Particle-based Parameter Inference in Stochastic Volatility Models: Batch vs. Online

ALBIN TOFT
Particle-based Parameter Inference in Stochastic Volatility Models: Batch vs. Online

ALBIN TOFT
Abstract

This thesis focuses on comparing an online parameter estimator to an offline estimator, both based on the PaRIS-algorithm, when estimating parameter values for a stochastic volatility model. By modeling the stochastic volatility model as a hidden Markov model, estimators based on particle filters can be implemented in order to estimate the unknown parameters of the model. The results from this thesis implies that the proposed online estimator could be considered as a superior method to the offline counterpart. The results are however somewhat inconclusive, and further research regarding the subject is recommended.
Sammanfattning

Partikelbaserad parameterskattning i stokastiska volatilitets modeller: batch vs. online

Acknowledgements

In this chapter I would like to express my gratitude towards the people who have helped me throughout my years at KTH, and who have aided me in the process of writing of this thesis. First and foremost I would like to say thanks to my supervisor Jimmy Olsson. Not only for all the valuable help and feedback he has given me while writing this thesis, but also for introducing me to the field of computational statistics. Also, a huge thanks goes to my family and my best friends for always being there and supporting me. Without you, these five years would have been a lot harder!
# Contents

1 Introduction ........................................... 1  
   1.1 Stochastic Volatility .............................. 1  
   1.2 Online Estimation ................................. 1  
   1.3 Parameter Estimation in Stochastic Volatility Models .... 2  
   1.4 Hidden Markov Models ............................. 3  
   1.5 Thesis Objective .................................. 4  
   1.6 Outline ............................................ 5  
   1.7 Research Question ................................. 6  

2 Background ........................................... 7  
   2.1 Hidden Markov Models and Parameter Estimation ............. 7  
      2.1.1 Hidden Markov models .......................... 7  
      2.1.2 Tangent filters ................................ 11  
      2.1.3 Parameter estimation in fully dominated HMMs ....... 11  
   2.2 Stochastic Volatility Models .......................... 13  
   2.3 Sequential Monte Carlo Methods ........................ 14  
      2.3.1 The bootstrap particle-filter .................... 14  
      2.3.2 Coefficient of variation and efficient sample size .. 16  

3 Methods ............................................. 18  
   3.1 Stochastic Volatility Models as Hidden Markov Models ..... 18  
   3.2 The PaRIS-algorithm ................................ 20  
      3.2.1 Tangent filter estimation ....................... 21  
   3.3 PaRIS-Based Online Parameter Estimation ................. 23  
   3.4 Batch EM Parameter Estimation ........................ 26  

4 Case Study .......................................... 30  
   4.1 Algorithm Validation ................................ 30  
      4.1.1 Validation of online algorithm simulated data ....... 30  
      4.1.2 Validation of offline algorithm ................... 35
# CONTENTS

4.2 Implementation on Actual Stock Prices .................................................. 38  
4.2.1 Online implementation ................................................................. 39  
4.2.2 Offline implementation ................................................................. 43  
4.3 Experiments Regarding Uncertainty of Estimates .................................... 46  
4.3.1 Experiment 1 .................................................................................. 47  
4.3.2 Experiment 2 .................................................................................. 49  

5 Comparison of Online and Offline Estimator ............................................ 52  
5.1 Comparison on Simulated Data ............................................................... 52  
5.2 Comparison on AAPL Data .................................................................... 53  
5.3 Comparison of Results from Experiment 1 and Experiment 2 ................. 54  

6 Discussion ............................................................................................... 56  
6.1 Case Study ......................................................................................... 56  
6.2 Stochastic Volatility Model Selection ..................................................... 57  
6.3 Online vs Offline Algorithm .................................................................. 58  

7 Conclusions and Future Work ................................................................... 59  
7.1 Conclusions ....................................................................................... 59  
7.2 Future Work ....................................................................................... 60  

Bibliography ............................................................................................... 62  

A Derivations of Quantities in Online Algorithm ........................................ 64  
A.1 Derivation of $\tilde{h}_{s,\theta}(x_s, x_{s+1})$ ................................................. 64  
A.2 Derivation of $\nabla_\theta g_{s,\theta}(x_s)$ ......................................................... 65  

B Derivation of Quantities in Offline Algorithm ........................................... 67  

C Maximization of Intermediate Quantity .................................................... 69  

D Validation of Online Algorithm With Reduced Particle Cloud ................. 71
Chapter 1

Introduction

1.1 Stochastic Volatility

When working with financial mathematics one often uses the formula proposed by Black and Scholes for pricing of European call options [1]. The pricing formula proposed by Black and Scholes is widely used in banking and is perhaps the most well-known formula for pricing of options. The formula is based on several assumptions regarding the dynamics of the price of the underlying asset, which is assumed to follow a log-normal distribution with constant drift and volatility over time. When examining data over stock prices, it has however become obvious that these assumptions do not coincide with the dynamics of actual stock prices. The assumption of constant volatility has especially proved to be unjustifiable. This has given rise to new and alternative models for stock prices and their volatility.

By removing the assumption of constant volatility, and viewing the volatility as a random process varying over time, several new models have been proposed. By introducing randomness to the volatility it is possible to formulate models which can capture the effects of a varying volatility. These models are known as stochastic volatility models, with some concrete examples being the Bates model [2], Heston model [3] and the Hull and White model [4].

1.2 Online Estimation

When working with stochastic processes, one is often interested in the task of parameter estimation. This is due to the fact that stochastic processes commonly are modeled using models including some parameters determining the
behaviour of the process. Given a set of observations, one then tries to find a model for the stochastic process which best resembles the observations. A crucial step in determining such a model is by estimating the parameters used for modeling the process. Depending on the observations, and what the application for the model is, there are several methods of estimating these parameters. If one for instance is interested in estimating the parameters using a fixed number of observations, the parameters could be estimated by using all available observations at once. This is what is often called a batch estimate. However, in many applications one faces the scenario where new data become available as time passes, or where the amount of data is extremely large. One might then be interested in repeatedly estimating the parameters as new observations become available. By online estimation it is possible to use previously estimated parameters and new observations to update the previous parameter estimates, without using previous observations.

1.3 Parameter Estimation in Stochastic Volatility Models

As introduced previously, stochastic volatility models are new and powerful models for modeling asset prices, exchange rates and other financial data. However, due to the complex nature of the models it can be problematic to estimate the parameters specifying the models. This problem arises from the fact that stochastic volatility models include so called missing data, meaning that only a fraction of the time series is observable. The problem of parameter estimation for missing data models can however be solved through the use of particle filters and the EM-algorithm, which is done by Kim and Stoffer in [5]. Combining particle filters and the EM-algorithm results in an offline estimator of the parameters. Today, offline parameter estimators are widely implemented in order to solve the problem of parameter estimation for stochastic volatility models. However, online estimators could also be implemented, and perhaps even be more effective than the offline alternatives. Today, the use of online estimators for stochastic volatility models seems to be sparse, even though the qualities of an online estimator makes it suitable for the problem. It is therefore of interest to investigate the performance of an online estimator as a solution to the parameter estimation problem for stochastic volatility models, and compare its effectiveness to that of an offline estimator.
1.4 Hidden Markov Models

Hidden Markov models are bi-variate stochastic processes characterized by an underlying un-observable Markov process \( \{ X_t \}_{t \in \mathbb{N}} \), and one observable process \( \{ Y_t \}_{t \in \mathbb{N}} \). The process \( \{ X_t \}_{t \in \mathbb{N}} \) is often referred to as the state, or hidden, process, and \( \{ Y_t \}_{t \in \mathbb{N}} \) as the observation process. The state process is only partially observable through the observation process, and the observations are conditionally independent given the corresponding states. One could think of a hidden Markov model as a house with lights on. For an observer standing outside, it is not possible to directly observe how many residents that are inside. However, by observing the amount of windows with lights on, the observer can make a qualified guess of the number of residents. In this example the hidden state process is the number of residents inside, and the observation process is the number of windows with light on. A general graphical illustration of a hidden Markov model can be seen in Figure 1.1.

![Graphical illustration of a hidden Markov model](image)

Figure 1.1: Graphical illustration of a hidden Markov model.

As Figure 1.1 illustrates, the observation at a certain time depends only on the hidden state at the same time instance. This is illustrated by the arrows in the figure. Further, it is also shown that the state process proceeds independently from the observation process.

Hidden Markov models are often used when one is interested in making inference regarding some Markov process by observing an associated process [6]. In recent years hidden Markov models have seen several applications, such as speech recognition, computational biology and signal processing. This can be seen by the multiple references to hidden Markov models in [7]. Hidden Markov models have also proved to be of great use in financial mathematics,
in particular when working with stochastic volatility models [8]. One of the major strengths of implementing hidden Markov models to stochastic volatility models is that it allows for volatility clustering, which will be discussed further in this thesis.

What makes hidden Markov models such a powerful tool in statistics is the combination of the general model structure and the fact that the hidden process possesses the Markov property. This makes the models applicable to multiple problems, while allowing for efficient algorithms for parameter estimation, simulation and prediction.

The fundamental problem when working with hidden Markov models is, given a specified model and a set of observations \( Y_1 \ldots Y_n \) one wants to make inference regarding the corresponding hidden states \( X_1 \ldots X_n \). Evaluating the conditional distribution of the hidden state \( X_k \) at time \( k \) given \( Y_{0:n} \) is what is referred to as smoothing in [6], and there are several ways of performing smoothing. Most of the early references on smoothing are based on a rather specific kind of models known as Gaussian linear state-space models. The use of these models partially arose from the pioneering work by Kalman and Bucy in 1961, resulting in the Kalman filter [9]. However, the lack of computational tools for handling nonlinear/non-Gaussian components prevented hidden Markov models for being used in their full potential for several years [8]. Today there exist several methods of handling smoothing for such models, for instance sequential Monte Carlo methods. These algorithms generate sequences of weighted particles in order to approximate the smoothing distributions of the targets, and have in recent years proved to be very useful. Known as particle filters, these methods approximate sequentially the smoothing distributions by means of repeated importance sampling.

### 1.5 Thesis Objective

The main objective of this thesis is to formulate and implement an online estimator for stochastic volatility. This will be done by modeling the stochastic volatility as a hidden Markov model, and by implementing a particle-based online estimator to the model. The PaRIS-algorithm, formulated by Olsson and Westerborn in [10], will be implemented as online estimator. The online estimator is to be implemented on log-returns of data on actual stock prices from the financial market. Further, the PaRIS-based online parameter estimator will be compared to a more conventional EM-based offline implementation.
of the PaRIS-algorithm. A major difference in the two estimators is that the online estimator approximates the “true” parameter, while the offline estimator approximates the maximum likelihood estimate of the “true” parameter. The main research question of the thesis is whether the online algorithm could be considered as an alternative, or even superior method, for parameter estimation of stocks modeled using a stochastic volatility model, to the offline EM-implementation. This will be evaluated by comparing accuracy, uncertainty and computational time for two algorithms based on the different approaches.

1.6 Outline

The outline of the thesis is as follows:

Chapter 2 provides the reader with the necessary background required for the remaining part of the thesis. It covers the concept of hidden Markov models, and explains terms such as smoothing, backward kernels, parameter estimation and tangent filters. Further, Chapter 2 also explains stochastic volatility models and sequential Monte Carlo methods.

In the following Chapter 3, the methods of deriving the particle-based online estimator of the parameters for the stochastic volatility model are presented. The chapter starts with rewriting the stochastic volatility model used in the thesis as a hidden Markov model. The PaRIS-algorithm is explained, and its application as an online parameter estimator is presented. Finally, the chapter explains the offline EM-implementation of the PaRIS-algorithm, which the online implementation is to be compared to.

Chapter 4 covers a case study where both the online and offline algorithm are implemented. The algorithms are first validated on a simulated data set, where the parameters to be estimated are known initially, and used to generate the data. Secondly, the algorithms are implemented on stock data from the financial market. Assuming that the used data follows the suggested stochastic volatility model, the two algorithms provide estimates of the model parameters, which are presented in this chapter. Also included in this chapter are two experiments where the performance of the two proposed algorithms are tested against each other. The experiments focus on comparing the uncertainty of the two algorithms, when the computational complexity for the algorithms are the same.
Following the results in Chapter 4 comes Chapter 5, where the results from the previous chapter are used in order to compare the two algorithms. The two algorithms are compared in terms of accuracy, precision, uncertainty and computational time. The comparisons made in this chapter are further discussed in Chapter 6. Finally, in Chapter 7 conclusions regarding the results and future work that might be of interest are stated.

1.7 Research Question

The main research question of the thesis is whether the online algorithm could be considered as an alternative, or even superior method, for parameter estimation of stocks modeled using a stochastic volatility model, to the offline EM-implementation. Answering this question is done by evaluating accuracy, precision, uncertainty and computational time for the two algorithms.
Chapter 2

Background

The purpose of this chapter is to provide necessary background regarding the main areas covered in the thesis. First, some definitions and explanations will be given regarding hidden Markov models. Parameter estimation for hidden Markov models will also be covered in this section. Secondly, some background regarding stochastic volatility models will be given, where the model covered in this thesis will be defined and explained. Thereafter sequential Monte Carlo methods will be introduced, and their application to hidden Markov models will be explained.

2.1 Hidden Markov Models and Parameter Estimation

2.1.1 Hidden Markov models

As previously mentioned in Section 1.4 a hidden Markov model (abbreviated HMM) consist of of a bivariate stochastic process \( \{(X_t, Y_t)\}_{t \in \mathbb{N}} \). The two processes \( \{X_t\} \) and \( \{Y_t\} \) takes values in two sets \( X \) and \( Y \). Accompanying each set is a \( \sigma \)-algebra \( \mathcal{X} \) and \( \mathcal{Y} \), which implies that \( (X, \mathcal{X}) \) and \( (Y, \mathcal{Y}) \) are both measureable spaces. We will now define a HMM by providing a definition of a transition kernel, and thereafter using the definition of a transition kernel in order to define a HMM. The following definitions are made by Cappé and Mouliens in [6].

**Definition 2.1.1** (Transition kernel). Let \( (X, \mathcal{X}) \) and \( (Y, \mathcal{Y}) \) be two measureable spaces. An un-normalized transition kernel from \( (X, \mathcal{X}) \) to \( (Y, \mathcal{Y}) \) is a function \( Q : X \times Y \rightarrow [0, \infty] \) that satisfies
(i) for all \( x \in X \), \( Q(x, \cdot) \) is a positive measure on \( (Y, \mathcal{Y}) \).

(ii) for all \( A \in \mathcal{Y} \), the function \( x \mapsto Q(x, A) \) is measure-able.

