PARAMETER ESTIMATION IN STOCHASTIC VOLATILITY MODELS WITH MISSING DATA USING PARTICLE METHODS AND THE EM ALGORITHM

by

Jeongeun Kim

BS, Seoul National University, 1998

MS, Seoul National University, 2000

Submitted to the Graduate Faculty of

the Department of Statistics in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2005

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF STATISTICS

This dissertation was presented

by

Jeongeun Kim

It was defended on

April 22, 2005

and approved by

David S. Stoffer, Professor

Ori Rosen, Professor

Wesley Thompson, Professor

Anthony Brockwell, Professor

Dissertation Director: David S. Stoffer, Professor

PARAMETER ESTIMATION IN STOCHASTIC VOLATILITY MODELS WITH MISSING DATA USING PARTICLE METHODS AND THE EM ALGORITHM

Jeongeun Kim, PhD

University of Pittsburgh, 2005

The main concern of financial time series analysis is how to forecast future values of financial variables, based on all available information. One of the special features of financial variables, such as stock prices and exchange rates, is that they show changes in volatility, or variance, over time. Several statistical models have been suggested to explain volatility in data, and among them *Stochastic Volatility models* or SV models have been commonly and successfully used. Another feature of financial variables I want to consider is the existence of several missing data. For example, there is no stock price data available for regular holidays, such as Christmas, Thanksgiving, and so on. Furthermore, even though the chance is small, stretches of data may not available for many reasons. I believe that if this feature is brought into the model, it will produce more precise results.

The goal of my research is to develop a new technique for estimating parameters of SV models when some parts of data are missing. By estimating parameters, the dynamics of the process can be fully specified, and future values can be estimated from them. SV models have become increasingly popular in recent years, and their popularity has resulted in several different approaches proposed regarding the problem of estimating the parameters of the SV models. However, as of yet there is no consensus on this problem. In addition there has been no serious consideration of the missing data problem. A new statistical approach based on the EM algorithm and particle filters is presented. Moreover, I expand the scope of application of SV models by introducing a slight modification of the models.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to acknowledge the advice, support, and friendship of a number of people who helped me during the writing of this thesis and my time as a graduate student. First, I would like to thank Professor David S. Stoffer, my advisor, for his suggestions and constant support during this research. His excellent guidance with endless patience always leads me to the right direction. I am honored to be his student.

I also would like to thank my other committee members, Professor Ori Rosen, Professor Wesley Thompson and Professor Anthony Brockwell for their helpful suggestions and encouragement. And I wish to express my appreciation to the rest of faculty, staff and students of the statistics department at University of Pittsburgh for their encouragement, support and friendship throughout my graduate years.

I am grateful to my parents and parents-in-law. They have always provided me with unconditional love and ceaseless support, in joy and in sorrow.

Last but not least, I would like to thank my husband, Taeryon, for being always there for me. I would not have been able to complete it without him. In addition, I appreciate my unborn baby for giving me limitless happiness. I'd like to dedicate this dissertation to my growing family.

1.0 INTRODUCTION

1.1 PROBLEM STATEMENT

This thesis treats the estimation problem of *stochastic volatility models* or *SV models* when some parts of the data are missing. Stochastic volatility models have been commonly and successfully used to explain the behavior of financial variables such as stock prices and exchange rates.

In the early stage of time series data analysis, people were interested mainly in mean behavior of the data and tried to find a model which could explain it effectively. Recently, concern about volatility or variance in the data has been raised because changes or patterns in volatility are observed in real data, especially in financial data, and knowledge of volatility can be a good piece of information for a decision-making process.

Figure 1.1 is the plot of *log returns*[1] of pound/dollar exchange rates from January 2nd, 1996, to December 31th, 1998, which are explained and analyzed in a later chapter. As you can easily notice in the first plot, there exist some patterns of behavior. For some periods of time, the data experience a small variance and for other periods of time, the data show a large variance. Hence, it is not reasonable to assume that the data have a constant variance as in the classical time series analysis. Furthermore, if it is possible to predict the future variance of financial variables, it would be very useful to control the risk. For example, if there are two stocks which have the same mean but different variances, people would prefer the stock

[1] A log return or continuously compounded return of exchange rates is

$$r_t = \log \frac{e_t}{e_{t-1}},$$

where e_t is the exchange rate at time index t.

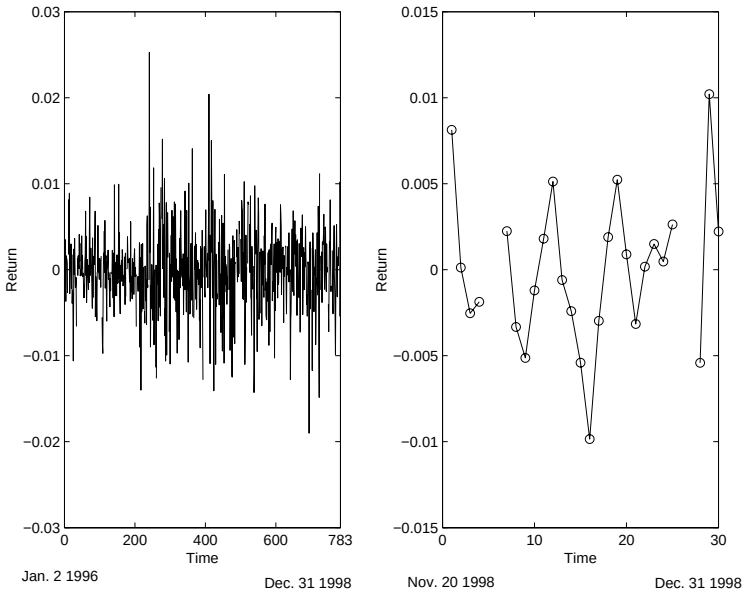

Figure 1.1: The log returns, r_t, of the pound/dollar exchange rates from January 2nd, 1996 to December 31st, 1998 (left), and the last 30 r_t's (right).

with lower variance in that it is less risky. Stochastic volatility models, which are time series models concerned with volatility, would provide people with this type of information.

In financial time series analysis, the main concern is how to forecast the future values of the variables. I adopt the SV model, which is a reasonable model for explaining variances, to the variables, and by estimating parameters in the model, I want to fully specify the dynamics of the process.

One more thing to consider in this thesis is the missing data problem. Thanks to the beauty of modern technology, there is less possibility that certain data will be missing. However, for example, there are regular holidays in stock markets. There is no stock price data available for holidays such as Thanksgiving Day and Christmas. This is true for exchange rates as you can see in the second plot of Figure 1.1, which is the plot of last 30 observations of the data. Over 30-day periods, there are 4 missing values (November 26, 27 for Thanksgiving Day and December 25, 26 for Christmas: note that one missing value returns two missing log returns). Moreover, there is still a chance that stretches of data are not available for many reasons even though that chance is small. This important fact is commonly ignored

and the data are typically analyzed as if there were no missing data. If this feature can be brought into the model, it will produce more precise estimations and forecasts.

The main objective for this thesis is to suggest a method that can handle the estimation problem of stochastic volatility models when some parts of data are missing.

1.2 STOCHASTIC VOLATILITY MODEL

1.2.1 Stochastic Volatility Model

Stochastic Volatility models belong to *state-space models* and they take the volatility or variance of the data into account. In a general SV model framework, the data, r, is generated from a probability model $f(r|x)$, where x is a vector of volatilities, and this unobserved vector x has a probabilistic structure $f(x|\theta)$ where θ is a vector of parameters. See Jacquier et al. (1994) or Shephard (1996) for more details.

In this thesis, I focus on the simple univariate model, which can be expressed with the following two equations:

$$x_t = \phi x_{t-1} + w_t, \tag{1.1}$$

$$r_t = \beta \exp(\frac{x_t}{2})\epsilon_t, \tag{1.2}$$

where $w_t \sim N(0, Q)$, $x_0 \sim N(\mu_0, \sigma_0^2)$ and $\epsilon_t \sim N(0, 1)$. Here, x_t is proportional to the log of volatility of the data, $\log(\sigma_t^2)$, where $\sigma_t^2 = \text{var}(r_t)$ and it has AR(1) structure without intercept. Squaring (1.2) and taking the logarithm of it results in a linear equation (1.3) that I use throughout this thesis.

$$y_t = \alpha + x_t + v_t, \tag{1.3}$$

where

$$\begin{aligned} y_t &= \log(r_t^2), \\ \alpha &= \log(\beta^2) + E(\log(\epsilon_t^2)), \\ v_t &= \log(\epsilon^2) - E(\epsilon_t^2) \sim \log(\chi_1^2) - E(\log(\chi_1^2)). \end{aligned}$$

Equation (1.1) and (1.3) form my version of a univariate SV model. The v_t is centered to become a zero mean variable. I use (1.3) instead of (1.2) since the normal mixture idea, which is explained in Chapter 3, can be more easily applied to (1.3). Many other authors also use a (1.3)-type equation and take advantage of its linearity. I call this SV model *Model A* throughout my thesis.

Equation (1.1) is called a *state equation* and equation (1.3) (or (1.2)) is called an *observation equation*. The main objective of this thesis is to estimate parameters in the model efficiently and to make forecasts for future volatility based on those parameters. Here, the target parameters are $\{\alpha, \phi, Q\}$.

Model A

$$\begin{aligned} x_t &= \phi x_{t-1} + w_t, \\ y_t &= \alpha + x_t + v_t, \end{aligned}$$

where $w_t \sim N(0, Q)$ and $v_t \sim \log(\chi_1^2) - E(\log(\chi_1^2))$.

1.2.2 ARCH/GARCH vs. SV Model

Stochastic Volatility models are not the first models that take volatility into consideration. Engle (1982) introduced *autoregressive conditionally heteroscedastic (ARCH)* models, and Bollerslev (1986) extended this idea to *generalized ARCH, or GARCH* models. These models let the variance be a function of previous observations and/or past variances. The difference between ARCH and GARCH models and SV models is that ARCH and GARCH models don't allow any noise in the volatility structure. In other words, their volatilities are deterministic when previous data are given. For example, GARCH(1,1) model is given by

$$r_t = \sigma_t \epsilon_t, \tag{1.4}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 r_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \tag{1.5}$$

where $\epsilon_t \sim N(0,1)$. Note that there is no error term for (1.5). Because of this property, the likelihood of ARCH or GARCH is easier to calculate and so parameter estimation in ARCH or GARCH is less burdensome. More specifically, one-step-ahead forecast density, $f(r_t|r_1, \cdots, r_{t-1})$ is available for ARCH or GARCH models and this enables us to get a likelihood of the data. Hence, estimation and testing are straightforward at least in principle. However, there is no analytic one-step-ahead forecast density for SV models. For that reason, either approximations have to be made, or numerical methods have to be used to get a likelihood for SV models.

The general comparison of the performance of the SV models and that of ARCH/GARCH is not simple because they came from different ideas and so represent different models. Empirical comparison was done by Pulgarin (2001). In this paper, based on the residuals the performance of the two models is assessed for two different data sets (Stock Dividend Yield Return data and New York Stock Exchange). Pulgarin (2001) concluded that the SV models better describe the shape of the real series based on the fact that the residuals from the GARCH models fail to achieve the assumed kurtosis (3 for normal distributions). However where the forecasting error is concerned the GARCH models give better performance.

Shephard (1996) also compared the performance of ARCH/GARCH and that of the SV models empirically. Four data sets were considered (daily exchange rate series of the Japanese yen and the German Deutsche Mark measured against the pound sterling, daily FTSE 100 and Nikkei 500 indexes, which are market indexes for the London and Tokyo equity markets) and the simple SV model and the GARCH(1,1) model were applied to these data sets. The results of Shephard (1996) suggest that the SV models are empirically more successful than the GARCH models: as measures of comparison, it uses log–likelihoods, Box-Ljung statistics with 30 lags and the fourth moments. It also comments that the SV model explains the fat-tailed behavior of returns better and this gives an explanation of the success of this model.

Kim et al. (1998) also reached similar conclusion with regard to the performance of the SV models and ARCH/GARCH models. They compared GARCH models with the SV models based on likelihood ratios and Bayes factors. According to their results, the SV model dominates the GARCH model. For details of their findings, see Kim et al. (1998).

