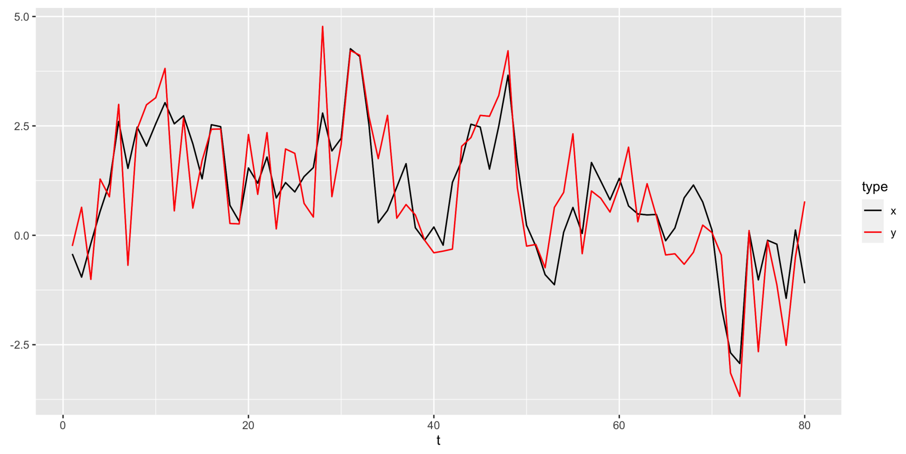


Figure 7.14: Simulated latent variables $x_{1:T}$ and observations $y_{1:T}$ from model 1, $T = 80$.

Results

The likelihood estimates obtained at the end of batch e are denoted by $\hat{Z}_{eL}$ and we denote the likelihood calculated with the Kalman filter up to iteration eL

by Z_{eL}. We therefore calculate

$$\mathrm{LR} = \frac{\hat{Z}_{eL}}{Z_{eL}} \tag{7.1}$$

for BP_{10} and BP_{20}. We first run BP_{10} 100 times. With $L = 10$, the algorithm use 8 batches in total. We obtain 100 estimates of the likelihood at the end of each of the 8 batches. The likelihood ratio in equation (7.1) is calculated for each of the batches and the result is in Figure 7.15. From Figure 7.15 we see that the likelihood estimates from BP_{10} are relatively close to the Z_{eL} obtained by the Kalman filter. From Figure 7.15 and Figure 7.16 we see that

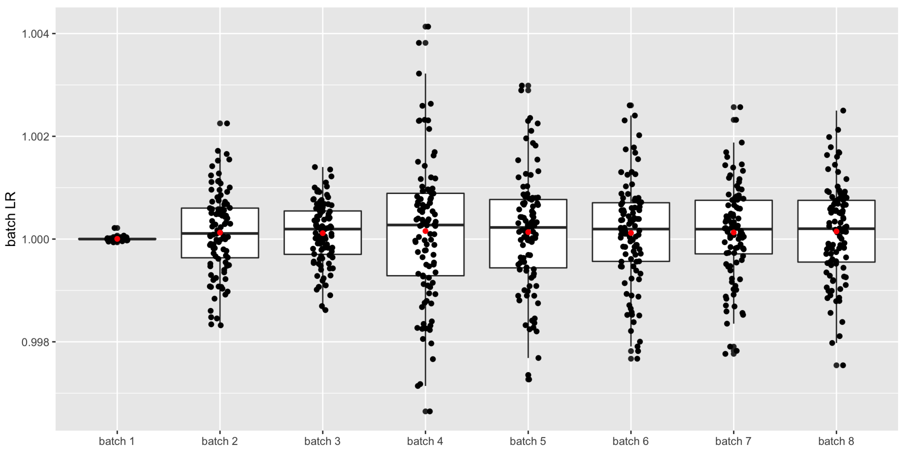


Figure 7.15: Box plots of 100 likelihood ratios with respect to the likelihood from a Kalman filter. Each box plot show the likelihood ratios at the end of the batches, the red circles represent the sample means. The likelihood estimates are obtained by the algorithm BP_{10}.

the variance of the likelihood estimates increase substantially from the end of batch 1 to the end of batch 2. This may be explained by the fact that in the first batch we are using all the observations in order to simulate all the latent variables. Because of the linear Gaussian model and the selected approximation $\bar{\psi}_t(x_t) = \mathcal{N}(x_t, \mu_t, \sigma_t^2)$ we have that the batch recursive structure from equation (5.8) should approximately hold which might help explain the low variance of the likelihood estimates obtained at the end of batch 1. In Figure 7.16 we are considering the same experiment with algorithm BP_{20}. By the scale of the y-axis, we see that the likelihood estimates when $L = 20$ have lower variance than the situation where $L = 10$. When $L = 20$, the BP is utilising 4 batches as seen in Figure 7.16. Here we also see that the variance of the likelihood estimates at the end of batch 1 is lower than after the subsequent batches. Comparing the likelihood ratios in Figure 7.15 to those in Figure 7.16, we see that given the same fixed observations $y_{1:80}$, having longer batches decrease the variance of the likelihood estimates.

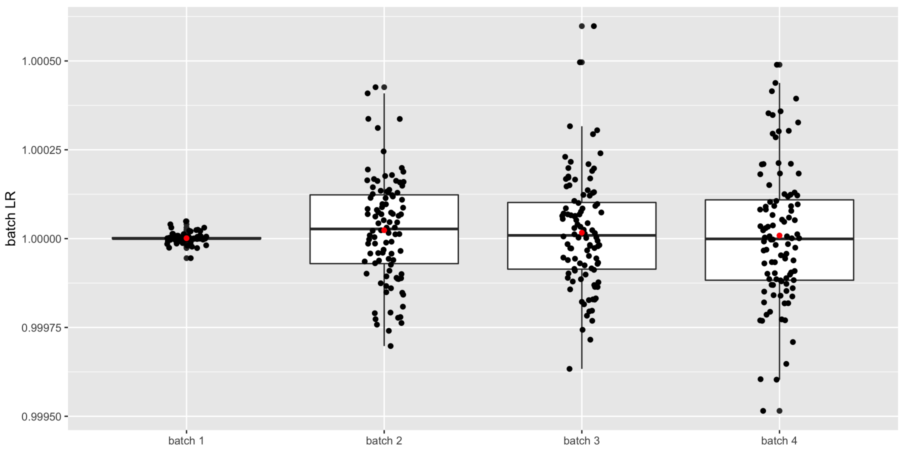


Figure 7.16: Box plots of 100 likelihood ratios with respect to the likelihood from a Kalman filter. Each box plot show the likelihood ratios at the end of the batches, the red circles represent the sample means. The likelihood estimates are obtained by the algorithm BP_{20}.