If \( Q(x, Y) = 1 \) for all \( x \in X \), then \( Q \) is called a transition kernel, or simply a kernel. If \( X = Y \) and \( Q(x, X) = 1 \) for all \( x \in X \), then \( Q \) will also be referred to as a Markov transition kernel on \( (X, \mathcal{X}) \).

Once the concept of a transition kernel has been defined, a formal definition of a hidden Markov model can be made. This is done in [6] by defining its transition kernel, and the definition is stated below.

**Definition 2.1.2 (Hidden Markov Model).** Let \( (X, \mathcal{X}) \) and \( (Y, \mathcal{Y}) \) be two measure-able spaces and let \( Q \) and \( G \) denote, respectively, a Markov transition kernel on \( (X, \mathcal{X}) \) and a transition kernel from \( (X, \mathcal{X}) \) to \( (Y, \mathcal{Y}) \). Consider the Markov transition kernel defined on the product space \( (X \times Y, \mathcal{X} \otimes \mathcal{Y}) \) by

\[
T[(x, y), C] = \int \int_C Q(x, dx') G(x', dy'), \quad (x, y) \in X \times Y, \ C \in \mathcal{X} \otimes \mathcal{Y}.
\]

(2.1)

The Markov chain \( \{X_t, Y_t\}_{t \geq 0} \) with Markov transition kernel \( T \) and initial distribution \( \nu \otimes G \), where \( \nu \) is a probability measure on \( (X, \mathcal{X}) \), is called a hidden Markov model.

Further, if one considers the case where there exists probability measures \( \mu, \lambda \) such that both \( G \) and \( Q \) has transition densities \( g \) and \( q \), the model is said to be fully dominated. This implies that for a fully dominated model we have

\[
G(x, A) = \int_A g(x, y) \mu(dy), \quad A \in \mathcal{Y}
\]

\[
Q(x, B) = \int_B q(x, x') \lambda(dx'), \quad B \in \mathcal{X}
\]

(2.2)

where \( t[(x, y), (x', y')] \triangleq q(x, x') g(x', y) \) is the transition density function of \( T \). For the remaining part of this thesis all HMMs will be considered to be fully dominated, since this makes statistical estimation of models with unknown parameters possible. The unknown parameters are found in \( g \) and \( q \), since one often define these transition densities as density functions from some parametric families with parameter \( \theta \). Here \( \theta \) is some parameter in a parameter space \( \Theta \), and to indicate the dependence on this parameter the transition densities will be assigned a
As mentioned briefly in Section 1.4 the fundamental problem in hidden Markov modeling is making inference of a hidden sequence \( X_0, \ldots, X_n \), given a fully specified model and the corresponding observations \( Y_0, \ldots, Y_n \). In particular, one is often interested in the expectation of \( X_{k:k'} \) given a sequence of observations \( Y_{0:n} = y_{0:n} \). First define the conditional distribution of \( X_{k:k'} \) given \( Y_{0:n} \) as \( \phi_{k:k'|n,\theta} \), and consider the expectation \( E_\theta[f(X_{k:k'})|Y_{0:n}] \). The expectation can then be defined as

\[
E_\theta[f(X_{k:k'})|Y_{0:n}] = \int \cdots \int f(x_{k:k'}) \phi_{k:k'|n,\theta}(Y_{0:n}, dx_{k:k'}),
\]

which makes it possible to express the expectation as

\[
\phi_{k:k'|n,\theta} f \triangleq E_\nu[f(X_{k:k'})|Y_{0:n}].
\]

If the HMM is assumed to be fully dominated, the conditional distribution \( \phi_{k:k'|n,\theta} \) can be expressed by using the joint distribution of \( X_{0:t} \) and \( Y_{0:t} \), which can be expressed as

\[
p_{\theta}(x_{0:t}, y_{0:t}) = \nu(x_0)g_{\theta}(x_0, y_0) \prod_{k=1}^{t} g_{\theta}(x_{k-1}, x_k)g_{\theta}(x_k, y_k).
\]

An expression for the conditional distribution can thereafter be obtained by

\[
p_{\theta}(x_{0:t}|y_{0:t}) = \frac{p_{\theta}(x_{0:t}, y_{0:t})}{p_{\theta}(y_{0:t})} = \frac{p_{\theta}(x_{0:t}, y_{0:t})}{\int_{X_t} p_{\theta}(x_{0:t}, y_{0:t})dx_{0:t}}
\]

\[
\phi_{0:t|t,\theta} = \frac{p_{\theta}(x_{0:t}, y_{0:t})}{\int_{X_t} p_{\theta}(x_{0:t}, y_{0:t})dx_{0:t}}.
\]

One is often in particular interested in the smoothing distribution, defined as \( \phi_{0:0|t,\theta} \). Using Equation (2.5) and 2.6 it is possible to express the joint-smoothing distribution as

\[
\phi_{0:t|t,\theta} = \frac{\nu(x_0)g_{\theta}(x_0, y_0) \prod_{k=1}^{t} g_{\theta}(x_{k-1}, x_k)g_{\theta}(x_k, y_k)}{\int_{X_t} \nu(x_0)g_{\theta}(x_0, y_0) \prod_{k=1}^{t} g_{\theta}(x_{k-1}, x_k)g_{\theta}(x_k, y_k)dx_{0:t}}.
\]

This distribution will prove to be important when making inference for HMMs. Further, the expression for the expectation \( \phi_{k:k'|n,\theta} f \triangleq E_\nu[f(X_{k:k'})|Y_{0:n}] \) can
be formulated as
\[ \phi_{k:k'}|n,\theta f = \left( \nu(dx_0) \prod_{t=0}^{\max(n,k')-1} Q_{\theta}(x_t, dx_{t+1}) \right) \]
\[ \int \cdots \int f(x_{k:k'}) \left( \prod_{m=0}^{n} g_{m;\theta}(x_m) \right) L_{\theta}(y_{0:n}) \]
(2.8)

with
\[ L_{\theta}(y_{0:n}) = \int \cdots \int g_{0;\theta}(x_0) \nu(dx_0) \prod_{t=0}^{n} g_{t+1;\theta}(x_{t+1}) Q_{\theta}(x_t, dx_{t+1}) \]
(2.9)

being the observed data likelihood. Closely related to the joint-smoothing distribution are the filter and predictor distributions. Define the filter distribution as \( \phi_{t,\theta} \equiv \phi_{t:t|t;\theta} \), and the predictor distribution as \( \pi_{t+1;\theta} \equiv \phi_{t+1:t+1|t;\theta} \). The filter recursion in [10] provides that for all \( t \in \mathbb{N} \) and \( f \in F(\mathcal{X}) \),
\[ \phi_{t,\theta} f = \frac{\pi_{t,\theta}(g_{t,\theta} f)}{\pi_{t,\theta} g_{t;\theta}} \]
(2.10)
\[ \pi_{t+1;\theta} f = \phi_{t;\theta} Q_{\theta} f, \]
(2.11)
where by convention \( \pi_{0;\theta} \equiv \nu \). These quantities will prove to be important in the following chapters, and in particular in Section 2.3.

One may also consider the time-reversed state process, which in [6] is proven to have the Markov property. If one consider a fully dominated model, the distribution of \( X_s \) given \( X_{s+1} \) and \( Y_{0:t} = y_{0:t} \) is give by a backward kernel \( \overrightarrow{Q}_{\phi_{s:\theta}} \). Given that the model is fully dominated, the backward kernel for \( x_{s+1} \in \mathbb{X} \) and \( f \in F(\mathcal{X}) \) can be written as
\[ \overrightarrow{Q}_{\phi_{s:\theta}} f(x_{s+1}) \equiv \frac{\int f(x_s) g_{0}(x_s, x_{s+1}) \phi_{s:\theta}(dx_s)}{\int q_{\theta}(x'_s, x_{s+1}) \phi_{s:\theta}(dx'_s)}. \]
(2.12)

Using the backward kernel, one can also express the joint-smoothing distribution \( \phi_{0:t|t;\theta} \) as
\[ \phi_{0:t|t;\theta} = \phi_{t;\theta} T_{t;\theta}. \]
(2.13)

Here the kernel \( T_{t;\theta} \) describes the in-homogeneous Markov chain initialized at \( x \in \mathbb{X} \), evolving backwards in time according to the backward kernels.

Often one is interested in computing smoothed expectations of some additive objective function \( h_t(x_{0:t}) \), which can be defined as
\[ h_t(x_{0:t}) \equiv \sum_{s=0}^{t-1} \tilde{h}_s(x_{s:s+1}). \]
(2.14)
This set up results in the possibility of recursive computation of \( \{T_t;\theta h_t\}_{t\in\mathbb{N}} \) by

\[
T_{t+1;\theta} h_{t+1}(x_{t+1}) = \int \{T_t;\theta h_t(x_t) + \tilde{h}_t(x_{t:t+1})\} \tilde{Q}_{\phi_t;\theta}(x_{t+1}, dx_t). \tag{2.15}
\]

Being able to recursively compute \( \{T_t;\theta h_t\}_{t\in\mathbb{N}} \) will prove to be essential in the following chapters.

### 2.1.2 Tangent filters

Tangent filters are defined as the gradient of the prediction filter \( \pi_t;\theta f_t \). That is, the tangent filter \( \eta_{f;\theta} \) is defined by

\[
\eta_{f;\theta} f_t \triangleq \nabla_{\theta} \pi_t;\theta f_t, \quad f_t \in F(\mathcal{X}). \tag{2.16}
\]

In order to obtain an expression for the gradient of the prediction distribution, one might first consider the gradient of \( \phi_{0:t|t-1;\phi} \), and thereafter obtain the marginal. This is done by Olsson and Westerborn in [10], where by first determining \( \nabla_{\theta} \phi_{0:t|t-1;\theta} f_{0:t} \) and thereafter marginalizing while using the backward decomposition in Equation (2.13), the following expression for the gradient of the prediction filter could be determined

\[
\nabla_{\theta} \pi_t;\theta f_t = \pi_t;\theta \{(T_t;\theta h_t;\theta - \pi_t;\theta T_t;\theta h_t;\theta) f_t\}. \tag{2.17}
\]

By expressing the tangent filters as in Equation (2.17) it is possible to recursively update the filter derivatives. This is due to the fact that both the predictors and the backward statistics \( \{T_t;\theta h_t\}_{t\in\mathbb{N}} \) are possible to update recursively through the filtering recursion in Equation (2.10) and the recursion in Equation (2.15). However, in most cases it is not possible to obtain closed form expressions for the predictors or the backwards statistics. Overcoming this problem can however be done by the use of sequential Monte Carlo methods, which will further be discussed in Section 2.3.

### 2.1.3 Parameter estimation in fully dominated HMMs

Since the main objective of this thesis is regarding parameter estimation for stochastic volatility models using HMMs, researching parameter estimation for HMMs is of utmost interest. As mentioned previously, fully dominated HMMs will be considered since statistical inference and parameter estimation for such models are in general simpler. Estimating parameters is done by likelihood optimization for so called incomplete data models, a concept that was
CHAPTER 2. BACKGROUND

proposed by Dempster et al. [11]. The term incomplete data refers to the fact that the observed data does not include observations of the hidden states. Given a \( \sigma \)-finite measure \( \lambda \) on \((X, \mathcal{X})\) and considering a family \( \{ f(\cdot; \theta) \}_{\theta \in \Theta} \) we consider the task of maximizing the likelihood \( L(\theta) \) defined as

\[
L(\theta) \triangleq \int f(x; \theta) \lambda(dx)
\]

with respect to \( \theta \). In most applications, \( f(x; \theta) \) is a simple function of \( \theta \). However, \( L(\theta) \) is usually determined by some high dimensional integral, which makes classical maximization methods difficult. In the case where one is interested in HMMs \( f \) is the joint probability density function of the pair of random variables \( X \) and \( Y \), with \( X \) being hidden and \( Y \) observed. Here \( f \) will be referred to as the complete data likelihood and \( L \) the density of \( Y \), which is the available likelihood for estimating \( \theta \). Note that since \( L(\theta) \) is assumed to be positive, maximizing \( L(\theta) \) is equivalent to maximizing \( l(\theta) \triangleq \log(L(\theta)) \).

A common and regularly implemented method of solving the optimization problem presented above is the expectation-maximization-algorithm (abbreviated EM). The EM is described by Dempster et al. [11], and is based on the concept of an intermediate quantity \( Q \), defined as

\[
Q(\theta, \theta') \triangleq \int \log f(x; \theta)p(x; \theta')\lambda(dx),
\]

where \( p(x; \theta) \triangleq f(x; \theta)/L(\theta) \) is the conditional density of \( X \) given \( Y \). The intermediate quantity is fully defined, and may be used as a surrogate for \( l(\theta) \). One can interpret \( Q(\theta; \theta') \) as \( E_\theta[\log(f(X; \theta))] \), given that \( X \) is distributed according to the probability density \( p(\cdot; \theta') \). Instead of maximizing \( l(\theta) \), which may be impossible, one maximizes \( Q \) with respect to \( \theta \), which gives a maximum likelihood estimate \( \theta^* \) of \( \theta \). The EM-algorithm can be summarized in two steps, that are repeated until the generated sequence of estimates \( \{ \theta^i \} \) converges. The algorithm is initialized by some initial guess \( \theta^0 \in \Theta \), then the two steps follows:

- **E-Step:** Determine \( Q(\theta; \theta^i) \)
- **M-Step:** Determine \( \theta^{i+1} \in \Theta \) which maximizes \( Q(\theta^{i+1}, \theta^i) \)

which are repeated until convergence is achieved.

There are however alternative methods to the EM-algorithm, such as gradient-based methods. These methods are based on the fact that for any model where
the EM-algorithm may be applied, it is possible to evaluate the derivatives of the objective function \( l(\theta) \) with respect to \( \theta \). If one considers the Robbins-Monro scheme at iteration \( n \) let

\[
\theta_n = \theta_{n-1} + \gamma_n Z_n,
\]

where \( Z_n \) is a noisy measurements of \( \nabla_{\theta} l(y_{0:t}|\theta=\theta_{n-1}) \), and \( \{\gamma_n\} \) a sequence satisfying the stochastic approximation requirements

\[
\sum_{n=1}^{\infty} \gamma_n = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \gamma_n^2 < \infty.
\]

If there is a large number of observations, approximating \( Z_n \) at each iteration will be computationally costly, since each observation is required in the approximation. Further, for every new observation available, \( Z_n \) needs to be recalculated, which means this is an offline algorithm. The algorithm can however be transformed into an online algorithm, which is described in [10], and this will be further explained in Section 3.3.

