Efficient Online Smoothing in General Hidden Markov Models

Johan Westerborn
Lund University

ABSTRACT

This thesis discusses the problem of estimating smoothed expectations of sums of additive functionals of sequences of hidden states in general hidden Markov models. To compute expectations of this sort, the smoothing distribution, i.e. the conditional distribution of the hidden states given the corresponding observations, needs to be approximated. This thesis proposes a new algorithm to achieve this in an efficient way. The method is based on an algorithm proposed by Del Moral et al. (2009) and contains as a key ingredient an accept-reject sampling step or a Metropolis-Hastings algorithm (depending on wether the transition density of the state sequence is bounded or not).

The proposed algorithm, which replaces certain expectations occurring in the original algorithm by Monte Carlo estimates, allows for online computation of these expectations. When a single Monte Carlo draw is used in these estimates a degeneracy problem occurs, which can be avoided completely by simply using two or more draws. The new algorithm is tested on three different models: a linear and Gaussian state-space model, a stochastic volatility model, and a model with non-bounded transition density. For the same input the new algorithm produces an output similar to the method proposed by Del Moral et al. (2009), but at a fraction of the computation time.

CONTENTS

ACKNOWLEDGEMENTS

I would like to thank my supervisor Jimmy Olsson for taking his time to explain and giving me this interesting problem. Without his help I would never have got this far.

I would also like to thank Vilhelm von Ehrenheim for his insight and helpful comments when discussing the problem and Emil Fredriksson for help with the report.

Lastly I want to thank my wonderful girlfriend Therese Alenlöv for being very supportive and understanding during this time.

NOTATIONS AND ABBREVIATIONS

$\mathbb{E}[\cdot]$	the statistical expectation
$\mathbb{V}[\cdot]$	the statistical variance
$\phi_{0:n\|n}$	the smoothing distribution
$\phi_{k\|k}$	the filtering distribution
$a \wedge b$	minimum of a and b
$\langle \cdot, \cdot \rangle$	Scalar product
$f(x\|y)$	The conditional probability density function of x given y
$X \sim f$	X is random variable with distribution f
$X \sim (a_i)_{i=1}^N$	X is a discrete random variable with probability function $\mathbb{P}(X = i) = a_i$

FoS	Forward implementation of FFBS
FoSwS	Forward implementation of FFBS with sampling
HMM	Hidden Markov Model
i.i.d.	Independent and identically distributed
MC	Monte Carlo
MH	Metropolis-Hastings
SIS	sequential importance sampling
SMC	sequential Monte Carlo

Chapter 1

INTRODUCTION

Hidden Markov models (HMMs) form a very popular and diverse group of models. They are used a lot in modeling in many different areas, some examples include, biological sequences (Churchill, 1992), speech recognition (Rabiner, 1989), and modeling daily returns from a stock (Hull and White, 1987). These are just a few examples of situations where modeling using a HMM is useful. For more examples of modeling with HMMs, Chapter 1 in Cappé et al. (2005) provides a good overview.

The word hidden in hidden Markov models describes that something is unknown, loosely speaking, a HMM can be described as a *Markov chain* being observed through noise. This also relates to the problems that are encountered when working with HMMs. The observed process is given but what is wanted is some function of the hidden process or estimation of the parameters in the model.

For instance when working with the daily returns from a stock it can be observed that volatility clustering is common, that is the volatility of the process is not constant but changes over time. To incorporate this into the model of the daily returns a hidden process is introduced. This hidden process is then observed through the volatility of the observation. For more detail on this model see Section 4.2.

Once the model is finalized parameter estimation can be done given the observed data. The most common method for estimating parameters in HMMs is the *expectation-maximization* (EM) algorithm (Baum et al., 1970). It turns out that this algorithm relies on the computations of the conditional distribution of the hidden process given all the values of the observed process. This distribution is called the *smoothing distribution*.

Estimating the smoothing distribution is a well-studied problem and there are many different solutions proposed throughout the literature. For the most simple models, the linear and Gaussian models described in Example 2.4, exact calculations of the smoothed distribution is possible using the *disturbance smoother* (see Cappé et al., 2005, Algorithm 5.2.15). Looking at more advanced models for which exact calculations are not possible *sequential Monte Carlo* (SMC) methods can be used. The idea behind SMC methods is to have something called particles, every particle can be thought of as an estimate of the hidden state. For each time point the particles are given a weight depending on the observed values, then the particles

that were given bad (small) weights are removed at the same time the particles with good (large) weights are multiplied. Once this selection step is through the particles are mutated to the next time step. These particles together with the weight at a specific time point are then used to estimate something called the *filtering distribution*. The filtering distribution is the conditional distribution of the hidden state at some time point given the observed values up to that time point.

One naive method for estimation of the smoothing distribution is looking at the trajectories that are given from the particles in the SMC method when estimation is done for the filtering distribution at the last time point. A problem with this method of estimating the smoothing-distribution is the selection step in the SMC methods. If two particles share the same ancestor their trajectories will collapse to one thus causing degeneration in the estimate. After a few time lags all the different trajectories will collapse to one single trajectory. A solution to this problem is the *fixed-lag smoothing* where the smoothing distribution is estimated using the trajectories from the filtering distribution a few time points ahead (see Olsson et al., 2008).

Another method is the *forward filtering backward smoothing* algorithm (Godsill et al., 2004). Here the idea is to first estimate the filtering distributions and then backwards through time estimate the smoothing distributions. This method is robust but the computational time needed increases greatly when increasing the number of particles forming the estimator. This problem was addressed by Douc et al. (2011) who manages to significantly lower the computational time. Compared to the fixed-lag smoothing this algorithm requires computations in both the forward and backward direction preventing an online implementation.

Another more resent method is the *forward implementation of FFBS* (FoS) presented by: Del Moral et al. (2009). This algorithm estimates expectations of a sum of additive functionals. These are typically the same kind of expectations that needs to be calculated in the EM-algorithm when working with HMMs.

The contribution of this thesis is the proposal of a new algorithm, *forward implementation of FFBS with sampling* (FoSwS), for computing smoothed sums of additive functionals in an online setting. This new method resembles closely the FoS algorithm but adds a sampling step to make the algorithm computationally more efficient at the cost of some additional variance in the estimates.

The idea with the algorithm is to use SMC methods to estimate the filtering distribution and for each particle calculate the mean of a function on all previous values. Instead of calculating this mean exactly *Monte Carlo methods* are used to estimate it. If only one particle is used in the expectation estimation a degeneracy problem arises similar to the naive method. This results in that the variance grows quadratically if only one particle is used. If two particles were used in the expectation estimation the variance of the estimates would only grow linearly.

Comparing FoS and FoSwS the given output is almost identical while the computational time needed for the FoSwS is much lower resulting in a more efficient algorithm.

1.1 Outline of the thesis

In this thesis Chapter 2 focuses on some of the theory needed for the rest of thesis. A brief overview on Markov chains, hidden Markov models, Monte Carlo methods and sampling methods is presented. Section 3.1 presents the forward filtering backward smoothing algorithm. In Section 3.2 the forward only smoothing algorithm is presented together with some discussion on the method. In Section 3.3 usage of *accept-reject* sampling in FoS is discussed. FoSwS is presented in Section 3.4. Chapter 4 is devoted to simulation studies using both of the methods for three different models. In Chapter 5 the thesis is summarized and the conclusions are presented.

Chapter 2

THEORY

This chapter is not a complete guide to the theory needed in this thesis but it is a good start. This chapter assumes basic knowledge in probability theory.

2.1 Markov Chain

A stochastic process is a family of random variables taking values on some space, this can be thought of as some function that generates random variables according to some rule. In this thesis I will use the notation $\{X_k\}$ for stochastic processes, the subindex k will in this thesis always belong to the non-negative integers $(0, 1, 2, 3, \ldots)$ and will also be called the time or the time point of the process.

The easiest way to describe a Markov chain is to think about it as a memoryless stochastic process. That the process is memoryless loosely means that the random evolution of the process in the future depends only on the current value.

A Markov chain can take its values either on a discrete or an uncountable set (or a combination of the two). An easy example of a Markov chain is the sum of dice rolls, i.e. you start at 0 and then repeatedly roll a dice and add the number of eyes to your sum every time. A Markov chain can be formally defined as follows.

Definition 2.1 (Markov chain)**.** *A stochastic process $\{X_n\}_{n\geq 0}$ taking values on some set $X \subseteq \mathbb{R}^d$ is a Markov chain if for all events $A \subseteq X$*

$$\mathbb{P}(X_{k+1} \in A | X_k = x_k, \ldots, X_0 = x_0) = \mathbb{P}(X_{k+1} \in A | X_k = x_k). \tag{2.1.1}$$

We also define the transition density $q_k(x', \cdot)$ such that

$$\mathbb{P}(X_{k+1} \in A | X_k = x_k) = \int_A q_k(x_k, x_{k+1}) \mathrm{d}x_{k+1}. \tag{2.1.2}$$

The process is called time homogeneous if the transition densities do not depend on k. For time homogeneous Markov chains we use the notation $q(x', \cdot)$ for the transition density.

The distribution of X_0 is called the initial distribution and is usually denoted by χ.

Equation (2.1.1) is called the Markov property. Equation (2.1.2) implies that given a state X_k the next value X_{k+1} is a random variable with density $q_k(x_k, \cdot)$, that is

$$X_{k+1}|X_k = x_{k+1} \sim q_k(x_k, \cdot).$$

We illustrate this definition with an example.

Example 2.2 (AR(1) process)**.** *An AR(1) process is defined by*

$$X_{k+1} = a + bX_k + V_{k+1} \tag{2.1.3}$$

where a, b are constants and $\{V_n\}_{n\geq 0}$ is a sequence of i.i.d. (independent and identically distributed) variables having a density f. The process $\{X_n\}_{n\geq 0}$ is a Markov chain since

$$\begin{aligned}
&\mathbb{P}(X_{k+1} \leq x_{k+1}|X_k = x_k, X_{k-1} = x_{k-1}, \dots X_0 = x_0) = \\
&\mathbb{P}(X_{k+1} \leq x_{k+1}|X_k = x_k) = &(2.1.4)\\
&\mathbb{P}(a + b \cdot X_k + V_{k+1} \leq x_{k+1}|X_k = x_k) = \\
&\mathbb{P}(V_{k+1} \leq x_{k+1} - a - b \cdot x_k), &(2.1.5)
\end{aligned}$$

where (2.1.4) *holds since the value of X_{k+1} does only depend on X_k according to* (2.1.3)*. Equation* (2.1.5) *implies that*

$$q_k(x_k, x_{k+1}) = f(x_{k+1} - a - b \cdot x_k).$$

It is clear that this is a time homogeneous process, since the distribution of X_{k+1} only depends on the current values of X_k it does not depend on the choice of k.

An interesting question that arises when studying Markov chains is if there is any distribution of X_k that will give the same distribution of X_{k+1}. Such a distribution is called a *stationary distribution.*

Definition 2.3 (Stationary distribution)**.** *If there is a distribution π such that*

$$\int \pi(x)q(x, z)\mathrm{d}x = \pi(z) \tag{2.1.6}$$

then we say that π is the stationary distribution to the Markov chain.

Equation (2.1.6) is also called the *global balance equation.* This implies that if X_k is distributed as π then X_{k+1} will also be distributed as π. So if a Markov chain reaches its stationary distribution it will stay there, but every Markov chain does not have a stationary distribution.

An interesting distribution is the distribution of all the values of a Markov chain, this distribution is called the joint density. The joint density $f_n(x_0, x_1, \dots, x_n)$ of $\{X_k\}_{k=0}^n$ with initial distribution χ and transition kernel q_k is

$$f_n(x_0, x_1, \dots, x_n) = \chi(x_0) \prod_{k=0}^{n-1} q_k(x_k, x_{k+1}). \tag{2.1.7}$$

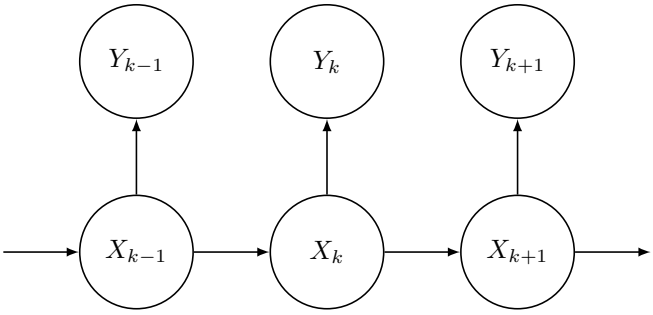

Figure 2.1: Hidden Markov model

Using this we can see that the n-step transition distribution has the density

$$f_n(x_n|x_0) = \int \dots \int \prod_{k=0}^{n-1} q_k(x_k, x_{k+1}) \mathrm{d}x_{n-1} \dots \mathrm{d}x_1$$

2.2 Hidden Markov Models

A hidden Markov model (HMM) can be seen as a model where we observe a Markov chain through another process. More formally, a HMM is a bivariate process $\{X_n, Y_n\}_{n\geq 0}$, where $\{X_n\}_{n\geq 0}$ is a Markov chain and $\{Y_n\}_{n\geq 0}$ is conditionally on $\{X_n\}_{n\geq 0}$ a sequence of independent variables such that the conditional distribution of Y_k only depends on X_k. This model can be viewed using the picture in Figure 2.1. Usually we are interested in the process $\{X_n\}_{n\geq 0}$ in some way but we can only observe the process $\{Y_n\}_{n\geq 0}$.