1.3 THESIS OUTLINE

This thesis is organized in the following fashion. I start by developing a new method to analyze a simple SV model with equations (1.1) and (1.3), along with giving brief reviews of EM algorithms and particle filters which are the main parts of my method in Chapter 2. In addition, I discuss other issues such as initial parameter selection, relative likelihood, a stopping rule and a method to get standard deviation of parameters, which are essential to perform the developed method. This chapter goes into an introduction to other authors' methods for a non-missing case, the stochastic EM algorithm and the MCEM algorithm. In Chapter 3, I present modification of the observation equation (1.3) which gives more flexibility to the SV model. I present normal mixtures as an observation error and show how the whole process in Chapter 2 can be modified. Chapter 4 is devoted to the missing data problem. Chapter 5 presents simulation results and application to the real data that validate the proposed method. Finally, Chapter 6 gives an overview of the results of the thesis and presents areas for future work.

1.4 CONTRIBUTION OF THESIS

SV models have become increasingly popular in recent years, and their popularity has resulted in several different approaches proposed regarding the problem of estimating the parameters of the SV models. See Section 2.5 for a brief review of previously proposed methods by other authors. However, as of yet there is no consensus on this problem. In addition, there has been no serious consideration of the missing data problem. Hence, I expect that my new parameter estimation technique which combines the EM algorithm along with particle filters and smoothers will help people use SV models more easily and effectively.

Also, every other article on the estimation problem of the SV model allows the observation error to be fixed as an assumed distribution (a normal distribution or a t-distribution). By introducing two normal mixtures with parameters estimated in the procedure, the range of application of the SV model can be widened. In other words, the parameter estimation method in this thesis can give a satisfactory estimation result even though the assumption

about the observation error is violated.

2.0 PARAMETER ESTIMATION IN SV MODEL

In this chapter, I leave the missing data problem out of consideration. After developing a proper method for the data without missing values in Chapter 2 and Chapter 3, I move on to the modification of the method for the missing data case in Chapter 4.

Basic idea to handle the parameter estimation problem in the SV model is to incorporate the EM algorithm and particle methods: particle filters and smoothers. Before presenting a new method, I start with brief reviews of the EM algorithm and particle methods in Section 2.1. In Section 2.2, an overview of a new estimation procedure is presented and details follow in Section 2.3. Section 2.4 discusses other issues that arise when the proposed estimation procedure is performed, such as a method to decide when to stop the procedure and a method to obtain suitable initial parameters which save computing time. Furthermore, it introduces a relative likelihood and how to calculate standard deviations of parameter estimates. In the last section, other approaches from various articles which analyze the SV model are briefly described and compared to the proposed method.

2.1 LITERATURE REVIEW

The new method in this thesis is basically based on the EM algorithm. When the EM algorithm is applied to the SV model, one can see that it is not possible to get the exact expected likelihood because of the complex dependent structure of the SV model. In order to solve this problem, I bring out particle methods and calculate an approximate expected likelihood by using the output of the particle methods.

2.1.1 Brief Review of EM Algorithm

Briefly speaking, the *Expectation-Maximization (EM) algorithm* is one of the parameter estimation tools to achieve maximum likelihood estimator (MLE) and it has been widely applied to the cases where the data is considered to be incomplete in the sense that it is not fully observable. Dempster et al. (1977) defined the EM algorithm and proved its properties. The following is a summary of their work.

Let $\mathcal{X}$ and $\mathcal{Y}$ be the two sample spaces where there exists a many-one mapping, $y^* : x \rightarrow y^*(x) = y \in \mathcal{Y}$ from $\mathcal{X}$ to $\mathcal{Y}$. The observed data y is a realization in $\mathcal{Y}$. The corresponding x in $\mathcal{X}$ is not observed and they call x the *complete data*. Suppose that there is a family of sampling densities $f(x|\theta)$ depending on parameters θ, then the density of observed data y is

$$g(y|\theta) = \int_{\mathcal{X}(y)} f(x|\theta)dx, \tag{2.1}$$

where $\mathcal{X}(y)$ is the pre-image of y in $\mathcal{X}$.

The problem to solve in this framework is to get MLE of θ by maximizing $l(\theta) = \log g(y|\theta)$. In many cases, $l(\theta)$ has no analytic solution. On the contrary, $f(x|\theta)$ or $\log f(x|\theta)$ is easy to handle. The EM algorithm is an iterative algorithm to find MLE by using $f(x|\theta)$ instead of $g(y|\theta)$. *Expectation step (E-step)* and *Maximization step (M-step)* form one iteration of the EM algorithm. At k^{th} iteration, the updated parameter $\theta^{(k)}$ is obtained from $\theta^{(k-1)}$ as follows:

E-step: Compute the *expected likelihood*, $Q(\theta|\theta^{(k-1)})$, where

$$Q(\theta|\theta') = E(\log f(x|\theta')|y, \theta). \tag{2.2}$$

M-step: Choose $\theta^{(k)}$ which maximizes $Q(\theta|\theta^{(k-1)})$.

They showed that the log-likelihood $l(\theta) = \log g(y|\theta)$ is non-decreasing at each iteration so that $\theta^{(k)}$ is expected to go close to local maximum as k increases. For more details, see Dempster et al. (1977) and Mclachlan and Krishnan (1997).

2.1.2 Brief Review of Particle Filters and Smoothers

The main idea of particle methods is to represent the probability distribution of z, which is usually hard to obtain directly, using M particles and associated weights, $\{z^{(j)}, w^{(j)}\}_{j=1}^{M}$, so that the empirical distribution of particles, $\frac{1}{\sum_{j=1}^{M} w^{(j)}} \sum_{j=1}^{M} w^{(j)} I(Z \leq z^{(j)})$, replaces the cdf of Z, $F(z) = Pr(Z \leq z)$ [1].

Particle filters and smoothers are sequential Monte Carlo methods grounded in particle representations, and they can be considered as generalizations of well-known Kalman filters and smoothers for general state-space models. Before I introduce particle filters and smoothers, I want to introduce related concepts first. Throughout this thesis, I use the following symbols:

- $\mathrm{f}_t^{(j)}$: jth particle filter at time t which approximates $f(x_t|Y_t)$[2].
- $\mathrm{s}_t^{(j)}$: jth particle smoother at time t which approximates $f(x_t|Y_n)$, $(n > t)$.
- $\mathrm{p}_t^{(j)}$: jth particle predictor at time t which approximates $f(x_t|Y_s)$, $(s < t)$.
- $\mathrm{w}_t^{(j)}$: a weight associated with $\mathrm{f}_t^{(j)}$ or $\mathrm{s}_t^{(j)}$.

2.1.2.1 State-Space Models and Related Concepts

General *state-space models* are represented by the following two equations: [3]

$$x_t = F_t(x_{t-1}, w_t), \tag{2.3}$$

$$y_t = H_t(x_t, v_t), \tag{2.4}$$

where F_t and H_t are known functions that may depend on parameters θ, and w_t and v_t are white noise processes. In this setting, x_t represents an unknown state at time t which has its own mechanism, and y_t represents the data observed indirectly from x_t. Hence, it is natural to name (2.3) the *state equation* and (2.4) the *observation equation*. Also, w_t is called the *state noise* or *state error*, and v_t is called the *observation noise* or *observation error*. When the state equation and the observation equation are as (2.5) and (2.6), the

[1] $I(x)$ is an indicator function whose value is 1 if x is true, 0 otherwise.
[2] $Y_k := \{y_1, \cdots, y_k\}$
[3] I follow the notation in Shumway and Stoffer (2000)

model is referred as a *linear state-space model*. In addition, if both w_t and v_t are normal distributions in (2.5) and (2.6), the model is called a *linear Gaussian state-space model*.

$$x_t = \Phi x_{t-1} + w_t, \tag{2.5}$$

$$y_t = A_t x_t + v_t. \tag{2.6}$$

In Section 1.2, I introduced SV models. As you can easily perceive, the SV model is a state-space model with (1.2) and (1.1) as an observation equation and a state equation, respectively. The modification of (1.2) to (1.3) makes the SV model a linear state-space model. However, the SV model is not a linear *Gaussian* state-space model.

In the state-space models, interests are usually in the estimation of unknown states x_t when the observed data $Y_s = \{y_1, \cdots, y_s\}$ are at hand. When $t > s$, x_t is estimated based on past observations, so this problem is called $forecasting$ or *prediction*. When $t < s$, the problem is called *smoothing*, and when $t = s$, the problem is called $filtering$. The *Kalman filter and smoother* are the methods to get the predictors, filters and smoothers for the linear Gaussian state-space models. The Kalman filter and smoother estimate x_t as its conditional expectation given the data Y_s, $x_t^s = E(x_t|Y_s)$, which is a best linear predictor, and exact x_t^s can be obtained sequentially when the parameters in the models are given.

2.1.2.2 Particle Filters and Smoothers Now, the problem is to get good filters and smoothers for the non-linear or non-Gaussian state-space models. Kalman filters and smoothers are based on assumptions of linearity and Gaussianity, so Kalman filters and smoothers do not work well when there are departures from these assumptions. Particle filters or Sequential Monte Carlo filters can be the solutions for non-linear or non-Gaussian cases. A good deal of articles on this topic have been published recently. Arulampalam et al. (2002) and Doucet et al. (2001) are good sources which cover particle filters and their several variants.

The main idea of particle filters and smoothers is this: Instead of giving a single estimate for the filter or the smoother, x_t^s, as in the Kalman filter and smoother, particle methods provide particles with associated weights to approximate the conditional density $f(x_t|Y_s)$. This set of particles makes it possible to approximate anything related to their true density.

For example, weighted particles $\{x_t^{(j)}, \mathrm{w}_t^{(j)}\}_{j=1}^M$ from $f(x_t|Y_s)$ enables us to approximate x_t^s as the weighted average of particles, $\frac{1}{\sum_{j=1}^M \mathrm{w}_t^{(j)}} \sum_{j=1}^M \mathrm{w}_t^{(j)} x_t^{(j)}$.

The basic strategy used to get particles from the desired density is based on sequential importance sampling and resampling. *Sequential importance sampling (SIS)* is a Monte Carlo method that forms the basis for most particle filtering methods. To approximate the conditional density of x_t given the previous states, X_{t-1}, and the past and present data, Y_t, $p(x_t|X_{t-1}, Y_t)$, SIS introduces a sequential importance density, $q(x_t|X_{t-1}, Y_t)$ where it is easier to sample from $q(x_t|X_{t-1}, Y_t)$ than $p(x_t|X_{t-1}, Y_t)$[4]. The joint conditional density of X_t given Y_t is

$$\begin{aligned} p(X_t|Y_t) &= \frac{p(y_t|X_t, Y_{t-1})p(X_t|Y_{t-1})}{p(y_t|Y_{t-1})} \\ &= \frac{p(y_t|X_t, Y_{t-1})p(x_t|X_{t-1}, Y_{t-1})}{p(y_t|Y_{t-1})} p(X_{t-1}|Y_{t-1}) \\ &= \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|Y_{t-1})} p(X_{t-1}|Y_{t-1}) \\ &\propto p(y_t|x_t)p(x_t|x_{t-1})p(X_{t-1}|Y_{t-1}) \\ &\propto \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q(x_t|X_{t-1}, Y_t)} q(x_t|X_{t-1}, Y_t)p(X_{t-1}|Y_{t-1}). \end{aligned} \tag{2.7}$$

Suppose that particle approximation of $p(X_{t-1}|Y_{t-1})$ is given as $\sum_{j=1}^M \mathrm{w}_{t-1}^{(j)} \delta(X_{t-1} - X_{t-1}^{(j)})$[5] and $x_t^{(j)}$ is a sample from $q(x_t|X_{t-1}^{(j)}, Y_t)$, for $j = 1, \cdots, M$. Then the particle approximation of $p(X_t|Y_t)$ is

$$\begin{aligned} p(X_t|Y_t) &\approx \sum_{j=1}^M \frac{p(y_t|x_t^{(j)})p(x_t^{(j)}|x_{t-1}^{(j)})}{q(x_t^{(j)}|X_{t-1}^{(j)}, Y_t)} \delta(x_t - x_t^{(j)}) \mathrm{w}_{t-1}^{(j)} \delta(X_{t-1} - X_{t-1}^{(j)}) \\ &\approx \sum_{j=1}^M \mathrm{w}_t^{(j)} \delta(X_t - X_t^{(j)}), \end{aligned} \tag{2.8}$$

where $\mathrm{w}_t^{(j)} = \frac{p(y_t|x_t^{(j)})p(x_t^{(j)}|x_{t-1}^{(j)})}{q(x_t^{(j)}|X_{t-1}^{(j)}, Y_t)} \mathrm{w}_{t-1}^{(j)}$ and $X_t^{(j)} = \{X_{t-1}^{(j)}, x_t^{(j)}\}$. Furthermore, if the importance density, $q(x_t|X_{t-1}, Y_t)$, does not depend on X_{t-2} ($q(x_t|X_{t-1}, Y_t) = q(x_t|x_{t-1}, Y_t)$), then the SIS particle filter can be summarized as follows:

[4] $X_t = \{x_0, \cdots, x_t\}$ and $Y_t = \{y_1, \cdots, y_t\}$

[5] $\delta(x)$ is an indicator function whose value is 1 if $x = 0$, 0 otherwise

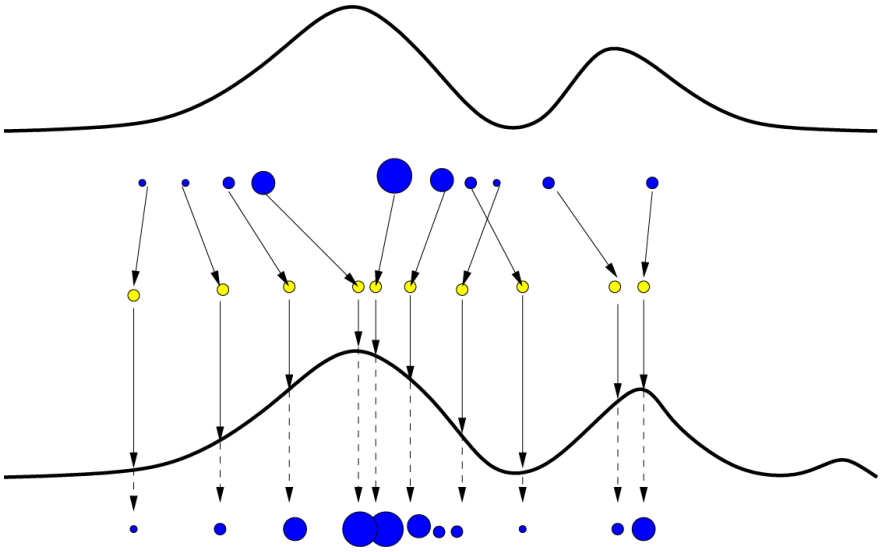

Figure 2.1: SIS particle filter.

SIS particle filter

Draw a set of random samples from $p(x_0)$ and let $\{x_0^{(j)}\}$ and let $w_0^{(j)} = \frac{1}{M}$, where M is the sample size.

For $t = 1, \cdots, n$, where n is the length of time series $\{x_t\}$,

1. For $j = 1, \cdots, M$
   a. Draw $x_k^{(j)} \sim q(x_t | x_{t-1}^{(j)}, Y_t)$.
   b. Calculate an updated weight, $\mathrm{w}_t^{(j)}$.

$$\mathrm{w}_t^{(j)} = \mathrm{w}_{t-1}^{(j)} \frac{p(y_t|x_t^{(j)}) p(x_t^{(j)}|x_{t-1}^{(j)})}{q(x_t^{(j)}|x_{t-1}^{(j)}, Y_t)}$$

2. $\{X_{t-1}^{(j)}, x_t^{(j)}\}_{j=1}^M$ with the weights $\{\mathrm{w}_t^{(j)}\}_{j=1}^M$ approximates $p(X_t|Y_t)$.

Using this SIS particle filter, the particles and the associated weights,$\{X_t^{(j)}, \mathrm{w}_t^{j)}\}_{j=1}^M$ can be obtained in a sequential manner (See Figure 2.1)[6]. However, the problem with this

[6]Figures 2.1 - 2.3 are taken from Maskell and Gordon (2002)

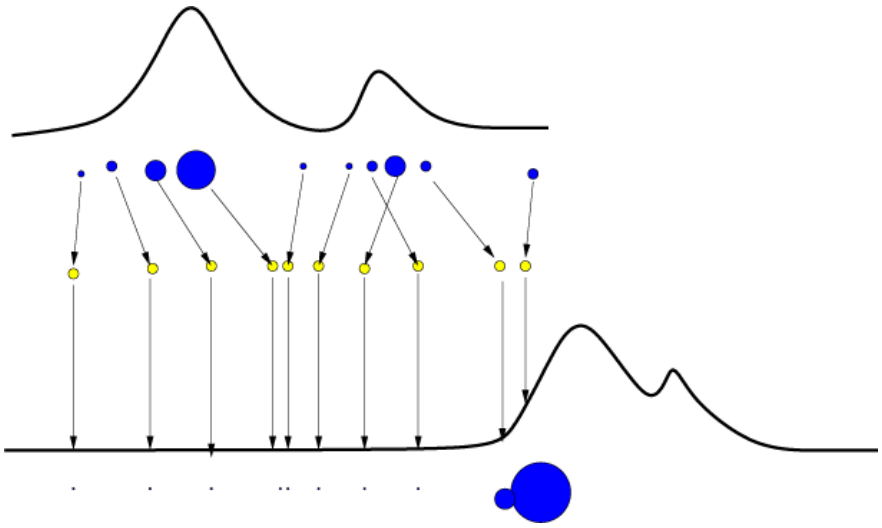

Figure 2.2: Problem of the SIS particle filter: degeneracy.

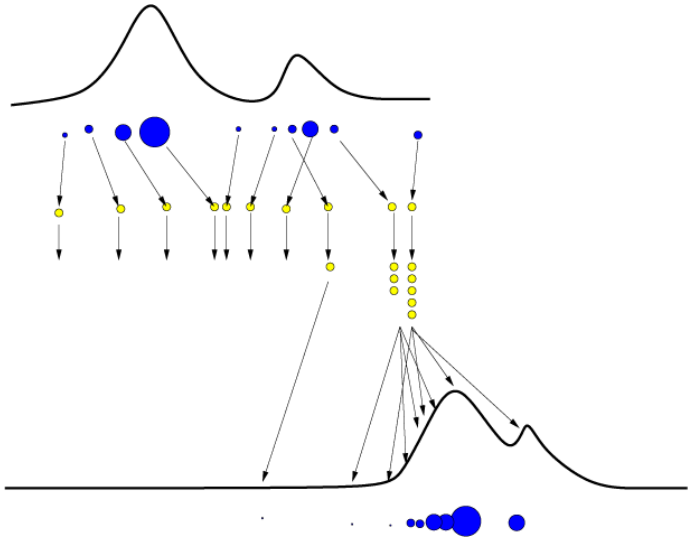

Figure 2.3: Resampling.

sequential importance sampling is degeneracy: after a few iterations, most of the particles have very small weights and they give little contribution to the desired density $p(x_t|Y_t)$. In Figure 2.2, two particles dominate the distribution and the others are useless since their weights are almost zero. Resampling is designed to solve this problem by removing particles with small weights, concentrating, instead, on particles with large weights. This resampling step involves generating a new set, $\{x_t^{(j*)}\}_{j=1}^M$, by resampling with replacement M times from $\{x_t^{(j)}\}_{j=1}^M$ so that $Pr(x_t^{(i*)} = x_t^{(j)}) = \mathrm{w}_t^{(j)}$. Figure 2.3 shows how resampling works. A generic particle filter draws $\{x_t^{(j)}\}_{j=1}^M$ using a SIS particle filter, and resamples $\{x_t^{(j*)}\}_{j=1}^M$ when degeneracy has occurred. For more details, see Arulampalam et al. (2002) or Doucet et al. (2001).

2.1.2.3 Particle Filtering Algorithm What I want to get through this filtering step are $\{\mathrm{f}_t^{(j)}, \mathrm{w}_t^{(j)}\}_{j=1}^M$, for $t = 1, \cdots, n$, where $\mathrm{f}_t^{(j)}$ represents jth realization of particle filter and $\mathrm{w}_t^{(j)}$ represents an associated weight, i.e. $f(x_t|Y_t) \approx \sum_{j=1}^M \mathrm{w}_t^{(j)} \delta(x_t - \mathrm{f}_t^{(j)})$. I get these particles in a sequential manner. Suppose that equally weighted particles $\{\mathrm{f}_{t-1}^{(j)}\}_{j=1}^M$ and associated weights $\{\mathrm{w}_{t-1}^{(j)}\}_{j=1}^M$ are given. First, I get the particles from $f(x_t|Y_{t-1})$, $\{\mathrm{p}_t^{(j)}\}_{j=1}^M$, and using them I get the desired particles, $\{\mathrm{f}_t^{(j)}\}_{j=1}^M$, where $\mathrm{p}_t^{(j)}$ denotes a particle drawn from $f(x_t|Y_{t-1})$.

Kitagawa and Sato (2001) (or Kitagawa (1996)) suggest an algorithm for filtering, where (2.3) and (2.4) are the model structures and $x_0 \sim p_0(x)$, $w_t \sim q(w)$. Here is a brief summary of their algorithm: since the predictive distribution, $f(x_t|Y_{t-1})$, can be expressed by

$$\begin{aligned} f(x_t|Y_{t-1}) &= \int\int f(x_t, x_{t-1}, w_t|Y_{t-1}) dw_t dx_{t-1} \\ &= \int\int f(x_t, |x_{t-1}, w_t) f(w_t) f(x_{t-1}|Y_{t-1}) dw_t dx_{t-1} \\ &= \int\int \delta(x_t - F_t(x_{t-1}, w_t)) f(w_t) f(x_{t-1}|Y_{t-1}) dw_t dx_{t-1}, \end{aligned} \tag{2.9}$$

where $\delta(x)$ denotes the delta function, $\mathrm{p}_t^{(j)} = F_t(\mathrm{f}_{t-1}^{(j)}, w_t^{(j)})$ is a random sample from $f(x_t|Y_{t-1})$ if $w_t^{(j)}$ is a realization of w_t. Therefore, if we repeat above procedure M times, we get $\{\mathrm{p}_t^{(1)}, \cdots \mathrm{p}_t^{(M)}\}$ which is an independent random sample from $f(x_t|Y_{t-1})$.

Now, given the observation y_t and the particle $\{\mathrm{p}_t^{(j)}\}_{j=1}^M$, we can get $\mathrm{f}_t^{(j)}$ as follows:

$$
\begin{aligned}
Pr(x_t = \mathrm{p}_t^{(j)} | Y_t) &= Pr(x_t = \mathrm{p}_t^{(j)} | Y_{t-1}, y_t) \\
&= \frac{p(y_t|\mathrm{p}_t^{(j)}) Pr(x_t = \mathrm{p}_t^{(j)}|Y_{t-1})}{\sum_{k=1}^{M} p(y_t|\mathrm{p}_t^{(k)}) Pr(x_t = \mathrm{p}_t^{(k)}|Y_{t-1})} \\
&= \frac{\mathrm{w}_t^{(j)}}{\sum_{k=1}^{M} \mathrm{w}_t^{(k)}} \qquad (2.10)
\end{aligned}
$$

Since $Pr(x_t = \mathrm{p}_t^{(j)}|Y_{t-1}) = \frac{1}{M}$, if we sample from $\{\mathrm{p}_t^{(j)}\}_{j=1}^{M}$ with weight $\mathrm{w}_t^{(j)} = p(y_t|\mathrm{p}_t^{(j)})$, we get $\{\mathrm{f}_t^{(j)}\}$.

The following is the summary of the algorithm:

Monte Carlo Filter

1. Generate a random number $\mathrm{f}_0^{(j)} \sim p_0(x)$ for $j = 1, \cdots, M$, where M is the number of particles.
2. Repeat the following steps for $t = 1, \cdots, n$, where n is the length of data.
   a. Generate a random number $w_t^{(j)} \sim q(w)$, for $j = 1, \cdots, M$.
   b. Compute $\mathrm{p}_t^{(j)} = F_t(\mathrm{f}_{t-1}^{(j)}, w_t^{(j)})$, for $j = 1, \cdots, M$.
   c. Compute $\mathrm{w}_t^{(j)} = p(y_t|\mathrm{p}_t^{(j)})$, for $j = 1, \cdots, M$.
   d. Generate $\mathrm{f}_t^{(j)}$, for $j = 1, \cdots, M$ by resampling of $\mathrm{p}_t^{(1)}, \cdots, \mathrm{p}_t^{(M)}$.
3. This Monte Carlo filter returns $\{\mathrm{f}_t^{(j)}, j = 1, \cdots, M, t = 1, \cdots, n\}$, so that $\sum_{j=1}^{M} \frac{1}{M} \delta(x_t - \mathrm{f}_t^{(j)}) \approx f(x_t|Y_t)$.