### 2.2 Stochastic Volatility Models

As introduced in Section 1.1, stochastic volatility models are a new and powerful models to describe the dynamics of stock prices. As mentioned previously, they are able to capture the stochastic behaviour of the volatility of stock prices, which results in models that closer resembles actual stock prices than models where the volatility is assumed to be constant. A general setup for a discretely observed stochastic volatility model is provided by Genon and Catalot in [12]

\[
\begin{align*}
    dY_t &= \varphi(V_t)dt + V_t^{1/2}dB_t, \quad Y_0 = 0, \\
    dV_t &= b(V_t)dt + a(V_t)dW_t, \quad V_0 = \eta,
\end{align*}
\]

(2.21)

where \( \{B_t, W_t\}_{t \geq 0} \) is a two dimensional Brownian Motion, \( \{V_t\} \) is a positive diffusion process and \( \eta \) a random variable independent of the Brownian motions. The process \( \{Y_t\} \) is discretely observed at times \( t = t_1, \ldots, t_n \), and is defined as \( Y_t \triangleq \log(S_t) \), with \( S_t \) being the asset price at time \( t \). By studying the empirical distribution of the observed increments of \( \{Y_t\} \) Genon and Catalot provide a proof of convergence and a central limit theorem in [12].

By defining \( \varphi(V_t), b(V_t) \) and \( a(V_t) \) in Equation (2.21) it is possible to identify known stochastic volatility models. Since most stochastic volatility models are applied to stock prices, \( \varphi(V_t) \) is commonly defined as \( \varphi(V_t) \triangleq \mu - \frac{1}{2}V_t \).

This is obtained by applying Itô’s lemma to the stochastic differential equation

\[
\begin{align*}
    dS_t &= \mu S_t dt + V_t^{1/2}S_t dB_t, \\
    S_0 &= s_0,
\end{align*}
\]

(2.22)
with the substitution \( Y_t \triangleq \log(S_t) \). Consider the \textit{stochastic volatility model} proposed by Hull and White in [4]

\[
\begin{align*}
    dS_t &= \mu S_t dt + V_t^{1/2} S_t dB_t, \\
    dV_t &= \phi V_t dt + \xi dW_t,
\end{align*}
\]

which by introducing the substitution \( Y_t \triangleq \log(S_t) \) and applying Itô’s lemma can be rewritten as

\[
\begin{align*}
    dY_t &= (\mu - \frac{1}{2} V_t) dt + V_t^{1/2} dB_t, \\
    dV_t &= \phi V_t dt + \xi dW_t.
\end{align*}
\]

### 2.3 Sequential Monte Carlo Methods

As mentioned in Section 2.1.2 one is required to have access to filter and prediction distributions in addition to the backward statistics, if filter derivative estimation is to be performed. This requirement is somewhat troublesome, since unless the state-space model is linear Gaussian or the state space \( X \) is finite, these quantities are not available in a closed form. Thus, one will have to rely on approximations of these quantities in order to make filter derivative estimation. Approximation can for instance be done by \textit{sequential Monte Carlo} methods, such as \textit{sequential importance sampling} proposed by Handchin in [13].

#### 2.3.1 The bootstrap particle-filter

In the following section it is assumed that all random variables are defined on a common probability space \((\Omega, \mathcal{F}, P)\). Given is a sequence of observations \( \{y_t\}_{t \in \mathbb{N}} \) and a particle sample \( \{\xi^i_t\}_{i=1}^N \). Approximating the predictor \( \pi_{t;\theta} f \) by \( \pi_{t;\theta}^N f \) can then be done as

\[
\pi_{t;\theta} f \approx \frac{1}{N} \sum_{i=1}^N f(\xi^i_t) \triangleq \pi_{t;\theta}^N f. \tag{2.25}
\]

Once an approximation of the predictor has been made, it is possible to derive an approximation of the filtering distribution. This is done by applying
the updating step of the filtering recursion in Equation (2.10). The filtering distribution $\phi_{t;\theta f}$ can then be approximated by $\phi_{t;\theta f}^N$ as
\[
\phi_{t;\theta f} \approx \sum_{i=1}^{N} \frac{w_i^t}{\Omega_t} f(\xi_i^t) \triangleq \phi_{t;\theta f}^N,
\]
where $N$ is the number of particles generated and $\Omega_t \triangleq \sum_{i=1}^{N} w_i^t$. The weights $\{w_i^t\}$ are generated by $w_i^t \triangleq g(\xi_i^t, y_t)$, and can be interpreted as the belief that the generated particle $\xi_i^t$ produced the observation $y_t$. If one were to take the approach of the naïve sequential importance sampler, the next step would be to generate new particles $\{\xi_{i+1}^t\}$ by propagating the current sample $\{\xi_i^t\}$, a step referred to as mutation. However this has a major flaw, which is weight degeneration. Weight degeneration means that as the algorithm is repeated over time, a substantial amount of the weights will decrease in magnitude to become close to infinitesimal. This is due to the fact that the algorithm mutates particles with small weights. These particles will most likely mutate into particles with smaller weights, which leads to the phenomena of weight degeneration. The issue with degenerating weights is due to the fact that as the weights degenerate, only a few of the generated particles will affect the approximations in Equation (2.25) and 2.26. Thus, the algorithm will become numerically unstable as time passes, and will yield inaccurate approximations. The bootstrap particle filter, proposed by Gordon et al. in [14] overcomes this flaw by including a selection step prior to the mutation. The selection step consists of replacing particles with small weights with particles with high weights. Mathematically this is done by for each particle drawing an index
\[
I_i^t \sim Pr(\{w_i^t\}_{i=1}^{N}) = \frac{w_i^t}{\Omega_t},
\]
In words, this means for each particle an index $j$ is drawn, where the probability of drawing index $j$ is $Pr(I_i^t = j) = w_i^t / \Omega_t$. Note that $Pr(\{w_i^t\}_{i=1}^{N})$ is known as the categorical distribution on $\{1, \ldots, N\}$ and is defined as just described. This generates a uniform particle sample $\{\xi_i^t, 1\}_{i=1}^{N}$ that targets $\phi_{t;\theta}$. Performing the selection step results in a numerically stable algorithm, with non-degenerating weights.

Following the selection step comes as previously mentioned the mutation step. Originally proposed by Gordon et al. in [14], this is done by propagating the particles according to the state process. This implies that a new particle sample $\{\xi_{t+1}^i\}_{i=1}^{N}$ is generated by
\[
\xi_{t+1}^i \sim q(\xi_i^t, \ldots), \quad i = 1, \ldots, N,
\]
with associated weights \( \{w^i_{t+1}\}_i \) computed by \( w^i_{t+1} = g_\theta(\xi^i_{t+1}, y_{t+1}) \), \( i = 1, \ldots, N \). The algorithm is initialized by drawing \( \{\xi^i_0\}_i \sim \nu^\otimes N \).

The bootstrap particle filter can also be used to obtain an approximation of the smoothed statistic \( \phi_{0:t;f} \) by storing the entire particle trajectory at each time step \( s + 1 \). This is done by \( \xi_{0:s+1}^i = (\xi_{0:s}^i, \xi_{s+1}^i) \), \( i = 1, \ldots, N \). As a result, the smoothing statistic can be approximated as

\[
\phi_{0:t;f} \approx \frac{1}{N} \sum_{i=1}^{N} w^i_t \Omega^i_t (\xi_{0:t}) \triangleq \phi^N_{0:t;f}.
\]  

However, this approximation suffers from a serious drawback called the particle path degeneracy phenomena. This problem arises from the multiple re-sampling steps in the bootstrap particle filter. The repeated re-sampling operations results in collapsing particle trajectories, meaning that prior to some random time \( T < t \) all trajectories are the same. This means that at a certain time \( t > T \), all particles \( \{\xi^i_t\}_i \) share a common ancestor at time \( T \). This results in inaccurate approximations of the joint filtering distribution, and thus an alternative approach is desirable without degenerating particle paths. One such solution proposed by Olsson and Alenlöv in [10] is by using the backward decomposition in Equation (2.13), and will be further explained in coming chapters.

\[\]  

**2.3.2 Coefficient of variation and efficient sample size**

Since an SMC-estimator can suffer from weight degeneration, it is of interest to obtain some measure of the performance of the algorithm. This is often done by determining the coefficient of variation \( CV_N \), and the efficient sample size \( N_{eff} \). The coefficient of variation is defined as

\[
CV_N \triangleq \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left( N \frac{w^i}{\sum_{l=1}^{N} w^l} - 1 \right)^2},
\]  

which is minimal when all weights are equal, and maximal when all weights except one is equal to zero. The efficient sample size is defined as

\[
N_{eff} \triangleq \frac{N}{1 + CV^2_N},
\]
and can be interpreted as the number of effective particles used by the algorithm in the estimation. Since it is desirable with an algorithm that uses as many particles as possible, a low value of the coefficient of variation and an efficient sample size close to the total number of particles $N$, imply that the algorithm does not suffer from the issue described above.
Chapter 3

Methods

In this chapter the methods for deriving the particle-based online estimator of the stochastic volatility will be presented. It provides a thorough explanation of how a general stochastic volatility model can be modeled as a HMM. Thereafter, the HMM model of the stochastic volatility is examined further, with important distributions, parameters and quantities defined. Once the setup for the stochastic volatility HMM is specified, a thorough description of the PaRIS-algorithm will be given. Thereafter online parameter estimation using the PaRIS-algorithm will be described. The last section of the chapter will describe a conventional offline parameter estimator using the PaRIS-algorithm.

3.1 Stochastic Volatility Models as Hidden Markov Models

Since online estimation is to be performed using SMC on HMMs, a crucial first step is rewriting a discretely observed stochastic volatility model, with sampling $\Delta t$, as a state-space model for a HMM. Consider the stochastic volatility model defined in Section 2.2 in Equation (2.23). By solving the stochastic differential equations in Equation (2.23) using Itô’s lemma we obtain

$$S_t = S_{t-\Delta t} \exp \left\{ \left( \mu - \frac{1}{2} V_t \right) \Delta t + U_1^t \sqrt{V_t \Delta t} \right\},$$

$$V_t = V_{t-\Delta t} \exp \left\{ \left( \phi - \frac{1}{2} \xi^2 \right) \Delta t + U_2^t \xi \sqrt{\Delta t} \right\},$$

(3.1)

where $\{U_1^t, U_2^t\}$ are two processes of independent standard normal random variables. By defining $Y_t \triangleq \log \left( \frac{S_t}{S_{t-\Delta t}} \right)$ and $X_t \triangleq \log(V_t)$, Equation (3.1)
can be rewritten as
\[ Y_t = \mu \Delta t - \frac{1}{2} \exp(X_t) \Delta t + \sqrt{\Delta t} \exp(X_t/2) U_{t_1}, \]
\[ X_t = X_{t_1-\Delta t} + (\phi - \frac{1}{2} \xi^2) \Delta t + \xi \sqrt{\Delta t} U_{t_1}. \]

(3.2)

This can be viewed as the state-space equations of a HMM, with \( \{X_t\} \) being the hidden process and \( \{Y_t\} \) being the observed. However, a simplified version of these state-space equations will be considered in this thesis. The simplified model is made similar to the stochastic volatility model defined by Cappé and Moulines in [6] in Example 80
\[ X_k = \phi X_{k-1} + \sigma U_{k_1} \]
\[ Y_k = \beta \exp(X_k/2) U_{k_2}, \]

(3.3)

where \((X_k, Y_k) \triangleq (X_{t_k}, Y_{t_k}), \quad k = 1, \ldots, n\). One can generalize the model specified in Equation (3.3) by including a constant drift in the log-returns, resulting in
\[ X_k = \phi X_{k-1} + \sigma U_{k_1} \]
\[ Y_k = \mu + \beta \exp(X_k/2) U_{k_2}. \]

(3.4)

One can argue that the simplified model specified above models the same behaviour as the more advanced model in Equation (3.2). For instance, the drift-term in the log-volatility in Equation (3.2) should be zero, since otherwise it would imply that the volatility would either become infinite or zero over time. The parameters that fully specify the model can be summarized as \( \theta = (\mu, \phi, \sigma^2, \beta^2) \). Some parameters worth extra consideration is \( \sigma^2 \), which is the volatility of the stochastic log-volatility, and is often called the vol-vol. Further, \( \mu \) is the drift of the log-returns, and \( \phi \) is a parameter regarding the behaviour of the volatility process. Some remarks regarding \( \phi \) is that if \( \phi \in (0, 1) \) the log-volatility is a weakly stationary process, and if \( \phi \geq 0 \) the model will have volatility clustering. Volatility clustering is a phenomena observed by Mandelbrot in [15], which was that large changes are often followed by large changes, and that small changes often were followed by small changes. Since this is a frequently occurring phenomena it is desirable with a model able to exhibit it.

Considering the model in Equation (3.4), it is possible to derive expressions for the transition densities \( q_0(x_{k-1}, x_k) \) and \( g_0(x_k, y_k) \). From Equation (3.4)
one can identify

\[ q_\theta(x_{k-1}, x_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ -\frac{(x_k - \phi x_{k-1})^2}{2\sigma^2} \right], \]

\[ g_\theta(x_k, y_k) = \frac{1}{\sqrt{2\pi\beta^2}} \exp \left[ -\frac{(y_k - \mu)^2}{2\beta^2} \exp(-x_k) - \frac{1}{2} x_k \right]. \] (3.5)

Deriving explicit expressions for the transition densities are of utmost interest, since they will be part of several vital quantities used in the algorithms. The parameters collected in \( \theta \) are unknown, and given a set of observations \( Y_{0:t} = y_{0:t} \) the model can be calibrated through estimating the parameters. This will be covered later in this chapter.

### 3.2 The PaRIS-algorithm

In Section 2.3 the bootstrap particle filter was introduced. Building on the theory covered regarding the bootstrap particle filter, the PaRIS-algorithm formulated in [10] will be presented in this section. As mentioned in Section 2.3, a problem with the bootstrap particle filter is the issue of degenerating particle trajectories. In [10] a solution to this problem is proposed by using the backward decomposition. In order to use the backward decomposition defined in Equation (2.13), one is required to approximate each backward kernel \( \overrightarrow{Q}_{\phi_{s,\theta}} \), \( s \in \mathbb{N} \). This can be done through Monte Carlo estimation as

\[ \overrightarrow{Q}_{\phi_{s,\theta}} f(x) = \sum_{i=1}^{N} \frac{w^s_i q_\theta(\xi^i_s, x)}{\sum_{l=1}^{N} w^s_l q_\theta(\xi^l_s, x)} f(\xi^i_s), \quad (x, f) \in X \times F(X), \] (3.6)

where \( f \in F(X) \). By replacing \( \overrightarrow{Q}_{\phi_{s,\theta}} \) with \( \overrightarrow{Q}_{\phi_{s,\theta}^N} \) in Equation (2.15), and assuming estimates \( \{\tilde{\tau}^i_t\}_{i=1}^{N} \) of \( \{T_t h_t(\xi^i_t)\}_{i=1}^{N} \) are available, it is possible to recursively update the estimates as

\[ \tilde{\tau}^i_{t+1} = \sum_{j=1}^{N} \frac{w^j_i q_\theta(\xi^j_t, \xi^i_{t+1})}{\sum_{l=1}^{N} w^j_l q_\theta(\xi^l_t, \xi^i_{t+1})} (\tilde{\tau}^j_t + \tilde{h}_t(\xi^j_t, \xi^i_{t+1})). \] (3.7)

In practice, the recursion is initialized by letting \( \tilde{\tau}^i_0 = 0, \quad i = 1, \ldots, N \). Using these estimates, the expectation \( \phi_{0:t+1|t;\theta} h_{t+1} \) can be approximated as

\[ \phi_{0:t+1|t;\theta} h_{t+1} = \frac{1}{N} \sum_{i=1}^{N} \tilde{\tau}^i_{t+1}. \] (3.8)
What makes this method promising is that it both allows for online estimation of $\phi_{0,t+1|t,\theta}h_{t+1}$ and also requires few variables to be stored in the memory. However, it suffers from a problem with computational complexity, since at every time step $N$ terms needs to be summed and computed for each particle. This results in a scenario where one is required to keep the number of particles low in order to avoid a slow algorithm, which results in low precision.