A general HMM can be described by the following equations

$$X_0 \sim \chi(\cdot), \tag{2.2.1}$$

$$X_k|X_{k-1} = x_{k-1} \sim q(x_{k-1}, \cdot), \qquad k \geq 1 \tag{2.2.2}$$

$$Y_k|X_k = x_k \sim g(x_k, \cdot). \qquad k \geq 0 \tag{2.2.3}$$

Here χ is the initial distribution, q is the transition density for the Markov chain $\{X_k\}_{k\geq 0}$ and g is the density from X_k to Y_k. These equations will now be illustrated in a small example

Example 2.4 (Linear Gaussian state-space model)**.** *The linear Gaussian state-space model is a simple HMM, the equations determining the system are the following*

$$\begin{aligned} X_k &= A \cdot X_{k-1} + R \cdot U_k, \\ Y_k &= B \cdot X_k + S \cdot V_k, \end{aligned}$$

where V_k and U_k are two sequences of i.i.d. standard Gaussian white noise. The initial distribution of X_0 is Gaussian with mean μ_ν and covariance matrix Γ_ν. A, B, R, and S are matrices of suitable sizes.

Comparing this to the Equations (2.2.1)–(2.2.3) *we find that*

$$\begin{aligned} X_0 &\sim \mathcal{N}(\mu_\nu, \Gamma_\nu) \\ X_{k+1}|X_k = x_k &\sim \mathcal{N}(A \cdot x_k, R^2) \\ Y_k|X_k = x_k &\sim \mathcal{N}(B \cdot x_k, S^2). \end{aligned}$$

That is, everything is drawn from normal distribution with the parameters specified in the model. A realization of this model with parameters $A = 0.7$, $R = 0.2$, $B = 1$, and $S = 1$, can be seen in Figure 2.2

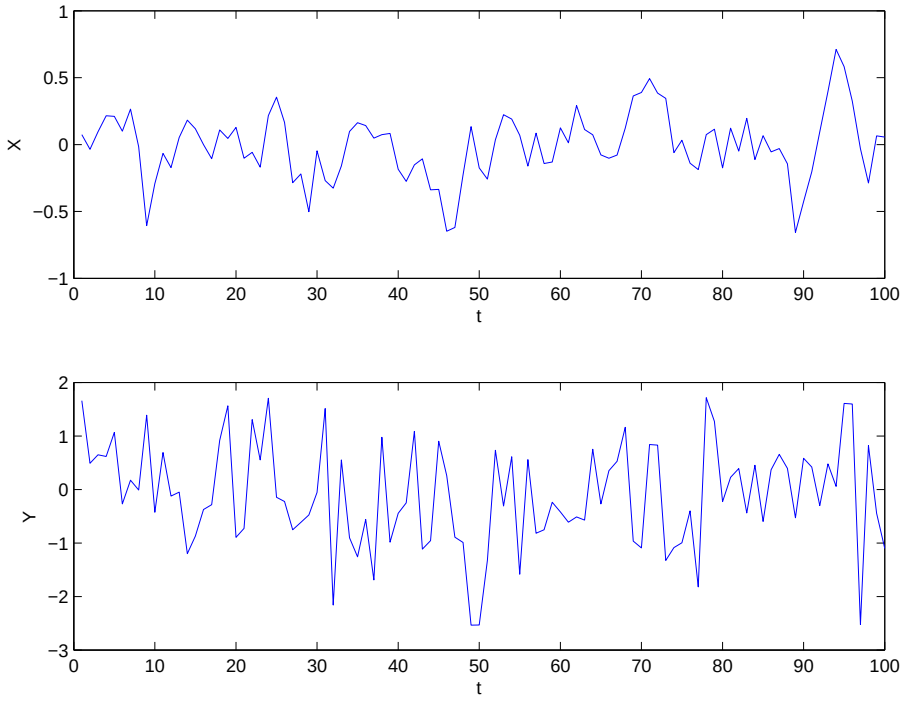

Figure 2.2: $\{X_k\}_{k=1}^{100}$ and $\{Y_k\}_{k=1}^{100}$ generated using the linear and Gaussian state-space model in Example 2.4 with parameters $A = 0.7$, $R = 0.2$, $B = 1$, and $S = 1$.

We denote by $x_{i:j}$ $(i \leq j)$ the set $\{x_i, x_{i+1}, \ldots, x_j\}$. The joint distribution of

$\{X_k, Y_k\}_{k=0}^n$ is defined by Equations (2.2.1)–(2.2.3) as

$$f(x_{0:n}, y_{0:n}) = \chi(x_0)g(x_0, y_0)\prod_{k=1}^{n} q(x_{k-1}, x_k)g(x_k, y_k). \tag{2.2.4}$$

In a large part of this paper will focus on the conditional distribution of some or all of the $\{X_k\}_{k=0}^n$ given all of the $\{Y_k\}_{k=0}^n$. These distributions are called the smoothing distributions, and the distribution of all $\{X_k\}_{k=0}^n$ given $\{Y_k\}_{k=0}^n$ is given by

$$f(x_{0:n}|y_{0:n}) = \frac{f(x_{0:n}, y_{0:n})}{f(y_{0:n})}. \tag{2.2.5}$$

Here we have that $f(x_{0:n}, y_{0:n})$ is the joint density given by Equation (2.2.4), and $f(y_{0:n})$ is then

$$f(y_{0:n}) = \int \dots \int f(x_{0:n}, y_{0:n})\mathrm{d}x_n \dots \mathrm{d}x_0. \tag{2.2.6}$$

The smoothing distribution is given its own notation as follows

$$\phi_{0:n|n}(x_{0:n}) = f(x_{0:n}|y_{0:n}) = \frac{f(x_{0:n}, y_{0:n})}{\mathcal{L}(y_{0:n})} \tag{2.2.7}$$

where $\mathcal{L}(y_{0:n})$ is called the *likelihood* and is defined by

$$\mathcal{L}(y_{0:n}) = f(y_{0:n}). \tag{2.2.8}$$

We will denote the filtering distribution, that is the conditional distribution of X_k given $Y_{0:k}$, by $\phi_{k|k}$.

2.2.1 The Smoothing Distribution and its Applications

The smoothing distribution (see Equation (2.2.7)) is important in many cases when it comes to hidden Markov models, especially if we want to calculate expectations on the form

$$\mathbb{E}\left[h(X_{0:n})|Y_{0:n}\right]. \tag{2.2.9}$$

These expectations are for instance used in the expectation-maximization algorithm for estimating parameters (more on that later). In the development that follow, focus will be set on sums of *smoothed additive functionals*,

$$\mathcal{S}_n = \mathbb{E}[S_n(X_{0:n})|Y_{0:n}], \tag{2.2.10}$$

where

$$S_n(X_{0:n}) = \sum_{k=1}^{n} s_k(x_{k-1}, x_k). \tag{2.2.11}$$

The functions $s_k(x_{k-1}, x_k)$ can be of many forms, a simple example is when $s_1(x_0, x_1) = x_0 + x_1$ and $s_k(x_{k-1}, x_k) = x_k$ for $k \geq 2$. This will result in the sum of the smoothed X values. This sum will be used later to validate the algorithms in Section 4.1.1

2.2.2 Expectation-Maximization Algorithm

One of the most common methods of estimating parameters in any statistical model is the *maximum likelihood estimator* (MLE). To perform the MLE in a HMM the likelihood defined by (2.2.8) is used. In this section a θ will be added to the notation to show that they depend on some parameters. The MLE of the parameters is then

$$\theta_{\text{MLE}} = \arg\max_{\theta} \mathcal{L}(y_{0:n}; \theta). \tag{2.2.12}$$

The likelihood is a positive function so the argument that maximizes the likelihood is the same argument that maximizes the *log-likelihood*

$$\ell(y_{0:n}; \theta) = \log(\mathcal{L}(y_{0:n}; \theta)). \tag{2.2.13}$$

To maximize the log-likelihood a new quantity $\mathcal{Q}(\theta; \theta')$ is introduced by

$$\mathcal{Q}(\theta; \theta') = \mathbb{E}_{\theta'}[\log(f(x, y; \theta))|Y_{0:n}], \tag{2.2.14}$$

$$= \int \log(f(x, y; \theta))\phi_{0:n|n}(x_{0:n}; \theta')\mathrm{d}x_{0:n}. \tag{2.2.15}$$

To maximize the log-likelihood is the same as maximizing $\mathcal{Q}(\theta; \theta')$. To perform the EM-algorithm the following two steps are iterated until the method converges

1: Determine $\mathcal{Q}(\theta; \theta^i)$.
2: Choose θ^{i+1} to be the value that maximizes $\mathcal{Q}(\theta; \theta^i)$ with respect to θ.

The initial value θ^0 is set to an arbitrary value. The EM-algorithm as described above will only be useful when it is possible to compute $\mathcal{Q}(\theta; \theta')$ given a value of θ' reasonably fast and the maximization can be done in closed form. These two requirements are satisfied when the density $f(\cdot; \theta)$ belong to an exponential family.

Definition 2.5 (Exponential Family)**.** *The family $\{f(\cdot; \theta)\}_{\theta \in \Theta}$ defines an exponential family if each member is on the form*

$$f(x; \theta) = \exp\{\langle \psi(\theta), S(x) \rangle - c(\theta)\}h(x), \tag{2.2.16}$$

where S and ψ are vector-valued functions, c is a real-valued function and h is a non-negative real-valued function. $\langle \cdot, \cdot \rangle$ is the scalar product.

$S(x)$ is known as the *sufficient statistics* and $\eta = \psi(\theta)$ is known as the *natural parameterization*. Most distributions we encounter belong to the exponential family. Here follows an example with the normal distribution.

Example 2.6 (Normal Distribution)**.** *Let X be distributed according to a normal distribution with mean μ and variance σ^2 i.e. the probability density function is given by*

$$f(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}.$$

This can be written on exponential family form (see Equation 2.2.16) using

$$\begin{aligned}\psi(\mu,\sigma^2) &= \left(-\frac{1}{2\sigma^2}, \frac{\mu}{\sigma^2}\right)\\ S(x) &= (x^2, x)\\ c(\mu,\sigma^2) &= \frac{\mu^2}{2\sigma^2} + \frac{1}{2}\log(\sigma^2)\\ h(x) &= \frac{1}{\sqrt{2\pi}}.\end{aligned}$$

When applying the EM algorithm to an exponential family distribution the algorithm reduces down to the algorithm presented in Algorithm 2.1.

Algorithm 2.1 Expectation-maximization in HMM with exponential family distribution

Require: Measurements $\{Y_k\}_{k\geq 0}$ from a HMM where the joint density belongs to an exponential family.

1: Set θ^0 arbitrary.
2: **for** $i \leftarrow 0$ until convergence **do**
3: Simulate $\mathcal{S}_t = \mathbb{E}[S_t(X)|Y_{0:t}]$ where $S_t(X)$ is the sufficient statistics.
4: Find θ^{i+1} that maximizes $\mathcal{Q}(\theta;\theta^i) = \langle\psi(\theta), \mathcal{S}_t\rangle - c(\theta)$
5: **end for**

2.3 Monte Carlo Methods

Say that we want to estimate en expectation $\tau(h) = \mathbb{E}[h(X)]$ for some random variable X having density function f. The expectation can be calculated using the integral

$$\mathbb{E}[h(X)] = \int h(x)f(x)\mathrm{d}x.$$

This integral can be quite complex and lacks a closed-form expression in general. One way of approximating this integral would be to use results from numerical analysis, but we can also note that by using a law of large numbers (see Gut, 2009, Section 6.5 for details) (given that our distribution has at least one finite moment) we may approximate the expectation as

$$\mathbb{E}[h(X)] \approx N^{-1}\sum_{i=1}^{N} h(\xi^i) \tag{2.3.1}$$

where the $\{\xi^i\}_{i=1}^N$ are an i.i.d. sample from the distribution of X. This estimator can be proved to be the quantity of interest. Also, if the function h is square integrable

and the distribution has at least two finite moments, we can get an estimation of the error using the central limit theorem.

The approximation (2.3.1) is referred to as the Monte Carlo (MC) method. The MC method can be used to calculate arbitrary integrals by realizing that

$$\int h(x)\mathrm{d}x = \int \frac{h(x)}{f(x)} f(x)\mathrm{d}x = \mathbb{E}\left[\frac{h(X)}{f(X)}\right],$$

where f is a density such that $f(x) = 0 \Rightarrow h(x) = 0$. So instead of calculating the integral we can draw random numbers ξ^i from the density f and then take the mean of the transformed variables $h(\xi^i)/f(\xi^i)$.

Comparing the convergence of MC methods with numerical analysis the later suffers from the *curse of dimensionality*. The curse of dimensionality is that as the dimensions grows the number of evaluations for a given precision grows exponentially. The MC methods does not suffer from this since the central limit theorem implies that the convergence rate is proportional to $\sqrt{N}$ independent of the dimension.

A lot of the Monte Carlo methods presented below are related to the problem of drawing variables from a distribution.

2.3.1 Accept-Reject Sampling

There are many ways of drawing i.i.d. random variables. The simplest method is the inverse method, which required the inverse of the cumulative density function to be known in closed form. Another method is accept-reject sampling where the main idea is to sample from a distribution that is easier to sample from and then accept this sample with some probability.