Univariate stochastic volatility model

We also consider the USV model. Compared to the definition of model 2 we here define $\alpha = 0.5$ instead of 0.8 as in Chapter 7. This implies lower variance in $p(x_1)$.

Setup

We use mainly the same setup as with the linear Gaussian model, but we increase both the number of particles and the number of repeats. Recall that the number of repeats represents the iteratively improving approximation $\bar{\psi}^b$.

Algorithm	Particles	Parameters
BP_{10}	$m = n = 200$	repeats $= 4$, $L = 10$, $\alpha = \beta = 0.5$
BP_{20}	$m = n = 200$	repeats $= 4$, $L = 20$, $\alpha = \beta = 0.5$

Data

We simulate $x_{1:80}$ latent variables and $y_{1:80}$ observations from model 2. These are shown in Figure 7.17.

Results

Comparing the likelihood ratios in Figure 7.18 with the likelihood ratios in Figure 7.15 we see the effect of not being in the optimal situation that we considered with the linear Gaussian model. The first thing we notice when comparing Figure 7.18 and Figure 7.15 is the sample variance of the likelihood ratios at the end of batch 1. With the linear Gaussian model and $\bar{\psi}_t(x_t) = \mathcal{N}(x_t, \mu_t, \sigma_t^2)$ we have the optimal situation. This resulted in the low variance of the likelihood estimates at the end of batch 1 in Figure 7.15. When not in the optimal situation, we see from Figure 7.18 that the variance of the

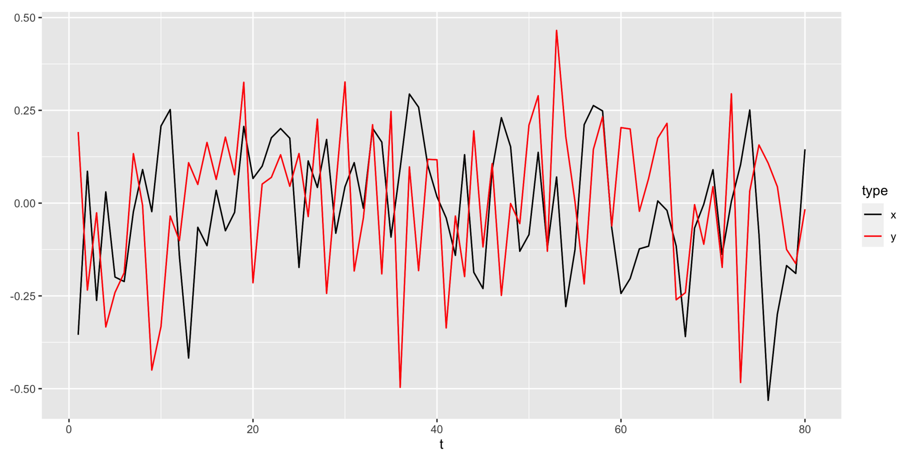


Figure 7.17: Simulated latent variables $x_{1:T}$ and observations $y_{1:T}$ from model 2, $T = 80$.

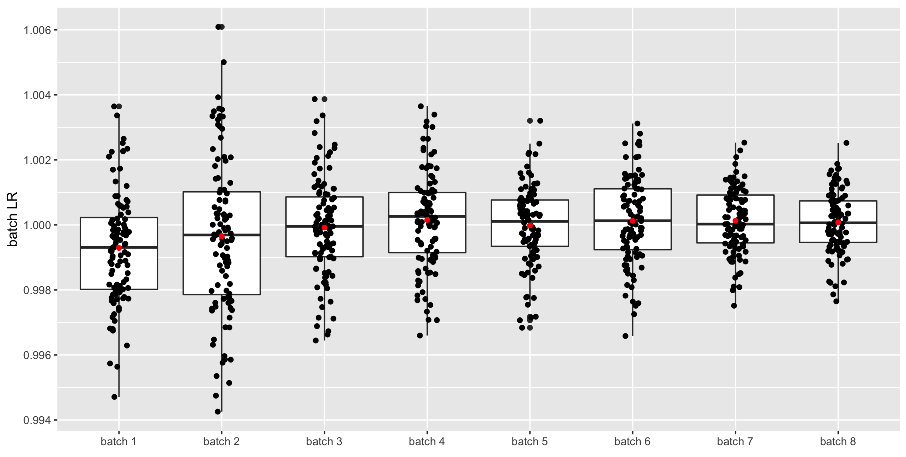


Figure 7.18: Box plots of 200 likelihood ratios with respect to the likelihood from a BPF using 500 000 particles. Each box plot show the likelihood ratios at the end of the batches, the red circles represent the sample means. The likelihood estimates are obtained by the algorithm BP_{10}.

likelihood estimates is higher at the end of batch 1. When increasing L to 20 we see that the variance of the likelihood estimates decrease compared to using $L = 10$. When comparing Figure 7.16 with Figure 7.19 we also here see that the variance is higher compared to the optimal situation. What seems to be a slight decrease of variance of the likelihood estimates at later batches in Figure 7.19 is somewhat unexpected.