The PaRIS-algorithm works around this problem by replacing Equation (3.7) with the Monte Carlo approximation

$$\tilde{\tau}_t^{j(i)} = \frac{1}{N} \sum_{j=1}^{N} \left( \tau_t^{j(i)} + \tilde{h}_t(\xi_t^{j(i)}; \xi_{t+1}^{i}) \right),$$

(3.9)

with $\tilde{N} \in \mathbb{N}^*$ being the sample size of the Monte Carlo estimate. Further, $\{J_t^{(i,j)}\}_{j=1}^{N}$ are i.i.d samples from $Pr(\{w_t^j q_\theta(\xi_t^j; \xi_{t+1}^i)\}_{j=1}^{N})$. Drawing these samples can be done through rejection-sampling [16], which is done by first drawing $J^* \sim Pr(\{w^*_t\}_{t=1}^{N})$ and thereafter accepting the draw with probability $q_\theta(\xi_{t+1}^j; \xi_{t+1}^i)/\tilde{\epsilon}$. Using the Monte Carlo estimate, it is possible to obtain an estimate of $\phi_{0,t+1|t,\theta}h_{t+1}$ as

$$\hat{\phi}_{0,t+1|t,\theta}h_{t+1} \triangleq \frac{1}{N} \sum_{i=1}^{N} \tilde{\tau}_t^{i}. \tag{3.10}$$

### 3.2.1 Tangent filter estimation

The PaRIS-Algorithm can effectively be implemented to estimate the tangent filter specified in Equation (2.17). The algorithm allows for online estimation of the flow $\{\eta_{t,\theta}\}_{t \in \mathbb{N}}$ as new observations become available. By targeting the predictor distributions through the use of a particle filter, while recursively updating estimates $\{\tau_t^i\}_{i=1}^{N} \approx \{T_{t,\theta}h_{t,\theta}(\xi_t^i)\}_{i=1}^{N}$ as outline in Equation (3.9), it is possible to derive estimates of the tangent filters. Note that $h_{t,\theta}$ is the complete data score function defined as

$$h_{t,\theta}(x_0,t) = \sum_{s=0}^{t-1} \tilde{h}_{s,\theta}(x_{s:s+1}), \quad \text{with} \quad \tilde{h}_{s,\theta}(x_{s:s+1}) = \nabla_\theta \log g_{s,\theta}(x_s) + \nabla_\theta \log q_\theta(x_s, x_{s+1}).$$

(3.11)

For any $f_t \in F(\mathcal{X})$, the PaRIS-algorithm returns an estimate

$$\hat{\eta}_{t,\theta}^N f_t = \frac{1}{N} \sum_{i=1}^{N} \left( \tau_t^i - \frac{1}{N} \sum_{j=1}^{N} \tau_t^j \right) f_t(\xi_t^i), \tag{3.12}$$
at each time step \( t \). This estimate targets \( \eta_{t, \theta} f_t \). In [10] the algorithm is summarized in pseudo-code, and the formulation is presented below. Note that every line is repeated for \( i \in (1, \ldots, N) \).

### Algorithm 1 PaRIS

1: **Input:** \((y_{0:t_{\text{end}}}, \theta)\)  
2: draw \( \xi^i_0 \sim \nu \)  
3: set \( \tau^i_0 \leftarrow 0 \)  
4: for \( t \leftarrow 0, 1, \ldots, t_{\text{end}} - 1 \) do  
   5: set \( w^i_t \leftarrow g_{t, \theta}(\xi^i_t) \)  
   6: draw \( I^i_t \sim Pr(\{w^l_t\}_{l=1}^N) \)  
   7: draw \( \xi^i_{t+1} \sim q_{\theta}(\xi^i_t, \ldots, \xi^i_{t+1}) \)  
   8: for \( j \leftarrow 1, 2, \ldots, N \) do  
      9: draw \( J^{(i,j)}_{t+1} \sim Pr(\{w^l_t q_{\theta}(\xi^l_t, \xi^i_{t+1})\}_{l=1}^N) \)  
   10: end for  
   11: set \( \tau^i_{t+1} \leftarrow \sum_{j=1}^N \left( \tau^l_{t+1} + h_{t, \theta}(\xi^l_{t+1}, \xi^i_{t+1}) \right) \)  
   12: set \( \bar{\tau}_{t+1} \leftarrow \sum_{l=1}^N \tau^l_{t+1} \)  
   13: set \( \eta_{t+1, \theta} \leftarrow \sum_{l=1}^N (\tau^l_{t+1} - \bar{\tau}_{t+1}) \delta^l_{t+1} \)  
   14: end for

Note that line 9 in the algorithm above is performed using the rejection sampling described previously. The rejection sampling can be summarized in the pseudo-code below.

### Algorithm 2 draw \( J^{(i,j)}_{t+1} \) using rejection sampling

1: **Input:** \(\{w^l_t\}_{l=1}^N, \{\xi^l_t\}_{l=1}^N, \xi^i_{t+1}\)  
2: set accepted\( \leftarrow \) FALSE  
3: while accepted=FALSE do  
   4: draw \( J^* \sim Pr(\{w^l_t\}_{l=1}^N) \)\(^1\)  
   5: draw \( U \sim Unif(0, 1) \)  
   6: if \( U \leq q_{\theta}(\xi^*, \xi^i_{t+1})/\epsilon \) then  
      7: set \( J^{(i,j)}_{t+1} \leftarrow J^* \)  
      8: set accepted\( \leftarrow \) TRUE  
   9: end if  
10: end while

\(^1\)Here \( Pr(\{w^l_t\}_{l=1}^N) \) denotes the categorical distribution on \( \{1, \ldots, N\} \), meaning that the probability of drawing index \( J^* = j \) is \( P(J^* = j) = w^j_t/\Omega_t, \ j = 1, \ldots, N. \)
3.3 PaRIS-Based Online Parameter Estimation

The algorithm outlined in Section 3.2 estimates the flow of filter derivatives for a fully specified model, with known parameter $\theta$. However, the state-space model for stochastic volatility introduced in Section 3.1 is not fully specified, since the parameters collected in $\theta$ are not initially unknown. As mentioned in Section 2.1.3, likelihood optimization of incomplete data models can be applied in order to obtain an estimate of $\theta$, given a state space model and observations $Y_{0:t} = y_{0:t}$. As described in Section 2.1.3 likelihood optimization of incomplete data models consists of maximizing the likelihood defined in Equation (2.18) with respect to $\theta$. Since the optimization problem is of complex nature, due to the fact that the log-likelihood $l_\theta(y_{0:t})$ is not possible to express in a closed form, an approximation of the optimization will be performed. As described previously, the two most common approaches to perform this approximation are the EM-algorithm and gradient-based methods. In [10] a gradient method using the PaRIS-algorithm is presented, which uses the approximated filter derivatives from the PaRIS in order to approximate the log-likelihood. The PaRIS-based parameter estimator proposed by Olsson and Westerborn in [10] is an online estimator, which is relevant to this thesis since an online estimator is to be compared to an offline.

The optimization problem to be solved is

$$\theta^* = \arg\max_{\theta \in \Theta} L_\theta(y_{0:t}),$$

(3.13)

which is equivalent to solving

$$\theta^* = \arg\max_{\theta \in \Theta} l_\theta(y_{0:t}),$$

(3.14)

where $l_\theta(y_{0:t}) \triangleq \log L_\theta(y_{0:t})$. Finding the optimal $\theta^*$ can therefore be done by determining the root to $\nabla_\theta l_\theta(y_{0:t})$, which can be done through stochastic approximation via the Robbins-Monro scheme. Consider the setup

$$\theta_{n+1} = \theta_n + \gamma_{n+1} \nabla_\theta l_\theta(y_{0:t})|_{\theta=\theta_n}.$$  

(3.15)

As mentioned in Section 2.1.3 this setup results in an offline estimator, since the gradients of the objective function are required to be recalculated as new observations become available. The estimator proposed by Olsson and Westerborn in [10] uses the known decomposition $l_\theta(y_{0:t}) = \sum_{s=0}^t l_\theta(y_s|y_{0:s-1})$.
with \( l_\theta(y_0|y_{0:-1}) = l_\theta(y_0) \). By using the decomposition it is possible to derive \( \nabla_\theta l_\theta(y_{0:t}) = \sum_{s=0}^{t} \nabla_\theta l_\theta(y_s|y_{0:s-1}) \). Since the objective function \( l_\theta(y_{0:t}) \) can be expressed as a sum of the one-step predictor likelihoods, the optimization problem could be cast into the framework of stochastic optimization through the use of stochastic sub-gradients. Instead of focusing on the gradient of \( l_\theta(y_{0:t}) \) one can instead focus on the gradient of each one-step predictor log-likelihood. Rewriting the updating scheme of \( \theta_n \) using the stochastic sub-gradient method consists of replacing \( \nabla_\theta l_\theta(y_{0:t})|_{\theta = \theta_{n-1}} \) with its stochastic sub-gradient \( \nabla_\theta l_\theta(y_n|y_{0:n-1})|_{\theta = \theta_{n-1}} \). This results in the following updating scheme

\[
\theta_n = \theta_{n-1} + \gamma_n \nabla_\theta l_\theta(y_n|y_{0:n-1})|_{\theta = \theta_{n-1}}. \tag{3.16}
\]

One now needs to compute the sub-gradient \( \nabla_\theta l_\theta(y_n|y_{0:n-1}) \). This can be done by first deriving an expression for \( L_\theta(y_n|y_{0:n}) \), proceeded by using the definition \( l_\theta(y_n|y_{0:n}) \triangleq \log(L_\theta(y_n|y_{0:n})) \), and finally differentiating the expression. Starting with \( L_\theta(y_n|y_{0:n}) \), one can rewrite the one step predictor likelihood as

\[
L_\theta(y_n|y_{0:n}) = \frac{L_\theta(y_{0:n})}{L_\theta(y_{0:n-1})}. \tag{3.17}
\]

Using Equation (2.8) one can then express \( L_\theta(y_{0:n}) \) and \( L_\theta(y_{0:n-1}) \) as

\[
L_\theta(y_{0:n}) = \int \cdots \int f(x_n) \left( \prod_{m=0}^{n} g_m(x_m) \right) (\nu(dx_0) \prod_{i=0}^{n-1} Q_\theta(x_i, x_{i+1})) \phi_{n:n|\theta}\int
\]

\[
L_\theta(y_{0:n-1}) = \int \cdots \int f(x_n) \left( \prod_{m=0}^{n-1} g_m(x_m) \right) (\nu(dx_0) \prod_{i=0}^{n-1} Q_\theta(x_i, x_{i+1})) \phi_{n:n-1|\theta}\int \tag{3.18}
\]

Inserting the expressions from Equation (3.18) in Equation (3.17) one then obtains the following

\[
L_\theta(y_n|y_{0:n}) = \frac{\phi_{n:n-1|\theta}\int g_{n|\theta}}{\phi_{n:n|\theta}\int}. \tag{3.19}
\]

Using the filter recursion from Equation (2.10), the expression can be rewritten as

\[
L_\theta(y_n|y_{0:n}) = \frac{\pi_{n:|\theta} g_{n|\theta}}{\pi_{n:|\theta} g_{n|\theta}} \tag{3.20}
\]
By taking the logarithm of the expression, it is possible to express the one step predictor log-likelihood as
\[ l_\theta(y_n | y_{0:n-1}) = \log(\pi_{n;\theta} g_{n;\theta}). \]
Finally differentiating the expression results in the sub-gradient, which can be written as

\[ \nabla_{\theta} l_\theta(y_n | y_{0:n-1}) = \frac{\pi_{n;\theta}(\nabla_{\theta} g_{n;\theta}) + \eta_{n;\theta} g_{n;\theta}}{\pi_{n;\theta} g_{n;\theta}}, \] (3.21)

with \( \eta_{n;\theta} \) being the tangent filter defined in Equation (2.16).

Since the quantities \( \pi_{n;\theta}(\nabla_{\theta} g_{n;\theta}), \eta_{n;\theta} g_{n;\theta} \) and \( \pi_{n;\theta} g_{n;\theta} \) in general may not be expressed in a closed form, the estimator proposed by Olsson and Westerborn in [10] proceeds by approximating the quantities, using the PaRIS-algorithm. The updating scheme for the parameter \( \theta \) proposed in [10] at observation \( n + 1 \) can be summarized as

\[ \theta_{n+1} = \theta_n + \gamma_{n+1} \hat{\zeta}_{n+1}, \] (3.22)

where \( \hat{\zeta}_{n+1} \) is an particle estimation, using the PaRIS-algorithm described in Section 3.2, of the gradient of the one step predictor log-likelihood in Equation (3.21).