Given that we want to sample from a distribution with density function f we need to find a constant $K < \infty$ and another density function g, called the proposal density, that is easy to sample from such that $f(x) \leq Kg(x)$ for every x. To draw one sample from f we start by drawing a candidate x from g and accepting it with the probability $f(x)/(Kg(x))$, if that candidate is rejected a new candidate is drawn from g until acceptance. The accepted candidates from the accept-reject method will form an i.i.d. sample from the distribution with density f.

We can also use accept-reject sampling when we only know the density f up to a normalizing constant c, that is $f(x) = z(x)/c$, by first choosing K' such that $z(x) \leq K'g(x)$ and then taking $K = K'/c$. As the unknown constant then appears in the numerator and denominator of the acceptance probability ratio and thus cancels out, we still obtain a practically valid algorithm. The algorithm for accept-reject sampling can be seen in Algorithm 2.2. For a proof that the output has the correct distribution and some more discussions on accept-reject sampling see Cappé et al. (2005, Section 6.2.1)

Algorithm 2.2 Accept-Reject step

Require: A target density $f(x)$ known up to a normalizing constant c ($f(x) = z(x)/c$), a density $g(x)$ that is easy to sample from, and a constant K' such that $z(x) \leq K'g(x)$.

1: **repeat**
2: Draw $x \sim g(\cdot)$
3: Draw $u \sim \mathcal{U}(0,1)$
4: **until** $u \leq z(x)/(g(x)K')$ **return** x

2.3.2 Sequential Monte Carlo

A problem that arises when working with HMMs and other kind of models is to estimate sequences $\{\tau_n\}_{n\geq 0}$ of expectations on the form

$$\tau_n(h) = \mathbb{E}_{f_n}\left[h(X_{0:n})\right] = \int h(x_{0:n}) f_n(x_{0:n}) \mathrm{d}x_{0:n}. \tag{2.3.2}$$

As n increases the dimension of this problem increases. This could of course be estimated using standard MC methods but as the dimension increases drawing from the distributions that is needed will become harder. Instead an idea is to use the estimates of τ_n when estimating τ_{n+1} this idea is called *sequential Monte Carlo* (SMC).

SMC methods works with sequentially in n updating N pairs $\{\xi^i_{0:n}, \omega^i_n\}_{i=1}^N$ of particles and weights that estimates $f_n(x_{0:n})$. The particles $\{\xi_{0:n}\}_{i=1}^N$ are generated using an instrumental distribution $r_n(x_{0:n})$, the weight ω^i_n of particle i at time n is calculated using a function $\omega_n(\xi^i_{0:n}) = f_n(\xi^i_{0:n})/r_n(\xi^i_{0:n})$ this gives an estimate of the expectation

$$\mathbb{E}_{f_n}\left[h(X_{0:n})\right] = \sum_{i=1}^N \frac{\omega^i_n}{\Omega_n} h(\xi^i_{0:n}), \tag{2.3.3}$$

where $\Omega_n = \sum_{i=1}^N \omega^i_n$. These estimates should then be updated recursively from $n = 1$ and onwards with the requirements that.

- Having $\{\xi^i_{0:n}, \omega^i_n\}_{i=1}^N$, the complexity of generating $\{\xi^i_{0:n+1}, \omega^i_{n+1}\}_{i=1}^N$ does not increase with n.
- The approximation remains stable with n.

A key observation to formulate a SMC algorithm is to chose an instrumental distribution $r_n(x_{0:n})$ such that

$$r_{n+1}(x_{0:n+1}) = r_{n+1}(x_{n+1}|x_{0:n}) r_n(x_{0:n}). \tag{2.3.4}$$

An instrumental distribution on this form allows us to calculate the weights at $n+1$

as

$$\omega_{n+1}^i := \frac{f_{n+1}(\xi_{0:n+1}^i)}{r_{n+1}(\xi_{0:n+1}^i)} \tag{2.3.5}$$

$$= \frac{f_{n+1}(\xi_{0:n+1}^i)}{r_{n+1}(\xi_{n+1}^i|\xi_{0:n}^i)r_n(\xi_{0:n}^i)} \frac{f_n(\xi_{0:n}^i)}{f_n(\xi_{0:n}^i)} \tag{2.3.6}$$

$$= \frac{f_{n+1}(\xi_{0:n+1}^i)}{r_{n+1}(\xi_{n+1}^i|\xi_{0:n}^i)f_n(\xi_{0:n}^i)} \omega_n^i. \tag{2.3.7}$$

The algorithm can now be formulated and is found in Algorithm 2.3 and it is known as the *sequential importance sampling* (SIS) algorithm. A problem with this method is something called *weight degeneracy*, that is after some samples only a few particles will be active in the estimations. The problem here is that once a particle gets a small weight the next weight will also be small (since they are multiplied), the weights will continue to get smaller and smaller. This problem is fixed in the *sequential importance sampling with resampling* (SISR) algorithm, also known has the particle filter. In this thesis the name particle filter will be used.

Algorithm 2.3 Sequential importance sampling (SIS)

Require: Estimate $\{\xi_{0:n}^i, \omega_n^i\}_{i=1}^N$ of $f_n(x_{0:n})$
1: **for** $i = 1 \to N$ **do**
2: Simulate $\xi_{n+1}^i \sim r_n(\cdot|\xi_{0:n}^i)$
3: Set $\xi_{0:n+1}^i = \{\xi_{0:n}^i, \xi_{n+1}^i\}$
4: Set $\omega_{n+1}^i = \frac{f_{n+1}(\xi_{0:n+1}^i)}{r_{n+1}(\xi_{n+1}^i|\xi_{0:n}^i)f_n(\xi_{0:n}^i)} \cdot \omega_n^i$
5: **end for**
6: Repeat recursively.

The Particle Filter

The idea behind the particle filter is to fix the problem with the weight degeneracy that was experienced in the SIS algorithm. To fix this problem a resampling step is added where particles are redrawn according to their weights. This implies that for time step k, N indices I_k^i between 1 and N is drawn, with the probability that

$$\mathbb{P}(I_k^i = j) = \frac{\omega_{k-1}^j}{\Omega_{k-1}},$$

where $\Omega_{k-1} = \sum_{j=1}^N \omega_{k-1}^j$. If we chose to propagate the particles according to the transition density of the HMM we get that the weight function is equal to the observation density. This algorithm is known as the bootstrap particle filter and can be seen in Algorithm 2.4, here ρ_0 is an initial distribution of the particles. The bootstrap particle filter was proposed by Gordon et al. (1993).

Algorithm 2.4 Bootstrap Particle filter

for $i = 1 \to N$ **do**
    simulate $\xi_0^i \sim \rho_0$
    set $\omega_0^i \leftarrow g_0(\xi_0^i, y_0)$
end for
for $k = 0 \to n-1$ **do**
    for $i = 0 \to N$ **do**
        simulate indices $I_{k+1}^i \sim (\omega_k^l)_{l=1}^N$
        simulate $\xi_{k+1}^i \sim q_k(\xi_k^{I_{k+1}^i}, \cdot)$
        set $\omega_{k+1}^i \leftarrow g_{k+1}(\xi_{k+1}^i, y_{k+1})$
    end for
end for
return $(\xi_{0:n}^i, \omega_{0:n}^i)_{i=1}^N$

2.3.3 Markov Chain Monte Carlo Methods

Markov Chain Monte Carlo (MCMC) methods draw random values from a high-dimensional distribution by simulating a Markov chain having the target distribution as stationary distribution. Like SMC methods MCMC methods are not limited to HMM but we will use them in this setting. As the case with the accept-reject method the target distribution only needs to be known up to a normalizing constant.

To be able to carry through this in practice we require that the procedure satisfies the following assumptions.

- Simulating the chain $\{\xi^i\}_{i\geq 1}$ is easy given an arbitrary initial value ξ^1
- The chain $\{\xi^i\}_{i\geq 1}$ has the desired stationary distribution π
- The distribution of the chain $\{\xi^i\}_{i\geq 1}$ converges towards π independent of initial value ξ^1

We will now discuss a method to generate chains $\{\xi^i\}_{i\geq 1}$ satisfying these conditions. A general rule of thumb when working with MCMC is to use a burn-in period; that is, if one want N values for their simulations one simulate $N+b$ values and discard the first b values. This is done to give the Markov chain some time to reach its stationary distribution.

Metropolis Hastings

The *Metropolis-Hastings* (MH) algorithm is an algorithm that generates a chain $\{\xi^i\}_{i\geq 1}$ with the properties mentioned above. It works by generating candidates from some density $r(x', \cdot)$ and then accepting this next step with a probability α. The algorithm is described in Algorithm 2.5.

Algorithm 2.5 Metropolis-Hastings

set ξ^0 to an arbitrary value
for $i = 1 \to N + b$ **do**
  simulate $\xi \sim r(\xi^i, \cdot)$
  set $\alpha(\xi^i, \xi) \leftarrow \dfrac{\pi(\xi) r(\xi, \xi^i)}{\pi(\xi^i) r(\xi^i, \xi)} \wedge 1$
  simulate $u \sim \mathcal{U}(0, 1)$
  if $u \leq \alpha(\xi^i, \xi)$ **then**
    set $\xi^{i+1} \leftarrow \xi$
  else
    set $\xi^{i+1} \leftarrow \xi^i$
  end if
end for
return $\{\xi^i\}_{i=b+1}^{N+b}$

It can be shown that the chain generated by the MH algorithm has π as stationary distribution (see Cappé et al., 2005, Section 6.2.3, for details).

There is some freedom in choosing the proposal distribution. Some popular choices are so-called independent and random walk proposals. No matter what proposal distribution is chosen it only needs to be known up to a normalizing constant. An independent proposal does not depend on the current state, so the acceptance probability reduces to

$$\alpha(\xi^i, \xi) = \frac{\pi(\xi)/r_{\text{ind}}(\xi)}{\pi(\xi^i)/r_{\text{ind}}(\xi^i)} \wedge 1.$$

A random walk proposal is of form $r(\xi^i, \xi) = h(\xi - \xi^i)$, where h is a symmetric density. The acceptance probability now reduces to

$$\alpha(\xi^i, \xi) = \frac{\pi(\xi)}{\pi(\xi^i)} \wedge 1.$$

But no matter what proposal is chosen we need to make sure that we really visit the whole space of the chain. That is every possible value ξ of the target distribution must be possible to reach from the proposal distribution.

Chapter 3

CALCULATION OF THE SMOOTHING DISTRIBUTION

In this chapter the focus will be on three different algorithms to compute the smoothing distribution. First the forward filtering backward smoothing (FFBS) algorithm is discussed followed by a forward implementation of the FFBS (FoS) and lastly a forward implementation of the FFBS with sampling (FoSwS). The functions $f(x|y)$ is used to note the probability density function of x given y.

3.1 Forward Filtering Backward Smoothing

If the particle filter (Algotihm 2.4) is performed estimates of the filter distribution is given. A naive way of trying to find the smoothing distribution is to look at the trajectories of the final particles, the particles that estimates $f_n(x_n|y_{0:n})$, these trajectories will degenerate into only one trajectory after some time point $k < n$ if n is large enough (see Olsson et al., 2008, for more on that topic). The FFBS algorithm uses the following decomposition to estimate the smoothing distribution (Cappé et al., 2005, Proposition 3.3.9)

(i) Compute forward in time the filtering distribution $\phi_{k|k}$ for $k = 1, \ldots, n$.

(ii) From $\phi_{n|n}$, compute for $k = n-1, \ldots, 0$

$$\phi_{k|n}(x_k) = \int \phi_{k+1|n}(x_{k+1}) b_k(x_{k+1}, x_k) \mathrm{d}x_{k+1}, \tag{3.1.1}$$

where $b_k(x_{k+1}, x_k)$ is a Markov transition density called the backward density.

The backward density b_k is the distribution of X_k given the values $X_{k+1:n}$ and $Y_{0:n}$, since it is assumed that $\{X_k\}_{k=0}^n$ is a Markov chain this distribution reduces down to

$$b_k(x_{k+1}, x_k) = f(x_k|x_{k+1}, y_{0:n}). \tag{3.1.2}$$

Taking these theoretical results into the world of SMC the N particles and weights $\{\xi^i_{0:n}, \omega^i_{0:n}\}_{i=1}^N$ that are given from the particle filter will estimate the filter distributions $\phi_{k|k}$. Using this the backward distribution can be estimated by (see Godsill et al., 2004)

$$f(x_k|x_{k+1}, y_{0:n}) \approx \sum_{i=1}^{N} \omega^i_{k|k+1} \delta_{\xi^i_k}(x_k), \tag{3.1.3}$$

where the new weights are

$$\omega^i_{k|k+1} = \frac{\omega^i_k q(x_{k+1}, \xi^i_k)}{\sum_{j=1}^N \omega^j_k q(x_{k+1}, \xi^j_k)}. \tag{3.1.4}$$

Using these equations R random samples from $\phi_{0:n|n}$ can be drawn using Algorithm 3.1. In step 2 and 5 of the algorithm sums of N variables needs to be calculated (to normalize the distributions), this gives each sample the computational complexity $\mathcal{O}(Nn)$, if we want N samples from the smoothing distribution then the total computational complexity is $\mathcal{O}(N^2 n)$. A complexity on this form says that if we double the number of particles the computation takes four times longer. Since SMC methods relies on the use of large numbers of particles this results in an impractical algorithm.

