MONTE CARLO APPROXIMATION OF THE SCORE FUNCTION AND HESSIAN MATRIX AND ITS USE IN LIKELIHOOD OPTIMISATION

Jingyi Ni

Supervisor: Doctor Feng Chen

School of Mathematics and Statistics
UNSW Sydney

August 2020

Submitted in partial fulfilment of the requirements of the degree of
Master of Statistics

Plagiarism statement

I declare that this thesis is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere.

I acknowledge that the assessor of this thesis may, for the purpose of assessing it:

- Reproduce it and provide a copy to another member of the University; and/or,
- Communicate a copy of it to a plagiarism checking service (which may then retain a copy of it on its database for the purpose of future plagiarism checking).

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct, and am aware of any potential plagiarism penalties which may apply.

By signing this declaration I am agreeing to the statements and conditions above.

Signed: Jingyi Mi Date: 07/08/2020

Acknowledgements

By far the greatest thanks must go to my supervisor, Dr.Feng Chen, for the guidance, care and support he provided. He always answer my questions patiently, guide me to read papers, think about problems, and solve problems. Without his guidance, I would not finish this thesis, nor would I learn a lot from this process.

Thanks must also go to Guangyu Lu, who is also under supervision by Dr.Chen. We work on similar topics and thus communicating with Guangyu is an important part of my study.

Jingyi Ni, 5 August 2020.

Abstract

In this thesis, we provide a general introduction of the State Space model and Particle Filter(PF) algorithm and then show how to implement the PF algorithm in state inference and parameter inference problems. In the maximum likelihood method for parameter inference, we introduce how to approximate the log-likelihood, the score function and Hessian matrix by PF algorithm. Some examples are presented to show the resulting estimates of these terms and the result of parameter inference.

Contents

Chapter 1

Introduction

In this chapter, we provide a general introduction of State Space Model(SSM), including the general expression, the applications and some examples of this model. We also talk about the inference problems of SSM and the developing process of filtering algorithms.

1.1 General Knowledge of State Space Model

State Space models, also known as hidden Markov models, are widely used models in various fields, including ecology, econometrics, engineering, environmental sciences, and finance; see [1, 2, 3, 4, 5]. The unobserved states of SSM which change over time often reflect the true states of our systems, so we are interested in obtaining some unknown state variables from measurement. By this kind of estimation of SSM, we can do orbit tracking for aerospace industry, position prediction for Radar tracking target, Beta coefficient analysis for stock,etc. Thus, it is of great significance to study on State Space models.

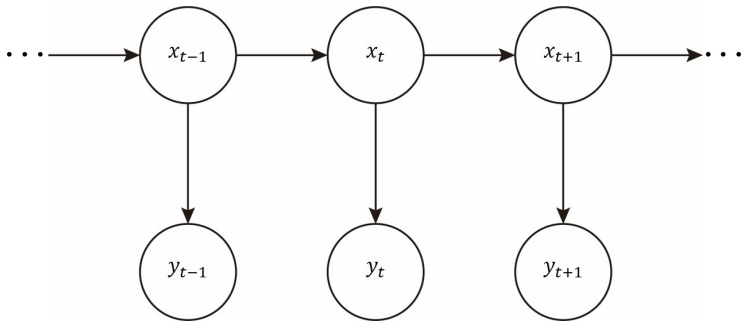

Figure 1.1: Graphical model of an SSM with latent process x and observed process y.

As shown in figure 1.1, a state space model consists of two stochastic processes $\{x_t\}_{t\geq 1}$ and $\{y_t\}_{t\geq 1}$, where x_t denotes the latent state (top) and y_t denotes the observation (bottom) from the system at time t. We assume that the states and observations are real-valued, i.e., $y_t \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ and $x_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$. The latent state $\{x_t\}_{t\geq 1}$ is modelled as a first-order Markov process of initial density $\mu_\theta(x)$, i.e., the latent state x_t only depends on the previous state x_{t-1}. That is, all the information in the past states $x_{1:t-1} \triangleq \{x_s\}_{s=1}^{t-1}$ is summarized in the most recent state x_{t-1}. The observations $y_{1:t}$ are conditionally independent as the observation y_t is only related

to the corresponding state x_t.

The densities of the latent state and the observation process are $f_\theta(x_t \mid x_{t-1})$ and $g_\theta(y_t \mid x_t)$, respectively, where $\theta \in \Theta \subset \mathbb{R}^p$ is the unknown parameter vector. The density $f_\theta(x_t \mid x_{t-1})$ describes the probability that the next state is x_t given the previous state x_{t-1}. And for the observation process, $g_\theta(y_t \mid x_t)$ describes the probability that the observation is y_t given the state x_t. Therefore, with the notation in place, a general SSM can be expressed as

$$x_1 \sim \mu_\theta(x_1), \quad x_t | x_{t-1} \sim f_\theta(x_t \mid x_{t-1}), \quad y_t | x_t \sim g_\theta(y_t \mid x_t) \tag{1.1.1}$$

Here are some examples of state space models.

1. ARMA Model
   The general $ARMA(p,q)$ model is

   $$y_t = \phi y_{t-1} + \cdots \phi_p y_{t-p} + \eta_t + \theta_1 \eta_{t-1} + \cdots + \theta_q \eta_{t-q} \tag{1.1.2}$$

   where η_t is white noise. Let $m = max(p, q+1)$, then we can rewrite the $ARMA(p,q)$ model as

   $$y_t = \phi y_{t-1} + \cdots \phi_p y_{t-m} + \eta_t + \theta_1 \eta_{t-1} + \cdots + \theta_{m-1} \eta_{t-m+1} \tag{1.1.3}$$

   where some of the AR or MA coefficients will be zero unless $p = q+1$. Define x_t as

   $$x_t = \begin{bmatrix} y_t \\ \phi_2 y_{t-1} + \cdots + \phi_p y_{t-m+1} + \theta_1 \eta_t + \cdots + \theta_{m-1} \eta_{t-m+2} \\ \vdots \\ \phi_m y_{t-1} + \theta_m \eta_t \end{bmatrix} \tag{1.1.4}$$

   Then the $ARMA(p,q)$ model can be put in state space form:

   $$x_t = \begin{bmatrix} \phi_1 & 1 & 0 & \cdots & 0 \\ \phi_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{m-1} & 0 & 0 & \cdots & 1 \\ \phi_m & 0 & 0 & \cdots & 0 \end{bmatrix} x_{t-1} + \begin{bmatrix} 1 \\ \theta_1 \\ \vdots \\ \theta_{m-2} \\ \theta_{m-1} \end{bmatrix} \eta_t \tag{1.1.5}$$

   $$y_t = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix} x_t \tag{1.1.6}$$

2. Time Varying Parameter Model
   A time varying parameter model can be used to simulate financial markets.

The general model is:

$$\begin{aligned} R_t &= \alpha_t + \beta_t R_{mt} + v_t \\ \alpha_t &= \alpha_{t-1} + w_{1t}, \quad w_{1t} \sim N(0, \sigma_1^2) \\ \beta_t &= \beta_{t-1} + w_{2t}, \quad w_{2t} \sim N(0, \sigma_2^2) \end{aligned} \tag{1.1.7}$$

where R_t is our return on assets and R_{mt} is the market interest rate. Define x_t as :

$$x_t = \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} \tag{1.1.8}$$

Then the time varying parameter model can be written as a state space model:

$$x_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_{t-1} + \begin{bmatrix} w_{1t} \\ w_{2t} \end{bmatrix} \tag{1.1.9}$$

$$R_t = [1 \quad R_{mt}]\, x_t + v_t \tag{1.1.10}$$

An overview of some concrete applications in many other fields including earthquake counts, polio counts, rainfall occurrence data, glacial varve data and daily returns on a share can be seen in [6].

1.2 Inference Problems for State Space Model

There are often two different inference problems connected to SSMs: the state inference problem and the parameter inference problem. The first problem is to infer the density of the latent state process given the observations and the model where the parameter θ is known. That is, we would like to determine the value of x_t given the information in the observations $y_{1:t}$, i.e., $p_\theta(x_t \mid y_{1:t})$. The second problem is to infer the value of the parameter vector θ from the observations $y_{1:T}$.

Actually, parameter inference is often the primary problem of interest. The existing inference methods can be classified into two categories: Bayesian or Maximum Likelihood (ML).

1. Maximum likelihood parameter inference

   The likelihood function of an SSM can be expressed as

$$\mathcal{L}(\theta) = p_\theta(y_{1:T}) = p_\theta(y_1) \prod_{t=2}^{T} p_\theta(y_t \mid y_{1:t-1}). \tag{1.2.1}$$

   To simplify calculations and improve the numerical stability of many algorithms, we often replace the likelihood function with the log-likelihood function given by

$$\ell(\theta) = \log p_\theta(y_{1:T}) = \log p_\theta(y_1) + \sum_{t=2}^{T} \log p_\theta(y_t \mid y_{1:t-1}). \tag{1.2.2}$$

We regard the likelihood as a function of the parameter and thus, the parameter which maximizes the likelihood or equivalently the log-likelihood is what we want. We should select the parameter that together with the model is the most likely to have been generated the observations. Therefore, the parameter inference problem in the ML setting is given by

$$\widehat{\theta}_{\mathrm{ML}} = \underset{\theta \in \Theta}{\operatorname{argmax}} \mathcal{L}(\theta) = \underset{\theta \in \Theta}{\operatorname{argmax}} \ell(\theta), \tag{1.2.3}$$

Where $\widehat{\theta}_{\mathrm{ML}}$ denotes the ML parameter estimate.

The score function and observed information matrix also play an important role in parameter inference problems. The score function is defined as the gradient of the log-likelihood,

$$\mathcal{S}(\theta') = \nabla \ell(\theta)|_{\theta=\theta'}. \tag{1.2.4}$$

The observed information matrix is defined as the negative Hessian of the log-likelihood,

$$\mathcal{J}(\theta') = -\nabla^2 \ell(\theta)\big|_{\theta=\theta'}. \tag{1.2.5}$$

The score function can be interpreted as the slope of the log-likelihood and the observed information matrix measures the amount of information in the data regarding the parameter θ. Since $\widehat{\theta}_{\mathrm{ML}}$ denotes the ML parameter estimate, the score function is zero when evaluated around this estimate and the observed information matrix should be positive definite.

2. Bayesian parameter inference

Another common method is to use Bayesian. Detailed introduction of Bayesian analysis can be seen in [7, 8]. Different from the ML parameter estimate, where we assume that the true parameter is a specific value, here we regard the true parameter as a random variable following a specific distribution. Our objective in Bayesian parameter inference is to infer an updated probability distribution called the posterior distribution given the information in the data described by the likelihood $p_\theta(y_{1:T})$ and prior distribution of θ denoted $p(\theta)$.

$$p(\theta \mid y_{1:T}) = \frac{p_\theta(y_{1:T})\, p(\theta)}{\int p_\theta(y_{1:T})\, p(\theta) d\theta} \propto p_\theta(y_{1:T})\, p(\theta). \tag{1.2.6}$$

From the above, we can see that both of the two methods are dependent on the likelihood function which satisfies,

$$p_\theta(y_{1:T}) = \int p_\theta(x_{1:T}, y_{1:T})\, dx_{1:T}, \tag{1.2.7}$$

where $p_\theta(x_{1:T}, y_{1:T})$ denotes the joint density and is given from equation (1.1.1) by,

$$p_\theta(x_{1:T}, y_{1:T}) = \mu_\theta(x_1) \prod_{t=2}^{T} f_\theta(x_t \mid x_{t-1}) \prod_{t=1}^{T} g_\theta(y_t \mid x_t). \tag{1.2.8}$$

For the state inference, the posterior density of the latent states can also be given by Bayesian method:

$$p_\theta(x_{1:t} \mid y_{1:t}) = \frac{p_\theta(x_{1:t}, y_{1:t})}{p_\theta(y_{1:t})}. \tag{1.2.9}$$

This posterior density is useful for computing the score vector of the log-likelihood according to Fisher's identity,

$$\nabla_\theta \ell_t(\theta) = \int \nabla_\theta \log p_\theta(x_{1:t}, y_{1:t}) \quad \cdot \quad p_\theta(x_{1:t} \mid y_{1:t})\, dx_{1:t}. \tag{1.2.10}$$

Therefore, in theory, it seems that it is possible to do the inference from the above equations. For the linear Gaussian model, it is easy to check that $p_\theta(x_{1:t}, y_{1:t})$ is a Gaussian distribution and Kalman filtering algorithm [9] is always the optimal solution of the estimation. However, in practice, most of the problems we need to deal with are nonlinear non-Gaussian state space models whose likelihood function is analytically intractable and the posterior might not be any known distribution. As a consequence, we cannot compute many of the integrals depending on the posterior distribution in closed-form. To solve these problems, a number of different numerical approaches and sampling methods have been developed. Thus, the main topic in this thesis is to deal with these problems by particle filter method, also known as Sequential Monte Carlo (SMC) method.

1.3 Filtering Algorithms

According to different time series of measurements and estimates, the inference problems in an SSM can be classified into mainly two types: filtering and smoothing. Both the filtering and smoothing problem can be marginal (inference on a single state), k-interval (inference of k states) or joint (inference on all states). In filtering, only observations $y_{1:t}$ collected until the current time step t are used to infer the current state x_t. Inferring the value of the current state x_t with $t \leq T$ by using all the collected observations including (possibly) future observations $y_{1:T}$ is called smoothing. In Table 1.1, we can see some common filtering and smoothing problems in SSMs.