This algorithm is the same as sequential importance resampling (SIR) filter, which can be easily derived from the SIS algorithm. The SIR filter chooses the conditional density of x_t given x_{t-1}, $f(x_t|x_{t-1})$, as an sequential importance density, $q(x_t|x_{t-1}, Y_t)$, and apply the resampling step at every time index.

2.1.2.4 Particle Smoothing Algorithm In the smoothing step, a primary goal is to get particle smoothers, $\{\mathrm{s}_t^{(j)}\}_{j=1}^M$, with associated weights, $\{\mathrm{w}_t^{(j)}\}_{j=1}^M$, where $f(x_t|Y_n) \approx \sum_{j=1}^M \mathrm{w}_t^{(j)}\delta(x_t - \mathrm{s}_t^{(j)})$. Theoretically, the trajectories of states $\{\mathrm{s}_1, \cdots, \mathrm{s}_n\}$ from the $f(x_1, \cdots, x_n|Y_n)$ can be obtained by a usual particle filtering method in the previous section. However, in practice, since the diversity of the paths of the particles is reduced by the resampling step, smoothed estimates degenerate. In other words, only small number of particles appear repeatedly in earlier time points if n gets bigger. Kitagawa and Sato (2001) suggested resampling only part of the data instead sampling whole path from 0 to t at time point t in order to prevent degeneracy.

Godsill et al. (2004) suggested a new smoothing method: *particle smoother using backwards simulation.* Unlike other smoothing methods, this method is free from degeneracy and concerns the whole trajectory of states $\{x_0, \cdots x_n\}$ from the joint density $f(x_0, \cdots, x_n|Y_n)$ not just the individual marginal smoothing density, $f(x_t|Y_n)$. This makes this method the most useful for my study as I need the random sample from the joint density of states to calculate $P_{t,t-1}^n = E\{(x_t - x_t^n)(x_{t-1} - x_{t-1}^n)|Y_n\}$. The particle smoother using backward simulation assumes that the filtering has already been performed so that the particles and associated weights, $\{\mathrm{f}_t^{(j)}\}_{j=1}^M$, $\{\mathrm{w}_t^{(j)}\}_{j=1}^M$, can approximate the filtering density, $f(x_t|Y_t)$, by $\frac{\sum \mathrm{w}_t^{(j)}\delta(x_t - \mathrm{f}_t^{(j)})}{\sum \mathrm{w}_t^{(j)}}$. This method can be justified by the following:

$$p(x_1, \cdots, x_n|Y_n) = p(x_n|Y_n)\prod_{t=1}^{n-1} p(x_t|x_{t+1}, \cdots, x_n, Y_n) \tag{2.11}$$

and

$$\begin{aligned} p(x_t|x_{t+1}, \cdots, x_n, Y_n) &= p(x_t|x_{t+1}, Y_t) \\ &= \frac{p(x_t|Y_t)f(x_{t+1}|x_t)}{p(x_{t+1}|x_t)} \\ &\propto p(x_t|Y_t)f(x_{t+1}|x_t) \end{aligned} \tag{2.12}$$

From the assumption of this method, particles from $p(x_t|Y_t)$ are available, so $p(x_t|\, x_{t+1}, \cdots, x_n, Y_n)$ can be approximated as follows:

$$p(x_t|x_{t+1}, \cdots, x_n, Y_n) \approx \sum_{j=1}^M \mathrm{w}_{t|t+1}^{(j)}\delta(x_t - \mathrm{f}_t^{(j)}), \tag{2.13}$$

where

$$\mathrm{w}_{t|t+1}^{(j)} = \frac{\mathrm{w}_t^{(j)} f(x_{t+1}|\mathrm{f}_t^{(j)})}{\sum_{k=1}^{M} \mathrm{w}_t^{(k)} f(x_{t+1}|\mathrm{f}_t^{(k)})}. \tag{2.14}$$

The idea is this: given a random sample $\{\mathrm{s}_{t+1}, \cdots \mathrm{s}_n\}$ approximately from $p(x_{t+1}, \cdots, x_n|Y_n)$, it is possible to get a sample s_t from $p(x_t|\mathrm{s}_{t+1}, \cdots, \mathrm{s}_n, Y_n)$ and the pair $\{\mathrm{s}_t, \mathrm{s}_{t+1}, \cdots, \mathrm{s}_n\}$ is approximately a random sample from $p(x_t, \cdots, x_n|Y_n)$. By repeating this process sequentially, $\{\mathrm{s}_t\}_{t=1}^n$ can be obtained at last. The following is the algorithm from Godsill et al. (2004). By repeating this algorithm M times, it is possible to get M trajectories of the states given the data.

Particle smoother using backwards simulation

Suppose weighted particles $\{(\mathrm{f}_t^{(j)}, \mathrm{w}_t^{(j)});\ j = 1, 2, \cdots, M\}$ are available for $t = 1, \cdots, n$.

For $j = 1, \cdots, M$,

1. Choose $\mathrm{s}_n^{(j)} = \mathrm{f}_n^{(i)}$ with probability $\mathrm{w}_n^{(i)}$.
2. For $n-1$ to 1,
   a. Calculate $\mathrm{w}_{t|t+1}^{(i)} \propto \mathrm{w}_t^{(i)} f(\mathrm{s}_{t+1}^{(j)}|\mathrm{f}_t^{(i)})$ for each i.
   b. Choose $\mathrm{s}_t^{(j)} = \mathrm{f}_t^{(i)}$ with probability $\mathrm{w}_{t|t+1}^{(i)}$.
3. $\mathrm{s}_{1:n}^{(j)} = (\mathrm{s}_1^{(j)}, \cdots, \mathrm{s}_n^{(j)})$ is an approximate realization from $p(X_n|Y_n)$. [7]

2.2 OVERVIEW OF THE ESTIMATION ALGORITHM

The main advantage of the EM algorithm is its ability to handle missing data. In the SV model, if $\{x_0, \cdots, x_n, y_1, \cdots, y_n\}$ is considered as a complete data set, only $\{y_1, \cdots, y_n\}$ is observed. In other words, $\{x_0, \cdots, x_n\}$ can be simply considered as missing data, and then the usual EM algorithm can be applied to the SV model. If the expected likelihood of the complete data given $Y_n = \{y_1, \cdots, y_n\}$ is available, parameter estimates can be obtained by

[7] $z_{a:b} = \{z_a, z_{a+1}, \cdots, z_{b-1}, z_b\}$

maximizing it. Shumway and Stoffer (2000) applied this idea to the parameter estimation of linear Gaussian state-space models and expand it to their missing data cases. They also suggested using this method for the SV model (see Chapter 4 of Shumway and Stoffer (2000)). However, since the observation noise of the SV model is not Gaussian, the well-known Kalman filter and smoother can not be adopted, and so computation of expected likelihood is difficult. In this paper, a simulation-based particle method is used to get the quantities needed in the calculation of the expected likelihood. The following is the algorithm of the whole estimation procedure: the details are given in the next section.

Parameter estimation algorithm for Model A: *Algorithm A*

Let the initial parameters be $\theta^{(0)} = \{\alpha^{(0)}, \phi^{(0)}, Q^{(0)}\}$.

For $i = 1, \cdots, i_{max}$, where i_{max} is the maximum number of iteration,

1. Filtering step: get particle filters $\mathrm{f}_t^{(j)}$ from $f(x_t|Y_t, \theta^{(i-1)})$, $j = 1, \cdots, M$, $t = 1, \cdots, n$, where M is the number of particles.
2. Smoothing step: get particle smoothers $\{\mathrm{s}_1^{(j)}, \cdots, \mathrm{s}_n^{(j)}\}$ from $f(x_1, \cdots, x_n|Y_n, \theta^{(i-1)})$, $j = 1, \cdots, M$ and save x_t^n, P_t^n, $P_{t,t-1}^n$ [8] for the estimation step (note that particle filters are needed to get smoothers).
3. Estimation step: get estimated parameters $\theta^{(i)} = \{\alpha^{(i)}, \phi^{(i)}, Q^{(i)}\}$ by maximizing the expected likelihood.

Repeat 1-3 until the process converges.

I call this algorithm *Algorithm A* throughout this thesis. At the beginning of each iteration, parameters are assumed to be known. For given parameters, I run a filtering step and a smoothing step. With the output of these two steps, I get updated parameter estimates by running an estimation step.

[8] $x_t^s = E(x_t|Y_s)$, $P_{t1,t2}^s = E\{(x_{t1} - x_{t1}^s)(x_{t2} - x_{t2}^s)|Y_s\}$, $P_t^s = P_{t,t}^s$

2.3 DETAILS OF THE PROCEDURE

Simply speaking, this proposed method is nothing but applying the EM algorithm. To get an expected likelihood, it is necessary to calculate quantities such as $x_t^n = E(x_t|Y_n)$. These quantities are so-called smoothers. To get smoothers, I apply particle smoothing method and to apply particle smoothing method, particle filters should be obtained in advance.

2.3.1 Filtering Step

In this step, I get particle filters which are a random sample from $f(x_t|Y_t)$. Since the SV model falls into the category of general state-space models, the algorithm explained in Section 2.1.2.3 can be applied directly. The following is the algorithm for the filtering step: through this filtering step, M samples from $f(x_t|Y_t)$ for each time point t, are in stock.

Particle filtering algorithm for Algorithm A

1. Generate $\mathrm{f}_0^{(j)} \sim N(\mu_0, \sigma_0^2),\ j = 1, \cdots, M$.
2. For $t = 1, \cdots, n$,
   a. Generate a random number $w_t^{(j)} \sim N(0, Q),\ j = 1, \cdots, M$.
   b. Compute $\mathrm{p}_t^{(j)} = \phi \mathrm{f}_{t-1}^{(j)} + w_t^{(j)}$ ($\mathrm{p}_t^{(j)}$ is random sample from $f(x_t|Y_{t-1})$).
   c. Compute $\mathrm{w}_t^{(j)}$=$p(y_t|\mathrm{p}_t^{(j)}) \propto \exp\left(-\frac{1}{2}\exp(y_t - \mathrm{p}_t^{(j)} - \gamma^* - \alpha)\right)\exp\left(\frac{1}{2}(y_t - \mathrm{p}_t^{(j)} - \gamma^* - \alpha)\right)$[9].
   d. Generate $\mathrm{f}_t^{(j)}$ by resampling with weight $\mathrm{w}_t^{(j)}$ ($\mathrm{f}_t^{(j)}$ is random sample from $f(x_t|Y_t)$).

2.3.2 Smoothing Step

In this step, I get $\hat{x}_t^n, \hat{P}_t^n, \hat{P}_{t,t-1}^n$and $E(\exp(y_t \widehat{-x_t} - \gamma^*)|Y_n)$ which are needed in the expectation step of the EM algorithm. To get these quantities, I use the particle smoothers which are random samples from $f(x_t|Y_n)$.

[9] $\gamma^* = E(\log(\chi_1^2)) \approx 1.2749$

The following is the algorithm to get particle smoothers, which is a slight modification of Godsill et al. (2004) in Section 2.1.2.4. Equally weighted particles $\{\mathrm{f}_t^{(j)}\}, j = 1, \cdots, M$, are available from the filtering step.