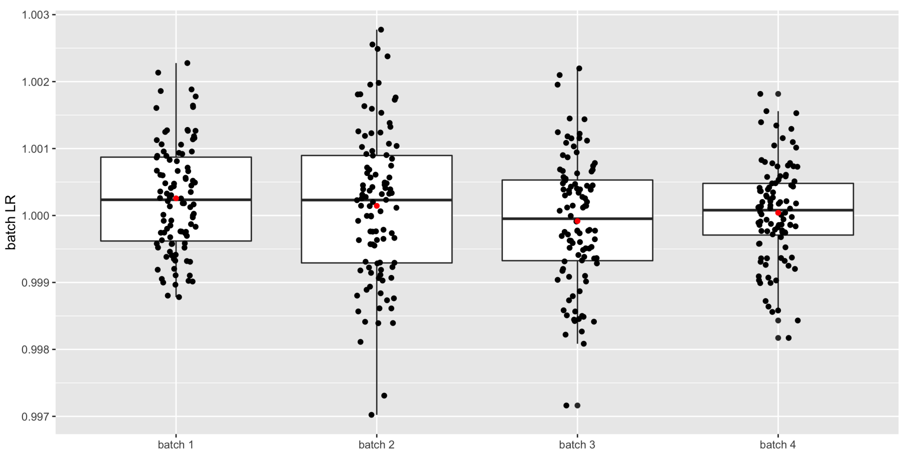


Figure 7.19: Box plots of 200 likelihood ratios with respect to the likelihood from a BPF using 500 000 particles. Each box plot show the likelihood ratios at the end of the batches, the red circles represent the sample means. The likelihood estimates are obtained by the algorithm BP_{20}.

CHAPTER 8

Conclusions

8.1 Summary

In Chapter 2 we briefly introduced state space models, Monte Carlo methods and related concepts. In Chapter 3 we considered some inferential aspects related to different Sequential Monte Carlo methods. We considered filtering and particle filter algorithms and in addition we briefly considered some aspects related to smoothing. Further we considered likelihood related to state space models. Specifically we considered estimating the likelihood of observations related to a state space models by using particle filter algorithms. The objective was to obtain likelihood estimates with low variance for further inference, for example in relation to parameter estimation. In Section 3.5 we saw that likelihood estimates can be obtained in an online setting by using e.g. a bootstrap particle filter. In that case we obtained a marginal likelihood estimate for each iteration by using approximations of likelihood factors.

Further, we considered twisting target distributions in Chapter 4 as a strategy to obtain likelihood estimates with low variance in an offline setting. We also considered $\Psi^*_{1:T}$, the optimal sequence with respect to minimising variance of likelihood estimates in Section 4.3. The optimal lookahead functions within the optimal sequence are generally intractable. For this reason we used approximations of the optimal lookahead functions in the offline setting utilising the backwards recursive structure. We saw in Section 4.7 that the recursive structure defined by the optimal twisting functions did not hold when we considered what we referred to as a constant lookahead setting. That is, lookahead functions on the form $\psi^c_t(x_t) = p(y_{t:t+C}|x_t)$ where C is a positive constant.

In Chapter 5 we considered twisting target distributions in a batch setting where we found the recursive structure within the batches which we denoted by batch recursive structure. This is in contrast with the constant lookahead setting. Twisting target distributions in the batch setting allowed us to obtain likelihood estimates at the end of each batch due to target equivalence with the joint distribution $p(x_{1:eL}, y_{1:eL})$ in batch e. Due to the batch recursive structure and how the batch twisting functions were defined, we could approximate the batch twisting functions by using the backwards recursive estimation within each batch. We also saw that we could have iteratively improving batch lookahead functions ψ^b in the same way Guarniero, Johansen, and Lee (2017) have iteratively improving lookahead functions ψ in an offline setting. Finally we presented an algorithm outline Algorithm 9 which can be used with parametric or nonparametric approximations $\bar{\psi}^b$. We considered the effect of using the batch setting in the numerical experiments of Chapter 7 and saw that using batch twisting functions and iteratively improving approximations also helped to reduce variance of the likelihood estimates in a batch setting.

In Chapter 6 we considered an alternative formulation of the optimal lookahead function ψ^* that we denoted by ψ'. We then considered approximations of this alternative lookahead function as an alternative to approximating the optimal lookahead function. The approximation of this alternative lookahead function was denoted by $\bar{\psi}'$. The main appeal of the approximation $\bar{\psi}'$ from Section 6.3 was that it do not require us to select a parametric class $\bar{\Psi}$ for the approximations of lookahead functions. The high computational complexity related to the approximation $\bar{\psi}'$ is however an issue. We considered some numerical experiments using $\bar{\psi}'$ in Chapter 7 where the variance of the likelihood estimates tended to be somewhat lower than those from the BPF, but the computational cost was far greater. We also considered what we referred to as Laplace-based approximations. For special cases of transition densities, we could utilise a Gaussian approximation of the observation density. Combined, this allowed us to calculate the twisting functions deterministically once we had the observations. We illustrated this in Section 6.5.

We considered a series of numerical experiments in Chapter 7. We considered different models and compared the variance of the likelihood estimates obtained. In summary, the results from the numerical experiments is in line with the numerical experiments from Guarniero, Johansen, and Lee (2017). The iAPF in an offline setting tended to provide likelihood estimates with the lowest variance. Followed by Algorithm 9 in a batch setting with parametric approximations $\bar{\psi}^b$ that tended to provide likelihood estimates with slightly higher variance than the iAPF. Followed by the bootstrap particle filter in an online setting, with a higher number of particles. The BPF tended to have higher variance than the batch algorithm. When the dimensionality of the problems increased, we however saw that the iAPF and batch algorithm obtained likelihood estimates with relatively low variance compared to the BPF using a higher number of particles which saw an large increase in variance. This is also in line with the numerical experiments from Guarniero, Johansen, and Lee (2017) where the iAPF tended to tackle high-dimensional problems better with respect to variance of the likelihood estimates.

8.2 Further work

As the objective was to obtain likelihood estimates with low variance, a main area of interest can therefore be to combine the batch algorithm with parameter estimation. Specifically a similar parameter estimation setting as in J. Liu and West (2001) where one could e.g. add artificial noise to the parameter θ only at the beginning of a new batch.

An interesting addition in Algorithm 9 would be a stochastic stopping criterion for the iteratively improvement of $\bar{\psi}^b$. Another area of interest would be an increase in the number of particles at the final iteration of batch $e-1$ and in the first iteration of batch e. This is based on the experiments from Chapter 7 where we saw significant decreases in effective sample size at the iterations around the start of a new batch.