A way of speeding up the algorithm is to change the draw of indices at step 5 to using accept-reject methods instead. Using this the computational complexity of drawing N samples from the smoothing distribution can be reduced to $\mathcal{O}(Nn)$ (see Douc et al., 2011). A similar method will be introduced in Section 3.3. This algorithm requires calculations in two directions, first the particle filter forward through time and then the smoothing backwards through time, preventing an online implementation.

Algorithm 3.1 SMC forward filtering backward smoothing

Require: Particles and weights $\{\xi^i_{0:n}, \omega^i_{0:n}\}_{i=1}^N$ estimating the filtering distributions.

1: **for** $r = 1 \to R$ **do**
2: Simulate an index $J_n \sim (\omega^i_n)_{i=1}^N$
3: Set $x^r_k \leftarrow \xi^{J_n}_n$
4: **for** $k = n-1 \to 0$ **do**
5: Simulate index $J_k \sim \left(\omega^i_k q(\xi^i_k, x^r_{k+1})\right)_{i=1}^N$
6: Set $x^r_k \leftarrow \xi^{J_k}_k$
7: **end for**
8: Set $x^r_{0:k} \leftarrow (x^r_0, \ldots, x^r_n)$
9: **end forreturn** $\{x^r_{0:n}\}_{r=1}^R$

3.2 Forward Implementation of FFBS

The forward implementation of FFBS (Del Moral et al., 2009) is a way of calculating smoothed sums of additive functionals of the sort introduced in Chapter 2.2.1. The approach here is to be able to compute these quantities using a forward only approach compared to the forward filtering backwards smoothing method. All densities and quantities depend on some parameter vector θ and this will be indicated using either a subscript or a superscript.

To summarize we want to calculate $\mathcal{S}_t^\theta$ using an online method. The notations used here will be the same as in Section 2.2.1 with addition of the dependence on the parameters.

To carry through forward implementation of FFBS of $\mathcal{S}_t^\theta$ we introduce an auxiliary function

$$T_t^\theta(x_t) := \int S_t(x_{0:t}) f_\theta(x_{0:t-1}|y_{0:t-1}, x_t) dx_{0:t-1},$$

and using this we get

$$\mathcal{S}_t^\theta = \int T_t^\theta(x_t) f_\theta(x_t|y_{0:t}) dx_t.$$

We can then update $T_t^\theta(x_t)$ recursively using

$$\begin{aligned}
T_t^\theta(x_t) &= \int S_t(x_{0:t}) f_\theta(x_{0:t-1}|y_{0:t-1}, x_t) dx_{0:t-1} \\
&= \int \left[S_{t-1}(x_{0:t-1}) + s_t(x_{t-1}, x_t)\right] f_\theta(x_{0:t-1}|y_{0:t-1}, x_t) dx_{0:t-1} \\
&= \int \left[\int S_{t-1}(x_{0:t-1}) f_\theta(x_{0:t-2}|y_{0:n-2}, x_{t-1}) \mathrm{d}x_{0:n-2}\right] f_\theta(x_{t-1}|y_{0:t-1}, x_t) \mathrm{d}x_{t-1} \\
&+ \int s_t(x_{t-1}, x_t) f_\theta(x_{t-1}|y_{0:t-1}, x_t) \mathrm{d}x_{t-1} \\
&= \int [T_{t-1}^\theta(x_{t-1}) + s_t(x_{t-1}, x_t)] f_\theta(x_{t-1}|y_{0:t-1}, x_t) dx_{t-1},
\end{aligned}$$

where we set $T_0^\theta(x_0) = 0$. The part $f_\theta(x_{t-1}|y_{0:t-1}, x_t)$ is the backward density introduced in Equation (3.1.2). The integral is just an expectation that needs to be calculated. The density of $f_\theta(x_{t-1}|y_{0:t-1}, x_t)$ can be estimated using the equations (3.1.3) and (3.1.4). With this in mind the algorithm for the forward implementation of the FFBS can be seen in algorithm 3.2, the algorithm is initialized by setting $\hat{T}_0^\theta(\xi_0^i) = 0$ for all i. Due to the two sums that needs to be calculated for each particle the computational complexity of the forward implementation of the FFBS is $\mathcal{O}(N^2 n)$ where N is the number of particles used and n is the number of time points.

Algorithm 3.2 Forward implementation of FFBS

Require: $\{\hat{T}^{\theta}_{t-1}(\xi^i_{t-1})\}_{i=1}^N$ and an SMC approximation $\{\omega^i_{t-1}, \xi^i_{t-1}\}_{i=1}^N$.

1: At time t compute the updated SMC approximation $\{\omega^i_t, \xi^i_t\}_{i=1}^N$ and set for each i

$$\hat{T}^{\theta}_t(\xi^i_t) = \sum_{j=1}^N \frac{\omega^j_{t-1} q_\theta(\xi^j_{t-1}, \xi^i_t)}{\sum_{j=1}^N \omega^j_{t-1} q_\theta(\xi^j_{t-1}, \xi^i_t)} \left[\hat{T}^{\theta}_{t-1}(\xi^j_{t-1}) + s_t(\xi^j_{t-1}, \xi^i_t)\right]$$

2: Now set $\hat{\mathcal{S}}^{\theta}_t = \sum_{i=1}^N \omega^i_t \hat{T}^{\theta}_t(\xi^i_t) / \sum_{i=1}^N \omega^i_t$

3.2.1 Some Notes on the Algorithm

The algorithm relies on computing the expectation

$$\begin{aligned}\hat{T}^{\theta}_t(\xi^i_t) &= \mathbb{E}\left[T^{\theta}_{t-1}(X_{t-1}) + s_t(X_{t-1}, \xi^i_t) \,\middle|\, Y_{0:t-1}, \xi^i_t\right] \\ &= \sum_{j=1}^N \frac{\omega^j_{t-1} f_\theta(\xi^j_{t-1}, \xi^i_t)}{\sum_{j=1}^N \omega^j_{t-1} f_\theta(\xi^j_{t-1}, \xi^i_t)} \left[\hat{T}^{\theta}_{t-1}(\xi^j_{t-1}) + s_t(\xi^j_{t-1}, \xi^i_t)\right].\end{aligned}$$

This is an expectation under the distribution of the backwards density, the same distribution that we draw from in the FFBS algorithm. The forward implementation of the FFBS is a special case of the FFBS algorithm that computes smoothed sums of additive functionals. The forward implementation is then possible since the computations are made in a different order.

A problem with this implementation is the computational complexity of the algorithm. When using SMC methods the number of particles is important to reduce the variance of the estimates. A computational complexity of order $\mathcal{O}(N^2)$ (omitting the notation of the time points) is very expensive since we want to use a large number of particles. To reduce this computational complexity and speeding up the algorithm the expectation in step 1 can be estimated using Monte Carlo methods, see Equation (2.3.1). Just as in the FFBS algorithm particles and weights are given, this reduces drawing a sample from the backwards distribution given a current particle ξ^ℓ_t to drawing an index J^ℓ_t depending on the particle ξ^ℓ_t such that

$$\mathbb{P}\left[J^\ell_t = i\right] = \frac{\omega^i_{t-1} q_\theta(\xi^i_{t-1}, \xi^\ell_t)}{c^\ell}, \tag{3.2.1}$$

here c^ℓ is a normalizing constant equal to

$$c^\ell = \sum_{i=1}^N \omega^i_{t-1} f_\theta(\xi^i_{t-1}, \xi^\ell_t). \tag{3.2.2}$$

This is again the same distribution that the FFBS algorithm draws from. To increase the computational speed the the accept-reject method from Douc et al. (2011) could be used.

3.3 Accept-Reject Implementation

As seen in the previous section one way of speeding up the algorithm could be to employ an accept-reject method to draw sample from the backward distribution and estimate the expectation using Monte Carlo methods. To recap for each particle ξ_t^ℓ at time t we want to draw indices J_t^ℓ from the distribution

$$\mathbb{P}(J_t^\ell = i) \propto \omega_{t-1}^i f_\theta(\xi_{t-1}^i, \xi_t^\ell).$$

To employ the accept-reject method (see Section 2.3.1) a new distribution that is easy to sample and a constant σ^+ is needed. If a constant σ^+ such that $q_\theta(\cdot, x_t^\ell) \leq \sigma^+$ can be found, then all is set since

$$\mathbb{P}(J_t^\ell = i) = \frac{1}{c^\ell}\omega_{t-1}^i q_\theta(\xi_{t-1}^i, \xi_t^\ell) \leq \frac{\sigma^+}{c^\ell}\omega_{t-1}^i, \forall_i (1 \leq i \leq N).$$

A proposal density can be chosen as the distribution proportional to the weights and then accept-reject can be applied using $K = \sigma^+/c^\ell$. For instance, when having the Gaussian models the distribution can be bounded by $\sigma^+ = 1/\sqrt{2\pi\sigma^2}$. The algorithm for one draw is then

Require: $\{\omega_{t-1}^i, \xi_{t-1}^i\}_{i=1}^N$ a particle ξ_t^ℓ and a constant $\sigma^+ = 1/\sqrt{2\pi\sigma^2}$.

1: **repeat**
2: Draw $J_t^\ell \sim \left(\omega_{t-1}^i\right)_{i=1}^N$
3: Draw $u \sim \mathcal{U}(0,1)$
4: **until** $u \leq q_\theta(\xi_{t-1}^{J_t^\ell}, \xi_t^\ell)/\sigma^+$

When using the accept-reject method the number of tries needed to find a sample depends on the constant K. In fact the expected number of tries before a sample is found is K. If K is large then the number of tries increases, this could result in the algorithm getting stuck trying to find one sample. An upper limit on the number of tries is implemented to fix this problem, if the upper limit is reached then the sample will be drawn using Equation (3.2.1). Implementing this accept-reject sampling scheme in the FFBS algorithm can be shown to have linear computational complexity (see Douc et al., 2011).

3.4 Forward Implementation of FFBS with Sampling

With the accept-reject sampling method available Algorithm 3.2 is changed to Algorithm 3.3. The idea behind this algorithm is to get a better computational complexity than the FoS algorithm, while not adding to much variance in the process. The parameter L in the algorithm is the number of samples drawn from the backward distribution for every particle. This parameter will be studied more later.

3.4.1 Notes on the new Algorithm

When studying this algorithm it is natural to look at the quantity $\hat{T}_t^\theta(\xi_t^\ell)$. This quantity depends on the particle it is placed at (ξ_t^ℓ) and on L particles from the

Algorithm 3.3 Forward implementation of FFBS with sampling

1: **for** $i = 1 \to N$ **do**
2: simulate $\xi_0^i \sim \chi$
3: set $\omega_0^i \leftarrow g_\theta(\xi_0^i, y_0)$
4: set $\hat{T}_0^\theta(\xi_0^i) \leftarrow 0$
5: **end for**
6: set $\hat{\mathcal{S}}_0^\theta \leftarrow 0$
7: **for** $k = 0 \to n-1$ **do**
8: **for** $i = 0 \to N$ **do**
9: simulate index $I_{k+1}^i \sim (\omega_k^l)_{l=1}^N$
10: simulate $\xi_{k+1}^i \sim q_\theta(\xi_k^{I_{k+1}^i}, \cdot)$
11: set $\omega_{k+1}^i \leftarrow g_\theta(\xi_{k+1}^i, y_{k+1})$
12: **for** $\ell = 1 \to L$ **do**
13: simulate index $J_{k+1}^{i,\ell} \sim (\omega_k^j q_\theta(\xi_k^j, \xi_{k+1}^i))_{j=1}^N$ using accept-reject sampling
14: **end for**
15: set $\hat{T}_{k+1}^\theta(\xi_{k+1}^i) \leftarrow \frac{1}{L}\sum_{\ell=1}^L \left[\hat{T}_{t-1}^\theta(\xi_k^{J_{k+1}^{i,\ell}}) + s_t(\xi_k^{J_{k+1}^{i,\ell}}, \xi_{k+1}^i)\right]$
16: **end for**
17: set $\hat{\mathcal{S}}_{k+1}^\theta = \sum_{i=1}^N \omega_{k+1}^i \hat{T}_{k+1}^\theta(\xi_{k+1}^i) / \sum_{i=1}^N \omega_{k+1}^i$
18: **end for**
19: **return** $\hat{\mathcal{S}}_{0:n}^\theta$

backward distribution. We will call the indices drawn in step 13 of Algorithm 3.3 backward indices.

When estimating $\hat{\mathcal{S}}_t^\theta$ the number of trajectories used is N. It is of interest to study how L will affect these estimates. Say that we set $L = 1$, then two particles that draw the same backward index will have the same trajectory from that point. This is similar to the problem we have when looking at the historical trajectories in the particle filter (see Olsson et al., 2008) and was the reason to implement the FFBS algorithm to find the smoothing distribution, see Section 3.1.

Degeneracy Problem

As noticed earlier, drawing only one particle from the backward distribution at each step implies degeneracy. To investigate this phenomenon, simulations were done, using both $L = 1$ and $L = 2$ to see if any significant difference in the variance of $\hat{\mathcal{S}}_t^\theta$ estimates can be noticed. The fraction of particles used in the estimates is also of interest.