Particle smoothing algorithm for Algorithm A

1. Choose $\mathrm{s}_n^{(j)} = \mathrm{f}_n^{(i)}$ with probability $\frac{1}{M}$.
2. For $n-1$ to 0,
   a. Calculate $\mathrm{w}_{t|t+1}^{(i)} \propto f(\mathrm{s}_{t+1}^{(j)}|\mathrm{f}_t^{(i)}) \propto \exp\left(-\frac{(\mathrm{s}_{t+1}^{(j)} - \phi \mathrm{f}_t^{(j)})^2}{2Q}\right)$ for each i.
   b. Choose $\mathrm{s}_t^{(j)} = \mathrm{f}_t^{(i)}$ with probability $\mathrm{w}_{t|t+1}^{(i)}$.
3. $\mathrm{s}_{1:n}^{(j)} = \{\mathrm{s}_1^{(j)}, \cdots, \mathrm{s}_n^{(j)}\}$ is the random sample from $f(x_1, \cdots, x_n|Y_n)$.
4. Repeat 1 – 3, for $j = 1, \cdots, M$, and calculate the followings:

$$\hat{x}_t^n = \frac{\sum_{j=1}^M \mathrm{s}_t^{(j)}}{M}, \quad \hat{P}_t^n = \frac{\sum_{j=1}^M (\mathrm{s}_t^{(j)} - \hat{x}_t^n)^2}{M-1}, \quad \hat{P}_{t,t-1}^n = \frac{\sum_{j=1}^M (\mathrm{s}_t^{(j)} - \hat{x}_t^n)(\mathrm{s}_{t-1}^{(j)} - \hat{x}_{t-1}^n)}{M},$$

$$E(\exp(y_t \widehat{-x_t} - \gamma^*)|Y_n) = \frac{\sum_{j=1}^M \exp(y_t - \mathrm{s}_t^{(j)} - \gamma^*)}{M}.$$

At the end of this smoothing step, $\hat{x}_t^n, \hat{P}_t^n, \hat{P}_{t,t-1}^n, E(\exp(y_t \widehat{-x_t} - \gamma^*)|Y_n)$ are saved and ready to use.

2.3.3 Estimation Step

To apply the EM algorithm, it is needed to calculate the *complete likelihood* and the *expected likelihood* given the data. The followings are the details.

The complete likelihood of $\{x_0, x_1, \cdots, x_n, y_1, \cdots, y_n\}$ [10] is

$$\begin{aligned} f(X,Y) &= f(x_0)\prod_{t=1}^{n} f(x_t|x_{t-1})\prod_{t=1}^{n} f(y_t|x_t) \\ &= \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(-\frac{(x_0-\mu_0)^2}{2\sigma_0^2}\right)\prod_{t=1}^{n}\frac{1}{\sqrt{2\pi Q}}\exp\left(-\frac{(x_t-\phi x_{t-1})^2}{2Q}\right) \\ &\quad \cdot\prod_{t=1}^{n} C_1\cdot\exp\left(-\frac{1}{2}\exp(y_t-x_t-\alpha-\gamma^*)\right)\exp\left(\frac{1}{2}(y_t-x_t-\alpha-\gamma^*)\right), \end{aligned} \tag{2.15}$$

where C_1 is constant, and

$$\begin{aligned} -2\log f(X,Y) &= \log\sigma_0^2 + \frac{(x_0-\mu_0)^2}{\sigma_0^2} + n\log Q + \sum_{t=1}^{n}\frac{(x_t-\phi x_{t-1})^2}{Q} \\ &\quad + \sum_{t=1}^{n}\{\exp(y_t-x_t-\alpha-\gamma^*) - (y_t-x_t-\alpha-\gamma^*)\} + C_2, \end{aligned} \tag{2.16}$$

where C_2 is constant. Hence, the expected likelihood of $\{x_0, x_1, \cdots x_n, y_1, \cdots y_n\}$ given $\{y_1, \cdots, y_n\}$ is

$$\begin{aligned} Q(\theta) &= E(-2\log f(X,Y)|Y) \\ &= \log\sigma_0^2 + \frac{(x_0^n-\mu_0)^2 + P_0^n}{\sigma_0^2} \\ &\quad + n\log Q + \sum_{t=1}^{n}\frac{(x_t^n-\phi x_{t-1}^n)^2 + P_t^n + \phi^2 P_{t-1}^n - 2\phi P_{t,t-1}^n}{Q} \\ &\quad + \sum_{t=1}^{n}\left\{\frac{1}{\exp(\alpha)}E(\exp(y_t-x_t^n-\gamma^*)|Y_n) - (y_t-x_t^n-\alpha-\gamma^*)\right\} + C, \end{aligned} \tag{2.17}$$

where $x_t^n = E(x_t|Y_n)$, $P_t^n = E[(x_t-x_t^n)^2|Y_n]$ and $P_{t,t-1}^n = E[(x_t-x_t^n)(x_{t-1}-x_{t-1}^n)|Y_n]$.

I get the EM estimate by minimizing (2.17). The following is the parameter estimates:

[10] $X = \{x_0, \cdots, x_n\}, Y = \{y_1, \cdots, y_n\}$

1. $[\phi, Q]$

$$\begin{aligned}
\frac{\partial Q(\theta)}{\partial \phi} &= \sum_{t=1}^{n} \frac{-2x_{t-1}^n(x_t^n - \phi x_{t-1}^n) + 2\phi P_{t-1}^n - 2P_{t,t-1}^n}{Q} \\
&= \sum_{t=1}^{n} \frac{2}{Q}[\phi((x_{t-1}^n)^2 + P_{t-1}^n) - (x_{t-1}^n x_t^n + P_{t,t-1}^n)] \\
&= \frac{2}{Q}[\phi S_{00} - S_{10}] \\
\frac{\partial Q(\theta)}{\partial Q} &= \frac{n}{Q} - \sum_{t=1}^{n} \frac{(x_t^n - \phi x_{t-1}^n)^2 + P_t^n + \phi^2 P_{t-1}^n - 2\phi P_{t,t-1}^n}{Q^2} \\
&= \frac{1}{Q^2}\left[nQ - \left\{\phi^2 \sum_{t=1}^{n}((x_{t-1}^n)^2 + P_{t-1}^n) - 2\phi \sum_{t=1}^{n}(x_{t-1}^n x_t^n + P_{t,t-1}^n)\right.\right. \\
&\quad \left.\left. + \sum_{t=1}^{n}((x_t^n)^2 + P_t^n)\right\}\right] \\
&= \frac{1}{Q^2}\left[nQ - \left\{\phi^2 S_{00} - 2\phi S_{10} + S_{11}\right\}\right]
\end{aligned}$$

$$\hat{\phi} = \frac{S_{10}}{S_{00}}, \tag{2.18}$$

$$\hat{Q} = \frac{1}{n}\left(S_{11} - \frac{S_{10}^2}{S_{00}}\right), \tag{2.19}$$

where $S_{00} = \sum_{t=1}^{n}((x_{t-1}^n)^2 + P_{t-1}^n)$, $S_{11} = \sum_{t=1}^{n}((x_t^n)^2 + P_t^n)$ and $S_{10} = \sum_{t=1}^{n}(x_{t-1}^n x_t^n + P_{t,t-1}^n)$.

2. $[\alpha]$

$$\frac{\partial Q(\theta)}{\partial \alpha} = \sum_{t=1}^{n}\left\{-\frac{1}{\exp(\alpha)}E(\exp(y_t - x_t - \gamma^*)|Y_n) + 1\right\}$$

$$\hat{\alpha} = \log\left(\frac{\sum_{t=1}^{n} E(\exp(y_t - x_t - \gamma^*)|Y_n)}{n}\right). \tag{2.20}$$

Since the smoothing step returns the quantities needed in these formulas, I can get EM estimates by plugging them in.

2.4 OTHER ISSUES

2.4.1 Initial Parameter Selection

A common criticism of the EM algorithm is that its convergence can be quite slow (see Mclachlan and Krishnan (1997)). This means that it may take long time to arrive to a maximum point. In order to save computing time, it is essential to start with good initial parameters. Anderson et al. (1969) suggested consistent estimates of the parameters of a linear system based on the idea of the method of moments. I apply their idea to get initial parameters.

1. $[\alpha]$

$$\begin{aligned} E(y_t) &= \alpha + E(x_t) + E(v_t) = \alpha \\ \rightarrow \quad & \alpha^{(0)} = \frac{\sum_{t=0}^{n} y_t}{n}. \end{aligned} \tag{2.21}$$

Here, the process x_t is assumed to be stationary from the beginning and so $E(x_t) = 0$.

2. $[\phi]$

$$\begin{aligned} E\left\{(y_t - E(y_t))(y_{t-2} - E(y_{t-2})\right\} &= E\left\{(\alpha + x_t + v_t - \alpha)(y_{t-2} - E(y_{t-2}))\right\} \\ &= E\left\{(\phi x_{t-1} + w_t + v_t)(y_{t-2} - E(y_{t-2}))\right\} \\ &= E\left\{(\phi(y_{t-1} - \alpha - v_{t-1}) + w_t + v_t)(y_{t-2} - E(y_{t-2}))\right\} \\ &= \phi\left\{(y_{t-1} - E(y_{t-1}))(y_{t-2} - E(y_{t-2}))\right\} \\ \rightarrow \quad & \phi^{(0)} = \frac{\sum_{t=3}^{n}(y_t - \bar{y}_t)(y_{t-2} - \bar{y}_{t-2})}{\sum_{t=3}^{n}(y_{t-1} - \bar{y}_{t-1})(y_{t-2} - \bar{y}_{t-2})}, \end{aligned} \tag{2.22}$$

where, $\bar{y}_{t-k}$ is the average of $\{y_{3-k}, \cdots, y_{n-k}\}$ for $k = 0, 1, 2$.

3. $[Q]$

$$\begin{aligned} & E[\{y_t - E(y_t) - \phi(y_{t-1} - E(y_{t-1}))\}^2] \\ = \; & E\{\alpha + x_t + v_t - \alpha - \phi(\alpha + x_{t-1} + v_{t-1} - \alpha)\}^2 \\ = \; & E\{w_t + v_t - \phi v_{t-1}\}^2 \\ = \; & Q + \mathrm{var}(v_t) + \phi^2 \mathrm{var}(v_{t-1}) \\ \rightarrow \quad & Q^{(0)} = \frac{\sum_{t=2}^{n}(y_t - \bar{y}_t - \phi(y_{t-1} - \bar{y}_{t-1}))^2}{n-1} - \widehat{\mathrm{var}(v_t)} - \hat{\phi}^2 \widehat{\mathrm{var}(v_{t-1})}. \end{aligned} \tag{2.23}$$

Since the above is the moment estimates, there is a possibility that the estimates do not meet a certain constraint. For example, ϕ should be between -1 and 1 to make the process stationary. As a result, I use the following as initial parameters: for the $\widehat{\text{var}(v_t)}$, I use 5, which is the approximate variance of $\log(\chi_1^2)$.

1. $\alpha^{(0)} = \dfrac{\sum_{t=0}^{n} y_t}{n}$.
2. $\phi^{(0)} = \text{sign}\left(\dfrac{\sum_{t=3}^{n}(y_t - \bar{y}_t)(y_{t-2} - \bar{y}_{t-2})}{n-2}\right) \cdot \min\left(\left|\dfrac{\sum_{t=3}^{n}(y_t - \bar{y}_t)(y_{t-2} - \bar{y}_{t-2})}{n-2}\right|, 0.99\right)$.
3. $Q^{(0)} = \max\left(\dfrac{\sum_{t=2}^{n}(y_t - \bar{y}_t - \phi(y_{t-1} - \bar{y}_{t-1}))^2}{n-1} - \widehat{\text{var}(v_t)} - \hat{\phi}^2\widehat{\text{var}(v_{t-1})}, 0.01\right)$.

2.4.2 Relative Likelihood

It is important to monitor the change of the likelihood (or the log-likelihood) at each iteration. In the EM algorithm, it is known that the likelihood of the observed data increases at every iteration. Although the E-step in this thesis, which use particles to calculate the expected likelihood, does not guarantee the monotone likelihood property, it is still worth watching the behavior of the relative likelihood. The relative likelihood is the ratio of the likelihoods at two adjacent iterations, and the relative likelihood at ith iteration, $\dfrac{f_{\theta^{(i)}}(y)}{f_{\theta^{(i-1)}}(y)}$, can be calculated by using the complete likelihood as follows:

$$\frac{f_{\theta^{(i)}}(y)}{f_{\theta^{(i-1)}}(y)} = \frac{f_{\theta^{(i)}}(x,y)}{f_{\theta^{(i)}}(x|y)} \cdot \frac{f_{\theta^{(i-1)}}(x|y)}{f_{\theta^{(i-1)}}(x,y)},$$

where y is observed data and x is complete data.