Name	Density
(Marginal) filtering	$p_\theta(x_t \mid y_{1:t})$
(Marginal) smoothing ($t \leq T$)	$p_\theta(x_t \mid y_{1:T})$
Joint smoothing	$p_\theta(x_{1:T} \mid y_{1:T})$
Fixed-interval smoothing ($s < t \leq T$)	$p_\theta(x_{s:t} \mid y_{1:T})$
Fixed-lag smoothing (for lag Δ)	$p_\theta(x_{t-\Delta-1:t} \mid y_{1:t})$

Table 1.1: Common filtering and smoothing densities in SSMs

Many different filtering algorithms have been developed to deal with these filtering and smoothing problems in SSMs.

1.3.1 Linear Filtering Algorithms

When we refer to optimal estimation in filtering and smoothing problems, we always means estimating the value of unknown states or parameters by some optimal chosen criterion. A well known method of parameter estimate was Least Squares Estimation raised by Gauss in 1795 [10, 11]. Norbert Wiener proposed Wiener Filtering Algorithm in 1940s in order to solve the problem of automatic aiming and firing control of anti-aircraft artillery system problem[12]. The discrete time model of this filtering algorithm was proposed by Andrey Nikolaevich Kolmogorov in 1941[13]. Thus, this is often referred to as Wiener-Kolmogorov filtering theory. Although Wiener Filtering Algorithm is an optimal filtering algorithm for linear systems, all the information in the past time is need in the algorithm, so it is difficult to perform on-line recursive calculation. In 1960, this problem was solved by Kalman Filter[14]. The KF algorithm only depends on one previous observation in the estimation procedure, easy to carry out on-line recursive calculation. And meantime, for linear Gaussian systems, KF algorithm is also an optimal filtering algorithm. An important breakthrough here is that State Space Model is used to define the KF algorithm, which directly result in the wide use of State Space model later in various fields. Most nonlinear filtering algorithms are also developed from this algorithm.

1.3.2 Nonlinear Filtering Algorithms

However, most systems are nonlinear in reality so that the KF algorithm cannot be used under this situation. The extended Kalman filtering algorithm(EKF)[15, 16] was developed based on the structure of Kalman filtering algorithm. It linearizes the nonlinear system by Taylor series expansion, which lead to a suboptimal solution of the state variable estimation. The extended Kalman filtering algorithm is widely used in weak nonlinear state space models because of the high efficiency of the calculation. But for the system with high nonlinearity, the linearization process makes the result not accuracy enough as the higher order terms of Taylor series expansion cannot be ignored directly. What's more, higher order Taylor series expansion needs more computing resources and will make the algorithm inefficient.

Another approximation method is called Unscented Kalman Filter(UKF) which is developed based on Unscented Transform(UT) [17]. This algorithm improves the calculation accuracy to some extent in comparison to EKF. However, the EKF and UKF algorithms both are only applicable to the systems where the noises follow the Gaussian distribution. Other nonlinear filtering algorithms include Central Difference Kalman Filter(CDKF), Divided Difference Kalman Filter(DDKF) and Quadrature Kalman Filter(QKF).

1.3.3 Monte Carlo Approximation

In order to work on non-Gaussian noise, a Particle Filter (PF) algorithm based on Sequential Monte Carlo (SMC) structure called Bootstrap Filter was created in 1993 [18]. This is the important milestone for the later rapid development of particle filter. The PF algorithm uses a set of weighted samples (also called particles) to represent the posterior distribution of the random process given noise and observations. The estimation of unknown variables is realized by recursively calculating and adjusting the weights of the particles.

The predecessor of the PF algorithm is the Monte Carlo (MC) method based on Sequential Importance Sampling (SIS). The MC integration method based on SIS was used in physics and statistics in 1950s [20]. However, this method had not been fully and effectively developed for a long period of time because of the particle degeneracy problem in SIS until Bootstrap Filter appeared. The solution is to add resampling process to the algorithm and thus the SIS method is improved to Sequential Importance Resampling method(SIR). However, SIR also leads to Sample Impoverishment problem, which means too-few samples at a certain moment. There are many methods to improve the PF algorithm around these issues , including choosing better importance density function, getting suitable resampling method, etc.

Chapter 2

State Inference Using Particle Methods

In this chapter we assume the parameter θ is known and we focus on the problem of estimating the latent process $\{x_t\}_{t\geq 1}$ sequentially given the observations. The estimation process also provides us with an on-line scheme to compute $\{p_\theta(y_{1:t})\}_{t\geq 1}$. The particle approximation of these terms are the main topic to be discussed in this chapter.

2.1 Recursive Bayesian Estimation

Generally speaking, regardless of linear systems or nonlinear systems, the optimal filtering problem can be solved by recursive Bayesian estimation [19]. The key idea of recursive Bayesian estimation is to estimate the posterior probability density function based on prior knowledge and data information. The complete algorithm often consists of two steps: the prediction process and the update process.

To simplify the problem, we first consider the task of estimating recursively in time the sequence of marginal posteriors $\{p(x_t \mid y_{1:t})\}_{t\geq 1}$. Assume that the posterior probability density $p(x_{t-1} \mid y_{1:t-1})$ is known, then the process of recursive Bayesian estimation is as follows:

1. Prediction: we predict $p(x_t \mid y_{1:t-1})$ from $p(x_{t-1} \mid y_{1:t-1})$. According to Chapman-Kolmogorov Equation, we can have

$$p(x_t \mid y_{1:t-1}) = \int p(x_t \mid x_{t-1})\, p(x_{t-1} \mid y_{1:t-1})\, dx_{t-1}, \tag{2.1.1}$$

   where $p(x_t \mid x_{t-1})$ is the Markov state transition density function $f_\theta(x_t \mid x_{t-1})$ which is decided by the model.

2. Update: we update $p(x_t \mid y_{1:t})$ when we have new observation at time t. By Bayes' theorem, we have

$$
\begin{aligned}
p\left(x_t \mid y_{1:t}\right) &= \frac{p\left(y_{1:t} \mid x_t\right) p\left(x_t\right)}{p\left(y_{1:t}\right)} \\
&= \frac{p\left(y_t, y_{1:t-1} \mid x_t\right) p\left(x_t\right)}{p\left(y_t, y_{1:t-1}\right)} \\
&= \frac{p\left(y_t \mid y_{1:t-1}, x_t\right) p\left(y_{1:t-1} \mid x_t\right) p\left(x_t\right)}{p\left(y_t \mid y_{1:t-1}\right) p\left(y_{1:t-1}\right)} \\
&= \frac{p\left(y_t \mid y_{1:t-1}, x_t\right) p\left(x_t \mid y_{1:t-1}\right) p\left(y_{1:t-1}\right) p\left(x_t\right)}{p\left(y_t \mid y_{1:t-1}\right) p\left(y_{1:t-1}\right) p\left(x_t\right)} \\
&= \frac{p\left(y_t \mid y_{1:t-1}, x_t\right) p\left(x_t \mid y_{1:t-1}\right)}{p\left(y_t \mid y_{1:t-1}\right)}
\end{aligned}
\tag{2.1.2}
$$

Since the observations $y_{1:t}$ are conditionally independent, we can have

$$
p\left(y_t \mid y_{1:t-1}, x_t\right) = p\left(y_t \mid x_t\right). \tag{2.1.3}
$$

Therefore, the posterior distribution can be written as

$$
p\left(x_t \mid y_{1:t}\right) = \frac{p\left(y_t \mid x_t\right) p\left(x_t \mid y_{1:t-1}\right)}{p\left(y_t \mid y_{1:t-1}\right)}, \tag{2.1.4}
$$

where $p\left(y_t \mid x_t\right)$ is defined in the model by $g_\theta\left(y_t \mid x_t\right)$, and $p\left(y_t \mid y_{1:t-1}\right)$ is the normalizing constant

$$
p\left(y_t \mid y_{1:t-1}\right) = \int p\left(y_t \mid x_t\right) p\left(x_t \mid y_{1:t-1}\right) \mathrm{d}x_t. \tag{2.1.5}
$$

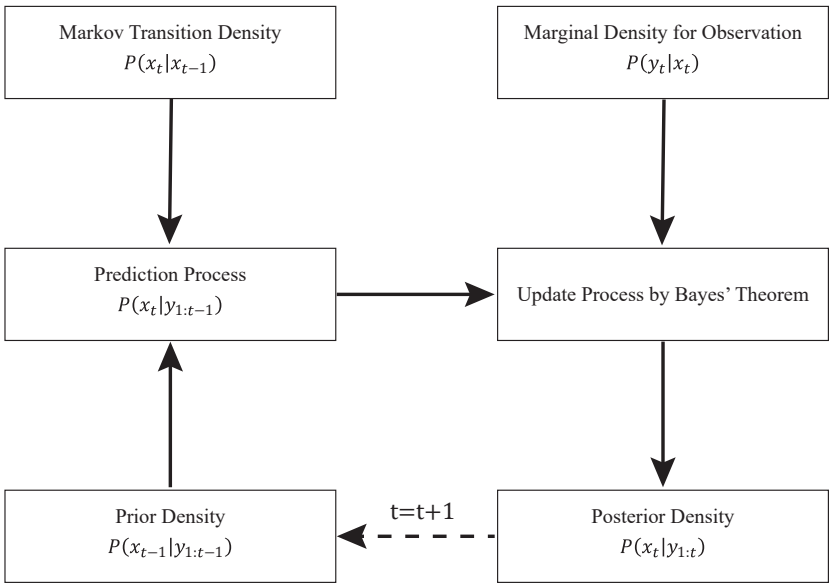

Figure 2.1: Process of the recursive Bayesian estimation.

Therefore, we have completed the recursive Bayesian estimation process as shown in Figure 2.1

In a similar manner, the smoothing problem can be solved by

$$p\left(x_{t+1} \mid y_{1:t}\right)=\int p\left(x_{t+1} \mid x_t\right) p\left(x_t \mid y_{1:t}\right) \mathrm{d} x_t, \tag{2.1.6}$$

$$p\left(x_t \mid y_{1:T}\right)=p\left(x_t \mid y_{1:t}\right) \int \frac{p\left(x_{t+1} \mid x_t\right) p\left(x_{t+1} \mid y_{1:T}\right)}{p\left(x_{t+1} \mid y_{1:t}\right)} \mathrm{d} x_{t+1}. \tag{2.1.7}$$

Since $p\left(x_t \mid y_{1:t}\right)$ is the marginal density of posterior density $p\left(x_{1:t} \mid y_{1:t}\right)$, the general equation of joint posterior density obtained by recursive Bayesian estimation is

$$p\left(x_{1:t} \mid y_{1:t}\right)=p\left(x_{1:t-1} \mid y_{1:t-1}\right) \frac{p\left(x_t \mid x_{t-1}\right) p\left(y_t \mid x_t\right)}{p\left(y_t \mid y_{1:t-1}\right)}, \tag{2.1.8}$$

where the normalizing constant is

$$p\left(y_t \mid y_{1:t-1}\right)=\int p\left(x_{1:t-1} \mid y_{1:t-1}\right) p\left(x_t \mid x_{t-1}\right) p\left(y_t \mid x_t\right) \mathrm{d} x_{1:t}. \tag{2.1.9}$$

Equation (2.1.1) is known as the prediction step and (2.1.4) is known as the update step. However, most particle filtering methods rely on a numerical approximation of recursion (2.1.8) and not of (2.1.1) and (2.1.4).

If we can compute $p\left(x_{1:t} \mid y_{1:t}\right)$ and thus $p\left(x_t \mid y_{1:t}\right)$ sequentially, then the marginal likelihood $p\left(y_{1:t}\right)$ can also clearly be evaluated recursively using

$$p\left(y_{1:t}\right)=p\left(y_1\right) \prod_{k=2}^{t} p\left(y_k \mid y_{1:k-1}\right) \tag{2.1.10}$$

where $p\left(y_k \mid y_{1:k-1}\right)$ is of the form (2.1.9).

Actually, the recursive process is the general framework of the filtering algorithms. Yu-Chi Ho and Robert C. K. Lee first studied on the recursive Bayesian filtering problem and pointed that Kalman filtering algorithm is a special case of Bayesian filtering[19]. As we discussed before, the recursions can only be solved analytically for two different classes of SSMs: linear Gaussian SSMs and SSMs with finite state processes. For the former, the recursions can be implemented using Kalman filter (KF).

If the system does not satisfy the linear Gaussian condition, it is difficult to obtain the analytical solution of Bayesian filtering. Two main methods are used to solve nonlinear systems. One is the analytical approximation for models with Gaussian noise, such as the EKF discussed in chapter 1. The other method is simulation

approximation based on Monte Carlo when noise is non-Gaussian, such as the Particle Filter(PF) method we will discuss later. Figure 2.2 shows the classification of Bayesian filtering.

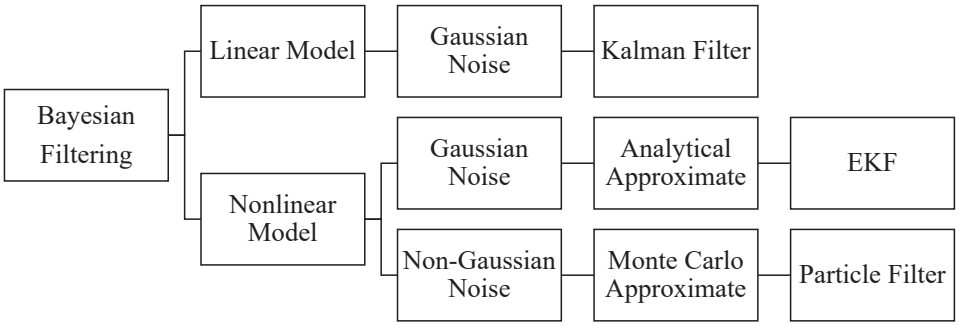

Figure 2.2: Classification of Bayesian filtering.