Given a set of observations \( Y_{0:t} = y_{0:t} \) and a state-space model with unknown parameter \( \theta \), the proposed estimator finds an estimate \( \theta^* \) of the true parameter \( \theta \) by “sweeping” through the observations. As new observations \( Y_{t+1:t'} = y_{t+1:t'} \) become available, the algorithm is able to update the parameter estimation by continuing from where it stopped. Hence, it is an online algorithm. Olsson and Westerborn provides a describing section of pseudo-code for the algorithm in [10], which is presented below.
Algorithm 3 PaRIS-based online parameter estimation

1: Input: $(y_{0:n})$
2: set $\theta_0$ arbitrarily
3: draw $\xi^i_0 \sim \nu$
4: set $\tau^i_0 \leftarrow 0$
5: for $t \leftarrow 0, 1, \ldots, t_{end} - 1$ do
6: set $w^i_t \leftarrow g_{\theta_t}(\xi^i_t, y_t)$
7: draw $I_i \sim Pr\{\{w^i_l\}_{l=1}^n\}$
8: draw $\xi^i_1 \sim q_{\theta_t}(\xi^i_t, \ldots)$
9: for $j \leftarrow 1, 2, \ldots, \tilde{N}$ do
10: draw $J^{(i,j)}_{t+1} \sim Pr\{\{w^i_lq_{\theta_t}(\xi^i_l, \xi^i_{l+1})\}_{l=1}^N\}$
11: end for
12: set $\tau^i_{t+1} \leftarrow \tilde{N}^{-1} \sum_{j=1}^{\tilde{N}} \left( \tau^{j(i,j)}_{t+1} + \tilde{h}_{t,\theta_t}(\xi^i_{t+1}, \xi^i_{t+1}) \right)$
13: set $\bar{\tau}_{t+1} \leftarrow N^{-1} \sum_{l=1}^N \tau^i_{l+1}$
14: set $\zeta^1_{t+1} \leftarrow N^{-1} \sum_{l=1}^N \nabla g_{\theta_l}(\xi^i_{l+1}, y_{t+1})$
15: set $\zeta^2_{t+1} \leftarrow N^{-1} \sum_{l=1}^N (\bar{\tau}^i_{t+1} - \tau^i_{t+1}) g_{\theta_l}(\xi^i_{l+1}, y_{t+1})$
16: set $\zeta^3_{t+1} \leftarrow N^{-1} \sum_{l=1}^N g_{\theta_l}(\xi^i_{l+1}, y_{t+1})$
17: set $\theta_{t+1} \leftarrow \theta_t + \gamma_{t+1} \frac{\zeta^1_{t+1} + \zeta^2_{t+1} \zeta^3_{t+1}}{\zeta^3_{t+1}}$
18: end for

As explained previously, rejection sampling is implemented to draw $J^{(i,j)}_{t+1}$ according to Algorithm 2.

3.4 Batch EM Parameter Estimation

As mentioned previously, there are mainly two approaches when one is interested in the task of parameter estimation. The estimator formulated in Section 3.3 is a so called online estimator, based on the approach where one is using the gradient of the objective function. However, parameter estimation can also be performed using an offline algorithm, based on the EM-algorithm described in Section 2.1.3. One is then required to first derive an expression for the intermediate quantity $Q(\theta; \theta^i)$, and secondly maximize the quantity with respect to $\theta$. Starting with the definition of the intermediate quantity, it is possible to derive

$$Q(\theta; \theta^i) \propto \sum_{k=0}^n E_{\theta^i} [\log g_k(X_k; \theta)|Y_{0:n}] + \sum_{k=0}^{n-1} E_{\theta^i} [\log q(X_k, X_{k+1}; \theta)|Y_{0:n}] .$$

(3.23)
In the particular case of the stochastic volatility model, using the definitions of the transition and emission densities in Equation (3.5), Equation (3.23) can be written as

\[
Q(\theta; \theta^i) \propto (n + 1) \log \beta^2 + \frac{1}{\beta^2} \left( S_1 - 2 \mu S_2 + \mu^2 S_3 \right) + \\
+ n \log \sigma^2 + \frac{1}{\sigma^2} \left( Z_1 - 2 \phi Z_2 + \phi^2 Z_3 \right),
\]

(3.24)

which is done in Appendix B. The statistics \( S_1, S_2, S_3, Z_1, Z_2, Z_3 \) are defined as

\[
S_1 = E_{\theta^i} \left[ \sum_{k=0}^{n} Y_k^2 e^{-X_k} | Y_{0:n} \right], \\
S_2 = E_{\theta^i} \left[ \sum_{k=0}^{n} Y_k e^{-X_k} | Y_{0:n} \right], \\
S_3 = E_{\theta^i} \left[ \sum_{k=0}^{n} e^{-X_k} | Y_{0:n} \right], \\
Z_1 = E_{\theta^i} \left[ \sum_{k=0}^{n-1} X_{k+1}^2 | Y_{0:n} \right], \\
Z_2 = E_{\theta^i} \left[ \sum_{k=0}^{n-1} X_{k+1} X_k | Y_{0:n} \right], \\
Z_3 = E_{\theta^i} \left[ \sum_{k=0}^{n-1} X_k^2 | Y_{0:n} \right].
\]

(3.25)

Following the E-step comes the M-step, where one in this case maximizes the intermediate quantity by searching the roots of the gradient of \( Q(\theta; \theta^i) \). Let \( \theta^* = (\mu^*, \phi^*, (\sigma^2)^*, (\beta^2)^*) \) such that

\[
\frac{\partial}{\partial \mu} Q(\theta^*; \theta^i) = 0 \\
\frac{\partial}{\partial \phi} Q(\theta^*; \theta^i) = 0 \\
\frac{\partial}{\partial \sigma^2} Q(\theta^*; \theta^i) = 0 \\
\frac{\partial}{\partial \beta^2} Q(\theta^*; \theta^i) = 0.
\]
Using simple rules of derivation, which are presented in Appendix C, one obtains

\[
\mu^* = \frac{S_2}{S_3}
\]

\[
\phi^* = \frac{Z_2}{Z_3}
\]

\[
(\sigma^2)^* = \frac{(Z_1 - 2\phi^*Z_2 + (\phi^*)^2Z_3}{n}
\]

\[
(\beta^2)^* = \frac{(S_1 - 2\mu^*S_2 + (\mu^*)^2S_3)}{n + 1}.
\] (3.26)

At each instance of the EM-algorithm the parameters are now updated according to its optimal values defined in Equation (3.26). However, in order to update the parameters one is required to estimate the statistics. Estimating the statistics can effectively be done implementing the PaRIS-algorithm in Algorithm 1. By letting

\[
\tilde{h}_{t,\theta'}(x_t, x_{t+1}) \triangleq \begin{bmatrix} y_t^2 e^{-x_t} \\ y_t e^{-x_t} \\ e^{-x_t} \\ x_{t+1}^2 \\ x_{t+1} x_t \\ x_t^2 \end{bmatrix},
\]

it is possible to obtain estimates of the statistics of interest. This can be done by using the output of the PaRIS-algorithm, giving an estimate of the statistics as \( \sum_{i=1}^{N} \frac{w_i}{w_n} \tau_i^n \). Using the estimations of the statistics one can now compute the intermediate quantity, and further compute the optimal parameter values. By repeating the EM-algorithm and for each instance estimating the statistics, it is possible to achieve an offline estimator of the maximum likelihood parameter values. The offline EM-estimator can be summarized in pseudo code below:
Algorithm 4 PaRIS-based offline parameter estimation

1: Input: \((y_{0:t_{\text{end}}})\)
2: set \(\theta_0\) arbitrarily
3: for \(k \leftarrow 0, 1, \ldots, \#EMs\) do
4:   draw \(\xi^i_0 \sim \nu\)
5:   set \(\tau^i_0 \leftarrow 0\)
6:   for \(t \leftarrow 0, 1, \ldots, t_{\text{end}} - 1\) do
7:     set \(w^i_t \leftarrow g_{\theta_k}(\xi^i_t, y_t)\)
8:     draw \(I^i_t \sim P_r\{w^i_1, \ldots, w^i_{t-1}\}\)
9:     draw \(\xi^i_{t+1} \sim q_{\theta_k}(\xi^i_t, I^i_t)\)
10:    for \(j \leftarrow 1, 2, \ldots, N\) do
11:       draw \(J^{i,j}(t+1) \sim P_r\{w^i_l, q_{\theta_k}(\xi^i_l, \xi^i_{t+1})\}\)
12:    end for
13:   set \(\tau^i_{t+1} \leftarrow N^{-1} \sum_{j=1}^N \left(\frac{J^{i,j}_{t+1}}{\tilde{r}} + \tilde{h}_{t, \theta_k}(\xi^i_{t+1}, \xi^i_{t+1})\right)\)
14: end for
15: set \(w^i_{t+1} \leftarrow g_{\theta_k}(\xi^i_{t+1}, y_{t+1})\)
16: set \(\Omega_{t+1} \leftarrow \sum_{l=1}^N w^i_{t+1}\)
17: set \(S_1 \leftarrow \sum_{l=1}^N \frac{w^i_{t+1}}{\Omega_{t+1}} r_{t+1}\)
18: set \(S_2 \leftarrow \sum_{l=1}^N \frac{w^i_{t+1}}{\Omega_{t+1}} r_{t+1}\)
19: set \(S_3 \leftarrow \sum_{l=1}^N \frac{w^i_{t+1}}{\Omega_{t+1}} r_{t+1}\)
20: set \(Z_1 \leftarrow \sum_{l=1}^N \frac{w^i_{t+1}}{\Omega_{t+1}} r_{t+1}\)
21: set \(Z_2 \leftarrow \sum_{l=1}^N \frac{w^i_{t+1}}{\Omega_{t+1}} r_{t+1}\)
22: set \(Z_3 \leftarrow \sum_{l=1}^N \frac{w^i_{t+1}}{\Omega_{t+1}} r_{t+1}\)
23: set \(\theta^{(1)}_{k+1} \leftarrow S_2/S_3\)
24: set \(\theta^{(2)}_{k+1} \leftarrow Z_2/Z_3\)
25: set \(\theta^{(3)}_{k+1} \leftarrow (Z_1 - 2\theta^{(2)}_{k+1} Z_2 + \theta^{(2)}_{k+1} Z_3)/t_{\text{end}}\)
26: set \(\theta^{(4)}_{k+1} \leftarrow (S_1 - 2\theta^{(1)}_{k+1} S_2 + \theta^{(1)}_{k+1} S_3)/(t_{\text{end}} + 1)\)
27: end for

This estimator will be used in Section 4.2.2 in order to validate the online estimated parameter values, and to provide an alternative solution which the online algorithm can be compared to.
Chapter 4

Case Study

The purpose of this chapter is to implement the algorithms described in Section 3.3 and Section 3.4 to the hidden Markov volatility model formulated in Section 3.1. Prior to using data of actual stock prices, data will be simulated using the HMM stochastic volatility model, with known parameters. By comparing the estimated parameters, obtained by applying the PaRIS-based parameter estimators to the simulated data, to the parameters used to simulate the data, one can determine whether the estimators finds the true parameters. Once it has been confirmed that the algorithms finds the true parameters, one can proceed to using actual stock prices. In order to validate the parameters estimated for the actual stock prices, an offline estimation of the parameters will also be performed. The offline estimation will be performed using a batch EM estimation. The implementations of the algorithms on both the simulated data and the actual stock data were carried out using R. Following the implementations on simulated and stock data comes two experiments where the uncertainty of the two algorithms are compared.

4.1 Algorithm Validation

4.1.1 Validation of online algorithm simulated data

In order to validate the online algorithm, parameter estimation was first performed on data generated using a model with specified parameter values. Consider the model specified in Equation (3.4), and let $\theta = (\mu, \phi, \sigma^2, \beta^2) = (0.75, 0.9, 0.2, 1)$. By assuming a initial distribution $\nu = \mathcal{N}(0, \sigma^2)$ of $X_0$, and drawing $x_0$ from $\nu$, it is possible to generate both unobserved data $X_{0:t_{end}} = x_{0:t_{end}}$ and observed data $Y_{0:t_{end}} = y_{0:t_{end}}$ for some time period $[0, t_{end}]$. Using
the generated data, it is now possible to run the parameter estimation described in Section 3.3 and obtain estimates of the parameters. The algorithm requires quantities \( \tilde{h}_{s, \theta}(x_s, x_{s+1}) \) and \( \nabla_{\theta} g_{s, \theta}(x_s) \), and the derivation of these quantities can be found in Appendix A.

The validation was performed using a simulated data set of 50,000 observations, with a fraction of the data presented in the figure below.

![First 500 observations of simulated data](image)

**Figure 4.1:** First 500 observations of the simulated log-returns, using the specified stochastic volatility model with parameters \( \theta = (\mu, \phi, \sigma^2, \beta^2) = (0.75, 0.9, 0.2, 1) \).

Using the simulated data, the PaRIS-based online estimator could now be implemented in order to obtain online estimates of the parameters. In order to verify the convergence of the estimator, the algorithm was implemented five times on the data set, each time with randomized initial values for \( \theta_0 \). Thus, resulting in five separate learning trajectories for each of the parameters. Trajectories converging to the true parameter values thus imply that the algorithm estimates the parameters of the model accurately, without
relying on specific values of the initial parameters in $\theta_0$. The parameter estimation was performed using $N = 1400$ particles, \(\{\gamma_t\}_{t=1}^{50,000} = \{t^{-0.6}\}_{t=1}^{50,000}\) and $\tilde{N} = 2$ as precision parameter. Further, the value of $\bar{\epsilon}$ in the rejection sampling was set to $\bar{\epsilon} = \frac{1}{\sqrt{2\pi\sigma_t^2}}$, with $\sigma_t^2$ being the estimated value of $\sigma^2$ at iteration $t$. This was done to satisfy the requirement $\bar{\epsilon} \geq q_\theta(\xi_t, \xi_{t+1})$. The results from the parameter estimation on the simulated data are presented below in graphs and tables.

![Graphs and Tables](image)

**Figure 4.2**: Trajectories of PaRIS-based online parameter estimator for 5 implementations with randomized initial values $\theta_0$. Note that red lines are representing the true parameter values. Each implementation is using $N = 1400$ particles and $\tilde{N} = 2$ as precision parameter.

<table>
<thead>
<tr>
<th>Run</th>
<th>$\mu$</th>
<th>$\phi$</th>
<th>$\sigma^2$</th>
<th>$\beta^2$</th>
<th>Run time [h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7793</td>
<td>0.8853</td>
<td>0.2184</td>
<td>0.9981</td>
<td>15.31</td>
</tr>
<tr>
<td>2</td>
<td>0.7772</td>
<td>0.8818</td>
<td>0.2221</td>
<td>1.0074</td>
<td>9.81</td>
</tr>
<tr>
<td>3</td>
<td>0.7783</td>
<td>0.8803</td>
<td>0.2231</td>
<td>1.0117</td>
<td>9.15</td>
</tr>
<tr>
<td>4</td>
<td>0.7783</td>
<td>0.8824</td>
<td>0.2121</td>
<td>1.0054</td>
<td>19.51</td>
</tr>
<tr>
<td>5</td>
<td>0.7783</td>
<td>0.8825</td>
<td>0.2250</td>
<td>1.0095</td>
<td>9.93</td>
</tr>
</tbody>
</table>

Table 4.1: Estimated parameters for each of the 5 estimations with corresponding computation times.
From Figure 4.2 one can clearly observe that the learning trajectories from the PaRIS-based online parameter estimator converge to the true parameter values. In both the learning trajectories for $\sigma^2$ and $\beta^2$ one can identify jumps in the beginning of the algorithm. These jumps might occur as a result of a negative updated value of the parameter, which might be the result of the algorithm updating the parameters with too large steps for small values of $t$. Since the variance of a random variable is non-negative, this is not allowed, thus the algorithm updates the parameter using the absolute value. However, as the algorithm proceeds through the data set, both $\sigma^2$ and $\beta^2$ converge towards the true parameter values. Further, Table 4.1 confirms the convergence of each parameter by providing numerical values of the estimates.