Multiply $f_{\theta^{(i)}}(x|y)$ and integrate out x, we get

$$\frac{f_{\theta^{(i)}}(y)}{f_{\theta^{(i-1)}}(y)} = E_{\theta^{(i-1)}}\left[\frac{f_{\theta^{(i)}}(x,y)}{f_{\theta^{(i-1)}}(x,y)}\middle| y\right].$$

So, the change in the log-likelihood is

$$\begin{aligned}\Delta l_Y(\theta^{(i-1)}, \theta^{(i)}) &= \log f_{\theta^{(i)}}(y) - \log f_{\theta^{(i-1)}}(y) \\ &= \log \frac{f_{\theta^{(i)}}(y)}{f_{\theta^{(i-1)}}(y)} \\ &= \log E_{\theta^{(i-1)}}\left[\frac{f_{\theta^{(i)}}(x,y)}{f_{\theta^{(i-1)}}(x,y)}\middle| y\right]. \end{aligned} \tag{2.24}$$

Hence, if we have samples $\{X_j\}_{j=1}^{M}$ from $f_{\theta^{(i-1)}}(x|y)$, the above can be estimated as follows:

$$\hat{\Delta} l_Y(\theta^{(i-1)}, \theta^{(i)}) \quad = \quad \log\left(\frac{1}{M}\sum_{j=1}^{M}\frac{f_{\theta^{(i)}}(X_j, y)}{f_{\theta^{(i-1)}}(X_j, y)}\right). \tag{2.25}$$

This is a slight modification of Chan and Ledolter (1995). They used the following equation to produce the relative likelihood for the MCEM algorithm:

$$\tilde{\Delta} l_Y(\theta^{(i-1)}, \theta^{(i)}) \quad = \quad -\log\left(\frac{1}{M}\sum_{j=1}^{M}\frac{f_{\theta^{(i-1)}}(X_j, y)}{f_{\theta^{(i)}}(X_j, y)}\right), \tag{2.26}$$

where $\{X_j\}_{i=1}^{M}$ is sample from $f_{\theta^{(i)}}(x|y)$.

The MCEM algorithm will be briefly introduced in Section 2.5.3

2.4.3 Stopping Rule and Selection of Particle Size

Theoretically, the suggested algorithm converges when the particle size, M, and the number of iterations, l, are large. Practically, it is not possible to use infinitely large M and l. Hence, to choose appropriate M and l is important in order to implement the suggested algorithm.

To decide where to stop the iterating estimation procedure is of importance because, the estimates from the procedure stopped too early may not be reliable, and it is waste of time and resources to run the procedure longer than necessary. Many numerical procedures involving iteration compare the two estimates at their previous iteration and their current iteration; if the two are close enough, the process is considered to be converged, and is stopped. Equivalently, the relative likelihood, the difference between two likelihood values at two adjacent iterations, can be considered as a measure in assessing convergence. In particular, a small relative likelihood signifies that a process approaches convergence. I use the relative likelihood to assess convergence in my thesis. More specifically, I conclude that the process is converged if the relative likelihood is less than some pre-determined tolerance, ϵ.

Regarding the selection of M, Tanner (1996) mentioned that it is inefficient to start with a large value of M when $\theta^{(i)}$ is far from the mode, and it is wise to start with a small M, increasing it as the current approximation moves closer to the MLE in the MCEM setting. I apply Tanner's method to save computing time.

2.4.4 Standard Deviation of Parameter Estimates

Variance-covariance matrix can be estimated by the inverse of observed information. Observed information is

$$-\frac{\partial^2 \log f(Y|\theta)}{\partial\theta\partial\theta'}\bigg|_{Y=y}.$$

According to Louis (1982), because $f(Y|\theta)$ is hard to handle in the EM setting, the observed information is calculated by using only the complete likelihood.

$$\begin{aligned}\frac{\partial^2 \log f(Y|\theta)}{\partial\theta\partial\theta'} &= \frac{\partial}{\partial\theta}\left[\frac{\frac{\partial}{\partial\theta'}\int f(X,Y|\theta)dX}{\int f(X,Y|\theta)dX}\right] \\ &= \frac{\frac{\partial^2}{\partial\theta\partial\theta'}\int f(X,Y|\theta)dX}{\int f(X,Y|\theta)dX} - \frac{\frac{\partial}{\partial\theta}\int f(X,Y|\theta)dX\frac{\partial}{\partial\theta'}\int f(X,Y|\theta)dX}{(\int f(X,Y|\theta)dX)^2}. \quad (2.27)\end{aligned}$$

Since

$$\begin{aligned}\frac{\frac{\partial^2}{\partial\theta\partial\theta'}\int f(X,Y|\theta)dX}{\int f(X,Y|\theta)dX} &= \int \frac{\frac{\partial^2}{\partial\theta\partial\theta'}f(X,Y|\theta)}{f(X,Y|\theta)}\cdot\frac{f(X,Y|\theta)}{\int f(X,Y|\theta)dX}dX \\ &= E\left[\frac{\frac{\partial^2}{\partial\theta\partial\theta'}f(X,Y|\theta)}{f(X,Y|\theta)}\bigg|Y\right], \quad (2.28)\end{aligned}$$

and

$$\begin{aligned}\frac{\frac{\partial}{\partial\theta}\int f(X,Y|\theta)dX}{\int f(X,Y|\theta)dX} &= \int \frac{\frac{\partial}{\partial\theta}f(X,Y|\theta)}{f(X,Y|\theta)}\cdot\frac{f(X,Y|\theta)}{\int f(X,Y|\theta)dX}dX \\ &= E\left[\frac{\frac{\partial}{\partial\theta}f(X,Y|\theta)}{f(X,Y|\theta)}\bigg|Y\right]. \quad (2.29)\end{aligned}$$

The second derivatives of the observed likelihood is

$$\frac{\partial^2 \log f(Y|\theta)}{\partial\theta\partial\theta'} = E\left[\frac{\frac{\partial^2}{\partial\theta\partial\theta'}f(X,Y|\theta)}{f(X,Y|\theta)}\bigg|Y\right] - E\left(\frac{\frac{\partial}{\partial\theta}f(X,Y|\theta)}{f(X,Y|\theta)}\bigg|Y\right)E\left(\frac{\frac{\partial}{\partial\theta'}f(X,Y|\theta)}{f(X,Y|\theta)}\bigg|Y\right). \quad (2.30)$$

On the other hand, the second derivatives of the complete likelihood is

$$\begin{aligned}\frac{\partial^2}{\partial\theta\partial\theta'}\log f(X,Y|\theta) &= \frac{\partial}{\partial\theta}\frac{\frac{\partial}{\partial\theta'}f(X,Y|\theta)}{f(X,Y|\theta)} \\ &= \frac{\frac{\partial^2}{\partial\theta\partial\theta'}f(X,Y|\theta)}{f(X,Y|\theta)} - \frac{(\frac{\partial}{\partial\theta}f(X,Y|\theta))(\frac{\partial}{\partial\theta'}f(X,Y|\theta))}{(f(X,Y|\theta))^2}, \end{aligned} \tag{2.31}$$

and the conditional expectation of (2.31) is

$$\begin{aligned}E\left[\frac{\partial^2}{\partial\theta\partial\theta'}\log f(X,Y|\theta)\middle|Y\right] &= E\left[\frac{\frac{\partial^2}{\partial\theta\partial\theta'}f(X,Y|\theta)}{f(X,Y|\theta)}\middle|Y\right] \\ &\quad -E\left[\frac{(\frac{\partial}{\partial\theta}f(X,Y|\theta))(\frac{\partial}{\partial\theta'}f(X,Y|\theta))}{(f(X,Y|\theta))^2}\middle|Y\right]. \end{aligned} \tag{2.32}$$

So

$$\begin{aligned}E\left[\frac{\frac{\partial^2}{\partial\theta\partial\theta'}f(X,Y|\theta)}{f(X,Y|\theta)}\middle|Y\right] &= E\left[\frac{\partial^2}{\partial\theta\partial\theta'}\log f(X,Y|\theta)\middle|Y\right] \\ &\quad +E\left[\left(\frac{\partial}{\partial\theta}\log f(X,Y|\theta)\right)\left(\frac{\partial}{\partial\theta'}\log f(X,Y|\theta)\right)\middle|Y\right]. \end{aligned} \tag{2.33}$$

By plugging (2.33) in (2.30), I get

$$\begin{aligned}\frac{\partial^2\log f(Y|\theta)}{\partial\theta\partial\theta'} &= E\left[\frac{\partial^2}{\partial\theta\partial\theta'}\log f(X,Y|\theta)\middle|Y\right] \\ &\quad +E\left[\left(\frac{\partial}{\partial\theta}\log f(X,Y|\theta)\right)\left(\frac{\partial}{\partial\theta'}\log f(X,Y|\theta)\right)\middle|Y\right] \\ &\quad -E\left[\frac{\partial}{\partial\theta}\log f(X,Y|\theta)\middle|Y\right]E\left[\frac{\partial}{\partial\theta'}\log f(X,Y|\theta)\middle|Y\right]. \end{aligned} \tag{2.34}$$

Hence, the observed information of observed data, y, is

$$\begin{aligned}-\frac{\partial^2\log f(Y|\theta)}{\partial\theta\partial\theta'} &= E\left[-\frac{\partial^2}{\partial\theta\partial\theta'}\log f(X,Y|\theta)\middle|Y\right] \\ &\quad -E\left[\left(\frac{\partial}{\partial\theta}\log f(X,Y|\theta)\right)\left(\frac{\partial}{\partial\theta'}\log f(X,Y|\theta)\right)\middle|Y\right] \\ &\quad +E\left[\frac{\partial}{\partial\theta}\log f(X,Y|\theta)\middle|Y\right]E\left[\frac{\partial}{\partial\theta'}\log f(X,Y|\theta)\middle|Y\right]. \end{aligned} \tag{2.35}$$

Therefore, the observed information can be obtained as follows:

If we have sample from $f(X|Y)$,

$$
\begin{aligned}
-\frac{\partial^2 \widehat{\log f}(Y|\theta)}{\partial\theta\partial\theta'} &= \frac{1}{M}\sum_{i=1}^{M} -\frac{\partial^2}{\partial\theta\partial\theta'}\log f(X_i, Y|\theta) \\
&\quad -\frac{1}{M}\sum_{i=1}^{M}\left(\frac{\partial}{\partial\theta}\log f(X_i, Y|\theta)\right)\left(\frac{\partial}{\partial\theta'}\log f(X_i, Y|\theta)\right) \\
&\quad +\left(\frac{1}{M}\sum_{i=1}^{M}\frac{\partial}{\partial\theta}\log f(X_i, Y|\theta)\right)\left(\frac{1}{M}\sum_{i=1}^{M}\frac{\partial}{\partial\theta'}\log f(X_i, Y|\theta)\right),
\end{aligned}
\tag{2.36}
$$

where X_i are sample from $f(X|Y)$.

Once I get the observed information matrix, the variance-covariance matrix can be obtained by taking the inverse of it. However, when I applied (2.36) to simulated data and real data sets, I met a practical problem; the information matrices are not positive definite. For this issue and a possible solution, see Appendix C

2.5 SUMMARY OF METHODS FROM OTHER AUTHORS

2.5.1 Parameter Estimation of SV Models Without Missing Data

SV models have become increasingly popular in recent years and their popularity has resulted in an enormous amount of articles being published about them. Many different approaches have been proposed and some of them are efficient and others are not.

As stressed in Section 1.2, the lack of analytic likelihoods makes the estimation problem difficult, and so either an approximation of the likelihoods or numerical methods has been considered.

Mellino and Turnbull (1990) used a generalized method of moments (GMM), which has a straightforward implementation, but is not efficient. Harvey et al. (1994) proposed a quasi maximum likelihood approach, which approximates the SV model to a linear Gaussian model, and used the well-developed estimation method for the linear Gaussian models (MLE based on Kalman filters and smoothers). This method is easy to perform but might not be efficient

since it ignores the fact that observation error v_t in (1.3) might have different characteristics with a normal distribution.

Durbin and Koopman (2000) used the idea of linearization of general state-space models and matched terms in the likelihood or posterior of a linearized model to those of a linear Gaussian model. As a result, the usual estimation skill for the linear Gaussian model can be applied to general (non-Gaussian and non-linear) state space models. They also explain their method from both classical and Bayesian perspectives.