The setup for testing this is a linear and Gaussian state-space model (see Example 2.4) with $A = 0.7$, $R = 0.5$, $B = 1$, and $S = 0.6$. The sum of the smoothed

X values are calculated, that is

$$S_t = \sum_{k=0}^{t} x_k.$$

First we regard the number of unique particles actually used in the samples. Start by saving all the trajectories that were used, that is look at what particles were used when estimating $\hat{T}_t^{\theta}(\xi_t^{\ell})$ for all ℓ. These trajectories for $L = 1$ and $L = 2$ are shown in Figure 3.1. Here we considered 100 time-steps and even though we used only 20 particles, the difference can easily be seen. After only a few time-points there is only one active particle used in the smoothing estimates when $L = 1$. When drawing two backward indices there are many particles active at each time-point.

If we now look at the fraction of unique particles used in the estimates instead,

$$\frac{\#(\hat{\mathcal{S}}_t^{\theta})}{N \cdot (t+1)},$$

where $\#(\cdot)$ is a function returning the number of unique particles used in the estimation. In Figure 3.2 the fraction of particles used is plotted over time.

Looking at the variances two tests were done. The first uses $t = 2000$ time points and $N = 500$ particles, this is then repeated 50 times to calculate the variances. The second test uses $t = 10000$ time points and $N = 2000$ particles, this is then repeated 100 times. To test the growth of the variance linear and quadratic curves are estimated to fit the variances. In Figure 3.3 and Figure 3.4 we can see that the variance when only drawing one particle grows quadratically while the variance only grows linearly when using two particles. Drawing two particles requires roughly twice the amount of computations compared to drawing one backward index. To make things fair then we could have $2N$ particles when we only draw one backward index this should theoretically reduce the variance by a factor 2. But this is just a constant and since the variance is growing faster it will always be better to draw two indices compared to one.

3.4.2 Non Limited Models

When using the accept-reject method of sampling from the backward distribution we need to find the constant σ^+, this constant does not exist if $q_{\theta}(\xi_{t-1}, \xi_t)$ is not bounded. Here the Metropolis-Hastings sampling that was introduced in section 2.3.3 can be used. The idea is to draw indices J_t^{ℓ} from the backwards kernel such that

$$\mathbb{P}(J_t^{\ell} = i) = \frac{\omega_{t-1}^{i} f_{\theta}(\xi_{t-1}^{i}, \xi_t^{\ell})}{c^{\ell}},$$

where c^{ℓ} is the normalizing constant. If the proposal distribution is chosen to be

$$r(i) \propto \omega_{t-1}^{i},$$

the acceptance probability of the new index J', given the previous value J, is given by

$$\alpha(J, J') = \frac{\omega_{t-1}^{J'} f_\theta(\xi_{t-1}^{J'}, \xi_t^\ell)}{\omega_{t-1}^{I} f_\theta(\xi_{t-1}^{J}, \xi_t^\ell)} \frac{\omega_{t-1}^{J}}{\omega_{t-1}^{J'}} \wedge 1,$$

this can be simplified to

$$\alpha(J, J') = \frac{f_\theta(\xi_{t-1}^{J'}, \xi_t^\ell)}{f_\theta(\xi_{t-1}^{J}, \xi_t^\ell)} \wedge 1.$$

The problem with using Metropolis-Hastings is that the given output is not i.i.d. and to compensate for this more samples needs to be drawn. There is also a need to have a burn-in to make sure that the chain has reached its stationary distribution.

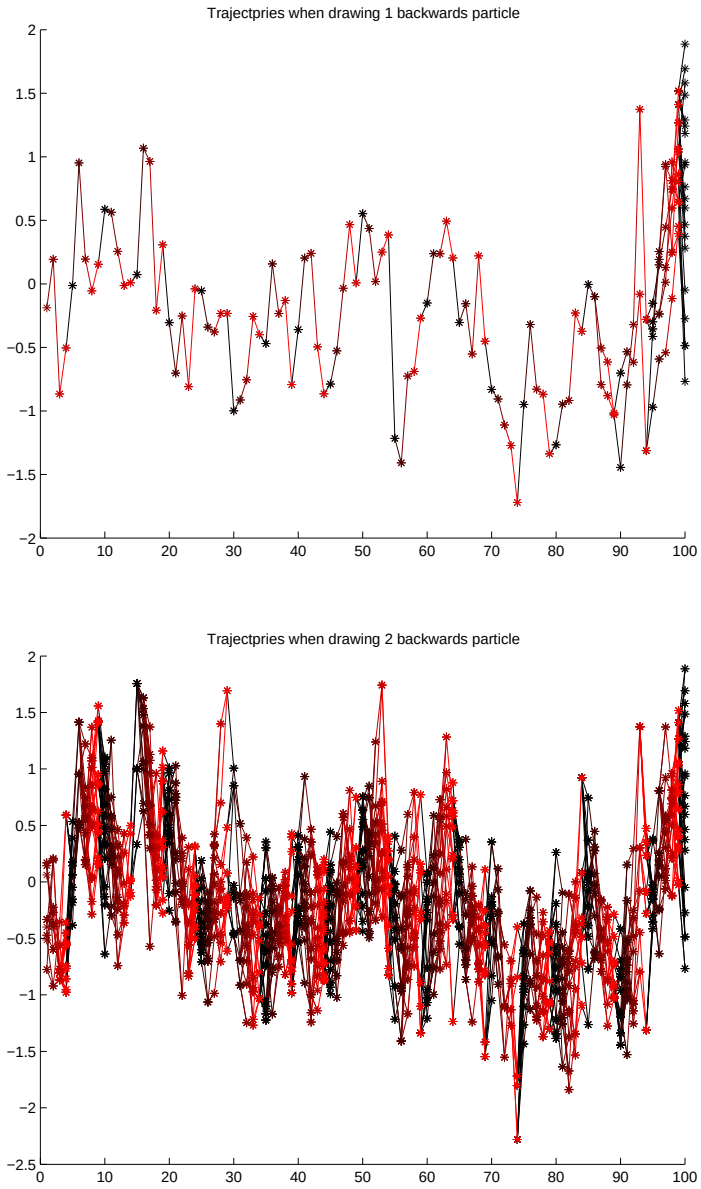

Figure 3.1: The trajectories used when computing the smoothed functionals. The top picture is when drawing 1 backward index and the bottom is when drawing 2 backward indices. It can be seen that more particles are active in the estimations when using 2 backward indices.

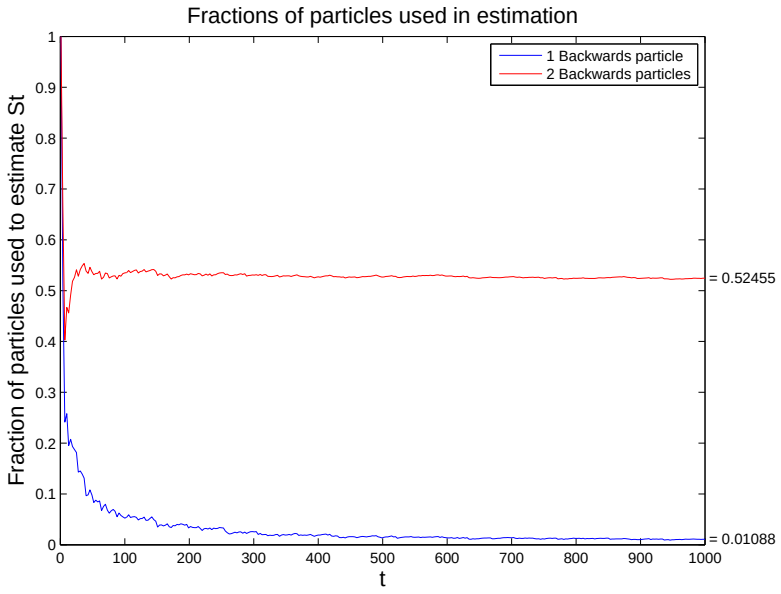

Figure 3.2: Fraction of the particles used in our estimations of S_t^θ using $N = 200$ particles. The value to the right of each graph is the end value of that line, that is the fraction of all particles that were used when estimating S_{1000}^θ.

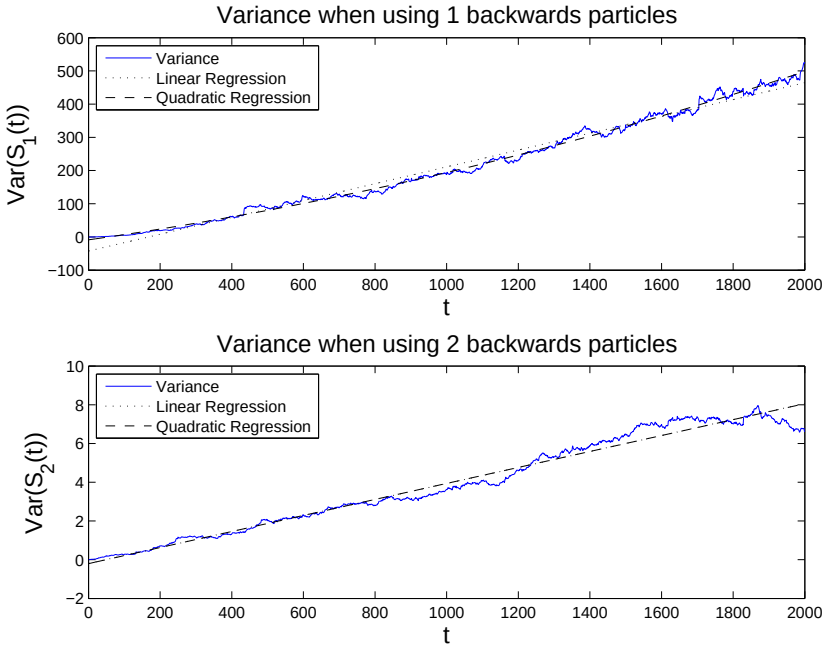

Figure 3.3: Variance for every time step for S_t^θ based on 50 iterations and 500 particles. In the top we use 1 backward index and in the bottom 2 backward indices. The dotted line is a linear fit to the data and the dashed line is a quadratic fit.

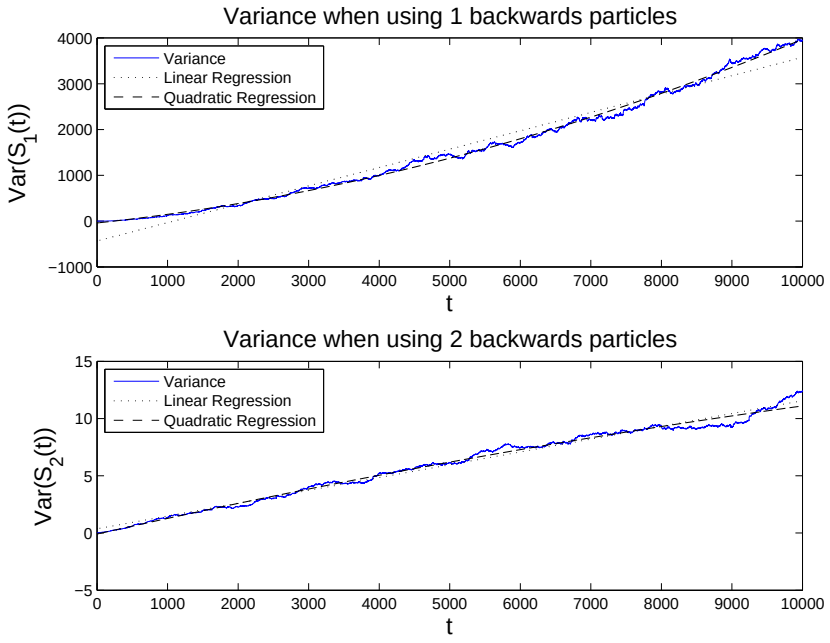

Figure 3.4: Variance for every time step for S_t^θ based on 100 iterations and 2000 particles. In the top we use 1 backward index and in the bottom 2 backward indices. The dotted line is a linear fit to the data and the dashed line is a quadratic fit.

Chapter 4

NUMERICAL RESULTS

In this section we compare the performance of FoSwS to that of FoS by applying the two algorithms to a linear Gaussian state-space model and the stochastic volatility model (Hull and White, 1987). It will also be tested on a non-limited model using the MH sampling method. For all these examples, the goal is to calculate some form of smoothed additive functionals (see Equation (2.2.10)).

4.1 Linear Gaussian Model

The first model that will be used is the linear Gaussian state-space model that was used in Example 2.4. It will be presented again below.

$$X_k = aX_{k-1} + \sigma_U U_k,$$
$$Y_k = bX_k + \sigma_V V_k,$$

where a, b, σ_U and σ_V, are constants. Here $\{U_k\}_{k\geq 1}$ and $\{V_k\}_{k\geq 0}$ are to mutually independent sequences of i.i.d. standard Gaussian white noise and X_0 is drawn from a standard Gaussian variable. The constant $b = 1$ will be used for all simulations and theory around this model. The joint density of $\{X_k, Y_k\}_{k=0}^n$ belongs to an exponential family. The sufficient statistics are then

$$S_{1,t} = \sum_{k=1}^{t} X_k^2, \qquad S_{2,t} = \sum_{k=1}^{t} X_k X_{k-1},$$
$$S_{3,t} = \sum_{k=1}^{t} X_{k-1}^2, \qquad S_{4,t} = \sum_{k=0}^{t} (Y_k - X_k)^2.$$

In addition, the vector of natural parameters is given by

$$\psi(\theta) = \left(-\frac{1}{2\sigma_U^2}, \frac{a}{\sigma_U^2}, -\frac{a^2}{2\sigma_U^2}, -\frac{1}{2\sigma_V^2} \right),$$

and

$$c(\theta) = \frac{t}{2}\log(\sigma_U^2) + \frac{t+1}{2}\log(\sigma_V^2),$$

where $\theta = (a, \sigma_U, \sigma_V)$.