Further, as described in Section 2.3.2 it is possible to identify potential issues with degenerating weights by calculating the $CV_N$ and $N_{eff}$, defined in Equation (2.30) and Equation (2.31). This was done for the last estimation run, and the results are presented below.

![Coefficient of variation and efficient sample size](image)

Figure 4.3: *Coefficient of variation and efficient sample size* for the last estimation run of the online implementation of the PaRIS-algorithm on the simulated data. An overall high efficient sample size $N_{eff}$ indicates a stable algorithm without degenerating weights. Final values after the last observation was $CV_N = 0.4011$ and $N_{eff} = 1205.9888$, with an average efficient sample size $\bar{N}_{eff} = 1210$, which can be considered close to the total number of particles, which was 1400.
Further, a histogram of the efficient sample size can be used to clearer visualize the distribution of the efficient number of particles in the algorithm.

![Histogram of Efficient Sample Size](image)

Figure 4.4: Histogram of efficient sample size for the last estimation run of the online implementation of the PaRIS-algorithm on the simulated data. A high frequency for the efficient sample size with values close to the total number of particles 1400 is desirable.

As can be seen in Figure 4.3, the algorithm in general has a low coefficient of variation, resulting in an efficient sample size close to the total number of particles. One can however identify some spikes in the $CV_N$, which could mean that some steps in the updating of $\theta$ could be inaccurate. However, combining the results from Figure 4.2 and Table 4.1 with Figure 4.3 and Figure 4.4, provides an overall impression that the algorithm is stable and provides accurate estimates.

In addition to the implementation above, the same validation was performed using a reduced particle sample with $N = 600$ and $t_{end} = 30,000$. This was done in order to validate that a smaller particle cloud and data set yielded
estimates converging to the true parameters. The results from this validation can be found in Appendix D.

4.1.2 Validation of offline algorithm

In addition to validating the online implementation of the algorithm on simulated data, the offline EM-estimator was also validated using a simulated data set. For the validation of the offline estimator, a smaller data set consisting of 1000 observations was generated, using both the same model and parameters as for the online-implementation. By applying the EM-implementation of the PaRIS-algorithm described in Section 3.4 on the simulated data set, maximum likelihood estimates of the true parameters were obtained. The estimates were generated from running the EM-algorithm 600 times, using $N = 300$ particles and $\tilde{N} = 2$ as precision parameter. By arbitrarily setting $\theta_0$ and updating $\theta_i, \quad i = 1, \ldots, 600$ according to the algorithm, it was possible to obtain estimates converging to the true parameters of the model. The results are presented in graphs below.
Figure 4.5: Learning trajectories of EM-estimations of model parameters $\theta = (\mu, \phi, \sigma^2, \beta^2)$. True parameter values $\theta^* = (0.75, 0.9, 0.2, 1)$ are indicated by red horizontal lines in each graph. Final estimation of parameters $\theta_{600} = (0.77, 0.89, 0.19, 0.95)$, with a run time of 5.70 hours.
Figure 4.6: Coefficient of variation and efficient sample size for offline EM-implementation of the PaRIS-algorithm on the simulated data. An overall high efficient sample size $N_{eff}$ indicates a stable algorithm without degenerating weights. The average efficient sample size for the last iteration of the EM-implementation could be computed as $\bar{N}_{eff} = 263$, which can be considered close to the total number of particles, which was 300.

Further, a histogram of the efficient sample size can be used to clearer visualize the distribution of the efficient number of particles in the algorithm.
Figure 4.7: Histogram of efficient sample size for the last estimation run of the offline EM-implementation of the PaRIS-algorithm on the simulated data. A high frequency for the efficient sample size with values close to the total number of particles 300 is desirable.

As can be seen in Figure 4.5 the EM-implementation of the PaRIS-algorithm is able to estimate the true parameters from a simulated data set. In the following section of this chapter, the offline EM-implementation will be compared to the online implementation of the PaRIS-algorithm when using actual stock data.

### 4.2 Implementation on Actual Stock Prices

Following the validation of the algorithms on simulated data, both the online and offline estimator were applied to real world stock data. The data used in the implementations were monthly price data of the Apple Inc, AAPL stock. By using the Yahoo Finance database it was possible to obtain a data set of monthly stock prices from the time-period (01-01-1980) up to the latest registered price. Note that before the data could be used in the algorithm, the
stock prices were required to be transformed into log-returns, which was done by \( Y_t = \log(S_t) - \log(S_{t-1}) \). The resulting log-return data set is presented below.

![Monthly log-Returns of AAPL](image)

**Figure 4.8:** Monthly log-returns from the AAPL stock from the time period (01-01-1980) to the latest registered price.

### 4.2.1 Online implementation

Once the stock prices had been transformed into log-returns, parameter estimation could be performed using the online algorithm as in Section 4.1.1, according to the algorithm described in Algorithm 3. Some minor alterations to the algorithm were made, in order to improve the performance. First of all, the number of particles was decreased from \( N = 1400 \) to \( N = 600 \) in order to decrease the computational time of the algorithm. Reducing the size of the particle cloud can be motivated by the results from Appendix D, which imply that a smaller number of particles and observations still results in converging learning trajectories. Secondly, to make sure that the learning trajectories converged, the algorithm was repeatedly applied to the data. This was done since the data set consisted of a relative small number of observations compared
to the simulated data set in Section 4.1.1. By repeating the data set, which consisted of approximately 400 observation, 75 times it was possible to create a larger set of observations consisting of approximately 35,000 observations. Letting the algorithm process this enlarged data set yielded in converging parameter estimates, which can be seen in Figure 4.9. Further, it showed that the algorithm suffered from some instability when estimating values of $\sigma^2$ and $\beta^2$ close to 0. Therefore, an alteration to the updating step of the estimation was made. By decreasing the step size of the online estimator for parameters $\sigma^2$ and $\beta^2$, if the current estimate of the parameter falls under 0.1, stable results were attainable. As for the implementation on simulated data, the algorithm was performed 5 times on the data. The results from the online implementation are presented graphically in Figure 4.9 and Figure 4.10, and numerically in Table 4.2.

![Figure 4.9: Learning trajectories of 5 online estimations of parameters. Using 5 randomized initial values for $\theta_0$, each individual implementation of the algorithm seems to converge towards the same value.](image-url)
Table 4.2: Estimated parameters for each of the 5 online estimations, with corresponding computation times.

<table>
<thead>
<tr>
<th>Run</th>
<th>( \mu )</th>
<th>( \phi )</th>
<th>( \sigma^2 )</th>
<th>( \beta^2 )</th>
<th>Run time [h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0156</td>
<td>0.9367</td>
<td>0.0775</td>
<td>0.0159</td>
<td>1.38</td>
</tr>
<tr>
<td>2</td>
<td>0.0165</td>
<td>0.9372</td>
<td>0.0825</td>
<td>0.0160</td>
<td>1.19</td>
</tr>
<tr>
<td>3</td>
<td>0.0156</td>
<td>0.9434</td>
<td>0.0711</td>
<td>0.0157</td>
<td>1.25</td>
</tr>
<tr>
<td>4</td>
<td>0.0155</td>
<td>0.9347</td>
<td>0.0764</td>
<td>0.0153</td>
<td>1.30</td>
</tr>
<tr>
<td>5</td>
<td>0.0151</td>
<td>0.9393</td>
<td>0.0754</td>
<td>0.0159</td>
<td>1.20</td>
</tr>
</tbody>
</table>

From Figure 4.9 one can observe that all 5 implementations of the algorithm converges towards similar parameter values. Further, the learning trajectories illustrate that the algorithm requires each implementation to be repeated over the data several times. This due to the fact that the data set only consists of approximately 400 observations, and the learning trajectories does not seem to converge until after at least 15,000 observations. Further, the numerical results presented in Table 4.2 provide additional information regarding the convergence of the learning trajectories. In order to investigate whether the algorithm suffers from problems with degenerating weights the efficient sample size and coefficient of variation were evaluated.
Figure 4.10: Coefficient of variation and efficient sample size for online implementation of the PaRIS-algorithm on the AAPL data. An overall high efficient sample size $N_{eff}$ indicates a stable algorithm without degenerating weights. The average efficient sample size for the last iteration of the online implementation could be computed as $\bar{N}_{eff} = 553$, which can be considered close to the total number of particles, which was 600.

Further, a histogram of the efficient sample size can be used to clearer visualize the distribution of the efficient number of particles in the algorithm.
Figure 4.11: Histogram of efficient sample size for the last estimation run of the online-implementation of the PaRIS-algorithm on the AAPL data. A high frequency for the efficient sample size with values close to the total number of particles 600 is desirable.

As can be seen in Figure 4.10 the overall efficient sample size is close to the total number of particles 600. This implies that the algorithm should not suffer problems with degenerating weights, which is the same conclusion that could be made regarding the implementation on simulated data.

4.2.2 Offline implementation

Once the online algorithm had been implemented on real stock data, the offline algorithm was implemented on the same data. This was done in order to provide some results which the results from the online implementation could be compared to. The main interests in comparing the two algorithms were, to first of all verify that both of the two algorithms estimates approximately the same parameter values, and secondly to evaluate which of the two implementations that approximates the parameters most effectively in regards of computational
time and precision. Using the same data set as in Section 4.2.1, and the offline algorithm validated in Section 4.1.2, it was possible to obtain parameter estimates from 5 implementations of the offline-algorithm. The results are presented below in tables and graphs.

![Figure 4.12: Learning trajectories of 5 offline estimations of parameters. Using 5 randomized initial values for \( \theta_0 \), each individual implementation of the offline algorithm seems to converge towards the same value.](image)

Table 4.3: Estimated parameters for each of the 5 offline estimations, with corresponding computation times.

<table>
<thead>
<tr>
<th>Run</th>
<th>( \mu )</th>
<th>( \phi )</th>
<th>( \sigma^2 )</th>
<th>( \beta^2 )</th>
<th>Run time [h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0249</td>
<td>0.9608</td>
<td>0.0443</td>
<td>0.0142</td>
<td>2.04</td>
</tr>
<tr>
<td>2</td>
<td>0.0241</td>
<td>0.9598</td>
<td>0.0468</td>
<td>0.0147</td>
<td>2.03</td>
</tr>
<tr>
<td>3</td>
<td>0.0245</td>
<td>0.9596</td>
<td>0.0452</td>
<td>0.0136</td>
<td>2.40</td>
</tr>
<tr>
<td>4</td>
<td>0.0244</td>
<td>0.9585</td>
<td>0.0486</td>
<td>0.0145</td>
<td>2.06</td>
</tr>
<tr>
<td>5</td>
<td>0.0236</td>
<td>0.9625</td>
<td>0.0443</td>
<td>0.0167</td>
<td>2.02</td>
</tr>
</tbody>
</table>
Figure 4.13: Coefficient of variation and efficient sample size for offline EM-implementation of the PaRIS-algorithm on the AAPL data. An overall high efficient sample size $N_{\text{eff}}$ indicates a stable algorithm without degenerating weights. The average efficient sample size for the last iteration of the EM-implementation could be computed as $\bar{N}_{\text{eff}} = 232$, which can be considered close to the total number of particles, which was 250.

Further, a histogram of the efficient sample size can be used to clearer visualize the distribution of the efficient number of particles in the algorithm.
Chapter 4. Case Study

Histogram of Efficient Sample Size

Figure 4.14: Histogram of efficient sample size for the last estimation run of the offline EM-implementation of the PaRIS-algorithm on the AAPL data. A high frequency for the efficient sample size with values close to the total number of particles 300 is desirable.

4.3 Experiments Regarding Uncertainty of Estimates

In this section two experiments will be performed. The main purpose of the two experiments is to investigate which of the two algorithms that provides the estimate with the least uncertainty. Experiment 1 focuses on the uncertainty with respect to the true parameter, while Experiment 2 focuses on the uncertainty with respect to the maximum likelihood estimate. During both of the experiments, the total number of particles was set to \( N = 300 \) and \( \tilde{N} = 2 \) for both the online and offline estimator. The true parameters were set as in Section 4.1.1. Setting the same \( N \) and \( \tilde{N} \) is of interest in order to being able to compare the performance of the two algorithms for the same computational budget. The performance will compared by evaluating the sample variance of
the estimates around the true parameter and the maximum likelihood estimate, which can be computed as

\[ s^2(\theta_{est}) = \frac{1}{M - 1} \sum_{m=1}^{M} (\theta_{est}^{(m)} - \theta^*)^2. \] (4.1)

In Equation (4.1), note that \( \theta_{est}^{(m)} \) is the estimated parameters of the \( m \):th implementation of the estimator, and \( \theta^* \) the true parameter values specified in Section 4.1.1 for Experiment 1. However, for Experiment 2 the value of \( \theta^* \) is a maximum likelihood estimate of the true parameters. This maximum likelihood estimate is produced by running a batch estimator with a large particle cloud and number of EM-iterations over the data. Further, \( M \) is the number of implementations of each algorithm to the data.

### 4.3.1 Experiment 1

The experiment was carried out in the following way. First \( M = 10 \) sets of 30,000 observations \( Y_{i,n} \) were generated as in Section 4.1.1. Thereafter, the online implementation of the PaRIS-algorithm was implemented once on each of the generated data sets. This resulted in 10 estimates of the true parameters. Secondly, \( M = 10 \) sets of 500 observations were generated. For each of the 10 data sets, the offline algorithm was implemented in order to estimate the parameters, which resulted in 10 estimates of the true parameters. The number of iterations for the offline algorithm was set to be 60. This was in order to make sure that the computational complexity of the two algorithms was the same. The total number of backwards samples performed for both algorithms were 18,000,000, since for the online algorithm \( N \cdot \tilde{N} \cdot 30,000 = 18,000,000 \) and for the offline \( N \cdot \tilde{N} \cdot 500 = 18,000,000 \). The resulting parameter estimates are presented in histograms below.
Figure 4.15: Histogram of final parameter estimates from online implementation. Sample variance could be computed as $s^2(\theta_{est}) = (0.00067, 0.00065, 0.00564, 0.00116)$, with each value corresponding to the sample variance of each parameter $s^2(\theta_{est}) = (s^2(\mu), s^2(\phi), s^2(\sigma^2), s^2(\beta^2))$. 