Shumway and Stoffer (2000) used an approximate likelihood in the normal mixtures setting. Their basic idea is to approximate the observation error, v_t, in equation (1.3) to two normal mixtures and then get the normal approximation of the conditional density of y_t given Y_{t-1} and I_t, where Y_{t-1} represents the previous observation and I_t is the indicator variable representing which of two normals the observation, y_t, comes from.

The methods listed above can be categorized into *approximation methods*. In general, they are easier to perform than *numerically intensive methods* and quicker, because they don't use any computer-intensive method. However, there is a possibility that estimation results might be inaccurate since they use approximations.

The other type of methods are numerically-intensive methods. These methods have recently become popular, because of the relatively cheap computing costs. Their strong points are that they use an exact likelihood. However, it takes a longer time to get the parameter estimates than with approximation methods.

A Bayesian approach, or MCMC method, was taken by Jacquier et al. (1994). Chib et al. (2002) and Kim et al. (1998) also adopted these methods. The researchers' basic idea is to get a random sample from the posterior density of parameters given the data. In their approach, parameters are assumed to have some prior density, while in the classical analysis (non-Bayesian analysis), parameters are fixed and unknown, as in my study. Usually, their algorithms are iterative algorithms consisting of the following steps:

1. To sample from $p(\theta|Y_n, X_n)$
2. To sample from $p(X_n|\theta, Y_n)$
3. By repeating above, a random sample $\{\theta, X_n\}$ from the joint posterior density $p(\theta, X_n|Y_n)$ can be obtained and this sample enables to estimate parameters and to get filters and

smoothers.

It is not obvious how to sample from the desired densities, so there are some techniques, which vary by author, needed to actually get random samples.

Recently, particle methods (Sequential Monte Carlo or Sequential Important Sampling) have been also applied to the SV model. Because particle filters are designed to get the samples of hidden states given parameters and data, in order to solve the estimation problem, it is necessary to either adopt other methods for the parameter estimation, or modify the particle filtering method to embrace the parameters as a part of the hidden states. A standard approach consists of setting a prior distribution on the unknown parameters and then considering the extended state, (X_n, θ). Estimation can be done by applying a filtering algorithm (for example, see Kitagawa and Sato (2001)). Using this method, the parameters are considered hidden states, like $\{x_t\}$. Doucet and Tadic (2003) combined particle filtering methods and gradient algorithms.

The proposed method in this chapter is free from approximation method criticisms by adopting the EM algorithms. Also, my proposed method deals with the fixed parameter problem, which most of the numerically-intensive methods do not. Also, the proposed approach will be more robust against the departure from the normality assumption of ϵ_t in equation (1.2) by adopting the normal mixtures idea in Chapter 3.

2.5.2 Stochastic EM Algorithm

Celeux and Diebolt (1985) introduced the *Stochastic EM algorithm*, a modified EM algorithm, to compute the MLE of finite mixture models. The only difference between the EM algorithm and the Stochastic EM algorithm is the way the E-step is performed. While the original EM algorithm calculates the expected likelihood $Q(\theta|\theta')$ as in (2.2), the Stochastic EM algorithm returns one realization of the likelihood by random sampling in the Stochastic E-step. In this regard, the Stochastic EM algorithm shares the same idea with the suggested method and MCEM algorithm, which will be introduced in the next section. Here is a brief summary of the Stochastic EM algorithm, using the notation from Section 2.1.1.

Stochastic expectation step (Stochastic E-step) and M-step form one iteration of the

Stochastic EM algorithm. At the k^{th} iteration, the updated parameter, $\theta^{(k)}$, is obtained from $\theta^{(k-1)}$ as follows:

Stochastic E-step: Compute $Q^*(\theta|\theta^{(k-1)})$, where

$$Q^*(\theta|\theta') = \log f(x^*, y|\theta) \text{ and } x^* \text{ is random sample from} f(x|y, \theta'). \tag{2.37}$$

M-step: Choose $\theta^{(k)}$ which maximizes $Q^*(\theta|\theta^{(k-1)})$.

2.5.3 MCEM Algorithm

In some applications of the EM algorithm, the E-step is hard to perform; and is sometimes impossible. Wei and Tanner (1990) introduced the *Monte Carlo EM algorithm* where the E-step is executed by Monte Carlo methods. Chan and Ledolter (1995) applied the MCEM algorithm to a time series model with count data, where the E-step is intractable even by numerical integration.

The MCEM algorithm can be considered as a generalization of the Stochastic EM algorithm. While the Stochastic EM algorithm returns one realization of the likelihood by random sampling in its E-step, the MCEM algorithm returns the expectation of M realizations of the likelihood by random sampling in its E-step. The following is a brief summary of the MCEM algorithm.

Monte Carlo expectation step (Monte Carlo E-step) and M-step form one iteration of the MCEM algorithm. At the k^{th} iteration, the updated parameter, $\theta^{(k)}$ is obtained from $\theta^{(k-1)}$ as follows:

Monte Carlo E-step: Compute $Q^*(\theta|\theta^{(k-1)})$, where

$$Q^*(\theta|\theta') = \frac{1}{M}\sum_{j=1}^{M} \log f(x_j^*, y|\theta) \text{ and } \{x_j^*\}_{j=1}^{M} \text{ is random sample from} f(x|y, \theta'). \tag{2.38}$$

M-step: Choose $\theta^{(k)}$ which maximizes $Q^*(\theta|\theta^{(k-1)})$.

My suggested method is similar to the MCEM algorithm through the use of the particle filter method to execute the E-step while the MCEM algorithm uses the Monte Carlo method. The particle filter method fully takes advantage of the sequential structure of the state-space models, thus it is easier to produce samples. In MCEM algorithms, it is not as obvious how to get samples from desired distributions, and is sometimes difficult.

3.0 SLIGHT MODIFICATION - NORMAL MIXTURES

One of the main assumptions that the stochastic volatility model holds is that ϵ_t in equation (1.2) has a normal distribution. Hence, in the previous chapter, I used (1.3) as an observation equation in the linear state space model setting and used centered $\log(\chi_1^2)$ as an observation noise. In this chapter, I will make use of two normal mixtures instead of $\log(\chi_1^2)$ as an observation error. In Section 3.1, I will introduce normal mixture models and their strong points compared to the model in the previous chapter. In Section 3.2, I will present the modification in details. Section 3.3 presents a method to obtain suitable initial parameters.

3.1 NORMAL MIXTURES AS AN OBSERVATION NOISE

3.1.1 Model Structures

In this chapter, I consider mixtures of two normal distributions as an observation noise of the linearized version of the SV model (see (1.1) and (1.3)). Therefore, the observation equation becomes

$$y_t = \alpha + x_t + v_t, \tag{3.1}$$

where

$$\begin{aligned} v_t &= I_t z_{t1} + (1 - I_t) z_{t0} - \mu\pi, \\ z_{t0} &\sim N(0, R_0), \\ z_{t1} &\sim N(\mu, R_1), \\ I_t &\sim Bernoulli(\pi). \end{aligned}$$

π is an unknown mixing probability, and z_{t0} and z_{t1} are normal random variables with means and variances $(0, R_0)$ and (μ, R_1), respectively. The last term in v_t, $-\mu\pi$, is added to make the v_t a zero mean variable, as in Chapter 2. The above equation is equivalent to the following equation:

$$y_t = x_t + v_t, \tag{3.2}$$

where

$$\begin{aligned} v_t &= I_t z_{t1} + (1 - I_t) z_{t0}, \\ z_{t0} &\sim N(m_0, R_0), \\ z_{t1} &\sim N(m_1, R_1), \\ I_t &\sim Bernoulli(\pi). \end{aligned}$$

When the observation equation is expressed as (3.2), the estimation step is easier to perform than when it is expressed as (3.1). Here, $m_0 = \alpha - \mu\pi$ and $m_1 = \alpha + (1 - \pi)\mu$. I call this model *Model B* throughout this thesis.

Model B

$$\begin{aligned} x_t &= \phi x_{t-1} + w_t, \\ y_t &= x_t + v_t, \end{aligned}$$

where $w_t \sim N(0, Q)$, $v_t = I_t z_{t1} + (1 - I_t) z_{t0}$, $I_t \sim Bernoulli(\pi)$ and $z_{ti} \sim N(m_i, R_i)$.

3.1.2 Advantages Over the Regular Model

The main advantage of the normal mixture distribution over the log of chi-square distribution for an observation error is its flexibility. In the previous chapter, v_t in the observation equation, (1.3), contains no parameter, which means that it leaves no room for deviation from the assumed model. People suspect that real data may have heavier tails than $\log(\chi_1^2)$ upon closer examination. If the real data is very different from the assumed distribution, $\log(\chi_1^2)$, the resulting parameter estimation may not be successful. An example will be shown in Chapter 5.

However, normal mixture distributions bring flexibility. Every probability distribution, including $\log(\chi_1^2)$, can be approximated by normal mixtures with an appropriate number of components. v_t has its own parameter $\{m_0, m_1, R_0, R_1, \pi\}$, which is estimated in the estimation step along with the other parameters, $\{\phi, Q\}$. Therefore, this modification will give better parameter estimates for $\{\phi, Q\}$ since it uses an observation error which reflects the data structure, rather than the observation error which had been derived just from the assumption.

3.2 MODIFICATION FOR THE NORMAL MIXTURE MODEL

In this setting, I consider $\{I_t\}$ as a part of the data. Like $\{x_t\}$ in the EM setting in Chapter 2, $\{I_t\}$ is also not observed, and it can be considered as missing data. The basic strategy is to apply the EM algorithm to the complete data $\{x_0, x_1, \cdots, x_n, y_1, \cdots, y_n, I_1, \cdots, I_n\}$, where $\{x_0, x_1, \cdots, x_n, I_1, \cdots, I_n\}$ are missing.

In addition to the symbols introduced in the previous chapter, I will use the following symbols:

- $\tilde{\mathrm{f}}_t^{(j)}$: jth particle filter of a indicator variable, I_t, at time t $((\mathrm{f}_t^{(j)}, \tilde{\mathrm{f}}_t^{(j)}) \sim f(x_t, I_t|Y_t))$.
- $\tilde{\mathrm{s}}_t^{(j)}$: jth particle smoother of a indicator variable, I_t, at time t $((\mathrm{s}_t^{(j)}, \tilde{\mathrm{s}}_t^{(j)}) \sim f(x_t, I_t|Y_n))$.
- $\tilde{\mathrm{p}}_t^{(j)}$: jth particle predictor of a indicator variable, I_t, at time t $((\mathrm{p}_t^{(j)}, \tilde{\mathrm{p}}_t^{(j)}) \sim f(x_t, I_t|Y_s)$, $s < t)$.

3.2.1 Overview of the Algorithm for Normal Mixture Cases

Parameter estimation algorithm for Model B: *Algorithm B*

Make initial guesses for the parameters: $\theta^{(0)} = \{\phi^{(0)}, Q^{(0)}, m_0^{(0)}, m_1^{(0)}, R_0^{(0)}, R_1^{(0)}, \pi^{(0)}\}$.

For $i = 1, \cdots, i_{max}$, where i_{max} is the maximum number of iterations,

1. Filtering step: get particle filters $(\mathrm{f}_t^{(j)}, \tilde{\mathrm{f}}_t^{(j)})$ from $f(x_t, I_t|Y_t, \theta^{(i-1)})$, $j = 1, \cdots, M$, $t = 1, \cdots, n$, where M is the number of particles.

2. Smoothing step: get particle smoothers $\{(\mathrm{s}_0^{(j)}, \cdots, \mathrm{s}_n^{(j)}), (\tilde{\mathrm{s}}_1^{(j)}, \cdots, \tilde{\mathrm{s}}_n^{(j)})\}$ from $f(x_0, \cdots, x_n, I_1, \cdots, I_n \ |Y_n, \theta^{(i-1)})$, $j = 1, \cdots, M$ and calculate and return quantities which are needed to get EM estimates.

3. Estimation step: get updated parameter estimates, $\theta^{(i)} = \{\phi^{(i)}, Q^{(i)}, m_0^{(i)}, m_1^{(i)}, R_0^{(i)}, R_1^{(i)}, \pi^{(i)}\}$, by maximizing the expected likelihood.

Repeat 1-3 until the process converges.