4.1.1 Comparing to the Exact Solution

Before testing with the sufficient statistics described above, a test of the validation of both algorithms is conducted. This test is conducted using the model described above, but instead of computing one of the sufficient statistics we let the algorithms produce estimates of the sum of the smoothed variables,

$$\mathcal{S}_t = \mathbb{E}\left[\sum_{i=0}^{t} X_i | Y_{0:t}\right]. \tag{4.1.1}$$

This quantity can be computed exactly using the disturbance smoother when working with the linear and Gaussian state-space model (see Cappé et al., 2005, algorithm 5.2.15). Figure 4.1 displays the values of $\hat{\mathcal{S}}_t$ when $a = 0.7$, $\sigma_U = 0.2$, and $\sigma_V = 1$. In Figure 4.2 box plots for a few time steps are shown. It can be noticed that both algorithms calculate the correct values and the increasing variance is to be expected since we are calculating the sum. In the box plots it can be noticed that the correct values are always within the box. Both algorithms appear to approximate well the correct values.

4.1.2 Estimation of Sufficient Statistics

Being convinced that the algorithms produce the correct output we look at estimating the sufficient statistics, we will be looking at $\mathcal{S}_{i,t}/t$ where

$$\mathcal{S}_{i,t} = \mathbb{E}\left[S_{i,t} | Y_{0:t}\right]. \tag{4.1.2}$$

The reason for looking at $\mathcal{S}_{i,t}/t$ instead of $\mathcal{S}_{i,t}$ is that the variance of $\mathcal{S}_{i,t}/t$ will decrease with time, while for $\mathcal{S}_{i,t}$ the variance will increase as seen in Figure 4.1. For the simulations the parameters were set to $a = 0.8$, $\sigma_U = 0.2$, and $\sigma_V = 1$, together with $T = 2000$ time points and $N = 500$ particles. This test was then repeated 50 times with the same $\{Y_k\}_{k=0}^{T}$ as input to produce the plots. For the FoSwS algorithm two backward indices were drawn. To compute the SMC approximations the bootstrap particle filter was used.

In Figure 4.3 the plots of $\hat{\mathcal{S}}_{i,t}/t$ is shown for the four different sufficient statistics. Looking at the figures it is hard to notice any difference between the methods. In Figure 4.4 box plots for the different sufficient statistics and time points are estimated. Table 4.1 shows numerical values of the estimates at the last time-point. There is no real difference between the two methods that can be seen in the plots, but when looking at the computational time needed for the two algorithms it can be seen that FoSwS was 4.88 times faster then the FoS algorithm.

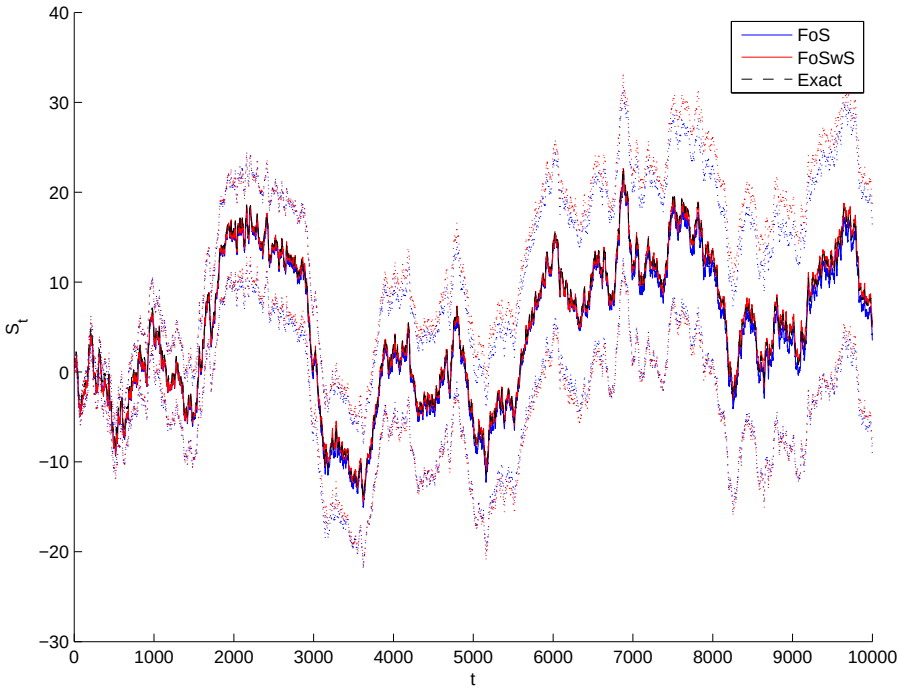

Figure 4.1: Estimation of the sum of the smoothed states. The increasing variance for both methods is to be expected. The blue line is the FoS algorithm and the red line is the FoSwS algorithm. The dotted lines are 95% confidence intervals and the black dashed line is the exact values.

i	$\hat{\mathcal{S}}_{i,T}/T$		$\mathbb{V}[\hat{\mathcal{S}}_{i,T}/T]$	
	FoS	FoSwS	FoS	FoSwS
1	0.1123	0.1120	$1.1713 \cdot 10^{-6}$	$6.8750 \cdot 10^{-7}$
2	0.0900	0.0898	$1.1141 \cdot 10^{-6}$	$6.6005 \cdot 10^{-7}$
3	0.1122	0.1119	$1.1733 \cdot 10^{-6}$	$6.8600 \cdot 10^{-7}$
4	1.0056	1.0059	$1.2280 \cdot 10^{-6}$	$1.3516 \cdot 10^{-6}$

Table 4.1: Values of $\hat{\mathcal{S}}_{i,t}/t$ for the four different sufficient statistics in the linear Gaussian model from Section 4.1

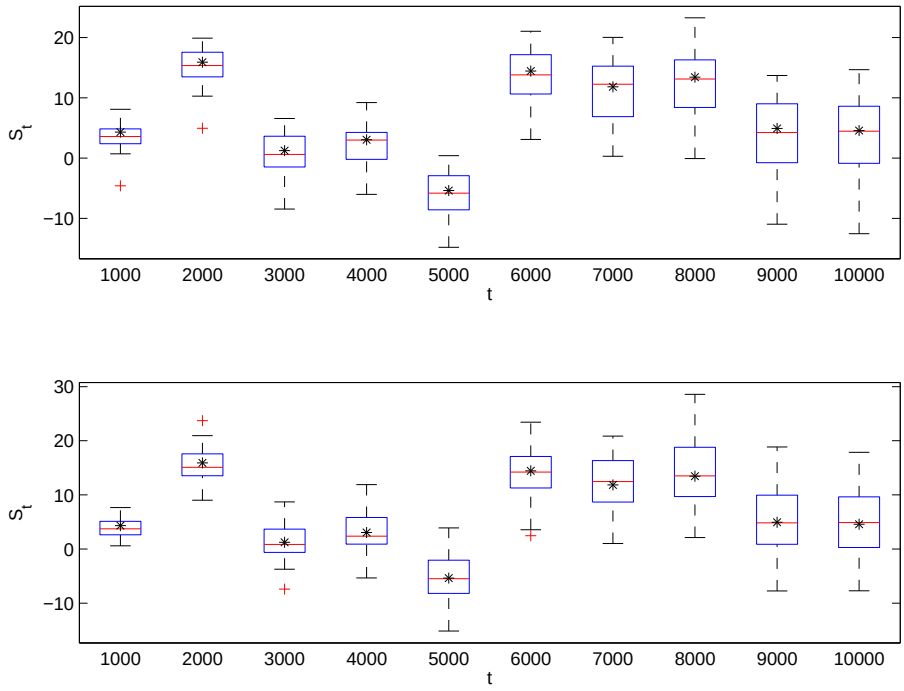

Figure 4.2: Box plots of the sum of the smoothed states. The top and bottom graph displays values produced using the FoS and FoSwS algorithms, respectively. The stars indicate the exact values.

4.2 Stochastic Volatility Model

The stochastic volatility model is an HMM defined by the equations

$$X_k = \phi X_{k-1} + \sigma U_k,$$
$$Y_k = \beta \exp(X_k/2)\, V_k,$$

where $\{U_k\}_{k\geq 0}$ and $\{V_k\}_{k\geq 0}$ are two mutually independent sequences of i.i.d. standard Gaussian noise that do not depend on the initial value X_0.

The following sufficient statistics are found by inspecting the complete data log-likelilhood

$$S_{1,t} = \sum_{k=1}^{t} X_k^2, \qquad S_{2,t} = \sum_{k=1}^{t} X_k X_{k-1},$$
$$S_{3,t} = \sum_{k=1}^{t} X_{k-1}^2, \qquad S_{4,t} = \sum_{k=0}^{t} Y_k^2 \exp(X_k).$$

In addition, the vector of natural parameters is given by

$$\psi(\theta) = \left(-\frac{1}{2\sigma^2}, \frac{\phi}{\sigma^2}, -\frac{\phi^2}{2\sigma^2}, -\frac{1}{2\beta^2} \right),$$

and

$$c(\theta) = \frac{t}{2}\log(\sigma^2) + \frac{t+1}{2}\log(\beta^2),$$

the parameter space for this model is $\theta = (\phi, \sigma, \beta)$.

This model allows for so-called volatility clustering, that is, the volatility of the $\{Y\}$ process is not constant and will for some time periods be considerably higher then for other time periods (compare for example with the linear and Gaussian model that has constant volatility). This makes the model useable in finance where it for instance can model the daily log-returns of a stock.

4.2.1 Estimation of Sufficient Statistics

We look again at $\mathcal{S}_{i,t}/t$ where $\mathcal{S}_{i,t}$ is defined in Equation (4.1.2). For the simulations the variables $\phi = 0.8$, $\sigma = 0.2$, and $\beta = 1$ were used. In this simulation $T = 2000$ time points were used and $N = 500$ particles. For the FoSwS algorithm $n = 2$ backwards particles were drawn in each step. The tests were repeated 50 times with the same sequence $\{Y_k\}_{k=1}^{T}$ as input. To compute the SMC estimates the bootstrap particle filter was used.

In Figure 4.5 the plots for $\hat{\mathcal{S}}_{i,t}/t$ is shown for the four different sufficient statistics given earlier. In the first three sufficient statistics, a difference between the two algorithms can be seen, at the same time as the confidence intervals for both methods contain each other. In Figure 4.6 the box plots for a few time points are shown. Table 4.2 gives numerical values for the estimations at the last time-point. In the box plots it can be noticed that the variance of estimates produces using the FoSwS algorithm is larger but acceptably so. When looking at the computation times the FoSwS algorithm is 13 times faster then the FoS algorithm.

i	$\hat{\mathcal{S}}_{i,t}/t$		$\mathbb{V}[\hat{\mathcal{S}}_{i,t}/t]$	
	FoS	FoSwS	FoS	FoSwS
1	0.1104	0.1096	$6.0889 \cdot 10^{-7}$	$6.3583 \cdot 10^{-7}$
2	0.0882	0.0871	$5.7817 \cdot 10^{-7}$	$6.1030 \cdot 10^{-7}$
3	0.1104	0.1095	$6.0779 \cdot 10^{-7}$	$6.3536 \cdot 10^{-7}$
4	1.0523	1.0521	$5.2856 \cdot 10^{-6}$	$7.2561 \cdot 10^{-6}$

Table 4.2: Values for $\hat{\mathcal{S}}_{i,t}/t$ for the four different sufficient statistics in the stochastic volatility model from Section 4.2

4.3 Non Limited Model

The model that we have studied so far have all been limited to the case of a bounded transition density (that is, the case where we can find a constant K). As discussed in Section 3.4.2, it is not always possible to find such a constant and in this case we propose a way of solving this problem by utilizing the Metropolis-Hastings algorithm. To test this method the following model is considered.

$$X_k = a + bX_{k-1} + \sigma_U X_{k-1} U_k, \tag{4.3.1}$$

$$Y_k = X_k + \sigma_V V_k. \tag{4.3.2}$$

As before $\{V_k\}_{k\geq 0}$ and $\{U_k\}_{k\geq 0}$ are two mutually independent sequences of standard Gaussian noise. There is also an initial distribution χ for X_0. In this case the transition density is evidently not bounded, which prevents the accept-reject method to be used. Instead the MH-sampling from Section 3.4.2 is used.

In this case we restrict ourselves to smoothing of sums of states, that is

$$\mathcal{S}_t = \mathbb{E}\left[\sum_{i=0}^{t} X_i | Y_{0:T}\right].$$

4.3.1 Estimation

Simulations in this model was carried through under the dynamics determined by the parameters $a = 0.1$, $b = 0.95$, $\sigma_U = 0.3$, and $\sigma_V = 1$. The length of the sample is $T = 2000$ time points and the particle smoother used $N = 500$. The initial state X_0 was set to the fixed value a. For the MH-step 8 indices were drawn and the last 3 of these were used in the estimates, that is a burn-in of length 5 was used. To generate confidence intervals and box plots, we generated 50 replicates using the same input data. The process generated by Equations (4.3.1)–(4.3.2) can be seen in Figure 4.7. With this choice parameters the state sequence is rather unstable, exhibiting jumps of significant size.