Another interesting aspect to consider is to introduce overlapping batches. One could think of temporarily storing the the last half of observations from the previous batch. These observations could perhaps be combined with the first half batch of new observations. We would then have a overlapping batch in which we run the batch algorithm. In this way one could utilise future observations also for the last iterations of the previous batch. It would be interesting to consider the effect of overlapping batches on the effective sample size.

The parametric approximations $\bar{\psi}^b$ were of a simple form in this thesis as it usually was set equal to a single Gaussian. There are here several possibilities for using e.g. mixture Gaussian or other parametric forms for the approximations. For the alternative approximation $\bar{\psi}'$ from Chapter 6 decreasing the computational complexity of these alternative approximations would be the primary goal.

Appendices

APPENDIX A

Extended calculations

A.1 Extended calculations chapter 2

Normalised IS estimator

We can write the target distribution as $p(x_{1:t}|y_{1:t}) = \frac{p(x_{1:t},y_{1:t})}{Z_t}$. We then have

$$E_p\left[k(x_{1:t})\right] = \frac{\int \frac{p(x_{1:t},y_{1:t})}{g(x_{1:t}|y_{1:t})} k(x_{1:t}) g(x_{1:t}|y_{1:t}) dx_{1:t}}{\int \frac{p(x_{1:t},y_{1:t})}{g(x_{1:t}|y_{1:t})} g(x_{1:t}|y_{1:t}) dx_{1:t}} = \frac{E_g\left[\frac{p(x_{1:t},y_{1:t})}{g(x_{1:t}|y_{1:t})} k(x_{1:t})\right]}{E_g\left[\frac{p(x_{1:t},y_{1:t})}{g(x_{1:t}|y_{1:t})}\right]}$$

This derivation was adapted from Chopin and Papaspiliopoulos (2020, p. 86).

Difference in MSE

Now we consider the difference in MSE between $\hat{\mu}_k^{\text{IS}}$ and $\tilde{\mu}_k^{\text{IS}}$. This is given by

$$\begin{aligned} \text{MSE}\left[\hat{\mu}_k^{\text{IS}}\right] &= V_g\left[\hat{\mu}_k^{\text{IS}}\right] + \left(E_g\left[\hat{\mu}_k^{\text{IS}}\right] - \mu_k\right)^2 \\ &\approx \frac{1}{n} V_g[u] - 2\mu_k \frac{1}{n} \text{Cov}[u,w] + \frac{1}{n}\mu_k^2 V_g[w]. \end{aligned}$$

Here we use the approximate expressions for the expectation and variance. We then consider the unnormalised estimator given by

$$\text{MSE}\left[\tilde{\mu}_k^{\text{IS}}\right] = V_g\left[\tilde{\mu}_k^{\text{IS}}\right] + \left(E_g\left[\tilde{\mu}_k^{\text{IS}}\right] - \mu_k\right)^2 = \frac{1}{n} V_g[u].$$

When we consider the difference in MSE we then have that

$$\text{MSE}\left[\hat{\mu}_k^{\text{IS}}\right] - \text{MSE}\left[\tilde{\mu}_k^{\text{IS}}\right] \approx \frac{1}{n}\mu_k^2 V_g[w] - 2\mu_k \frac{1}{n}\text{Cov}[u,w].$$

We want to consider the case where $\text{MSE}\left[\hat{\mu}_k^{\text{IS}}\right] < \text{MSE}\left[\tilde{\mu}_k^{\text{IS}}\right]$. That is, when we should use the normalised estimator instead of the unnormalised estimator. We follow the assumptions of Givens and Hoeting (2013) and assume that $\mu_k > 0$. We then have

$$\frac{1}{n}\mu_k^2 V_g[w] - 2\mu_k \frac{1}{n}\text{Cov}[u,w] < 0 \to \frac{\mu_k V_g[w]}{2\sqrt{V_g[u]}\sqrt{V_g[w]}} < \frac{\text{Cov}[u,w]}{\sqrt{V_g[u]}\sqrt{V_g[w]}}$$

$$\frac{\frac{\sqrt{V_g[w]}}{1}}{2\frac{\sqrt{V_g[u]}}{\mu_k}} < \text{Corr}[u,w] \to \frac{\text{CV}[w]}{2\text{CV}[u]} < \text{Corr}[u,w].$$

Sequential weight structure

The second equality is given by multiplying by 1 and substituting the decomposition of (2.5) into the expression. We then have

$$\tilde{w}_t^i = \frac{p(x_{1:t}^i|y_{1:t})}{g(x_{1:t}^i|y_{1:t})} = \frac{p(x_{1:t-1}^i|y_{1:t-1})}{g(x_{1:t-1}^i|y_{1:t-1})} \frac{p(x_{1:t}^i|y_{1:t})}{p(x_{1:t-1}^i|y_{1:t-1})g(x_t^i|y_{1:t}, x_{1:t-1}^i)}$$

A.2 Extended calculations chapter 4

Optimal sequence of lookahead functions

Recall first the assumption $\tilde{\psi}_T(x_T) \equiv 1$ and Definition 4.2.1,

$$\begin{aligned}
\psi_T^*(x_T) &= p(y_T|x_T)\tilde{\psi}_T^*(x_T) = p(y_T|x_T) \\
\psi_{T-1}^*(x_{T-1}) &= p(y_{T-1}|x_{T-1})p(y_T|x_{T-1}) \\
&= p(y_{T-1}|x_{T-1}) \int p(y_T|x_T)p(x_T|x_{T-1})dx_T \\
&= p(y_{T-1}|x_{T-1}) \int \psi_T^*(x_T)p(x_T|x_{T-1})dx_T \\
&= p(y_{T-1}|x_{T-1})\tilde{\psi}_{T-1}^*(x_{T-1}) \\
\psi_{T-2}^*(x_{T-2}) &= p(y_{T-2}|x_{T-2})p(y_{T-1:T}|x_{T-2}) \\
&= p(y_{T-2}|x_{T-2}) \int\int \prod_{s=T-1}^{T} p(y_s|x_s)p(x_s|x_{s-1})dx_{T-1:T} \\
&= p(y_{T-2}|x_{T-2}) \int \\
&p(y_{T-1}|x_{T-1}) \left[\int p(y_T|x_T)p(x_T|x_{T-1})dx_T\right] p(x_{T-1}|x_{T-2})dx_{T-1} \\
&= p(y_{T-2}|x_{T-2}) \int \psi_{T-1}^*(x_{T-1})p(x_{T-1}|x_{T-2})dx_{T-1} \\
&= p(y_{T-2}|x_{T-2})\tilde{\psi}_{T-2}^*(x_{T-2}) \\
&\vdots
\end{aligned}$$