2.2 Monte Carlo and Importance Sampling

Generally speaking, the Particle Filter method can be seen as the application of Monte Carlo method in Bayesian estimation and is accessible to any nonlinear system which can be described by State Space model. Since PF algorithm is developed from the Monte Carlo (MC) method based on Sequential Importance Sampling (SIS) and improved by Sequential Importance Resampling method(SIR), we will introduce PF starting from MC method and importance sampling in this section.

2.2.1 Monte Carlo method

MC methods are a collection of statistical simulation methods based on sampling and the strong law of large numbers (SLLN). The idea is to use a large number of sample points in the state space to approximate the posterior probability distribution function of the variable. As shown in Figure 2.3, the circles in the figure represent discrete sample points, thus the integration problem is transformed into a summation problem of finite sample points.

Here, we consider estimating the expected value (an integral) of a function $\varphi(x)$

$$\widehat{\varphi} = \mathbb{E}_\pi[\varphi(x)] = \int \varphi(x)\pi(x)\mathrm{d}x \tag{2.2.1}$$

where $\pi(x)$ denotes a (normalised) distribution that x follows. If we can sample a set of independently identically distribution(IID) particles $\{x_1, x_2, \cdots, x_N\}$ from $\pi(x)$, as a consequence of the SLLN, we can estimate the expectation by the sample average

$$\widehat{\varphi}_{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} \varphi\left(x^{(i)}\right), \quad x^{(i)} \sim \pi(x). \tag{2.2.2}$$

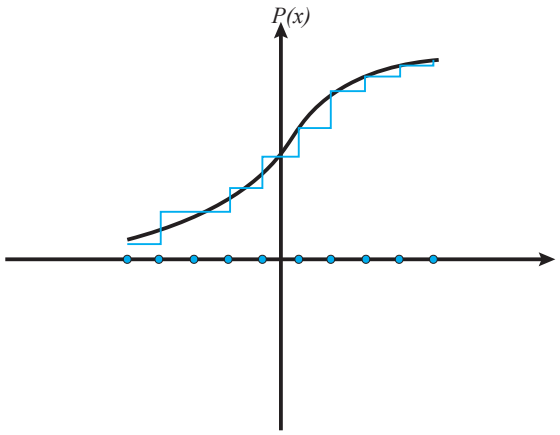

Figure 2.3: MC method for integration problem.

Similarly, for Bayesian filtering problem, we can sample IID particles $\{x_{1:t}^i\}_{i=1}^N$ from posterior probability density $p\left(x_{1:t} \mid y_{1:t}\right)$. Then the posterior probability density can be approximated by the following formula,

$$\hat{p}\left(x_{1:t} \mid y_{1:t}\right) = \frac{1}{N}\sum_{i=1}^{N}\delta\left(x_{1:t} - x_{1:t}^{(i)}\right), \tag{2.2.3}$$

where $\delta(\cdot)$ is Dirac-delta function defined by

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{2.2.4}$$

and is also constrained to satisfy the identity,

$$\int_{-\infty}^{\infty}\delta(x)dx = 1. \tag{2.2.5}$$

Based on this approximation, the conditional expectation of function $\varphi(x_{1:t})$

$$E\left[\varphi\left(x_{1:t}\right)\right] = \int \varphi\left(x_{1:t}\right) p\left(x_{1:t} \mid y_{1:t}\right) dx_{1:t} \tag{2.2.6}$$

can be approximated by

$$\begin{aligned} \hat{E}\left[\varphi\left(x_{1:t}\right)\right] &= \frac{1}{N}\sum_{i=1}^{N}\int \varphi\left(x_{1:t}\right)\delta\left(x_{1:t} - x_{1:t}^{(i)}\right) dx_{1:t} \\ &= \frac{1}{N}\sum_{i=1}^{N}\varphi\left(x_{1:t}^{(i)}\right) \end{aligned} \tag{2.2.7}$$

The convergence of this estimator is guaranteed by SLLN, i.e.

$$\hat{E}\left[\varphi\left(x_{1:t}\right)\right] \xrightarrow{a.s} \mathbb{E}\left[\varphi\left(x_{1:t}\right)\right], \quad N \to \infty. \tag{2.2.8}$$

Therefore, in theory, we can use the mean value of the sample particles to estimate the expectation. The main advantage of Monte Carlo methods is that the variance of the approximation error decreases at a rate of $\mathcal{O}(1/N)$ regardless of the dimension of the space. However, the problem of this approach is also obvious. Since we don't know the posterior probability density $p\left(x_{1:t} \mid y_{1:t}\right)$, we certainly cannot directly sample from it. In this case, importance sampling (IS) [21] can be used to sample from another distribution called the proposal distribution (or importance distribution) $q\left(x_{1:t} \mid y_{1:t}\right)$ and adapt the particles using a weighting scheme.

2.2.2 Importance Sampling

Importance sampling is the basis of all the algorithms developed later on. By using the particle system $\left\{w_t^{(i)}, x_t^{(i)}\right\}_{i=1}^{N}$, where $x_t^{(i)}$ and $w_t^{(i)}$ denote particle i at time t and its corresponding (unnormalised) importance weight, we can approximate the posterior probability density by

$$\widehat{p}\left(x_{1:t} \mid y_{1:t}\right) \triangleq \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta\left(x_{1:t} - x_{1:t}^{(i)}\right) \tag{2.2.9}$$

$$\widetilde{w}_t^{(i)} \triangleq \frac{w_t^{(i)}}{\sum_{k=1}^{N} w_t^{(k)}}. \tag{2.2.10}$$

In practice, sampling from posterior probability density is impossible, so we need an importance distribution $q\left(x_{1:t} \mid y_{1:t}\right)$ which is easily to be sampled from. To see how it works, first we rewrite the conditional expectation of function $\varphi(x_{1:t})$ as

$$\begin{aligned} E\left[\varphi\left(x_{1:t}\right)\right] &= \int \varphi\left(x_{1:t}\right) p\left(x_{1:t} \mid y_{1:t}\right) dx_{1:t} \\ &= \int \varphi\left(x_{1:t}\right) \frac{p\left(x_{1:t} \mid y_{1:t}\right)}{q\left(x_{1:t} \mid y_{1:t}\right)} q\left(x_{1:t} \mid y_{1:t}\right) dx_{1:t} \\ &= \int \varphi\left(x_{1:t}\right) \frac{p\left(y_{1:t} \mid x_{1:t}\right) p\left(x_{1:t}\right)}{q\left(x_{1:t} \mid y_{1:t}\right) p\left(y_{1:t}\right)} q\left(x_{1:t} \mid y_{1:t}\right) dx_{1:t} \\ &= \int \varphi\left(x_{1:t}\right) \frac{w_t\left(x_{1:t}\right)}{p\left(y_{1:t}\right)} q\left(x_{1:t} \mid y_{1:t}\right) dx_{1:t} \end{aligned} \tag{2.2.11}$$

where $w_t(x_{1:t})$ denotes the weight of particle $x_{1:t}$ and is in the form

$$\begin{aligned} w_t(x_{1:t}) &= \frac{p(y_{1:t} \mid x_{1:t})\, p(x_{1:t})}{q(x_{1:t} \mid y_{1:t})} \\ &= \frac{p(x_{1:t} \mid y_{1:t})\, p(y_{1:t})}{q(x_{1:t} \mid y_{1:t})} \\ &\propto \frac{p(x_{1:t} \mid y_{1:t})}{q(x_{1:t} \mid y_{1:t})}. \end{aligned} \tag{2.2.12}$$

If we write $p(y_{1:t})$ in the following integral,

$$p(y_{1:t}) = \int p(y_{1:t} \mid x_{1:t})\, p(x_{1:t})\, dx_{1:t} \tag{2.2.13}$$

then the conditional expectation of function $\varphi(x_{1:t})$ can be transformed to the ratio of two expectations on distribution $q(x_{1:t} \mid y_{1:t})$. The detailed process of deduction is as follows.

$$\begin{aligned} E[\varphi(x_{1:t})] &= \int \varphi(x_{1:t}) \frac{w_t(x_{1:t})}{p(y_{1:t})} q(x_{1:t} \mid y_{1:t})\, dx_{1:t} \\ &= \frac{1}{p(y_{1:t})} \int \varphi(x_{1:t})\, w_t(x_{1:t})\, q(x_{1:t} \mid y_{1:t})\, dx_{1:t} \\ &= \frac{\int \varphi(x_{1:t})\, w_t(x_{1:t})\, q(x_{1:t} \mid y_{1:t})\, dx_{1:t}}{\int p(y_{1:t} \mid x_{1:t})\, p(x_{1:t})\, dx_{1:t}} \\ &= \frac{\int \varphi(x_{1:t})\, w_t(x_{1:t})\, q(x_{1:t} \mid y_{1:t})\, dx_{1:t}}{\int w_t(x_{1:t})\, q(x_{1:t} \mid y_{1:t})\, dx_{1:t}} \\ &= \frac{E_q[\varphi(x_{1:t})\, w_t(x_{1:t})]}{E_q[w_t(x_{1:t})]} \end{aligned} \tag{2.2.14}$$

Therefore, now we can sample IID particles $\{x_{1:t}^i\}_{i=1}^N$ from importance distribution $q(x_{1:t} \mid y_{1:t})$ and apply the MC method to get

$$\hat{q}(x_{1:t} \mid y_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta\left(x_{1:t} - x_{1:t}^{(i)}\right), \tag{2.2.15}$$

and

$$\begin{aligned}\hat{E}\left[\varphi\left(x_{1:t}\right)\right] &= \frac{\frac{1}{N}\sum_{i=1}^{N}\varphi\left(x_{1:t}^{(i)}\right)w_t\left(x_{1:t}^{(i)}\right)}{\frac{1}{N}\sum_{i=1}^{N}w_t\left(x_{1:t}^{(i)}\right)} \\ &= \sum_{i=1}^{N}\varphi\left(x_{1:t}^{(i)}\right)\frac{w_t\left(x_{1:t}^{(i)}\right)}{\sum_{i=1}^{N}w_t\left(x_{1:t}^{(i)}\right)} \\ &= \sum_{i=1}^{N}\varphi\left(x_{1:t}^{(i)}\right)\tilde{w}_t\left(x_{1:t}^{(i)}\right),\end{aligned} \tag{2.2.16}$$

where $\tilde{w}_t\left(x_{1:t}^{(i)}\right)$ is the normalized importance weight, given by

$$\widetilde{w}_t\left(x_{1:t}^{(i)}\right) = \frac{w_t\left(x_{1:t}^{(i)}\right)}{\sum_{i=1}^{N}w_t\left(x_{1:t}^{(i)}\right)} \triangleq \widetilde{w}_t^{(i)} \quad i = 1, \ldots, N \tag{2.2.17}$$

and satisfies

$$\tilde{w}_t^i \in [0,1], \quad \sum_{i=1}^{N}\tilde{w}_t^i = 1. \tag{2.2.18}$$

2.2.3 Sequential Importance Sampling

In the method of importance sampling, all the observation data until the current time step t are required to estimate the posterior probability density. That is, when new observation data arrives, the importance weight of the entire state sequence needs to be recalculated again. Thus, to avoid the computational complexity, we can apply the sequential analysis to MC method to achieve a recursive algorithm called Sequential Importance Sampling(SIS).

This solution involves selecting an importance distribution which has the following structure

$$\begin{aligned}q\left(x_{1:t} \mid y_{1:t}\right) &= q\left(x_t \mid x_{1:t-1}, y_{1:t}\right)q\left(x_{1:t-1} \mid y_{1:t}\right) \\ &= q\left(x_t \mid x_{1:t-1}, y_{1:t}\right)q\left(x_{1:t-1} \mid y_{1:t-1}\right).\end{aligned} \tag{2.2.19}$$

From equation (2.1.8) we have,

$$p\left(x_{1:t} \mid y_{1:t}\right) \propto p\left(x_{1:t-1} \mid y_{1:t-1}\right)p\left(x_t \mid x_{t-1}\right)p\left(y_t \mid x_t\right). \tag{2.2.20}$$

Thus, the associated unnormalised weights can be computed recursively using the

decomposition

$$\begin{aligned} w_t^{(i)} &\propto \frac{p\left(x_{1:t}^{(i)} \mid y_{1:t}\right)}{q\left(x_{1:t}^{(i)} \mid y_{1:t}\right)} \\ &\propto \frac{p\left(x_{1:t-1}^{(i)} \mid y_{1:t-1}\right) p\left(x_t^{(i)} \mid x_{t-1}^{(i)}\right) p\left(y_t \mid x_t^{(i)}\right)}{q\left(x_t^{(i)} \mid x_{1:t-1}^{(i)}, y_{1:t}\right) q\left(x_{1:t-1}^{(i)} \mid y_{1:t-1}\right)} \\ &= w_{t-1}^{(i)} \frac{p\left(x_t^{(i)} \mid x_{t-1}^{(i)}\right) p\left(y_t \mid x_t^{(i)}\right)}{q\left(x_t^{(i)} \mid x_{1:t-1}^{(i)}, y_{1:t}\right)}. \end{aligned} \tag{2.2.21}$$

To choose a suitable importance distribution, we assume $q\left(x_t^{(i)} \mid x_{1:t-1}^{(i)}, y_{1:t}\right) = q\left(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t\right)$, then equation (2.2.21) can be written as

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p\left(x_t^{(i)} \mid x_{t-1}^{(i)}\right) p\left(y_t \mid x_t^{(i)}\right)}{q\left(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t\right)}. \tag{2.2.22}$$

The normalised weight $\widetilde{w}_t^{(i)}$ is calculated by equation (2.2.17). And the posterior probability density can be approximated by

$$\hat{p}\left(x_t \mid y_{1:t}\right) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta\left(x_t - x_t^{(i)}\right). \tag{2.2.23}$$

So far, we have completed the process of SIS algorithm. We randomly sample particles from the proposal distribution, sequentially calculate the corresponding importance weight of particles according to equation (2.2.21), and finally approximate the posterior probability distribution of the system state in the form of weighted summation of particles, so as to get the recursive estimate of state expectation:

$$\hat{x}_t = \sum_{i=1}^{N} x_t^{(i)} \cdot \tilde{w}_t^{(i)}. \tag{2.2.24}$$

In the classical bootstrap particle filter algorithm, the Markov transition density $p\left(x_t \mid x_{t-1}\right)$ is chosen to be the proposal distribution, i.e.

$$q\left(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t\right) = p\left(x_t^{(i)} \mid x_{t-1}^{(i)}\right). \tag{2.2.25}$$

Thus the importance weight is calculated sequentially by

$$w_t^{(i)} \propto w_{t-1}^{(i)} p\left(y_t \mid x_t^{(i)}\right). \tag{2.2.26}$$

2.2.4 Resampling

It appears that we have provided a feasible solution to the filtering problem, however, it is important to be aware that the method presented here suffers from severe drawbacks. The main problem of this algorithm is that the variance of the estimate increases exponentially with t [22]. This results from that the particle weights deteriorate over time and in the limit most particle weights become very small (almost zero) after a few iterations. Hence, many ineffective number of particles consume a lot of computational resources but contribute a little to the posterior estimate, even influence the accuracy of the algorithm. This phenomenon is called Particle Degeneracy.