Considering estimation of $\hat{\mathcal{S}}_t/t$ the estimates produced by the FoS algorithm and the FoSwS with MH-sampling algorithm, displayed in Figure 4.8, the estimates are close to indistinguishable. Also the box plots in Figure 4.8 are close to indistinguishable. There is however (as before) a difference in the computational times needed for the two methods with the FoSwS algorithm being three times faster than the FoS algorithm.

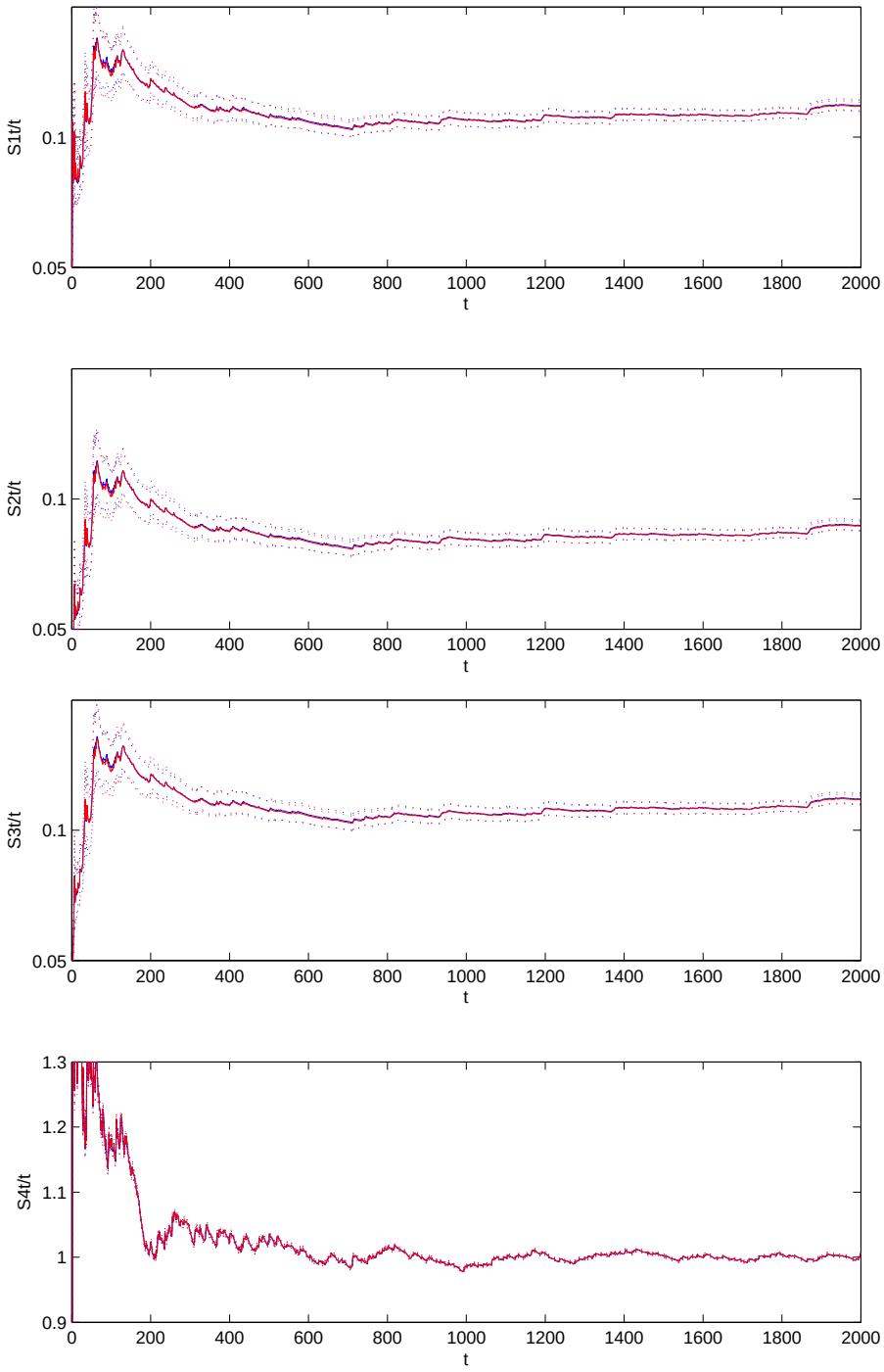

Figure 4.3: Estimation of smoothed sufficient statistics in the linear and Gaussian model, $\hat{\mathcal{S}}_t/t$. The blue line is the FoS algorithm. The red line is the new FoSWS algorithm. The dotted lines are 95% confidence intervals. All plots have been cut to increase the resolution.

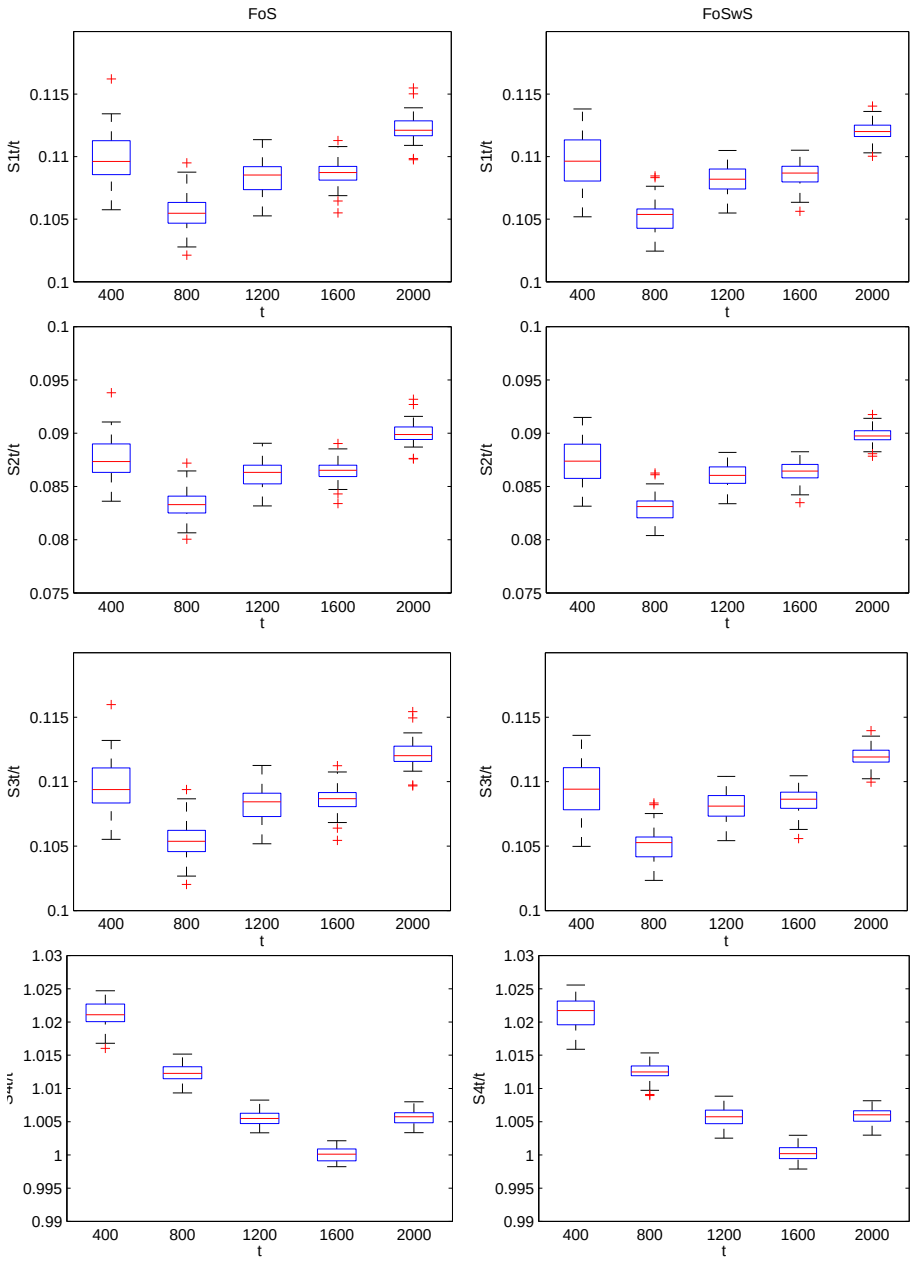

Figure 4.4: Box plots for time averaged smoothed sufficient statistics estimates $\hat{\mathcal{S}}_t/t$ for different time points. The left and right column displays values produced using the FoS and FoSwS algorithms, respectively.

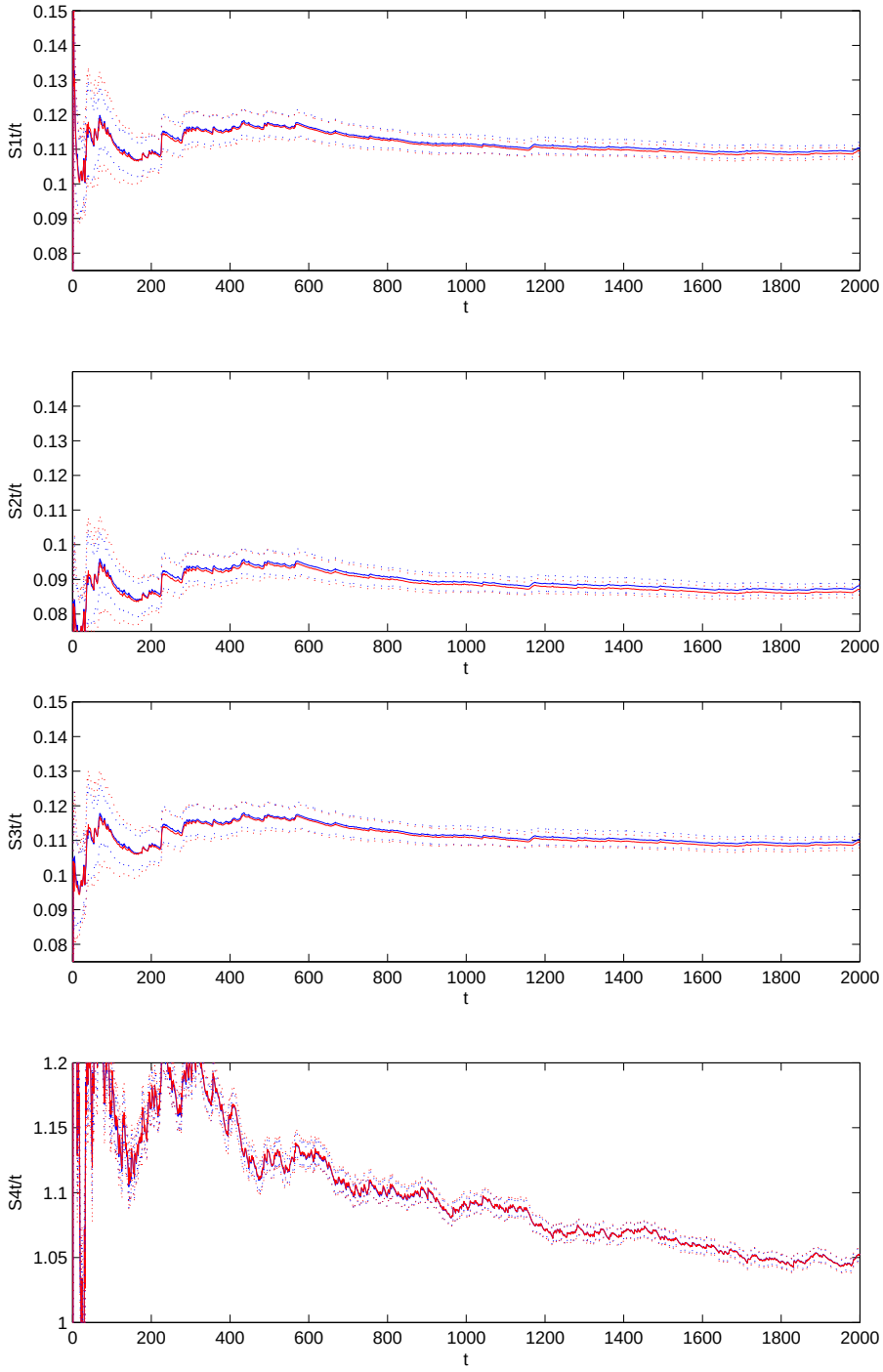

Figure 4.5: Estimation of the time averaged smoothed sufficient statistics ($\hat{\mathcal{S}}_t/t$) for the stochastic volatility model. The blue line is the FoS algorithm, the red line is the FoSwS algorithm and the dotted lines are 95% confidence intervals. The plots are cut to increase resolution.