Approximations of optimal twisting functions

Recall the assumption $\tilde{\psi}_T^*(x_T) \equiv 1$. For the last approximate lookahead function we then get

$$\begin{aligned}
\tilde{\psi}_{T-1}^*(x_{T-1}) &= \int p(y_T|x_T)p(x_T|x_{T-1})\tilde{\psi}_T^*(x_T)dx_T \\
&= \int p(y_T|x_T)\tilde{\psi}_T^*(x_T)p(x_T|x_{T-1})dx_T \\
\tilde{\bar{\psi}}_{T-1}(x_{T-1}) &= \int \bar{\psi}_T(x_T)p(x_T|x_{T-1})dx_T.
\end{aligned}$$

This implies $\bar{\psi}_T(x_T) \approx p(y_T|x_T)\tilde{\psi}^*_T(x_T) \approx p(y_T|x_T)$. Then for $\bar{\psi}_{T-1}(x_{T-1})$ we get

$$\begin{aligned}
\tilde{\psi}^*_{T-2}(x_{T-2}) &= \int p(y_{T-1}|x_{T-1})p(x_{T-1}|x_{T-2})\tilde{\psi}^*_{T-1}(x_{T-1})dx_{T-1} \\
&= \int p(y_{T-1}|x_{T-1})\tilde{\psi}^*_{T-1}(x_{T-1})p(x_{T-1}|x_{T-2})dx_{T-1} \\
\tilde{\bar{\psi}}_{T-2}(x_{T-2}) &= \int \bar{\psi}_{T-1}(x_{T-1})p(x_{T-1}|x_{T-2})dx_{T-1}.
\end{aligned}$$

This implies that

$$\begin{aligned}
\bar{\psi}_{T-1}(x_{T-1}) &= p(y_{T-1}|x_{T-1})\tilde{\psi}^*_{T-1}(x_{T-1}) \\
&\approx p(y_{T-1}|x_{T-1})\tilde{\bar{\psi}}_{T-1}(x_{T-1}).
\end{aligned}$$

Numerical optimisation

Here we will denote component j of θ as θ_j. We want to minimise the distance function with respect to the composite vector $(\theta, \lambda_t)^T$. We focus on the distance function D_{iAPF}. We then have

$$\begin{aligned}
\nabla D(\theta) &= \begin{pmatrix} \frac{\partial}{\partial\theta_1}D(\theta) \\ \frac{\partial}{\partial\theta_2}D(\theta) \\ \vdots \\ \frac{\partial}{\partial\lambda_t}D(\theta) \end{pmatrix} \\
&= \begin{pmatrix} 2\sum_{i=1}^n \left[\bar{\psi}_t(x_t^i,\theta) - \lambda_t\dot{\psi}_t(x_t^i)\right]\frac{\partial\bar{\psi}_t(x_t^i,\theta)}{\partial\theta_1} + \frac{\partial}{\partial\theta_1}r_t(x_t^{1:n},\theta,\lambda_t) \\ 2\sum_{i=1}^n \left[\bar{\psi}_t(x_t^i,\theta) - \lambda_t\dot{\psi}_t(x_t^i)\right]\frac{\partial\bar{\psi}_t(x_t^i,\theta)}{\partial\theta_2} + \frac{\partial}{\partial\theta_2}r_t(x_t^{1:n},\theta,\lambda_t) \\ \vdots \\ -2\sum_{i=1}^n \left[\bar{\psi}_t(x_t^i,\theta) - \lambda_t\dot{\psi}_t(x_t^i)\right]\dot{\psi}_t(x_t^i) + \frac{\partial}{\partial\lambda_t}r_t(x_t^{1:n},\theta,\lambda_t) \end{pmatrix}.
\end{aligned}$$

To consider the Hessian, we need the second derivatives

$$H = \begin{pmatrix} \frac{\partial^2}{\partial\theta_1^2}D(\theta) & \dots & \frac{\partial^2}{\partial\theta_1\partial\lambda_t}D(\theta) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial\lambda_t\partial\theta_1}D(\theta) & \dots & \frac{\partial^2}{\partial\lambda_t^2}D(\theta) \end{pmatrix}.$$

These are found by taking the derivatives of the terms in the gradient. Here we have the general form for component j in θ

$$\begin{aligned}
\frac{\partial^2}{\partial\theta_j^2}D(\theta) = 2\sum_{i=1}^n &\frac{\partial\bar{\psi}_t(x_t^i,\theta)}{\partial\theta_j}\frac{\partial\bar{\psi}_t(x_t^i,\theta)}{\partial\theta_j} \\
&+ \left[\bar{\psi}_t(x_t^i,\theta) - \lambda_t\dot{\psi}_t(x_t^i)\right]\frac{\partial^2\bar{\psi}_t(x_t^i,\theta)}{\partial\theta_j^2} \\
&+ \frac{\partial^2}{\partial\theta_j^2}r_t(x_t^{1:n},\theta,\lambda_t) \\
\frac{\partial^2}{\partial\lambda_t\partial\theta_j}D(x_t,\theta) = -2\sum_{i=1}^n &\frac{\partial\bar{\psi}_t(x_t^i,\theta)}{\partial\theta_j}\dot{\psi}_t(x_t^i) + \frac{\partial^2}{\partial\lambda_t\partial\theta_j}r_t(x_t^{1:n},\theta,\lambda_t)
\end{aligned}$$

$$\frac{\partial^2}{\partial \lambda_t^2} D(x_t, \theta) = 2 \sum_{i=1}^{n} \dot{\psi}_t(x_t^i)\dot{\psi}_t(x_t^i) + \frac{\partial^2}{\partial \lambda_t^2} r_t(x_t^{1:n}, \theta, \lambda_t).$$

The cross terms are found by similar calculations.