Resampling techniques are a key ingredient of SMC methods which solve particle degeneracy problem in some important scenarios. By including a resampling step into the SIS algorithm, we can focus the computational efforts on the "promising" regions of the state space. The resampling step essentially duplicates particles with high weights and discard particles with low weights, while keeping the total number of particles fixed.

The degree of particle degeneracy is measured by effective sample size [23, 24]. For a sample set with N sample points, the effective sample size N_{eff} can be approximated by

$$N_{eff} = \frac{N}{1 + \mathrm{Var}\left(w_k^j\right)}. \tag{2.2.27}$$

However, the calculation of the above equation is too complex. Thus, another simpler approximation is used in actual application

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}_k^i\right)^2}. \tag{2.2.28}$$

The smaller the effective sample size N_{eff}, the more serious the particle degeneracy phenomenon. In order to decide the time of applying resampling algorithm, we often set a threshold value N_{th} to compare with $\hat{N}_{eff}$. When the inequality $\hat{N}_{eff} < N_{th}$ happens, which means the particle degeneracy problem is serious and unacceptable, the resampling algorithm should be used to increase the effective sample size.

Although the resampling step partially solves the particle degeneracy problem, it also results in another problem called Sample Impoverishment [25]. A lot of particles with high weights are duplicated in the resampling step, thus the particles in

the resampled sample set are no longer independent. More and more of the same particles cause the sample set to lose its diversity. To balance between the particle degeneracy problem and sample impoverishment problem, researches suggested various improved resampling method [26, 27, 28, 29]. In this thesis, we apply the simplest multinomial resampling (also known as simple random resampling), which is the core idea of Bootstrap Filter algorithm.

2.2.5 Particle Filter Algorithm

Combination of SIS and resampling is the classical standard Particle Filter algorithm. The algorithm can thus be summarised as follows and Figure 2.4 vividly demonstrate the complete process of Particle Filter algorithm.

Algorithm 1 Standard Particle Filter Algorithm

At time $t = 1$**, for all** $i \in \{1, \cdots, N\}$**:**

1. sample $x_1^{(i)} \sim p(x_1) = \mu_\theta(x_1)$.
2. initial weight $\tilde{w}_1^{(i)} = \frac{1}{N}$.

At time $t \geq 2$**, for all** $i \in \{1, \cdots, N\}$**:**

1. sample
   $x_t^{(i)} \sim q\left(x_t^{(i)} \mid x_{t-1}^{(i)}, y_t\right) = p\left(x_t^{(i)} \mid x_{t-1}^{(i)}\right) = f_\theta(x_t \mid x_{t-1})$.
2. update and normalize particle weights
   $w_t^{(i)} = w_{t-1}^{(i)} p\left(y_t \mid x_t^{(i)}\right)$
   $\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^N w_t^{(i)}}$.
3. resample to obtain N new equally-weighted particles
   $\left\{\tilde{x}_t^j, \frac{1}{N}; j = 1, 2, \cdots, N\right\}$.
4. state estimation
   $\hat{x}_t = \sum_{i=1}^N x_t^{(i)} \cdot \tilde{w}_t^{(i)}$.

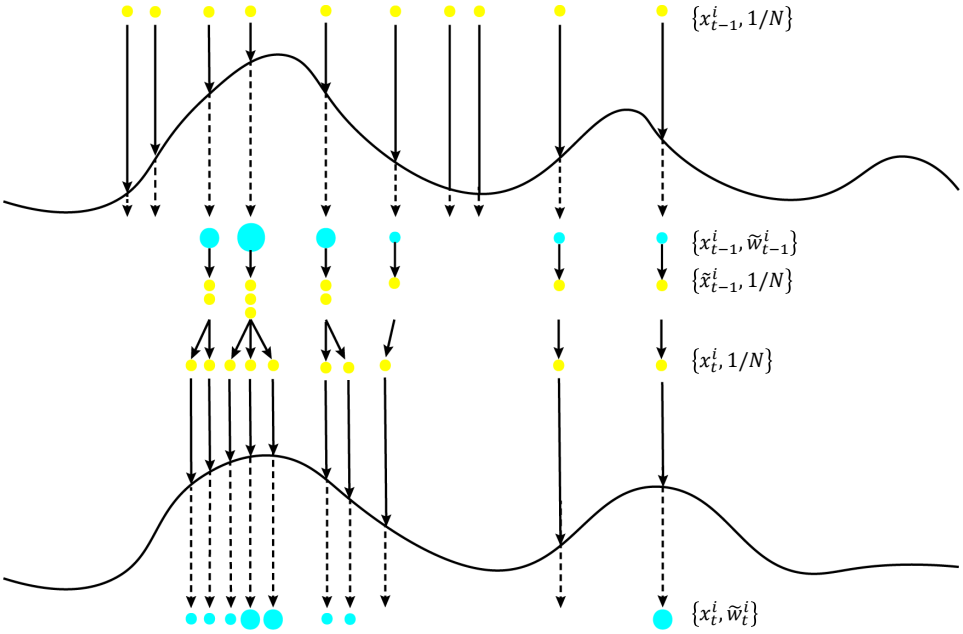

Figure 2.4: Particle Filter Process.

Since we can estimate the posterior probability density and posterior state by particle filter now, the corresponding likelihood (2.1.10) then can also be approximated. For estimating the likelihood, we introduce the auxiliary particle filter(APF) algorithm developed from the above standard PF algorithm. The APF is a popular approach covering as special cases a large class of particle algorithms, such as the bootstrap filter.

Given the particles at time $t-1$, the APF proceeds to time t by three steps.

1. Resampling: The particles are resampled according to their auxiliary weights. The result is an equally-weighted particle system $\left\{\tilde{x}_t^{(i)}, 1/N; i = 1, 2, \cdots, N\right\}$.

2. Propagation: The particles are propagated to time t by sampling from a proposal kernel $x_t^{(i)} \sim R_\theta\left(x_t \mid \widetilde{x}_{t-1}^{(i)}, y_t\right)$.

3. Weighting: The (unnormalised) particle weight is calculated for each particle. The importance weights are given by

$$w_t^{(i)} = W_\theta\left(x_t^{(i)}, \tilde{x}_{t-1}^{(i)}\right) = \frac{g_\theta\left(y_t \mid x_t^{(i)}\right) f_\theta\left(x_t^{(i)} \mid \widetilde{x}_{t-1}^{(i)}\right)}{R_\theta\left(x_t^{(i)} \mid \widetilde{x}_{t-1}^{(i)}, y_t\right)}. \tag{2.2.29}$$

   The bootstrap PF is just a special case of APF when $R_\theta\left(x_t^{(i)} \mid \widetilde{x}_{t-1}^{(i)}, y_t\right) = f_\theta\left(x_t^{(i)} \mid \widetilde{x}_{t-1}^{(i)}\right)$, and thus $w_t^{(i)} = g_\theta\left(y_t \mid x_t^{(i)}\right)$. More sophisticated alternatives of R_θ exist, for example, the fully-adapted PF.

Then we can calculate equation (2.1.9) by

$$\begin{aligned} p_\theta\left(y_t \mid y_{1:t-1}\right) &= \int p_\theta\left(x_{1:t-1} \mid y_{1:t-1}\right) p_\theta\left(x_t \mid x_{t-1}\right) p_\theta\left(y_t \mid x_t\right) \mathrm{d}x_{1:t} \\ &= \int p_\theta\left(x_{t-1} \mid y_{1:t-1}\right) p_\theta\left(x_t \mid x_{t-1}\right) p_\theta\left(y_t \mid x_t\right) \mathrm{d}x_{t-1:t} \\ &= \int W_\theta\left(x_t, x_{t-1}\right) R_\theta\left(x_t \mid x_{t-1}, y_t\right) p_\theta\left(x_{t-1} \mid y_{1:t-1}\right) \mathrm{d}x_{t-1:t}. \end{aligned} \tag{2.2.30}$$

To approximate the integral, we note that the (unweighted) particle pairs $\left\{\widetilde{x}_{t-1}^{(i)}, x_t^{(i)}\right\}_{i=1}^N$ are approximately drawn from $R_\theta\left(x_t \mid x_{t-1}, y_t\right) p_\theta\left(x_{t-1} \mid y_{1:t-1}\right)$. Consequently, the MC approximation of the above equation is obtained by

$$p_\theta\left(y_t \mid y_{1:t-1}\right) \approx \frac{1}{N} \sum_{i=1}^N w_t^{(i)}. \tag{2.2.31}$$

By inserting this approximation into equation (2.1.10), we obtain the particle estimate of the likelihood

$$\widehat{\mathcal{L}}(\theta) = \prod_{t=1}^T \left(\frac{1}{N} \sum_{i=1}^N w_t^{(i)} \right). \tag{2.2.32}$$

However, working directly with the likelihood typically results in numerical difficulties. To avoid this problem in practice, we instead use an estimate of the log-likelihood

$$\widehat{\ell}(\theta) = \log \widehat{\mathcal{L}}(\theta) = \sum_{t=1}^T \log \left[\sum_{i=1}^N w_t^{(i)} \right] - T \log N. \tag{2.2.33}$$

The algorithm for approximating the likelihood(log-likelihood) can thus be summarised as follows.

Algorithm 2 APF for log-likelihood estimation

Input: An SSM, observations: $y_{1:T}$, no. particles: N.
Output: MC estimation of the log-likelihood: $\widehat{\ell}(\theta)$.

1: Initialise particles $x_1^{(i)}$for $i = 1$ to N.
2: for $t = 1$ to T do
3: Resample the particles with weights $\left\{w_{t-1}^{(i)}\right\}_{i=1}^N$.
4: Propagate the particles using $R_\theta(\cdot)$.
5: Calculate new importance weights $\left\{w_t^{(i)}\right\}_{i=1}^N$.
6: end for
7: Compute log-likelihood $\widehat{\ell}(\theta)$.

As previously discussed, the likelihood $\mathcal{L}(\theta)$ and log-likelihood $\ell(\theta)$ play important roles in both ML and Bayesian parameter inference. Details relating to parameter

inference will be discussed in the next chapter.

In practice, any physical system is nonlinear, as long as it is analyzed with sufficient precision. Therefore, PF algorithms are developing rapidly in various fields because of the good estimation performance in nonlinear and non-Gaussian systems.

2.3 State Estimation in Linear Gaussian SSM

In this section, we start to implement the PF algorithm based on the process we discussed in the previous sections. To simplify, we consider the basic linear Gaussian state-space (LGSS) model.

The particular LGSS model considered is given by

$$x_1 \sim \mu_\theta(x_1), \quad x_t \left| x_{t-1} \sim \mathcal{N}\left(x_t; \phi x_{t-1}, \sigma_v^2\right), \quad y_t \right| x_t \sim \mathcal{N}\left(y_t; x_t, \sigma_e^2\right), \tag{2.3.1}$$

where parameters are denoted by $\theta = \{\phi, \sigma_v, \sigma_e\}$ and $\mathcal{N}(x; \mu, \sigma^2)$ denote the Gaussian density with mean μ and standard deviation $\sigma > 0$. $\phi \in (-1, 1)$ determines the persistence of the state, while $\sigma_v, \sigma_e \in \mathbb{R}_+$ denote the standard deviations of the state transition noise and the observation noise, respectively.

We begin with generating data from the model with $T = 250$ observations and $\theta = \{0.75, 1.00, 0.10\}, x_1 = 0$. Figure 2.5 shows the simulated data record.

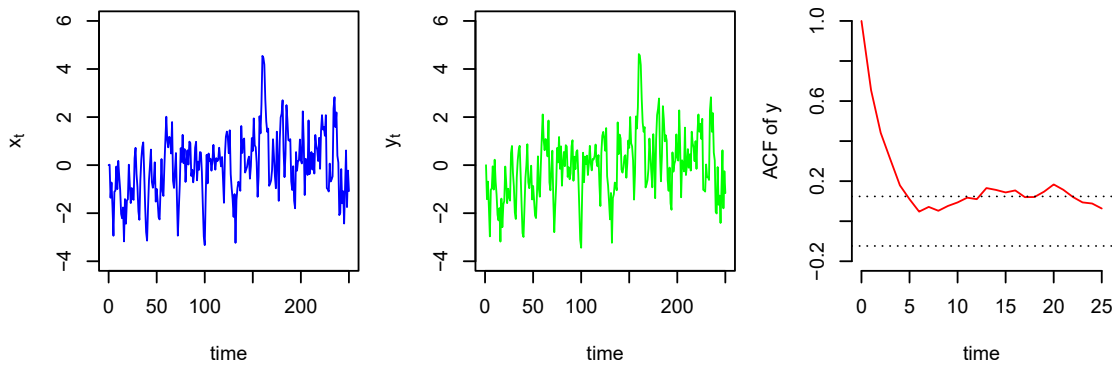

Figure 2.5: Simulated data from the LGSS model with latent state (left), observations (middle), and autocorrelation function (ACF) of the observations (right).