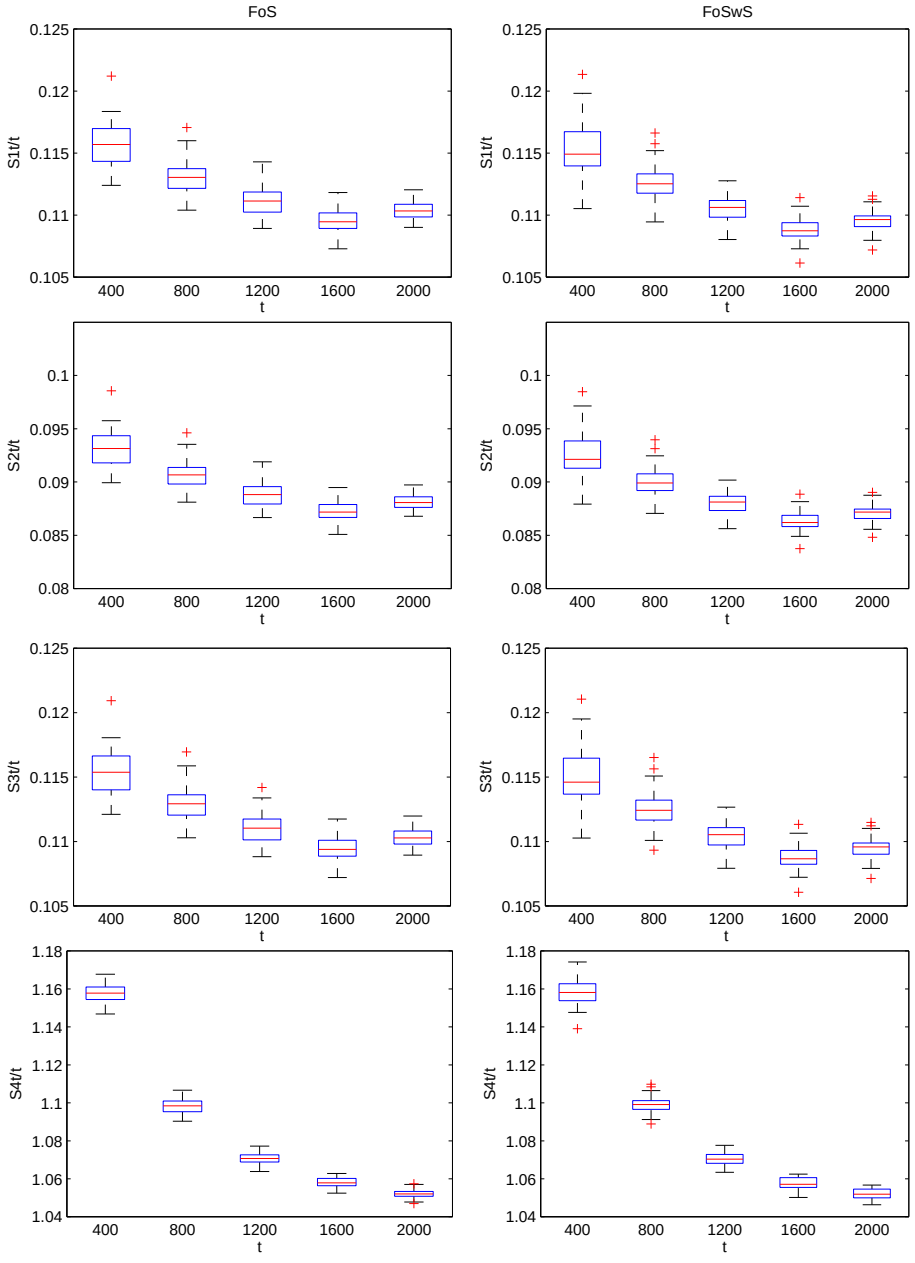

Figure 4.6: Box plots for time averaged smoothed sufficient statistics estimates $\hat{\mathcal{S}}_t/t$ for different time points. The left and right column displays values produced using the FoS and FoSwS algorithms, respectively.

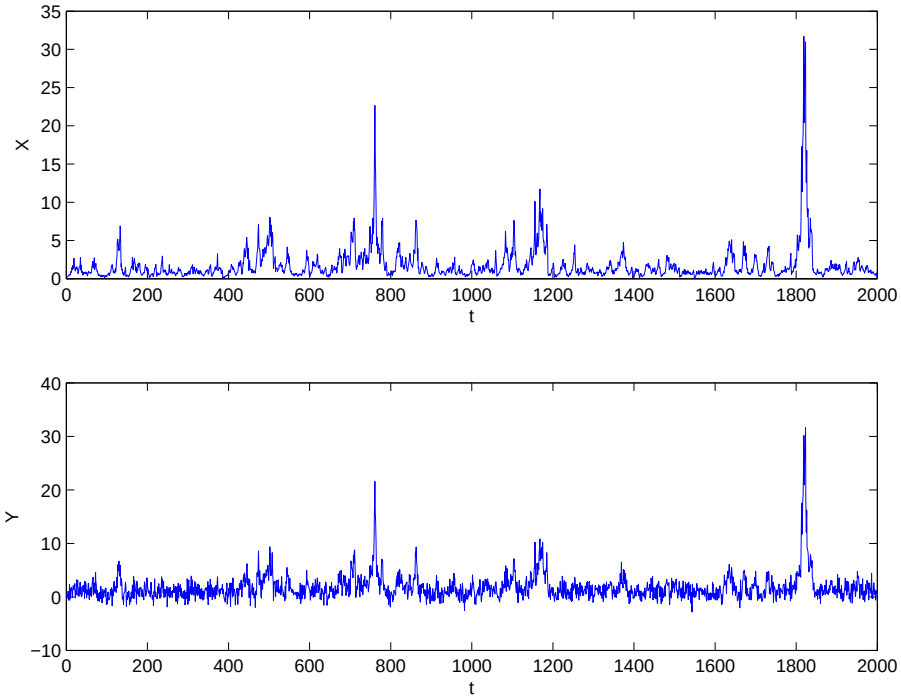

Figure 4.7: The X and Y processes generated by the non-limited model given by Equations (4.3.1)–(4.3.2). The top and bottom graph shows the X and Y process, respectively.

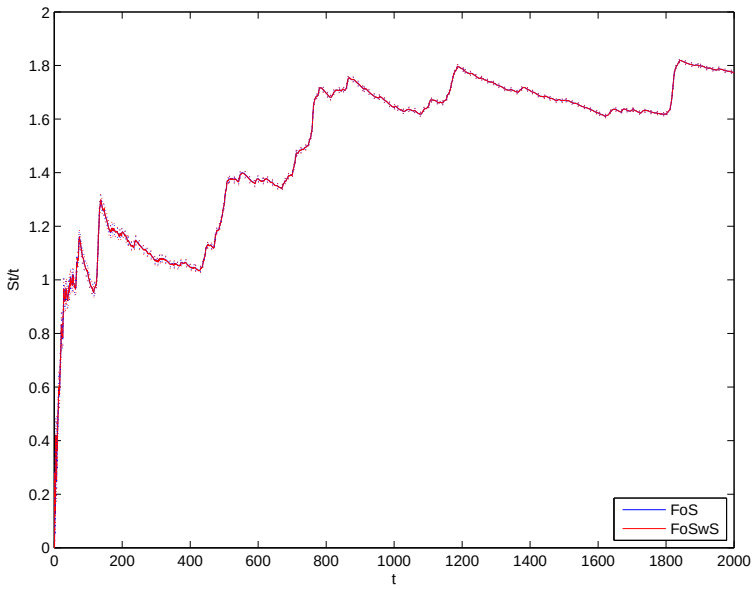

Figure 4.8: $\hat{\mathcal{S}}_t/t$ for t from 0 to 2000 for the non-limited model. The blue line is from the FoS algorithm and the red line from the FoSwS algorithm, the dotted lines are 95% confidence interval. The FoSwS algorithm uses the MH-sampling method.

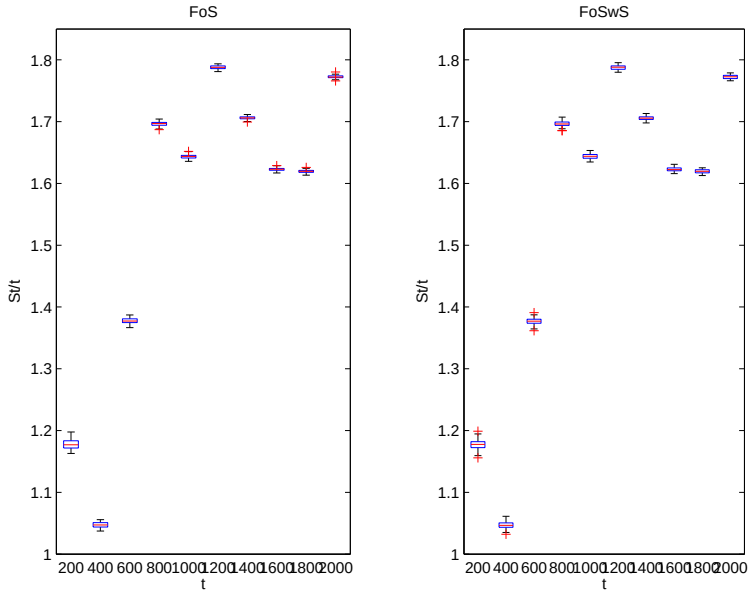

Figure 4.9: Box plots for $\hat{\mathcal{S}}_t/t$ for every 200 values of t from 200 to 2000. The left and right plots refer to the FoS and the FoSwS algorithms, respectively.

Chapter 5

CONCLUSIONS

This master's thesis proposes a version of the forward implementation of FFBS (introduced by Del Moral et al., 2009) with sampling (FoSwS). This algorithm uses Monte Carlo methods to estimate expectations that were computed exactly in the forward implementation of FFFBS (FoS) algorithm. When using this algorithm it is important to draw more then one backward index, as seen in Section 3.4.1. For all the simulations done, two indices were drawn, which seems to be enough to avoid any form of degeneracy. The precision of the estimates produced by FoSwS do not differ from that of the FoS. When working with the more complex stochastic volatility model, the FoS algorithm has the lower variance but not by any large margin, see Table 4.2. What is more interesting is the computational time needed for the two algorithms; the FoSwS is almost 5 times faster for the linear and Gaussian model, 13 times faster for the stochastic volatility model, and 3 times faster for the non-limited model. These numbers are of course dependent on the implementation, but should still indicate a gain of computational efficiency for the FoSwS algorithm in general.

Using Metropolis-Hastings sampling the algorithm was able to handle the non-limited model where accept-reject sampling is not possible. In the non-limited model both algorithms produce equivalent output, but again the FoSwS algorithm is faster than the FoS algorithm. The thesis provides no discussion on how long burn-in and how many indices that needs to be generated when using Metropolis-Hastings. One index would probably have the same devastating effect as in the accept-reject case. For the simulations in Section 3.4.2, 8 indices were drawn and the last three of these were accepted. But here more research is needed to gain a better understanding.

To move on from this thesis topics of research could include a closer study on the computational times needed. For instance, could the FoSwS algorithm be proved to have a computational complexity of order $\mathcal{O}(N)$? When only drawing one index the degeneration is present, but can we obtain qualitative results explaining this? Another interesting question is to apply the proposed algorithm in an online EM setting (Cappé, 2011; Del Moral et al., 2009; Poyiadjis et al., 2011).

Chapter 6

Bibliography

Baum, L. E., Petrie, T., Soules, G. and Weiss, N. (1970), "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains", *The Annals of Mathematical Statistics* , Vol. 41, pp. pp. 164–171.

Cappé, O. (2011), "Online em algorithm for hidden markov models", *Journal of Computational and Graphical Statistics* , Vol. 20, pp. 728–749.

Cappé, O., Moulines, E. and Rydén, T. (2005), *Inference in Hidden Markov Models*, Springer, New York.

Churchill, G. A. (1992), "Hidden markov chains and the analysis of genome structure", *Computers & Chemistry* , Vol. 16, pp. 107 – 115.

Del Moral, P., Doucet, A. and S. Singh, S. (2009), Forward smoothing using sequential monte carlo, Technical report, Cambridge University.

Douc, R., Garivier, A., Moulines, E. and Olsson, J. (2011), "Sequential monte carlo smoothing for general state space hidden markov models", *The Annals of Applied Probability* , Vol. 21, pp. 2109–2145.

Godsill, S. J., Doucet, A. and West, M. (2004), "Monte carlo smoothing for nonlinear time series", *Journal of the American Statistical Association* , Vol. 99.

Gordon, N. J., Salmond, D. and Smith, A. (1993), "Novel approach to nonlinear/non-gaussian bayesian state estimation", *IEE Proceedings F. Radar & Signal Processing* , Vol. 40, pp. 107–113.

Gut, A. (2009), *An Intermediate Course in Probability Theory*, Springer, New York.

Hull, J. and White, A. (1987), "The pricing of options on assets with stochastic volatilities", *The Journal of Finance* , Vol. 42, pp. pp. 281–300.

Olsson, J., Cappé, O., Douc, R. and Moulines, E. (2008), "Sequential monte carlo smoothing with application to parameter estimation in nonlinear state space models", *Bernoulli* , Vol. 14, pp. 155–179.

Poyiadjis, G., Doucet, A. and Singh, S. S. (2011), "Particle approximations of the score and observed information matrix in state space models with applications to parameter estimation", *Biometrika* , Vol. 98, pp. 65–80.

Rabiner, L. R. (1989), "A tutorial on hidden markov models and selected applications in speech recognition", *Proceedings of the IEEE* , Vol. 77, pp. 257 – 286.