A.3 Extended calculations chapter 5

The batch sequence

We consider the batch sequence with one row per batch. Recall that we denote the first iteration of a generic batch e by $q+1$. We also denote the first iteration of the last batch E by $r+1$. Recall that eL is the last iteration in a batch, we then have

$$\begin{aligned}
\Psi_{1:T}^s =& \psi_1(x_1), \psi_2(x_2) \dots, \psi_{L-1}(x_{L-1}), \psi_L(x_L) \\
& \vdots \\
& \psi_{q+1}(x_{q+1}, x_q), \psi_{q+2}(x_{q+2}) \dots, \psi_{eL-1}(x_{eL-1}), \psi_{eL}(x_{eL}) \\
& \vdots \\
& \psi_{r+1}(x_{r+1}, x_r), \psi_{r+2}(x_{r+2}) \dots, \psi_{EL-1}(x_{EL-1}), \psi_{EL}(x_{EL}) \\
=& \psi_1^b(x_1), \psi_2^b(x_2) \dots, \psi_{L-1}^b(x_{L-1}), \psi_L^b(x_L) \\
& \vdots \\
& \frac{\psi_{q+1}^b(x_{q+1})}{\tilde{\psi}_{0e}^b(x_q)}, \psi_{q+2}^b(x_{q+2}) \dots, \psi_{eL-1}^b(x_{eL-1}), \psi_{eL}^b(x_{eL}) \\
& \vdots \\
& \frac{\psi_{r+1}^b(x_{r+1})}{\tilde{\psi}_{0E}^b(x_r)}, \psi_{r+2}^b(x_{r+2}) \dots, \psi_{EL-1}^b(x_{EL-1}), \psi_{EL}^b(x_{EL})
\end{aligned}$$

Batch target equivalence

We here consider batch target equivalence when using the batch twisted models. Recall that batch e contains the iterations $(e-1)L+1, (e-1)L+2, \dots, eL-1, eL$. Here we consider batch target equivalence at the end of batch e. Recall that the batch normalising function $\tilde{\psi}_{eL}^b \equiv 1$ from equation (5.7). Each of the lines below represent twisted transition densities and twisted observation functions from the corresponding batch.

$$\begin{aligned}
& p_1^\psi(y_1|x_1) p_1^\psi(x_1) \prod_{s=2}^{L} p_s^\psi(y_s|x_s) p_s^\psi(x_s|x_{s-1}) \\
& p_{L+1}^\psi(y_{L+1}|x_{L+1}, x_L) p_{L+1}^\psi(x_{L+1}|x_L) \prod_{s=L+2}^{2L} p_s^\psi(y_s|x_s) p_s^\psi(x_s|x_{s-1}) \\
& \vdots
\end{aligned}$$

$$
\begin{aligned}
&p^{\psi}_{q+1}(y_{q+1}|x_{q+1},x_q)p^{\psi}_{q+1}(x_{q+1}|x_q)\prod_{s=q+2}^{eL} p^{\psi}_s(y_s|x_s)p^{\psi}_s(x_s|x_{s-1})\\
=&\frac{p(y_1|x_1)\tilde\psi^b_1\tilde\psi^b_{01}}{\psi^b_1}\frac{p(x_1)\psi^b_1}{\tilde\psi^b_{01}}\prod_{s=2}^{L}\frac{p(y_s|x_s)\tilde\psi^b_s}{\psi^b_s}\frac{p(x_s|x_{s-1})\psi^b_s}{\tilde\psi^b_{s-1}}\\
&\frac{p(y_{L+1}|x_{L+1})\tilde\psi^b_{L+1}\tilde\psi^b_{02}}{\psi^b_{L+1}}\frac{p(x_{L+1}|x_L)\psi^b_{L+1}}{\tilde\psi^b_{02}}\prod_{s=L+2}^{2L}\frac{p(y_s|x_s)\tilde\psi^b_s}{\psi^b_s}\frac{p(x_s|x_{s-1})\psi^b_s}{\tilde\psi^b_{s-1}}\\
&\vdots\\
&\frac{p(y_{q+1}|x_{q+1})\tilde\psi^b_{q+1}\tilde\psi^b_{0E}}{\psi^b_{q+1}}\frac{p(x_{q+1}|x_q)\psi^b_{q+1}}{\tilde\psi^b_{0E}}\prod_{s=q+2}^{eL}\frac{p(y_s|x_s)\tilde\psi^b_s}{\psi^b_s}\frac{p(x_s|x_{s-1})\psi^b_s}{\tilde\psi^b_{s-1}}\\
=&\left[p(y_1|x_1)p(x_1)\prod_{s=2}^{L}p(y_s|x_s)p(x_s|x_{s-1})\right]\tilde\psi^b_L\\
&\left[p(y_{L+1}|x_{L+1})p(x_{L+1}|x_L)\prod_{s=L+2}^{2L}p(y_s|x_s)p(x_s|x_{s-1})\right]\tilde\psi^b_{2L}\\
&\vdots\\
&\left[p(y_{q+1}|x_{q+1})p(x_{q+1}|x_q)\prod_{s=q+2}^{eL}p(y_s|x_s)p(x_s|x_{s-1})\right]\tilde\psi^b_{eL}\\
=&\left[p(y_1|x_1)p(x_1)\prod_{s=2}^{L}p(y_s|x_s)p(x_s|x_{s-1})\right]\\
&\left[\prod_{s=L+1}^{2L}p(y_s|x_s)p(x_s|x_{s-1})\right]\\
&\vdots\\
&\left[\prod_{s=q+1}^{eL}p(y_s|x_s)p(x_s|x_{s-1})\right]\\
=&p(y_1|x_1)p(x_1)\prod_{s=2}^{eL}p(y_s|x_s)p(x_s|x_{s-1}).
\end{aligned}
$$