Then we make use of the implementation of particle filter algorithm to estimate the filtered state and to investigate the properties of this estimate. The estimates from the particle filter are compared with the corresponding estimates from the Kalman filter. As we have discussed in the previous sections, KF algorithm is an optimal filtering algorithm for LGSS model. Thus, comparing with Kalman filter is a sensible way to measure the performance of particle filter algorithm.

On the top of Figure 2.6, is the set of observations simulated from the LGSS model we mentioned before. In the middle we present the difference between the state estimate from the Kalman filter and the estimate from the particle filter algorithm using N = 2000 particles. The accuracy of particle filter algorithm can also be measured by the bias (absolute error) and the mean square error (MSE) of the state estimation. The bias is defined by

$$\text{Bias}\left(\widehat{x}_t^N\right) = \frac{1}{T}\sum_{t=1}^{T}\left|\widehat{x}_t^N - \widehat{x}_t\right|, \tag{2.3.2}$$

and the mean square error is denoted by

$$\text{MSE}\left(\widehat{x}_t^N\right) = \frac{1}{T}\sum_{t=1}^{T}\left(\widehat{x}_t^N - \widehat{x}_t\right)^2, \tag{2.3.3}$$

where $\widehat{x}_t^N$ denotes the estimation from PF algorithm and $\widehat{x}_t$ is the estimation obtained by KF algorithm. Their logarithms for different values of N are presented in the bottom of Figure 2.6. We can see that the bias and the MSE decrease rapidly as N increases.

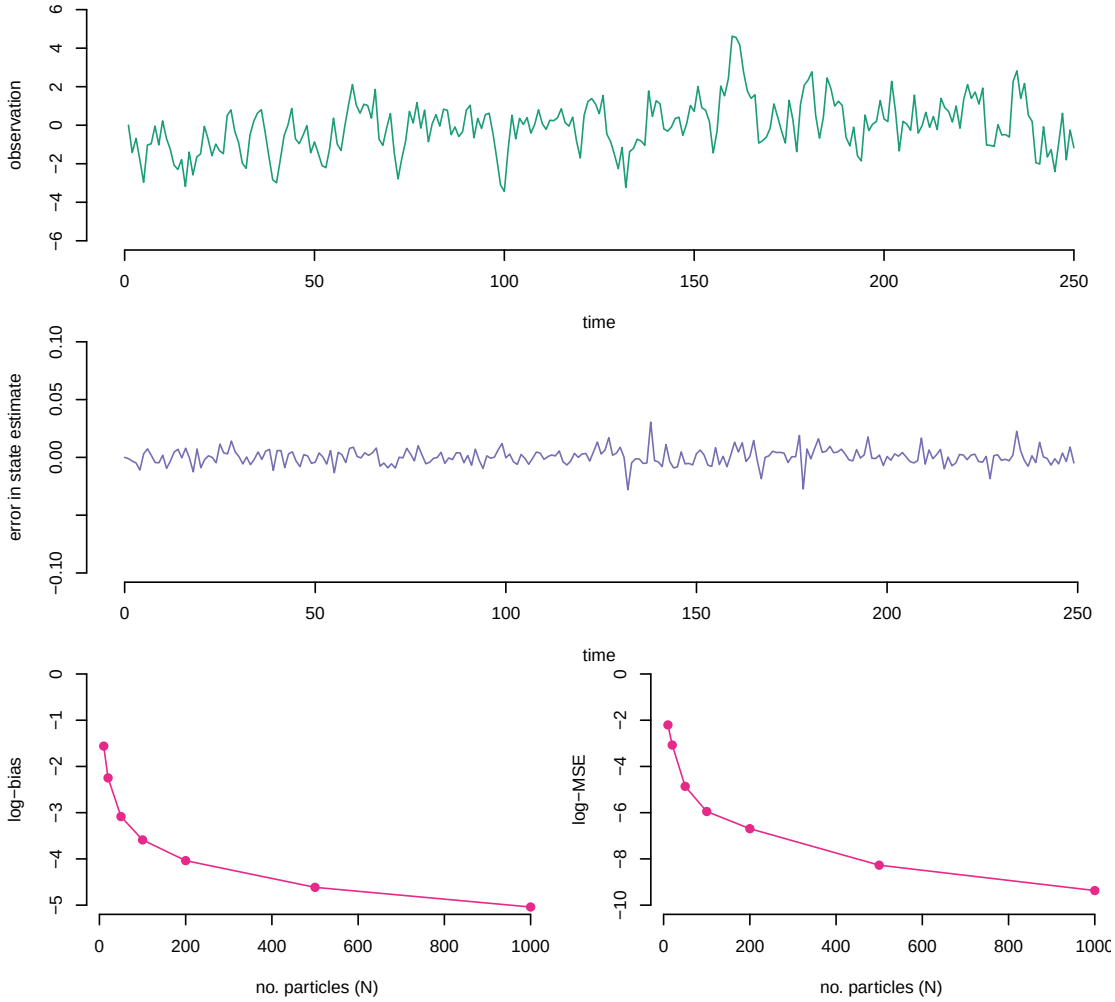

Figure 2.6: Top: a simulated set of observations y_t from the LGSS model. Middle: the error in latent state estimate using particle filter algorithm with N = 2000. Bottom: the estimated log-bias (left) and log-MSE (right) for the particle filter algorithm with different number of particles N.

Chapter 3

Parameter Inference Using Particle Methods

In this chapter, We describe how the particle filtering algorithm can be used to implement parameter inference techniques. We give some overview of Bayesian parameter inference and maximum likelihood parameter inference first and then proceeds with the implementation of these methods with some examples.

3.1 Bayesian parameter inference

3.1.1 Overview of Bayesian methods

Sampling based approaches or approximate analytical computations are main ideas to do Bayesian parameter inference. Examples of sampling based approaches are MCMC methods or SMC methods [35]. Examples of analytical approximations are the integrated nested Laplace approximation (INLA) [36], variational Bayes (VB) [37], and expectation propagation (EP) [38].

In this section, we are primarily interested in the particle Metropolis-Hastings (PMH) algorithm for Bayesian parameter inference and some examples will be shown here. A tutorial article [39] provides a comprehensive introduction to the particle Metropolis-Hastings (PMH) algorithm for parameter inference. We do not introduce the details about PMH in this thesis. We just do parameter inference by PMH method mentioned in that tutorial article first in this section and then compare with the results from ML methods in next section.

3.1.2 Estimating the parameters in LGSS model

To simplify the parameter inference problem in the LGSS model (2.3.1), we assume $\theta = \phi$ is the only unknown parameter and keep $\{\sigma_v, \sigma_e\} = \{1.00, 0.10\}$ fixed to their true values. Following the PMH algorithm in article [39], setting the initial value of the Markov chain by $\theta_0 = 0.5$, the number of iterations by $K = 2000$, and the number of particles by $N = 5000$, we can get the results of parameter inference by PMH algorithm as shown in figure 3.1. Three runs of PMH are presented using different step sizes $\epsilon = 0.01$ (left), 0.10 (center) and 0.50 (right). To make sure that the Markov chain has reached its stationary regime, we discard the first 1000 iterations as burn-in and only use the last 1000 iterations to approximate the parameter posterior.

The resulting estimate of the posterior mean is $\widehat{\phi} = 0.66$ for $\epsilon = 0.01$ and 0.1, and $\widehat{\phi} = 0.65$ for $\epsilon = 0.5$. Actually, the choice of ϵ influences the correlation in the Markov chain, thus a good choice of ϵ is important to obtain an efficient algorithm.

We note that the posterior estimate of unknown parameter differs slightly from the true value 0.75. From the asymptotic theory of the Bayesian estimator, we know that this is due to the relatively small sample size T and a finite K. If we set number of observations $T = 500$, we will get the result $\widehat{\phi} = 0.72$, which is much closer to the true value 0.75. And also, the estimated posterior variance will be smaller, which means the results are more stable when T goes larger.

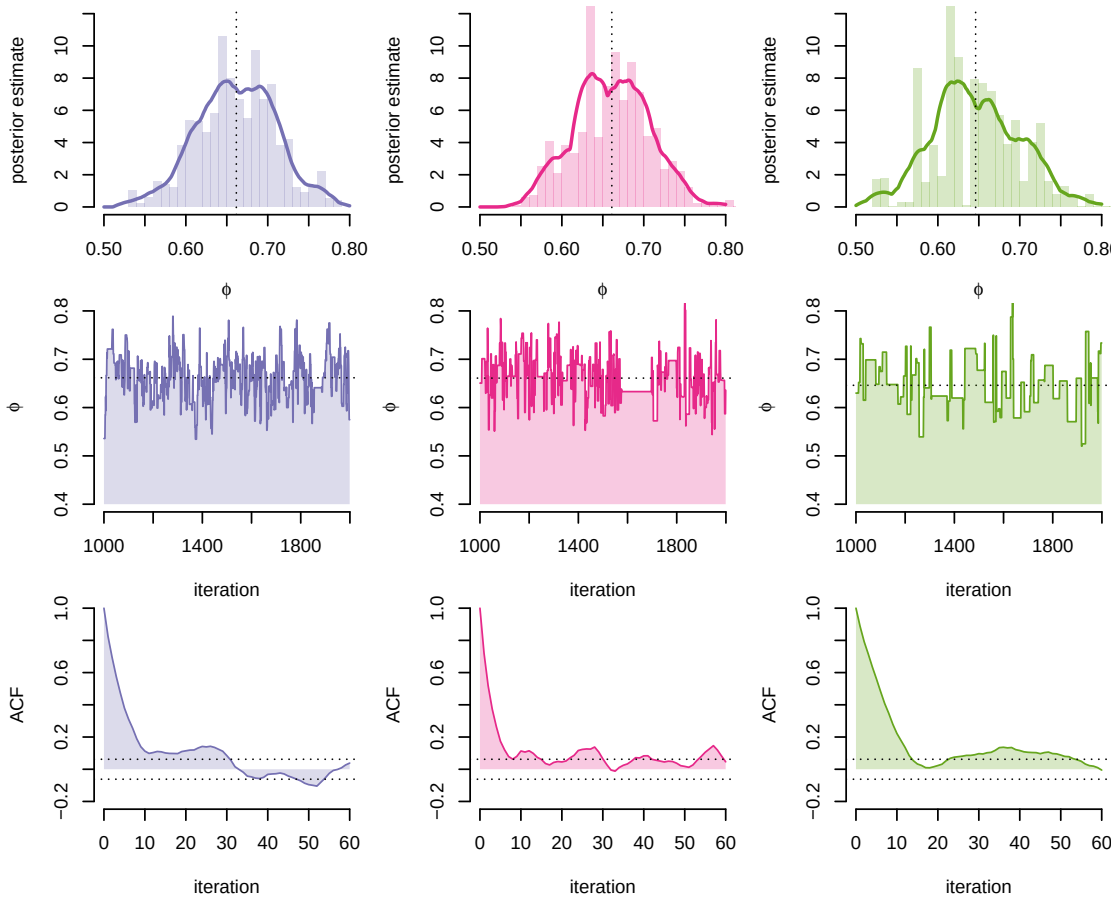

Figure 3.1: The parameter inference results using PMH methods for different step sizes:$\epsilon = 0.01$ (left), 0.10 (center) and 0.50 (right). Top: histogram and kernel density estimate. Middle: the state of the Markov chain after the 1000 burn-in. Bottom: the estimated ACF of the Markov chain. Dotted lines in the top and middle plots indicate mean value and the dotted lines in the bottom plot indicate the 95% confidence intervals of the ACF coefficients.

3.1.3 Estimating the parameters in nonlinear model

We now proceeds with application of the PMH algorithm to infer the parameters of a nonlinear SSM called stochastic volatility model [40]. The SV model is defined by

$$\begin{aligned} x_1 &\sim \mathcal{N}\left(x_1; \mu, \frac{\sigma_v^2}{1-\phi^2}\right), \\ x_{t+1} \mid x_t &\sim \mathcal{N}\left(x_{t+1}; \mu + \phi\left(x_t - \mu\right), \sigma_v^2\right), \\ y_t \mid x_t &\sim \mathcal{N}\left(y_t; 0, \exp\left(x_t\right)\right), \end{aligned} \tag{3.1.1}$$

where the parameters are denoted by $\theta = \{\mu, \phi, \sigma_v\}$. Here, $\mu \in \mathbb{R}, \phi \in [-1, 1]$ and $\sigma_v \in \mathbb{R}_+$ denote the mean value, the persistence and standard deviation of the state process, respectively.

This model is important in econometrics, where the latent state x_t is known as the log-volatility and the observations y_t are so-called log-returns. The log-volatility is useful for risk management and to price various financial contracts [41].

In the tutorial article [39], the data from Quandl for the period between January 2, 2012 and January 2, 2014 are extracted to do parameter inference and the resulting estimate are shown in Figure 3.2.