Estimates from online algorithm
Figure 4.16: Histogram of final parameter estimates from offline implementation. Sample variance could be computed as $s^2(\theta_{est}) = (0.00240, 0.00131, 0.00854, 0.04291)$, with each value corresponding to the sample variance of each parameter $s^2(\theta_{est}) = (s^2(\mu), s^2(\phi), s^2(\sigma^2), s^2(\beta^2))$.

### 4.3.2 Experiment 2

In the second experiment one data set of 500 observations was generated. Thereafter, the offline algorithm was repeated $M = 10$ times on the data set, resulting in 10 estimates. Once again 60 iterations were performed to ensure the same computational complexity. Thereafter the data set was repeated 60 times, similar to what was done to the stock data in Section 4.2.1. The online algorithm was then performed on the repeated data $M = 10$ times, resulting in 10 estimates. As for the previous experiment, the results are presented in histograms.
Figure 4.17: Histogram of final parameter estimates from online implementation. Sample variance could be computed as $s^2(\theta_{est}) = (0.00003, 0.00045, 0.00477, 0.00020)$, with each value corresponding to the sample variance of each parameter $s^2(\theta_{est}) = (s^2(\mu), s^2(\phi), s^2(\sigma^2), s^2(\beta^2))$. 

Estimates from online algorithm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>0.792, 0.796, 0.800</td>
</tr>
<tr>
<td>$\phi$</td>
<td>0.835, 0.845, 0.855</td>
</tr>
<tr>
<td>$\sigma^2$</td>
<td>0.18, 0.20, 0.22</td>
</tr>
<tr>
<td>$\beta^2$</td>
<td>0.96, 0.97, 0.98, 0.99, 1.00</td>
</tr>
</tbody>
</table>
Estimates from offline algorithm

Figure 4.18: Histogram of final parameter estimates from offline implementation. Sample variance could be computed as \( s^2(\theta_{est}) = (0.00002, 0.01351, 0.03036, 0.00237) \), with each value corresponding to the sample variance of each parameter \( s^2(\theta_{est}) = (s^2(\mu), s^2(\phi), s^2(\sigma^2), s^2(\beta^2)) \).
Chapter 5

Comparison of Online and Offline Estimator

In this chapter the results from the case study will be used to compare the algorithms. The results from the two algorithms on both simulated and the AAPL data will be compared in terms of computational time, stability, accuracy and precision. Further, the results from the two experiments in Section 4.3 will be compared in terms of the accuracy and variance of the estimated parameters.

5.1 Comparison on Simulated Data

Prior to comparing the performance of the two algorithms on the AAPL data, it is of interest to compare the results from the simulated data. As can be seen in Section 4.1.1 and Section 4.1.2, both of the two algorithms were able to estimate the parameters generating the simulated data with a relative high accuracy. From the results for the online implementation presented in Table 4.1, one can observe that for the 5 online implementations, the drift parameter $\mu$ is estimated with at least an accuracy of 3.2% to the true value and $\phi$ with at least 2.2% accuracy. Further, $\sigma^2$ is estimated with at least 12.5% accuracy, and finally $\beta^2$ with at least 1.2% accuracy. The results from the offline implementation presented in Figure 4.5, indicates that the offline implementation also was able to obtain accurate estimates, estimating $\mu$ with an accuracy of 2.7% and $\phi$ with 1.1%. The offline algorithm was also able to estimate both $\sigma^2$ and $\beta^2$ with an accuracy of 5%. Comparing the accuracy of the two implementations suggests, that on simulated data both the online and the offline implementation are able to estimate the parameters of interest with a similar accuracy. However, the run time of the offline implementation was consider-
ably shorter than for the online implementation. The fastest of the 5 online implementations was able to finish in just under 10 hours, compared to the offline implementation that finished in just under 6 hours. However, from the learning trajectories in Figure 4.2, one can observe that the trajectories seems to converge to the true parameter values after 25,000 iterations already. If one instead were to terminate the algorithm at this time, instead of after 50,000 iterations, the run time could be decreased to a time compare able to the offline algorithm. Further, one should also take in consideration that the online algorithm used a considerably larger number of particles than the offline algorithm. While the online algorithm used $N = 1400$ particles, the offline algorithm used a much smaller particle sample consisting of only $N = 300$ particles. By optimizing the number of particles and number of iterations for the online algorithm, it could therefore be possible to reduce the run time, without impacting the accuracy of the results considerably.

Since both of the two algorithms are based on the PaRIS-algorithm, they perform similarly in terms of efficient sample size. The average efficient sample size for the $5^{th}$ online implementation was $\bar{N}_{eff} = 1210$, which was $86.4\%$ of the total number of particles used. For the offline implementation the last iteration had an average efficient sample size of $\bar{N}_{eff} = 263$, which was $87.7\%$ of the total number of particles. This implies that neither the online, nor the offline implementation suffered from instability due to degenerating weights.

### 5.2 Comparison on AAPL Data

Comparing the performance of the algorithms on actual data is not as straightforward as when comparing the results from the simulated data. This due to the fact that the “true” parameter values for $\mu$, $\phi$, $\sigma^2$ and $\beta^2$ are unknown. The comparison will therefore instead be based on comparing the behaviour of the learning trajectories for the two algorithms, comparing the parameter estimations and the run times of the algorithms. First, comparing the parameters estimated from the 5 online implementations in Table 4.2 to the estimated parameters from the offline implementations in Table 4.3, one can observe that both of the two algorithms provide similar estimates of the parameters. One could argue that the estimates of the parameters $\mu$ and $\sigma^2$ varies somewhat between the online and offline implementation, with the online implementations providing an average estimation of $\bar{\mu} = 0.0157$ and $\bar{(\sigma^2)} = 0.0766$ while the offline implementations yielded average estimations of $\bar{\mu} = 0.0242$ and $\bar{(\sigma^2)} = 0.0458$. However, both the online and offline implementation resulted
in estimates of similar magnitude, with estimates close to 0.

Regarding the precision of the algorithms, one can argue that the parameter estimates from both algorithms were precise. This argument can be made since for both the online and offline algorithm, the parameter estimates from the individual implementations did not vary significantly. For instance, the empirical standard deviation of the estimate of $\phi$ for the online algorithm was 0.0033, and 0.0015 for the offline algorithm.

Evaluating the run times of the algorithms one can observe that the run times of the online implementation were shorter than for the offline implementation. It is worth to take in consideration that the online implementations used more than twice as many particles as the offline implementations, while still resulting in shorter run times. When comparing the run times of the algorithms, one should also keep in mind the fact that the online algorithm is able to update the parameter estimates without processing previous data, if new data should become available. Updating the parameter estimates for the online algorithm would therefore not be as time consuming as if the entire data set would be required. However, the offline algorithm would require both the previous observations and the new data in order to update the parameter estimates, resulting in run times similar or longer than those presented in Table 4.2.

Regarding the coefficient of variation and efficient sample size, a similar comparison as for the simulated data can be made. Since both algorithms are based on the PaRIS-algorithm, the results are similar, with the average efficient sample size for the online implementation being 92.2% of the total number of particles and 92.8% for the offline algorithm. Thus one can argue that neither of the algorithms suffers from weight degeneration.

### 5.3 Comparison of Results from Experiment 1 and Experiment 2

Starting with the results from Section 4.3.1, one can observe that the sample variance for the estimated parameters by the online algorithm are smaller than those from the offline. This implies that for the same computational budget, the online algorithm produces less uncertain estimates with respect to the true parameter than the offline algorithm. One could therefore argue that the most effective algorithm is the online algorithm in this experiment.
Regarding Section 4.3.2, the results are not as clear as those from Section 4.3.1, since neither of the algorithms have a smaller sample variance for all parameters. The offline algorithm yields in estimates with smaller sample variances in the parameter $\mu$, while the online algorithm gives a smaller sample variance for the estimate of $(\phi, \sigma^2, \beta^2)$. It is therefore hard to give a definitive argument for which of the two algorithm that is most effective in this experiment. One could however argue that the online algorithm yields in a relative low variance for all parameters, while the offline algorithm yields in a relative high variance for the parameter $(\phi, \sigma^2)$. This could be used as an argument for the statement that the online algorithm is preferable over the offline.

Looking at the combined results from Experiment 1 and 2, it can be said that both algorithms yield similar variances. However, if one were to choose one algorithm over the other in order to minimize the variance, the most effective choice would be the online algorithm. This argument can be made since in Experiment 1, the online algorithm was superior to the offline in terms of minimizing the variance. The somewhat inconclusive results from Experiment 2 do not imply that the offline algorithm could be considered superior to the online, and thus the online algorithm must be considered the most effective estimator.
Chapter 6

Discussion

In this chapter the comparisons made between the online and offline algorithms stated in Chapter 5 will be discussed. Further, the results presented in Chapter 4 will be discussed for both the online and offline algorithm, in addition to what was said in Chapter 5. A section discussing the stochastic volatility model used in the thesis will also be included in this chapter.

6.1 Case Study

Regarding the results presented in Chapter 4, most of the discussion have already been made in Chapter 5. Both of the algorithms were able to provide accurate estimates of the hidden parameters using simulated data, as can be seen in Figure 4.1 and Figure 4.5. The fact that both algorithms were able to find the hidden parameters for simulated data meant, that if stock prices could be modeled as the simulated data were generated, the algorithms should be able to estimate the parameters for stock prices modeled using the stochastic volatility model specified in Equation (3.4). However, the assumption that the stochastic volatility model describes the log-returns of the AAPL stock prices does not have to be completely accurate, which will be further discussed in Section 6.2. Following the results from the simulated data, the results from the AAPL data were presented. As mentioned previously, both algorithms provided similar results, with the online algorithm having slightly shorter run times. One modification which was made to the online implementation was the implementation of a reduced step size of the updating of the parameter estimates of $\sigma^2$ and $\beta^2$. As mentioned, this was done in order to eliminate problems with instability when the parameter estimates got close to 0. One could argue that this modification of the algorithm is somewhat of a patchwork, and
a “quick-fix” to a problem with the algorithm. However, the problem arises from the fact that if the data used in the case study is modeled as the suggested stochastic volatility model, the parameter estimates are close to 0. This can be seen from the results from the offline implementation of the same data in Table 4.3. If a different data set was to be used, where the parameter estimates did not come as close to 0, the problem would not be present. This can be seen by looking at the results from the simulated data, where the hidden parameters were of greater magnitude, and thus the instability problem was not present.

The comparison made in Section 5.3 resulted in the argument that the online algorithm was superior in terms of minimizing the variance around the true parameter, for a fixed computational budget. The results regarding variance around the maximum likelihood estimate were however somewhat inconclusive, and one could argue that experiments with a larger set of implementations of each algorithm should have been performed.

6.2 Stochastic Volatility Model Selection

The stochastic volatility model considered in the case study of the thesis was as mentioned based on the Hull-White model. Further, the model was simplified in order to facilitate the use of hidden Markov models as a modeling tool. First of all, one could argue that it would have been of interest to use a different stochastic volatility model, such as the Heston model for instance. This could have been done in order to provide some comparison regarding which model that was most suitable for the data used in the case study. However, since the main focus of this thesis is to compare an online solution to an offline solution, this was not done. Further on, using an alternative stochastic volatility model, with a different set of parameters, might solve the instability problem for the online algorithm. Since the problem occurred due to parameter estimates close to 0, an alternative model with a different structure might yield parameter estimates of larger magnitude, and thus eliminating the problem.

Secondly, one could argue that the simplification of the Hull-White model is unjustifiable, since the hidden process is present in the drift term of the log-return, which can be seen in Equation (3.2). One could argue that a more suitable model should be

\[
X_k = \phi X_{k-1} + \sigma U^1_k \\
Y_k = (\mu - \exp(X_k)) + \beta \exp(X_k/2)U^2_k.
\]
However, as the results in Table 4.2 and Table 4.3 indicate, the parameter estimates of $\mu$ for both the online and offline implementations are close to 0. Therefore, one could instead argue that it would be sufficient to use the even simpler model suggested by Cappé and Mouliens in [6] in Example 80, where the drift parameter $\mu$ is not present.

6.3 Online vs Offline Algorithm

The main research question of the thesis was whether an online implementation of the PaRIS-algorithm could be considered as an alternative, or even superior method, for parameter estimation of stocks modeled using stochastic volatility models, compared to a more classical offline EM algorithm. From the comparisons between the results from the two algorithms in Chapter 5, it is suggested that the two algorithms performs almost equally well on a fixed data set, with the online algorithm being somewhat faster. However as mentioned in Chapter 5, as new observations become available the online algorithm is able to update the parameter estimates without using the previous observations. This makes the online implementation useful when working with data such as stock prices, since the stock market provides a constant stream of observations. Therefore, one could argue that the online algorithm is superior to the offline, when the intended application is for financial data. Further, the results from Experiment 1 and 2 implied that the online estimator was more effective in terms of minimizing the variance around the true parameter than the offline algorithm.
Chapter 7

Conclusions and Future Work

7.1 Conclusions

The main purpose of this thesis was to investigate and evaluate the use of online parameter estimation using the PaRIS-algorithm. Primary, the main interest of the thesis was how the online implementation of the PaRIS-algorithm performed as a method of parameter estimation on a stochastic volatility model, compared to an offline EM-algorithm. From a case study, where the online algorithm was compared to a more conventional offline EM-implementation of the PaRIS-algorithm, several conclusion could be made.

i) The online algorithm was able to estimate the true parameters of data simulated using a stochastic volatility model.

ii) The online and offline algorithm had similar performance on the simulated data.

iii) On real stock data, both the online and offline algorithm yielded similar parameter estimates, with the online algorithm having slightly better performance. However, the online algorithm suffered from some instability.

iv) The online algorithm was superior in terms of minimizing the variance around the true parameter compared to the offline algorithm, for a fixed computational budget.

Summarizing the list above as a final conclusion, one can conclude that the online implementation of the PaRIS-algorithm can be considered as an alternative, and in some applications a superior method to the offline EM-algorithm.
In particular when the intended field of use is financial data, or a similar situation where a constant flow of new observations is available. A noteworthy remark is however that the algorithm can be somewhat unstable for variance parameters close to 0.

7.2 Future Work

Future work that might be worth taking in consideration could be to implement the algorithm on an alternative stochastic volatility model, such as the Heston model for instance. As mentioned in the previous chapter, see Section 6.2, a different stochastic volatility model might yield parameter estimates more suitable for the online algorithm.

Further future work that might be worth taking into consideration could be to implement a combination of an offline and online algorithm. Since the main benefit of an online algorithm is that the algorithm is good at handling new observations, it might be a good idea to use an offline algorithm when estimating parameters from historical data, and then switching over to an online algorithm when new observations become available. Using the estimated parameter values from the offline algorithm as initial values for the online algorithm, the online algorithm can continue where the offline algorithm stopped, in order to update the estimations using the new observations. Such a scenario could for instance be when estimating parameters in a stochastic volatility model for a certain stock price. Using all available data up to the current date, an offline algorithm can be implemented in order to estimate the parameters. As time passes and new observations become available, the online algorithm could be implemented to update the estimates.