Optimal batch twisting functions

We use the same intuition as in the offline setting, following Naesseth, Lindsten, and Schön (2019). Iteration $q+1$ is here the first iteration and iteration eL the last in batch e. We then start by inserting the recursively defined $\tilde f^b_{t+1}(x_{1:t+1})$ into the same integral that the unnormalised optimal batch twisting target distribution is defined by. This gives the recursive structure

$$\tilde f^b_t(x_{1:t}) = \int \tilde f^b_{t+1}(x_{1:t+1})dx_{t+1} \tag{A.1}$$

$$
\begin{aligned}
&= \int \tilde{f}_t^b(x_{1:t})\phi_{t+1}(x_{t+1}, x_t)\frac{\tilde{\psi}_{t+1}^b(x_{t+1})}{\tilde{\psi}_t^b(x_t)}dx_{t+1} \\
\tilde{\psi}_t^{b*}(x_t) &= \int \phi_{t+1}(x_{t+1}, x_t)\tilde{\psi}_{t+1}^{b*}(x_{t+1})dx_{t+1}. \qquad \text{(A.2)}
\end{aligned}
$$

We consider the iteration $t = eL$ which is the last in the batch. Recall that the recursively defined twisting target distribution is equal to the twisting target distributions from the twisted model. We then have because of batch target equivalence for iteration eL that

$$
\begin{aligned}
\tilde{f}_{eL}^b(x_{1:eL}) &= p(x_{1:eL}, y_{1:eL}) = p(x_{1:eL-1}, y_{1:eL-1})p(y_{eL}|x_{eL})p(x_{eL}|x_{eL-1}) \\
\tilde{f}_{eL-1}^b(x_{1:eL-1}) &= p(x_{1:eL-1}, y_{1:eL-1})\tilde{\psi}_{eL-1}^b(x_{eL-1}).
\end{aligned}
$$

We substitute these expressions into equation (A.1). Then we have

$$
\begin{aligned}
\tilde{f}_{eL-1}^b(x_{1:eL-1}) &= \int \tilde{f}_{eL}^b(x_{1:eL})dx_{eL} \\
\tilde{\psi}_{eL-1}^{b*}(x_{eL-1}) &= \int p(y_{eL}|x_{eL})p(x_{eL}|x_{eL-1})dx_{eL} \\
&= p(y_{eL}|x_{eL-1}).
\end{aligned}
$$

For the remaining iterations $t = eL-1, \dots, q+1$ we use the recursive structure in equation (A.2) and have that

$$
\tilde{\psi}_t^{b*}(x_t) = p(y_{t+1:eL}|x_t) \qquad \text{for } t = eL-1, \dots, q+1.
$$

By setting the standard definition of the normalising function from Definition 4.2.1 equal to the optimal $\tilde{\psi}_t^{b*}(x_t)$, that is $\tilde{\psi}_t(x_t) = \tilde{\psi}_t^{b*}(x_t)$ we have that

$$
\psi_t^{b*}(x_t) = p(y_{t:eL}|x_t) \qquad \text{for } t = eL, \dots, q+1.
$$

By continuing the recursive structure we also get

$$
\begin{aligned}
\tilde{\psi}_{0e}^{b*}(x_q) &= \int p(y_{q+1}|x_{q+1})p(x_{q+1}|x_q)\tilde{\psi}_{q+1}^{b*}(x_{q+1})dx_{q+1} \\
&= \int p(y_{q+1}|x_{q+1})p(x_{q+1}|x_q)p(y_{q+1:eL}|x_{q+1})dx_{q+1} \\
&= p(y_{q+1:eL}|x_q).
\end{aligned}
$$

APPENDIX B

Calculations for numerical examples

B.1 Alternative psi estimation for Poisson observations

Mode

We have that

$$\frac{\partial}{\partial x} p(y_t|x) = \frac{\partial}{\partial x} \frac{(e^x)^{y_t}}{y_t!} e^{-e^x} = \frac{1}{y_t!} e^{-e^x + xy_t}(y_t - e^x)$$

and

$$\begin{aligned} \frac{\partial}{\partial x} p(y_t|x) &= 0 \\ \frac{1}{y_t!} e^{-e^x + xy_t}(y_t - e^x) &= 0 \\ y_t e^{-e^x + xy_t} &= e^{-e^x + xy_t + x} \\ x &= \log y_t. \end{aligned}$$

Laplace approximation

We consider the derivatives needed for the function $f(x_t) = \log p(y_t|x_t)$

$$\begin{aligned} f^{(0)}(x_t) &= y_t x_t - e^{x_t} - \log y_t! \\ f^{(1)}(x_t) &= y_t - e^{x_t} \\ f^{(2)}(x_t) &= -e^{x_t}. \end{aligned}$$

Bibliography

Ala-Luhtala, J. et al. (2016). “An Introduction to Twisted Particle Filters and Parameter Estimation in Non-Linear State-Space Models”. eng. In: *IEEE transactions on signal processing* vol. 64, no. 18, pp. 4875–4890.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). “Particle Markov chain Monte Carlo methods”. eng. In: *Journal of the Royal Statistical Society. Series B, Statistical Methodology* vol. 72, no. 3, pp. 269–342.

Brewer, M. J. (2000). “A Bayesian model for local smoothing in kernel density estimation”. eng. In: *Statistics and Computing* vol. 10, no. 4, pp. 299–309.

Briers, M., Doucet, A., and Maskell, S. (2010). “Smoothing algorithms for state–space models”. eng. In: *Annals of the Institute of Statistical Mathematics* vol. 62, no. 1, pp. 61–89.

Cappé, O., Moulines, E., and Rydén, T. (2005). *Inference in Hidden Markov Models*. eng. First Edition. Springer Series in Statistics. New York, NY: Springer New York.

Chopin, N. (2004). “Central Limit Theorem for Sequential Monte Carlo Methods and Its Application to Bayesian Inference”. eng. In: *The Annals of statistics* vol. 32, no. 6, pp. 2385–2411.

Chopin, N. and Papaspiliopoulos, O. (2020). *An Introduction to Sequential Monte Carlo*. eng. Springer Series in Statistics. Cham: Springer International Publishing AG.