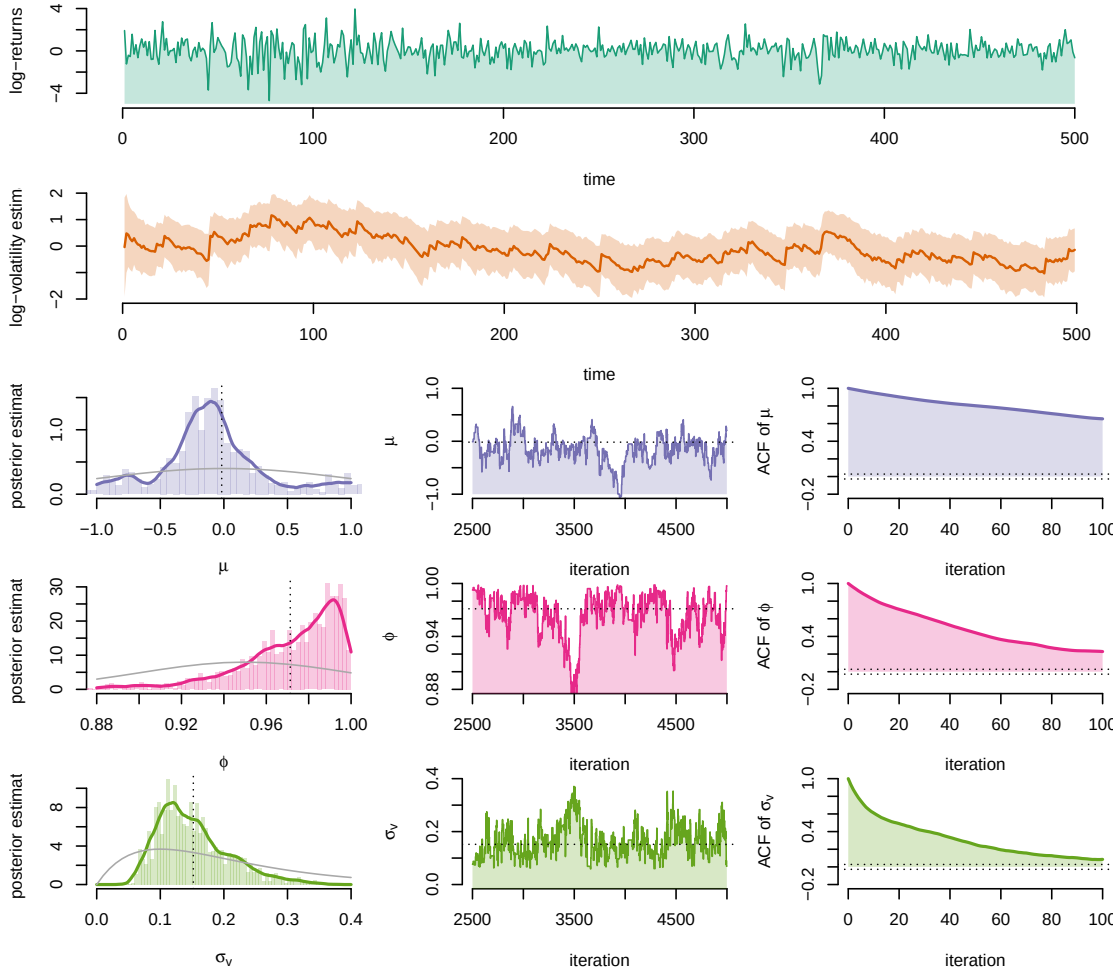

Figure 3.2: First two lines: the daily log-returns and estimated log-volatility with 95% confidence intervals of the NASDAQ OMXS30 index for the period between January 2, 2012 and January 2, 2014. Bottom: the posterior estimate (left), the trace of the Markov chain (middle) and the corresponding ACF (right) of the parameters obtained from PMH.

3.2 Maximum likelihood parameter inference

3.2.1 Overview of ML methods

The parameter inference problem in the ML framework is described in equation (1.2.3). In most situations, it cannot be solved by analytical calculations. Instead popular approaches are mainly based on optimisation and other iterative algorithms. Gradient-based optimisation algorithms [30, 31] are widely used to maximise the log-likelihood and thereby estimate the parameters of the model. This method operates by an iterative application of

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \epsilon_k \widehat{\mathcal{S}}\left(\theta_k\right). \tag{3.2.1}$$

where $\widehat{\theta}_k$ denotes the current estimate of the parameter vector and $\widehat{\theta}_{k+1}$ is the proposed next estimate. ϵ_k means the step length and $\widehat{\mathcal{S}}\left(\theta_k\right)$ is the score function which we will discuss in detail and show how to calculate later. Furthermore, if we can estimate the Hessian matrix, Newton optimisation algorithm can be implemented by iteration of

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \epsilon_k \widehat{\mathcal{J}}^{-1}\left(\theta_k\right) \widehat{\mathcal{S}}\left(\theta_k\right). \tag{3.2.2}$$

where $\widehat{\mathcal{J}}$ is the information matrix (negative Hessian).

Another popular method for ML inference is the expectation maximisation algorithm where the latent states are seen as missing information and are estimated together with the parameter vector of the model. An additive functional which can be obtained by smoothing is required in this algorithm. By using different smoothing method, we can obtain different kinds of expectation maximisation algorithm [32, 33, 34].

Here we are mainly interested in the Newton optimisation algorithm. Before we implement the algorithm, from equation (3.2.2) we know that we should first calculate the score function and the observed information matrix by particle filter algorithm.

Fisher's identity for the score vector of the log-likelihood is

$$\nabla \log p_\theta\left(y_{1:t}\right) = \int \nabla \log p_\theta\left(x_{1:t}, y_{1:t}\right) p_\theta\left(x_{1:t} \mid y_{1:t}\right) dx_{1:t}. \tag{3.2.3}$$

Similarly, the observed information matrix satisfies Louis' identity

$$-\nabla^2 \log p_\theta\left(y_{1:t}\right) = \nabla \log p_\theta\left(y_{1:t}\right) \nabla \log p_\theta\left(y_{1:t}\right)^{\mathrm{T}} - \frac{\nabla^2 p_\theta\left(y_{1:t}\right)}{p_\theta\left(y_{1:t}\right)}, \tag{3.2.4}$$

where

$$\begin{aligned} \frac{\nabla^2 p_\theta\left(y_{1:t}\right)}{p_\theta\left(y_{1:t}\right)} = & \int \nabla \log p_\theta\left(x_{1:t}, y_{1:t}\right) \nabla \log p_\theta\left(x_{1:t}, y_{1:t}\right)^{\mathrm{T}} p_\theta\left(x_{1:t} \mid y_{1:t}\right) dx_{1:t} \\ & + \int \nabla^2 \log p_\theta\left(x_{1:t}, y_{1:t}\right) p_\theta\left(x_{1:t} \mid y_{1:t}\right) dx_{1:t}. \end{aligned} \tag{3.2.5}$$

To recursively compute the score and observed information matrix, we define the vector $\alpha_t^{(i)} = \nabla \log p_\theta \left(x_{1:t}^{(i)}, y_{1:t}\right)$, and the matrix $\beta_t^{(i)} = \nabla^2 \log p_\theta \left(x_{1:t}^{(i)}, y_{1:t}\right)$. Then the particle approximation process for score vector and information matrix proceeds as follows.

1. Resample the particle set $\left\{x_{1:t-1}^{(i)}, \alpha_{t-1}^{(i)}, \beta_{t-1}^{(i)}\right\}_{i=1}^{N}$ using the weights $\left\{w_{t-1}^{(i)}\right\}$ to obtain a set of N new particles.
2. For $i = 1, \cdots, N$, sample $x_t^{(i)} \sim q_\theta \left(\cdot \mid y_t, x_{t-1}^{(i)}\right)$, and compute the weights

$$w_t^{(i)} \propto \frac{g_\theta \left(y_t \mid x_t^{(i)}\right) f_\theta \left(x_t^{(i)} \mid x_{t-1}^{(i)}\right)}{q_\theta \left(x_t^{(i)}, y_t \mid x_{t-1}^{(i)}\right)}. \tag{3.2.6}$$

3. Update $\left\{\alpha_t^{(i)}, \beta_t^{(i)}\right\}_{i=1}^{N}$, the score estimate S_t and observed information matrix estimate Σ_t:

$$\begin{aligned}
\alpha_t^{(i)} &= \alpha_{t-1}^{(i)} + \nabla \log g_\theta \left(y_t \mid x_t^{(i)}\right) + \nabla \log f_\theta \left(x_t^{(i)} \mid x_{t-1}^{(i)}\right) \\
\beta_t^{(i)} &= \beta_{t-1}^{(i)} + \nabla^2 \log g_\theta \left(y_t \mid x_t^{(i)}\right) + \nabla^2 \log f_\theta \left(x_t^{(i)} \mid x_{n-1}^{(i)}\right) \\
S_t &= \sum_{i=1}^{N} w_t^{(i)} \alpha_t^{(i)}, \quad \text{and} \quad \Sigma_t = S_t S_t^{\mathrm{T}} - \sum_{i=1}^{N} w_t^{(i)} \left(\alpha_t^{(i)} \alpha_t^{(i)\mathrm{T}} + \beta_t^{(i)}\right).
\end{aligned} \tag{3.2.7}$$

The estimates are obtained by substituting $\hat{p}_\theta \left(x_{1:t} \mid y_{1:t}\right)$ for $p_\theta \left(x_{1:t} \mid y_{1:t}\right)$ into equations (3.2.3) - (3.2.5).

3.2.2 Estimating the parameters in LGSS model

Consider the LGSS model (2.3.1) with the true parameter vector $\theta^* = \{0.75, 1.00, 0.10\}$. We generate data of length $T = 500$ using the initial value $x_1 = 0$. As assumed in section 3.1.2, here ϕ is chosen to be the single unknown parameter and $\{\sigma_v, \sigma_e\} = \{1.00, 0.10\}$ fixed to their true values. We set the number of particles by $N = 5000$ and calculate the log-likelihood, the score function and the expected information matrix for different value of ϕ. The results are presented in Figure 3.3.

It can be seen obviously in the figure that the maximum of the log-likelihood is near the true parameter and the score function is zero close to this point. Since the score function is the slope of the log-likelihood, the zero of the score function results in a maximum of the log-likelihood function when the expected information (negative Hessian) is positive.

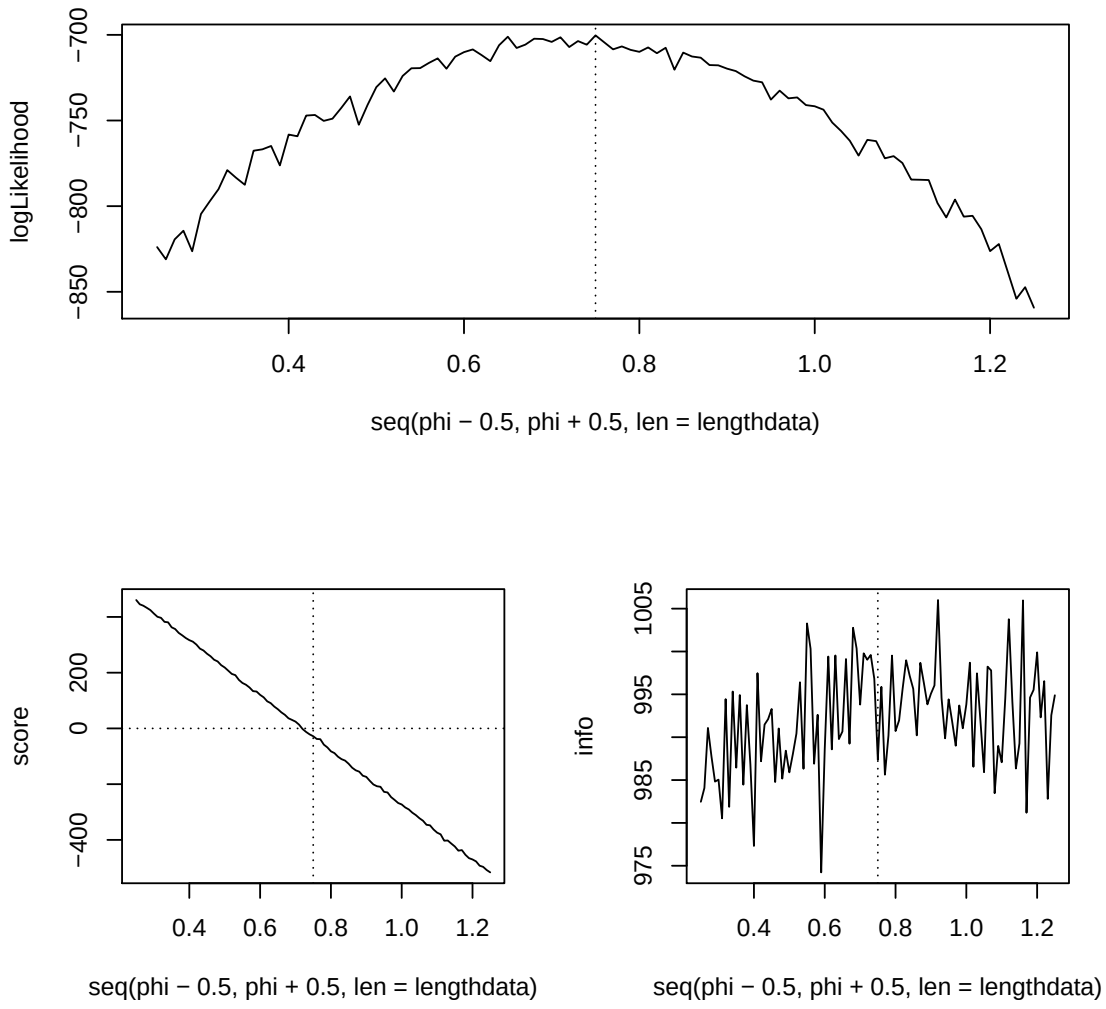

Figure 3.3: The estimates of the log-likelihood (top) the score function (bottom left) and the expected information matrix (bottom right) of the LGSS model by PF algorithm. The dotted lines indicate the true value of the parameter ϕ (0.75).