One could also consider changing the distribution of the noise terms for the hidden or observed process. In this thesis, the noise terms are assumed to follow a standard Gaussian distribution. One could however instead make an assumption that the noise terms follow Student’s t-distribution, with \( \nu \) degrees of freedom. The Student’s t-distribution is similar to the Gaussian distribution, however the choice of \( \nu \) affects how heavy the tails of the distribution are. Since the Student’s t-distribution allows heavier tails, modeling of log-returns of stock prices are often done using the Student’s t-distribution, and thus one could argue that it might be of interest to investigate how the algorithm performs on a model with such noise terms.
Implementing the online algorithm on an alternative kind of financial data might also be of interest. One could for instance try modeling exchange rates using a suitable stochastic volatility model. The algorithm could also be implemented on a market with high volatility, such as the cryptocurrency market for instance.
Bibliography


Appendix A

Derivations of Quantities in Online Algorithm

In this appendix expressions for necessary quantities in the online algorithm are derived. In the first section, an expression for $\tilde{h}_{s,\theta}(x_s, x_{s+1})$ is derived, and in the following section an expression for $\nabla_{\theta} g_{s,\theta}(x_s)$ is derived.

A.1 Derivation of $\tilde{h}_{s,\theta}(x_s, x_{s+1})$

The derivation of $\tilde{h}_{s,\theta}(x_s, x_{s+1})$ follows below.

$$\tilde{h}_{s,\theta}(x_s, x_{s+1}) = \nabla_{\theta} \log q_{\theta}(x_s, x_{s+1}) + \nabla_{\theta} \log g_{s,\theta}(x_s) =$$

$$= \nabla_{\theta} \left[ \log \left( \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left[ -\frac{(x_{s+1} - \phi x_s)^2}{2\sigma^2} \right] \right) \right] +$$

$$+ \nabla_{\theta} \left[ \log \left( \frac{1}{\sqrt{2\pi \beta^2}} \exp \left[ -\frac{(y_s - \mu)^2}{2\beta^2} \exp(-x_s) - \frac{1}{2} x_s \right] \right) \right] =$$

$$= \nabla_{\theta} \left[ -\frac{(x_{s+1} - \phi x_s)^2}{2\sigma^2} - \frac{1}{2} \log(2\pi \sigma^2) \right] +$$

$$+ \nabla_{\theta} \left[ -\frac{(y_s - \mu)^2}{2\beta^2} \exp(-x_s) - \frac{1}{2} \log(2\pi \beta^2) \right].$$
Now obtain gradient using $\nabla_{\theta} = \left( \frac{\partial}{\partial \mu}, \frac{\partial}{\partial \phi}, \frac{\partial}{\partial \sigma^2}, \frac{\partial}{\partial \beta^2} \right)$.

\[
\frac{\partial}{\partial \mu} \left( \log g_{s,\theta}(x_s) \right) = \frac{y_s - \mu}{\beta^2} \exp(-x_s)
\]
\[
\frac{\partial}{\partial \phi} \left( \log q_{\theta}(x_s, x_{s+1}) \right) = \frac{x_s(x_{s+1} - \phi x_s)}{\sigma^2}
\]
\[
\frac{\partial}{\partial \sigma^2} \left( \log q_{\theta}(x_s, x_{s+1}) \right) = \frac{(x_{s+1} - \phi x_s)^2 - \sigma^2}{2(\sigma^2)^2}
\]
\[
\frac{\partial}{\partial \beta^2} \left( \log g_{s,\theta}(x_s) \right) = \frac{(y_s - \mu)^2 \exp(-x_s) - \beta^2}{2(\beta^2)^2}.
\]

Finally an expression for $\tilde{h}_{s,\theta}(x_s, x_{s+1})$ can be written as

\[
\tilde{h}_{s,\theta}(x_s, x_{s+1}) = \left[ \begin{array}{c} \frac{y_s - \mu}{\beta^2} \exp(-x_s) \\ \frac{x_s(x_{s+1} - \phi x_s)}{\sigma^2} \\ \frac{(x_{s+1} - \phi x_s)^2 - \sigma^2}{2(\sigma^2)^2} \\ \frac{(y_s - \mu)^2 \exp(-x_s) - \beta^2}{2(\beta^2)^2} \end{array} \right]
\]

(A.1)

### A.2 Derivation of $\nabla_{\theta} g_{s,\theta}(x_s)$

The derivation of $\nabla_{\theta} g_{s,\theta}(x_s)$ follows below.

\[
\nabla_{\theta} g_{s,\theta}(x_s) = \nabla_{\theta} \left[ \frac{1}{\sqrt{2\pi\beta^2}} \exp \left[ -\frac{(y_s - \mu)^2}{2\beta^2} \exp(-x_s) - \frac{1}{2} x_s \right] \right] =
\]
\[
= \begin{bmatrix} \frac{\partial}{\partial \mu} \\ 0 \\ 0 \\ \frac{\partial}{\partial \beta^2} \end{bmatrix} \left[ \frac{1}{\sqrt{2\pi\beta^2}} \exp \left[ -\frac{(y_s - \mu)^2}{2\beta^2} \exp(-x_s) - \frac{1}{2} x_s \right] \right].
\]

First focus on $\frac{\partial}{\partial \mu} (g_{s,\theta}(x_s))$.

\[
\frac{\partial}{\partial \mu} (g_{s,\theta}(x_s)) = g_{s,\theta}(x_s) \frac{y_s - \mu}{\beta^2} \exp(-x_s) =
\]
\[
= \frac{y_s - \mu}{\sqrt{2\pi(\beta^2)^3}} \exp \left[ -\frac{(y_s - \mu)^2}{2\beta^2} \exp(-x_s) - \frac{3}{2} x_s \right].
\]
Now proceed with \( \frac{\partial}{\partial \beta^2} (g_{s,\theta}(x_s)) \).

\[
\frac{\partial}{\partial \beta^2} (g_{s,\theta}(x_s)) = - \frac{\exp \left[ -\frac{(y_s-\mu)^2}{2\beta^2} \exp(-x_s) - \frac{1}{2}x_s \right]}{2\sqrt{2\pi}(\beta^2)^3} + \frac{g_{s,\theta}(x_s)(y - \mu)^2}{2(\beta^2)^2} \exp(-x_s) =
\]

\[
= \left( \frac{(y - \mu)^2 \exp(-x_s) - \beta^2}{\sqrt{2^3\pi(\beta^2)^5}} \right) \exp \left[ -\frac{(y_s - \mu)^2}{2\beta^2} \exp(-x_s) - \frac{1}{2}x_s \right] =
\]

\[
= \left( y - \mu \right)^2 - \beta^2 \exp(x_s) \frac{(y - \mu)^2}{\sqrt{2^3\pi(\beta^2)^5}} \exp \left[ -\frac{(y_s - \mu)^2}{2\beta^2} \exp(-x_s) - \frac{3}{2}x_s \right].
\]

Finally an expression for \( \nabla_{\theta} g_{s,\theta}(x_s) \) can be written as

\[
\nabla_{\theta} g_{s,\theta}(x_s) = \begin{bmatrix}
\frac{y_s-\mu}{\sqrt{2\pi(\beta^2)^3}} \exp \left[ -\frac{(y_s-\mu)^2}{2\beta^2} \exp(-x_s) - \frac{3}{2}x_s \right] \\
0 \\
\frac{(y-\mu)^2-\beta^2 \exp(x_s)}{\sqrt{2^3\pi(\beta^2)^5}} \exp \left[ -\frac{(y_s-\mu)^2}{2\beta^2} \exp(-x_s) - \frac{3}{2}x_s \right]
\end{bmatrix}
\]  (A.2)
Appendix B

Derivation of Quantities in Offline Algorithm

In this appendix follows the derivation of the statistics $S_1, S_2, S_3, Z_1, Z_2, Z_3$ used in the estimation of the intermediate quantity $Q$. Starting with Equation (3.23), it is possible to derive the definitions in Equation (3.25) in the following way.

$$Q(\theta, \theta^i) \propto \sum_{k=0}^{n} E_{\theta^i} [\log g_k(X_k; \theta)|Y_{0:n}] + \sum_{k=0}^{n-1} E_{\theta^i} [\log q(X_k, X_{k+1})|Y_{0:n}] .$$

Now first define $G \triangleq \sum_{k=0}^{n} E_{\theta^i} [\log g_k(X_k; \theta)|Y_{0:n}]$ in order to derive $S_1, S_2, S_3$.

$$G = \sum_{k=0}^{n} E_{\theta^i} \left[ \log \left( \frac{1}{\sqrt{2\pi}\beta^2} \exp \left[ -\frac{(Y_k - \mu)^2}{2\beta^2} e^{-X_k} - \frac{1}{2} X_k \right] \right) |Y_{0:n}] \right] \propto$$

$$\propto -\frac{1}{2} (n + 1) \log(\beta^2) -$$

$$- \frac{1}{2\beta^2} \left( E_{\theta^i} \left[ \sum_{k=0}^{n} Y_k e^{-X_k} |Y_{0:n}] \right] - 2\mu E_{\theta^i} \left[ \sum_{k=0}^{n} Y_k e^{-X_k} |Y_{0:n}] \right] + \mu^2 \right) E_{\theta^i} \left[ \sum_{k=0}^{n} e^{-X_k} |Y_{0:n}] \right] \right].$$

From the expression above it is possible to identify the statistics $S_1, S_2, S_3$ as defined in Equation (3.25).

In order to derive expressions for the remaining statistics $Z_1, Z_2, Z_3$, define $T \triangleq \sum_{k=0}^{n+1} E_{\theta^i} [\log q(X_k, X_{k+1})|Y_{0:n}]$. Now using a similar procedure as for
It is possible to define the statistics $Z_1, Z_2, Z_3$ using $T$.

$$
T = \sum_{k=0}^{n-1} E_{\theta i} \left[ \log \left( \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ -\frac{(X_{k+1} - \phi X_k)^2}{2\sigma^2} \right] \right) | Y_{0:n} \right] \propto
$$

$$
\propto -\frac{1}{2} n \log(\sigma^2) -
$$

$$
- \frac{1}{2\sigma^2} \left( E_{\theta i} \left[ \sum_{k=0}^{n-1} X_{k+1}^2 | Y_{0:n} \right] - 2\phi E_{\theta i} \left[ \sum_{k=0}^{n-1} X_{k+1} X_k | Y_{0:n} \right] + \phi^2 E_{\theta i} \left[ \sum_{k=0}^{n-1} X_k^2 | Y_{0:n} \right] \right).
$$

From the expression above it is possible to identify the statistics $Z_1, Z_2, Z_3$ as defined in Equation (3.25).
Appendix C

Maximization of Intermediate Quantity

In this appendix, the maximization of $Q(\theta; \theta^i)$ with respect to $\theta$ is explained. This maximization is used in the EM-algorithm, in order to update the parameter estimates. The maximization is performed by finding the roots of the gradient $\nabla_{\theta} Q(\theta; \theta^i)$. Below follows the differentiation of the expression stated in Equation (3.24), with respect to the parameters $\theta = (\mu, \phi, \sigma^2, \beta^2)$.

\[
\frac{\partial}{\partial \mu} Q(\theta; \theta^i) \propto \frac{1}{\beta^2} (2\mu S_3 - 2S_2) \quad (C.1)
\]

\[
\frac{\partial}{\partial \phi} Q(\theta; \theta^i) \propto \frac{1}{\sigma^2} (2\phi Z_3 - 2Z_2) \quad (C.2)
\]

\[
\frac{\partial}{\partial \sigma^2} Q(\theta; \theta^i) \propto \frac{n}{\sigma^2} - \frac{1}{(\sigma^2)^2} (Z_1 - 2\phi Z_2 + \phi^2 Z_3) \quad (C.3)
\]

\[
\frac{\partial}{\partial \beta^2} Q(\theta; \theta^i) \propto \frac{n+1}{\beta^2} - \frac{1}{(\beta^2)^2} (S_1 - 2\mu S_2 + \mu^2 S_3). \quad (C.4)
\]

Following the differentiation, expressions for the roots of the partial derivatives are obtained. For Equation (C.1)

\[
\frac{1}{(\beta^2)^*} (2\mu^* S_3 - 2S_2) = 0
\]

\[
2\mu^* S_3 = 2S_2
\]

\[
\mu^* = \frac{S_2}{S_3}.
\]
For Equation (C.2)

\[
\frac{1}{(\sigma^2)^*} (2\phi^* Z_3 - 2Z_2) = 0
\]

\[
2\phi^* Z_3 = 2Z_2 \\
\phi^* = Z_2/Z_3.
\]

For Equation (C.3)

\[
\frac{n}{(\sigma^2)^*} - \frac{1}{((\sigma^2)^*)^2} (Z_1 - 2\phi^* Z_2 + (\phi^*)^2 Z_3) = 0
\]

\[
\frac{n}{(\sigma^2)^*} = \frac{(Z_1 - 2\phi^* Z_2 + (\phi^*)^2 Z_3)}{((\sigma^2)^*)^2}
\]

\[
(\sigma^2)^* = \frac{(Z_1 - 2\phi^* Z_2 + (\phi^*)^2 Z_3)}{n}.
\]

For Equation (C.4)

\[
\frac{n + 1}{(\beta^2)^*} - \frac{1}{((\beta^2)^*)^2} (S_1 - 2\mu^* S_2 + (\mu^*)^2 S_3) = 0
\]

\[
\frac{n + 1}{(\beta^2)^*} = \frac{(S_1 - 2\mu^* S_2 + (\mu^*)^2 S_3)}{((\beta^2)^*)^2}
\]

\[
(\beta^2)^* = \frac{(S_1 - 2\mu^* S_2 + (\mu^*)^2 S_3)}{n + 1}.
\]
Appendix D

Validation of Online Algorithm With Reduced Particle Cloud

In this appendix the results from the validation of the online algorithm using $N = 600$ and $t_{end} = 30,000$ are presented. The validation is performed as in Section 4.1.1, with the same true parameters. The results are presented in the following figures.
Figure D.1: Learning trajectories of 5 implementations of the online algorithm using $N = 600$ and $t_{end} = 30,000$. Red lines correspond to the true parameter values, and as the trajectories imply the algorithm is able to estimate the true parameter values.
Figure D.2: Coefficient of variation and efficient sample size for the last implementation of the online estimator. Values of $N_{eff}$ close to the total number of particles $N = 600$ indicate a stable algorithm not suffering from issues with degenerating weights.
Figure D.3: Histogram of *efficient sample size* for the last implementation of the online estimator. A high frequency for the *efficient sample size* with values close to 600 is desirable.