Creal, D. (2012). “A Survey of Sequential Monte Carlo Methods for Economics and Finance”. eng. In: *Econometric Reviews* vol. 31, no. 3, pp. 245–296.

Crisan, D. and Miguez, J. (2014). “Particle-kernel estimation of the filter density in state-space models”. eng. In: *Bernoulli : official journal of the Bernoulli Society for Mathematical Statistics and Probability* vol. 20, no. 4, pp. 1879–1929.

Devore, J. L. and Berk, K. N. (2012). *Modern Mathematical Statistics with Applications*. eng. Second Edition. Springer Texts in Statistics. New York, NY: Springer New York.

Douc, R. et al. (2011). “SEQUENTIAL MONTE CARLO SMOOTHING FOR GENERAL STATE SPACE HIDDEN MARKOV MODELS”. eng. In: *The Annals of Applied Probability* vol. 21, no. 6, pp. 2109–2145.

Doucet, A., Freitas, N. d., and Gordon, N. (2001). “An Introduction to Sequential Monte Carlo Methods”. eng. In: *Sequential Monte Carlo Methods in Practice*. Ed. by Doucet, A., Freitas, N. d., and Gordon, N. Information Science and Statistics. New York, NY: Springer New York, pp. 3–14.

Doucet, A., Godsill, S., and Andrieu, C. (2000). “On sequential Monte Carlo sampling methods for Bayesian filtering”. eng. In: *Statistics and Computing* vol. 10, no. 3, pp. 197–208.

Fearnhead, P., Wyncoll, D., and Tawn, J. (2010). “A sequential smoothing algorithm with linear computational cost”. eng. In: *Biometrika*. Biometrika vol. 97, no. 2, pp. 447–464.

Gelman, A. et al. (2013). *Bayesian Data Analysis*. eng. Third Edition. CRC Press.

Givens, G. H. and Hoeting, J. A. (2013). *Computational Statistics*. eng. Second Edition. Wiley Series in Computational Statistics. Hoboken, N.J: Wiley.

Guarniero, P., Johansen, A. M., and Lee, A. (2017). “The Iterated Auxiliary Particle Filter”. eng. In: *Journal of the American Statistical Association* vol. 112, no. 520, pp. 1636–1647.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. eng. Second Edition. Springer Series in Statistics. New York: Springer.

Heng, J. et al. (2020). “Controlled sequential Monte Carlo”. eng. In: *The Annals of Statistics* vol. 48, no. 5, pp. 2904–2929.

Johansen, A. M. and Doucet, A. (2008). “A note on auxiliary particle filters”. eng. In: *Statistics & Probability Letters* vol. 78, no. 12, pp. 1498–1504.

Kantas, N. et al. (2015). “On Particle Methods for Parameter Estimation in State-Space Models”. eng. In: *Statistical Science* vol. 30, no. 3, pp. 328–351.

Kitagawa, G. (1987). “Non-Gaussian State-Space Modeling of Nonstationary Time Series”. eng. In: *Journal of the American Statistical Association* vol. 82, no. 400, pp. 1032–1041.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. eng. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press.

Koller, D. and Lerner, U. (2001). “Sampling in Factored Dynamic Systems”. eng. In: *Sequential Monte Carlo Methods in Practice*. Ed. by Doucet, A., Freitas, N. d., and Gordon, N. Information Science and Statistics. New York, NY: Springer New York, pp. 445–464.

Kong, A., Liu, J. S., and Wong, W. H. (1994). “Sequential Imputations and Bayesian Missing Data Problems”. eng. In: *Journal of the American Statistical Association* vol. 89, no. 425, pp. 278–288.

Lin, M., Chen, R., and Liu, J. S. (2013). “Lookahead Strategies for Sequential Monte Carlo”. eng. In: *Statistical Science* vol. 28, no. 1, pp. 69–94.

Lindsten, F., Helske, J., and Vihola, M. (2018). “Graphical model inference: Sequential Monte Carlo meets deterministic approximations”. eng. In: *Advances in Neural Information Processing Systems*. Ed. by Bengio, S. et al. Vol. 31.

Liu, J. and West, M. (2001). “Combined Parameter and State Estimation in Simulation-Based Filtering”. eng. In: *Sequential Monte Carlo Methods in Practice*. Ed. by Doucet, A., Freitas, N. d., and Gordon, N. Information Science and Statistics. New York, NY: Springer New York, pp. 197–223.

Luethi, D. et al. (2021). *FKF: Fast Kalman Filter*. R package version 0.2.2.

Lutz, M. (2014). *Python pocket reference*. eng. Sebastopol, California.

Metropolis, N. and Ulam, S. (1949). “The Monte Carlo Method”. eng. In: *Journal of the American Statistical Association* vol. 44, no. 247, pp. 335–341.

Naesseth, C. A., Lindsten, F., and Schön, T. B. (2019). “Elements of Sequential Monte Carlo”. eng. In: *FOUNDATIONS AND TRENDS IN MACHINE LEARNING* vol. 12, no. 3, pp. 307–392.

Pitt, M. K. and Shephard, N. (1999). “Filtering via Simulation: Auxiliary Particle Filters”. eng. In: *Journal of the American Statistical Association* vol. 94, no. 446, pp. 590–599.

R Core Team (2021). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria.

Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods.* eng. Second Edition. Springer Texts in Statistics. New York, NY: Springer New York.

Rue, H., Martino, S., and Chopin, N. (2009). “Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations”. eng. In: *Journal of the Royal Statistical Society. Series B, Statistical Methodology* vol. 71, no. 2, pp. 319–392.

Skare, Ø., Bølviken, E., and Holden, L. (2003). “Improved Sampling-Importance Resampling and Reduced Bias Importance Sampling”. eng. In: *Scandinavian Journal of Statistics.* Scandinavian Journal of Statistics vol. 30, no. 4, pp. 719–737.

Spall, J. C. (2003). *Introduction to stochastic search and optimization: estimation, simulation, and control.* eng. Wiley-interscience series in discrete mathematics and optimization. Hoboken, N.J: Wiley.

Wickham, H. (2016). *ggplot2: elegant graphics for data analysis.* eng. 2nd ed. 2016. Use R! Cham: Springer International Publishing.