Now that we have the estimates of log-likelihood, score function and expected information, we can infer the value of ϕ by Newton optimisation. Here we set a stopping point $|\widehat{\theta}_{k+1} - \widehat{\theta}_k < 0.001|$ and initial value $\theta_1 = 0.5$ for the iteration equation (3.2.2). That is, when this condition is arrived, we stop the iteration and return $\widehat{\theta}_{k+1}$ as our estimate for the parameter. This method converges much fater than the PMH method and the iteration will often be stopped in ten times. However, in the PMH algorithm, we should discard the first 1000 iterations to ensure the stability.

To get a more convincing results, we generate 500 data sets from the same true parameter vector and then run the estimation procedure on each of the simulated data set to get 500 estimated parameter vectors. The results are presented in Figure 3.4. The mean value of the 500 estimated parameter is $\widehat{\phi} = 0.746$ with

standard deviation = 0.03, which is very close to the true value 0.75. What's more, if we change the initial value of the iteration, both the convergence rate and the estimated results are consistent(almost no difference). Therefore, it seems that Newton optimisation based on PF algorithm is an efficient way to do parameter inference in SSMs.

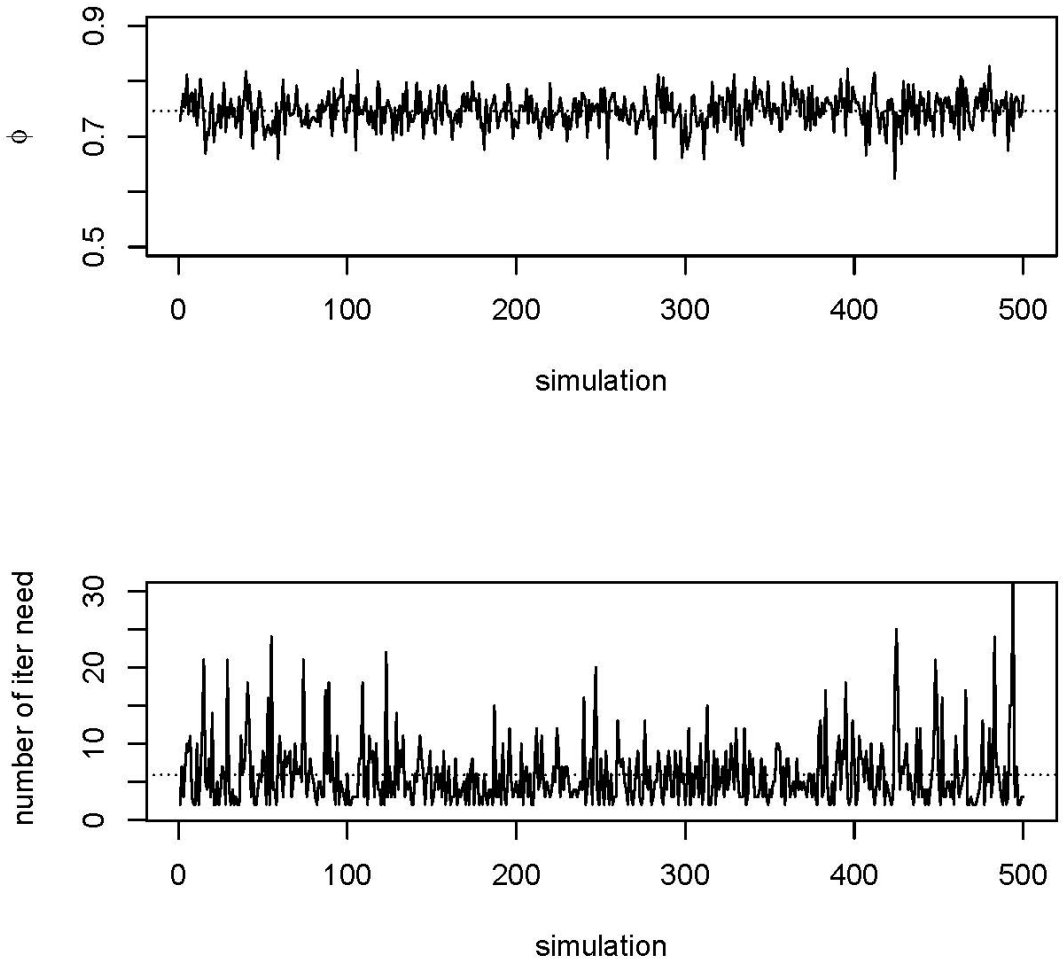

Figure 3.4: Estimated results by Newton optimisation. Top: estimated value of parameter. Bottom: number of iterations to converge. The dotted lines are the mean value of 500 simulations.

3.2.3 Estimating the parameters in nonlinear model

Similarly, to work on the SV model defined by equation (3.1.1), we generate data of length $T = 1000$ with true parameter vector $\theta^* = \{\mu^*, \phi^*, \sigma_v^*\} = \{-1.02, 0.95, 0.25\}$.

If we choose ϕ to be the unknown parameter, the estimates of the log-likelihood, the score function and the expected information are presented in Figure 3.5. Similar to the results of the LGSS model, at the position close to the true value of the

parameter, the log-likelihood has a maximum, the score function is zero and the expected information is positive.

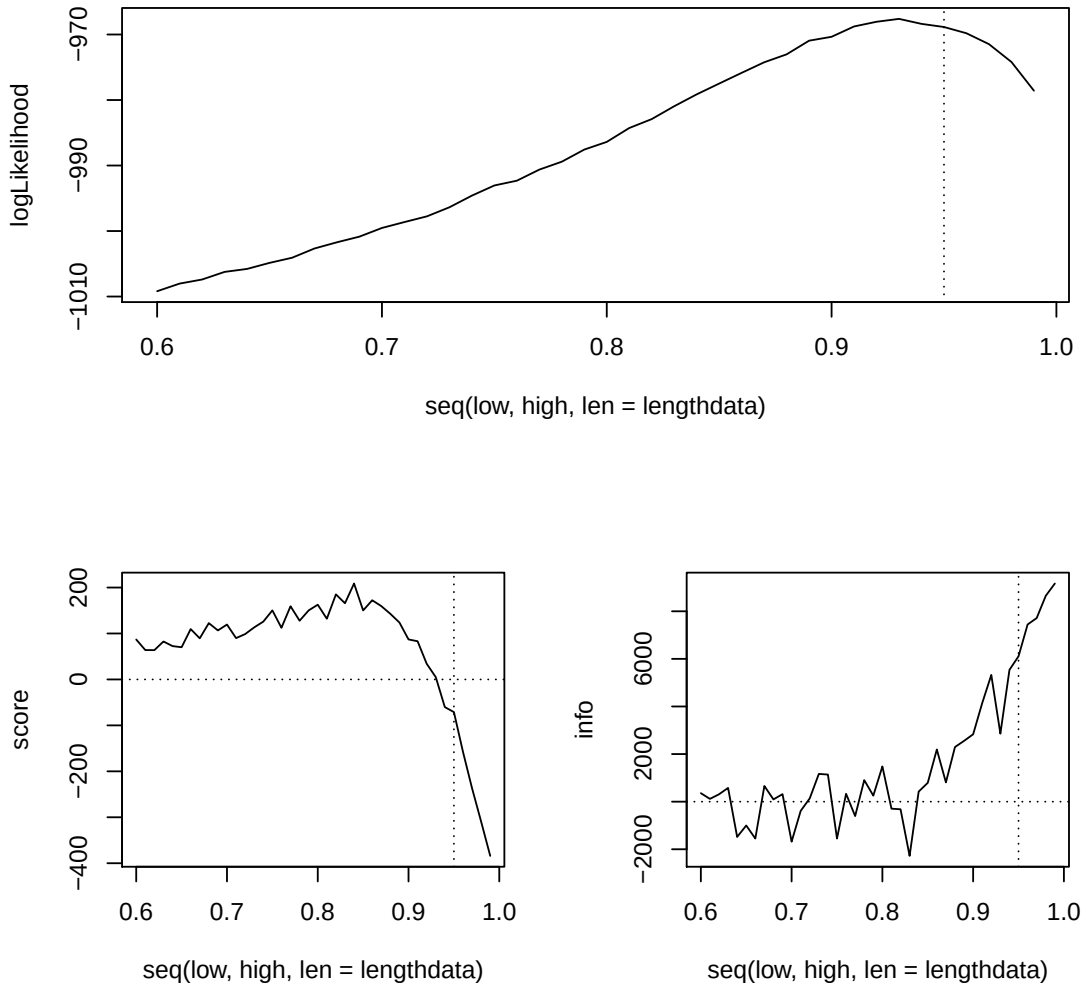

Figure 3.5: The estimates of the log-likelihood (top) the score function (bottom left) and the expected information matrix (bottom right) of the SV model by PF algorithm for different values of ϕ. The dotted lines indicate the true value of the parameter ϕ (0.95).

The estimates based on choosing μ to be the unknown parameter also returns the same results, see Figure 3.6. We note that the curve of log-likelihood here is not as smooth as other examples we mentioned before. We can make it smooth by choosing a larger value of T.

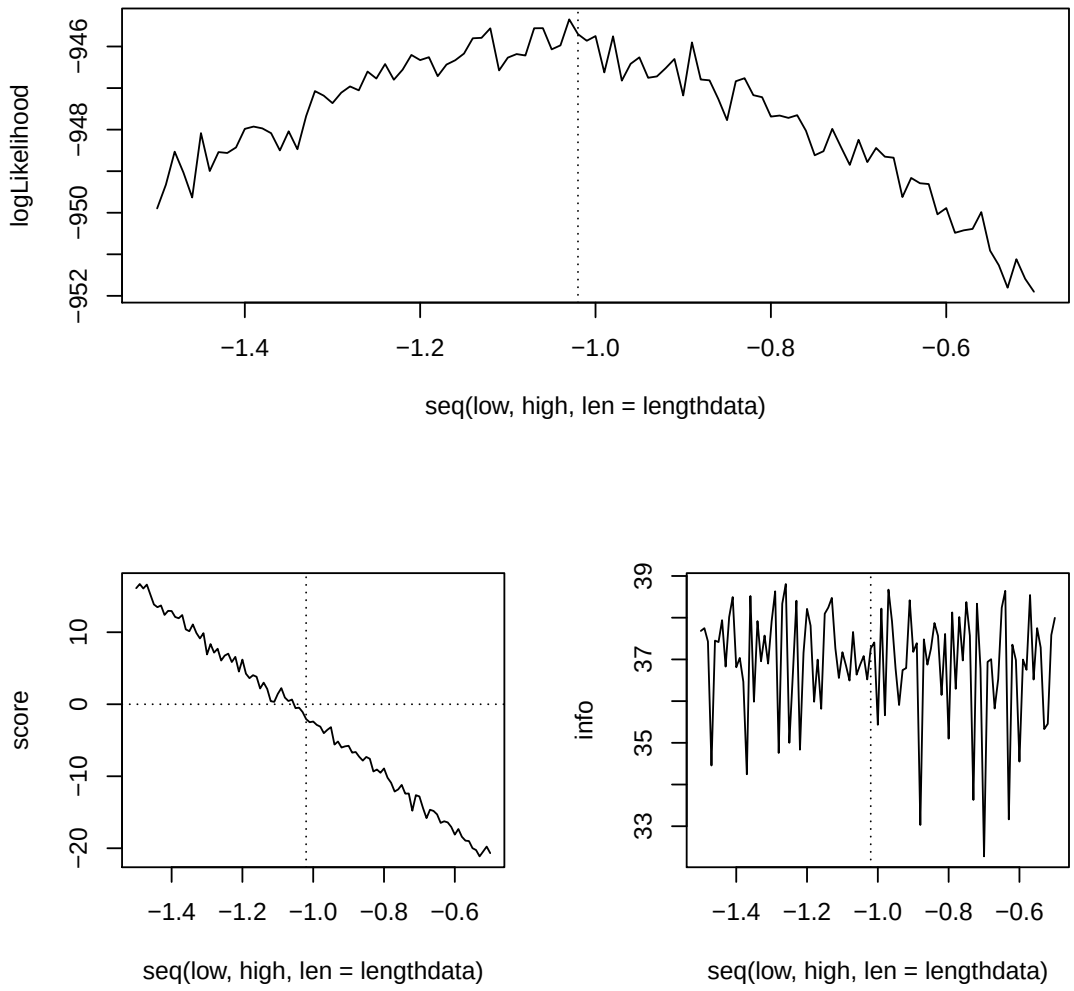

Figure 3.6: The estimates of the log-likelihood (top) the score function (bottom left) and the expected information matrix (bottom right) of the SV model by PF algorithm for different values of μ. The dotted lines indicate the true value of the parameter μ (-1.02).

Then we simulate 500 data set to infer the parameter. Results for ϕ and μ are presented in Figure 3.7 and Figure 3.8 respectively. The mean value of the estimated parameter is $\widehat{\phi} = 0.9495$, with 95% confidence interval [0.9339, 0.9652] and $\widehat{\mu} = -1.0254$, with 95% confidence interval [-1.3505, -0.7004]. We note that for both ϕ and μ, the Newton optimisation based on particle filter method gives a good resulting estimate of the true value. Compared with the PMH algorithm, Newton optimisation convergences much faster and returns more accurate results.

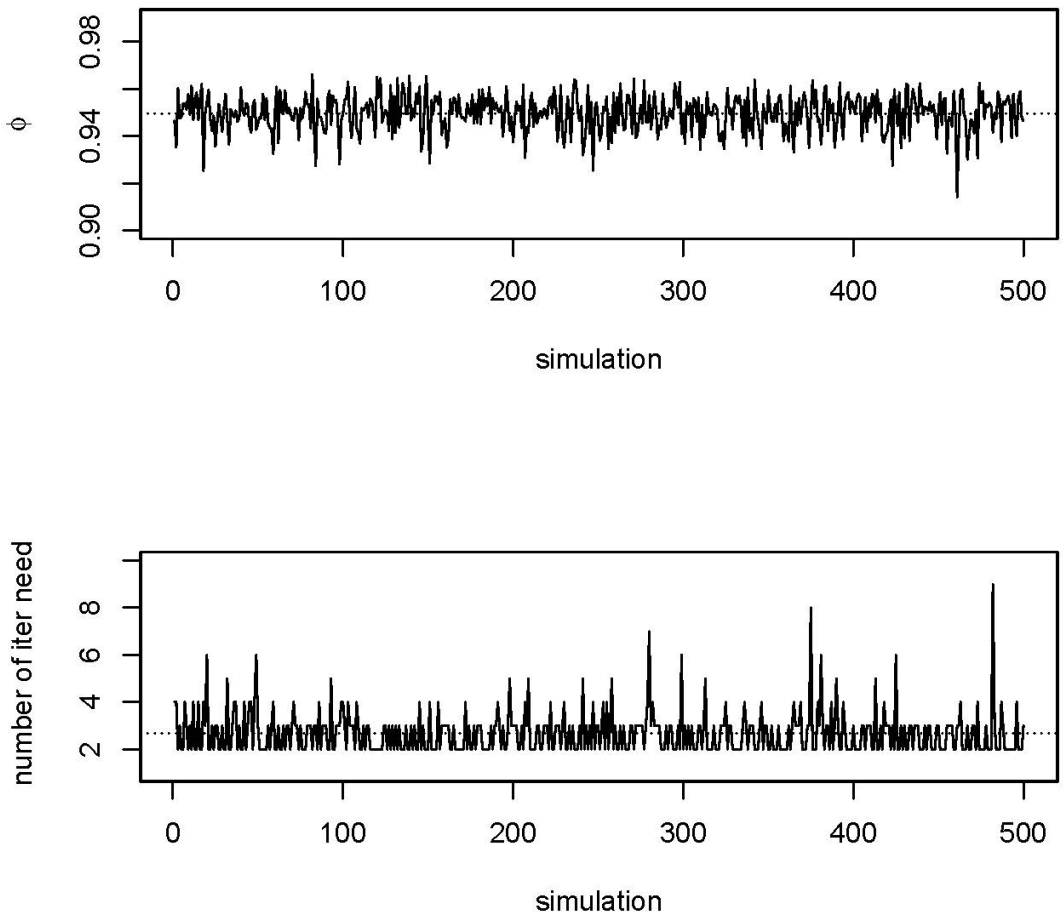

Figure 3.7: Estimated results for ϕ by Newton optimisation. Top: estimated value of parameter. Bottom: number of iterations to converge. The dotted lines are the mean value of 500 simulations.

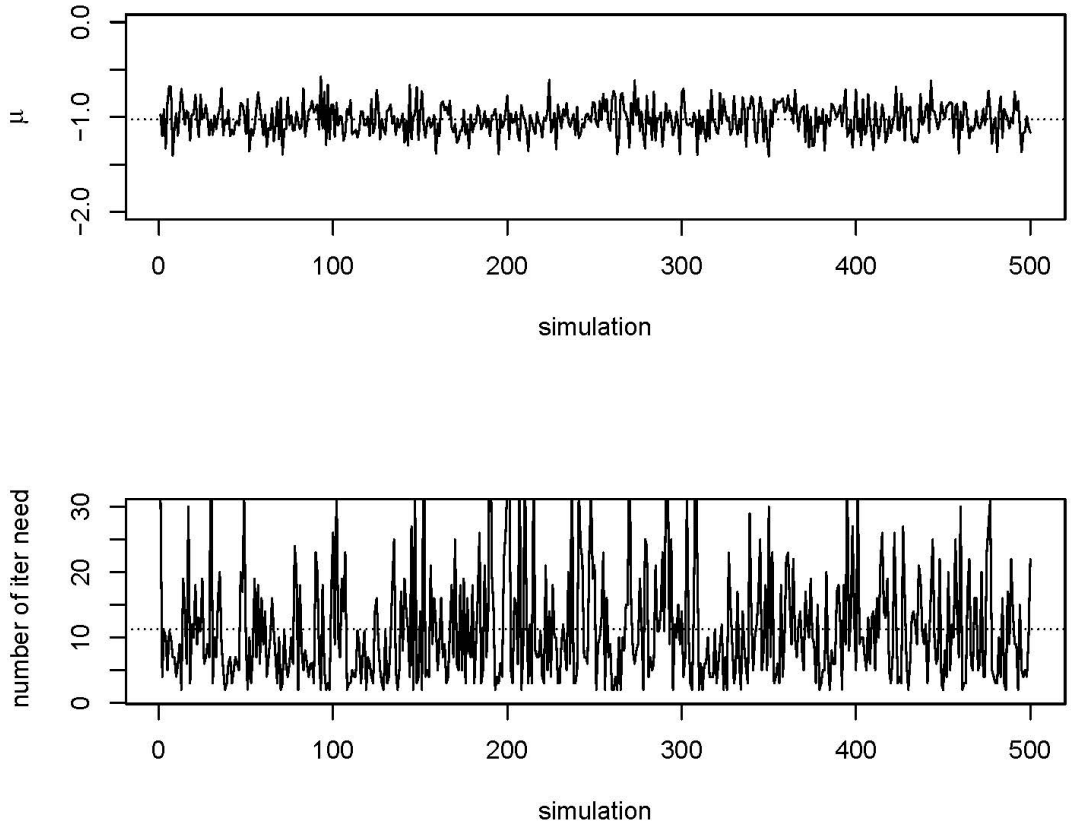

Figure 3.8: Estimated results for μ by Newton optimisation. Top: estimated value of parameter. Bottom: number of iterations to converge. The dotted lines are the mean value of 500 simulations.

Chapter 4

Conclusion

We have described the Particle Filter algorithm to get the Monte Carlo approximation of the latent state in State Space models. What's more, in parameter inference problems, we apply this method to approximate the likelihood, score function and Hessian matrix and thus get the maximum likelihood estimation of the unknown parameter vector by Newton optimisation.

However, there are some topics we do not finish as planned and can be investigated in the future. We have tried the LGSS model and SV model in this thesis and the next step we planned is to work on the COGARCH model, which doesn't have a tractable log-likelihood.

The particle algorithm we implement in this thesis to compute the score function and Hessian matrix is from the paper [30]. This paper also introduces another particle algorithm which has higher computational complexity but some other advantages. What we can do in future is to apply the second algorithm in ML parameter inference problems and compare the two methods.

References

[1] Cappé, Olivier and Moulines, Eric and Rydén, Tobias, Inference in hidden Markov models, *Springer Science & Business Media*, 2006.

[2] Arnaud, Doucet and de Freitas, Nando and Gordon, Neil, Sequential Monte Carlo methods in practice, *Information Science and Statistics (Springer New York, 2001)* (2001).

[3] Elliott, Robert J and Aggoun, Lakhdar and Moore, John B, Hidden Markov models: estimation and control, *Springer Science & Business Media* **29**,(2008).

[4] West, Mike and Harrison, Jeff, Bayesian forecasting and dynamic models *Springer Science & Business Media* (2006).

[5] Durbin, James and Koopman, Siem Jan,Time series analysis by state space methods, *Oxford university press* (2012).

[6] Langrock, Roland, Some applications of nonlinear and non-Gaussian state–space modelling by means of hidden Markov models, *Journal of Applied Statistics* **38**,12,(2011),2955–2970.

[7] Berger, James O, Statistical decision theory and Bayesian analysis, *Springer Science & Business Media* (2013).

[8] Robert, Christian, The Bayesian choice: from decision-theoretic foundations to computational implementation, *Springer Science & Business Media* (2007).

[9] Anderson, Brian DO and Moore, John B, Optimal filtering, *Courier Corporation* (2012).

[10] Stigler, Stephen M, Gauss and the invention of least squares, *The Annals of Statistics* (1981),465–474.

[11] Agrawal, Akshay and Verschueren, Robin and Diamond, Steven and Boyd, Stephen, A rewriting system for convex optimization problems, *Journal of Control and Decision* **5**(1),(2018),42–60.

[12] Wiener, Norbert, Extrapolation, interpolation, and smoothing of stationary time series, vol. 2, *MIT press Cambridge, MA* (1949).

[13] Kolmogoroff, A, Interpolation und extrapolation von stationaren zufalligen folgen, *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya* **5**(1),(1941),3–14.

[14] Kalman, Rudolph Emil, A new approach to linear filtering and prediction problems, 1960.

[15] Bucy, Richard S and Senne, Kenneth D, Digital synthesis of non-linear filters, *Automatica* **7**(3),(1971),287–298.

[16] Sunahara, Yoshifumi, An approximate method of state estimation for nonlinear dynamical systems, (1970).

[17] Julier, Simon J and Uhlmann, Jeffrey K, New extension of the Kalman filter to nonlinear systems, *Signal processing, sensor fusion, and target recognition VI* **3068**,(1997),182–193.

[18] Gordon, Neil J and Salmond, David J and Smith, Adrian FM, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE proceedings F (radar and signal processing)* **140**(2),(1993),107–113.

[19] Ho, YC and Lee, RCKA, A Bayesian approach to problems in stochastic estimation and control, *IEEE transactions on automatic control* **9**(4),(1964),333–339.

[20] Hammersley, John M and Morton, K William, Poor man's monte carlo, *Journal of the Royal Statistical Society: Series B (Methodological)* **16**(1),(1954), 23–38.

[21] Marshall, Andrew W, The use of multi-stage sampling schemes in Monte Carlo computations, *RAND CORP SANTA MONICA CALIF* **(**1954).

[22] Kong, Augustine and Liu, Jun S and Wong, Wing Hung, Sequential imputations and Bayesian missing data problems, *Journal of the American statistical association* **89**(425),(1994),278–288.

[23] Kong, Augustine, A note on importance sampling using standardized weights, *University of Chicago, Dept. of Statistics, Tech. Rep* **348**,(1992).

[24] Liu, Jun S and Chen, Rong, Sequential Monte Carlo methods for dynamic systems, *Journal of the American statistical association* **93**(443),(1998),1032–1044.

[25] Bolic, Miodrag and Djuric, Petar M and Hong, Sangjin, Resampling algorithms and architectures for distributed particle filters, *IEEE Transactions on Signal Processing* **53**(7),(2005),2442–2450.

[26] Fox, Dieter, Adapting the sample size in particle filters through KLD-sampling, *The international Journal of robotics research* **22**(12),(2003),985–1003.

[27] Sankaranarayanan, Aswin C and Srivastava, Ankur and Chellappa, Rama, Algorithmic and architectural optimizations for computationally efficient particle filtering, *IEEE Transactions on Image Processing* **17**(5),(2008),737–748.

[28] Li, Tiancheng and Sattar, Tariq Pervez and Sun, Shudong, Deterministic resampling: unbiased sampling to avoid sample impoverishment in particle filters, *Signal Processing* **92**(7),(2012),1637–1645.

[29] Balasingam, Balakumar and Bolić, Miodrag and Djurić, Petar M and Miguez, Joaquin, Efficient distributed resampling for particle filters, *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011),3772–3775.

[30] Poyiadjis, George and Doucet, Arnaud and Singh, Sumeetpal S, Particle approximations of the score and observed information matrix in state space models with application to parameter estimation, *Biometrika* **98**(1),(2011),65–80.

[31] Yıldırım, Sinan and Singh, Sumeetpal S and Dean, Thomas and Jasra, Ajay, Parameter estimation in hidden Markov models with intractable likelihoods using sequential Monte Carlo, *Journal of Computational and Graphical Statistics* **24**(3),(2015),846–865.

[32] Del Moral, Pierre and Doucet, Arnaud and Singh, Sumeetpal, Forward smoothing using sequential Monte Carlo, *arXiv preprint arXiv:1012.5390* (2010).

[33] Lindsten, Fredrik and Schön, Thomas B, Backward simulation methods for Monte Carlo statistical inference, *Foundations and Trends® in Machine Learning* **6**(1),(2013),1–143.

[34] Schön, Thomas B and Wills, Adrian and Ninness, Brett, System identification of nonlinear state-space models, *Automatica* **47**(1),(2011),39–49.

[35] Chopin, Nicolas and Jacob, Pierre E and Papaspiliopoulos, Omiros, SMC2: an efficient algorithm for sequential analysis of state space models, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **75**(3),(2013),397–426.

[36] Rue, Håvard and Martino, Sara and Chopin, Nicolas, Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations, *Journal of the royal statistical society: Series b (statistical methodology)* **71**(2),(2009),319–392.

[37] Bishop, Christopher M, Pattern recognition and machine learning, *Springer* 2006.

[38] Minka, Thomas P, Expectation propagation for approximate Bayesian inference, *arXiv preprint arXiv:1301.2294* (2013).

[39] Dahlin, Johan and Schön, Thomas B, Getting started with particle Metropolis-Hastings for inference in nonlinear dynamical models, *Journal of Statistical Software* **88**(CN2),(2019),1–41.

[40] Hull, John and White, Alan, The pricing of options on assets with stochastic volatilities, *The journal of finance* **42**(2),(1987),281–300.

[41] Hull, John C, Options, Futures, and other Derivatives (ed.), *Upper Saddle River, NJ: Prentice Hall*, 2009