As you may notice, the algorithm looks the same as that of Chapter 2. However, many of the details are different. I want to present details of the estimation step first because quantities needed to be saved in the smoothing step can be identified through the estimation step. I call this algorithm for Model B *Algorithm B* throughout this thesis.

3.2.2 Estimation Step

Minor modification is needed in the estimation step. The complete likelihood of $\{x_0, \cdots, x_n, y_1, \cdots, y_n, I_1, \cdots, I_n\}$ is[1]

[1] $X = \{x_0, \cdots, x_n\}, Y = \{y_1, \cdots, y_n\}, I = \{I_1, \cdots, I_n\}$.

$$
\begin{aligned}
f(X,Y,I) &= f(x_0)\prod_{t=1}^{n} f(x_t|x_{t-1})\prod_{t=1}^{n} f(I_t|I_{t-1},X)\prod_{t=1}^{n} f(y_t|y_{t-1},I,X) \\
&= f(x_0)\prod_{t=1}^{n} f(x_t|x_{t-1})\prod_{t=1}^{n} f(I_t)\prod_{t=1}^{n} f(y_t|x_t,I_t) \\
&= \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(-\frac{(x_0-\mu_0)^2}{2\sigma_0^2}\right)\prod_{t=1}^{n}\frac{1}{\sqrt{2\pi Q}}\exp\left(-\frac{(x_t-\phi x_{t-1})^2}{2Q}\right) \\
&\quad \prod_{t=1}^{n}\pi^{I_t}(1-\pi)^{1-I_t}\prod_{t=1}^{n}\frac{1}{\sqrt{2\pi R_t^*}}\exp\left(-\frac{(y_t-x_t-\mu_t^*)^2}{2R_t^*}\right),
\end{aligned}
$$

where $R_t^* = I_t R_1 + (1-I_t)R_0,\ \mu_t^* = I_t m_1 + (1-I_t)m_0$. The log-likelihood is

$$
\begin{aligned}
-2\log f(X,Y,I) &= \log\sigma_0^2 + \frac{(x_0-\mu_0)^2}{\sigma_0^2} + \sum_{t=1}^{n}\left[\log Q + \frac{(x_t-\phi x_{t-1})^2}{Q}\right] \\
&\quad -2\sum_{t=1}^{n}\left[I_t\log\pi + (1-I_t)\log(1-\pi)\right] \\
&\quad +\sum_{t=1}^{n}\left[\log R_t^* + \frac{(y_t-x_t-\mu_t^*)^2}{R_t^*}\right] + C,
\end{aligned}
$$

where C is a constant. Now, the expected likelihood given data, $\{y_1,\cdots,y_n\}$, is

$$
\begin{aligned}
Q(\theta) &= E(-2\log f(X,Y,I)|Y) \\
&= \log\sigma_0^2 + \frac{(x_0^n-\mu_0)^2+p_0^n}{\sigma_0^2} + \sum_{t=1}^{n}\left[\log Q + \frac{(x_t^n-\phi x_{t-1}^n)^2 + P_t^n + \phi^2 P_{t-1}^n - 2\phi P_{t,t-1}^n}{Q}\right] \\
&\quad -2\left[\log\pi\sum_{t=1}^{n}\pi_t^n + \log(1-\pi)\left(n-\sum_{t=1}^{n}\pi_t^n\right)\right] \\
&\quad +\sum_{i=0}^{1}\left[\sum_{t=1}^{n}\pi_{ti}^n\log R_i\right] + \sum_{i=0}^{1}\left[\frac{1}{R_i}\sum_{t=1}^{n}\{E(I_{ti}(y_t-x_t-m_i)^2|Y_n)\}\right], \qquad (3.3)
\end{aligned}
$$

where $\pi_{t1}^n = \pi_t^n = E[I_t|Y_n]$, $\pi_{t0}^n = 1-\pi_t^n$, $I_{t1} = I_t$, $I_{t0} = 1-I_t$.

By minimizing (3.3), I get the following estimates.

1. $[\phi, Q]$

$$\begin{aligned}
\frac{\partial Q(\theta)}{\partial \phi} &= \sum_{t=1}^{n} \frac{-2x_{t-1}^n(x_t^n - \phi x_{t-1}^n) + 2\phi P_{t-1}^n - 2P_{t,t-1}^n}{Q} \\
&= \sum_{t=1}^{n} \frac{2}{Q}[\phi((x_{t-1}^n)^2 + P_{t-1}^n) - (x_{t-1}^n x_t^n + P_{t,t-1}^n)] \\
&= \frac{2}{Q}[\phi S_{00} - S_{10}] \\
\frac{\partial Q(\theta)}{\partial Q} &= \frac{n}{Q} - \sum_{t=1}^{n} \frac{(x_t^n - \phi x_{t-1}^n)^2 + P_t^n + \phi^2 P_{t-1}^n - 2\phi P_{t,t-1}^n}{Q^2} \\
&= \frac{1}{Q^2}\left[nQ - \sum_{t=1}^{n}\{\phi^2((x_{t-1}^n)^2 + P_{t-1}^n) - 2\phi(x_{t-1}^n x_t^n + P_{t,t-1}^n) + ((x_t^n)^2 + P_t^n)\}\right] \\
&= \frac{1}{Q^2}[nQ - \{\phi^2 S_{00} - 2\phi S_{10} + S_{11}\}]
\end{aligned}$$

$$\hat{\phi} = \frac{S_{10}}{S_{00}}, \tag{3.4}$$

$$\hat{Q} = \frac{1}{n}\left(S_{11} - \frac{S_{10}^2}{S_{00}}\right), \tag{3.5}$$

where

$$\begin{aligned}
S_{00} &= \sum_{t=1}^{n}((x_{t-1}^n)^2 + P_{t-1}^n), \qquad (3.6) \\
S_{11} &= \sum_{t=1}^{n}((x_t^n)^2 + P_t^n), \\
S_{10} &= \sum_{t=1}^{n}(x_{t-1}^n x_t^n + P_{t,t-1}^n).
\end{aligned}$$

2. $[\pi]$

$$\begin{aligned}
\frac{\partial Q(\theta)}{\partial \pi} &= -2\left\{\frac{1}{\pi}\sum_{t=1}^{n}\pi_t^n - \frac{1}{1-\pi}\left(n - \sum_{t=1}^{n}\pi_t^n\right)\right\} \\
&= \frac{2}{\pi(1-\pi)}\left\{\pi\left(\sum_{t=1}^{n}\pi_t^n + n - \sum_{t=1}^{n}\pi_t^n\right) - \sum_{t=1}^{n}\pi_t^n\right\}
\end{aligned}$$

$$\hat{\pi} = \frac{\sum_{t=1}^{n}\pi_t^n}{n}. \tag{3.7}$$

3. $[m_0, m_1, R_0, R_1]$

$$\begin{aligned}
\frac{\partial Q(\theta)}{\partial m_0} &= -2\frac{1}{R_0}\left\{\sum_{t=1}^{n} E[(1-I_t)(y_t - x_t - m_0)|Y_n]\right\} \\
\frac{\partial Q(\theta)}{\partial m_1} &= -2\frac{1}{R_1}\left\{\sum_{t=1}^{n} E[I_t(y_t - x_t - m_1)|Y_n]\right\} \\
\frac{\partial Q(\theta)}{\partial R_0} &= \frac{1}{R_0}\left(n - \sum_{t=1}^{n}\pi_t^n\right) - \frac{1}{R_0^2}\left\{\sum_{t=1}^{n} E[(1-I_t)(y_t - x_t - m_0)^2|Y_n]\right\} \\
\frac{\partial Q(\theta)}{\partial R_1} &= \frac{1}{R_1}\left(\sum_{t=1}^{n}\pi_t^n\right) - \frac{1}{R_1^2}\left\{\sum_{t=1}^{n} E[I_t(y_t - x_t - m_1)^2|Y_n]\right\}
\end{aligned}$$

$$\hat{m_0} = \frac{\sum_{t=1}^{n} E[(1-I_t)(y_t - x_t)|Y_n]}{n - \sum_{t=1}^{n}\pi_t^n}, \tag{3.8}$$

$$\hat{m_1} = \frac{\sum_{t=1}^{n} E[I_t(y_t - x_t)|Y_n]}{\sum_{t=1}^{n}\pi_t^n}, \tag{3.9}$$

$$\hat{R_0} = \frac{\sum E[(1-I_t)((y_t - x_t - \hat{m}_0)^2|Y_n)]}{n - \sum_{t=1}^{n}\pi_t^n}, \tag{3.10}$$

$$\hat{R_1} = \frac{\sum E[(I_t)((y_t - x_t - \hat{m}_1)^2)|Y_n]}{\sum_{t=1}^{n}\pi_t^n}. \tag{3.11}$$

From the above, $\hat{x}_t^n$, $\hat{P}_t^n$, $\hat{P}_{t,t-1}^n$, $\hat{\pi}_t^n$, $E[I_t(\widehat{y_t - x_t^n})|Y_n]$, $E[(1-I_t)(\widehat{(y_t} - x_t)^2|Y_n)]$ and $E[(I_t)((y_t - \widehat{x_t} - \hat{\mu}_1)^2)|Y_n]$ are needed to get parameter estimates. A smoothing step returns these quantities.

3.2.3 Filtering Step

The result of using this filtering step will be getting particle filters which are the random samples from $f(x_t, I_t|Y_t)$. These particle filters are necessary in order to be able to perform the smoothing step. The difference from Chapter 2 is that $\{I_t\}$ is also sampled by assuming $\{I_t\}$ as another state variable. Hence, the state equation of this model can be re-expressed as the following:

$$\begin{aligned}
x_t &= \phi x_{t-1} + w_t, \\
I_t &= 0 \cdot I_{t-1} + B_t,
\end{aligned} \tag{3.12}$$

or

$$\begin{pmatrix} x_t \\ I_t \end{pmatrix} = \begin{pmatrix} \phi & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_{t-1} \\ I_{t-1} \end{pmatrix} + \begin{pmatrix} w_t \\ B_t \end{pmatrix}, \tag{3.13}$$

where $w_t \sim N(0, Q)$, $B_t \sim Bernoulli(\pi)$. The following is the algorithm for the filtering step.

Particle filtering algorithm for Algorithm B

1. Generate $\mathrm{f}_0^{(j)} \sim N(\mu_0, \sigma_0^2)$.
2. For $t = 1, \cdots, n$,
   a. Generate a random number $w_t^{(j)} \sim N(0, Q)$, $j = 1, \cdots, M$.
   Generate a random number $B_t^{(j)} \sim Bernoulli(\pi)$, $j = 1, \cdots, M$.
   b. Compute $\mathrm{p}_t^{(j)} = \phi \mathrm{f}_{t-1}^{(j)} + w_t^{(j)}$.
   Compute $\tilde{\mathrm{p}}_t^{(j)} = B_t^{(j)}$.
   c. Compute $\mathrm{w}_t^{(j)} = p(y_t | \mathrm{p}_t^{(j)}, \tilde{\mathrm{p}}_t^{(j)}) \propto \frac{1}{R_t^{*(j)}} \exp\left(\frac{(y_t - \mathrm{p}_t^{(j)} - \mu_t^{*(j)})^2}{2R_t^{*(j)}} \right)$,
   where $\mu_t^{*(j)} = \tilde{\mathrm{p}}_t^{(j)} m_1 + (1 - \tilde{\mathrm{p}}_t^{(j)}) m_0$ and $R_t^{*(j)} = \tilde{\mathrm{p}}_t^{(j)} R_1 + (1 - \tilde{\mathrm{p}}_t^{(j)}) R_0$.
   d. Generate $[\mathrm{f}_t^{(j)}, \tilde{\mathrm{f}}_t^{(j)}]$ by resampling with weights, $\mathrm{w}_t^{(j)}$.

At the end of the filtering step, I will have M pairs of samples from $f(x_t, I_t | Y_t)$ for $t = 0, \cdots n$.

3.2.4 Smoothing Step

In this step, some quantities which are needed in the estimation step of the EM algorithm will be obtained. To get these quantities, I use the particle smoothers, $(\mathrm{s}_t, \tilde{\mathrm{s}}_t)$, which are a random sample from $f(x_t, I_t | Y_n)$. Again, I follow the method of Godsill et al. (2004). The smoothing step returns the following, which is known to be essential from the Estimation step.

- $\hat{x}_t^n = \dfrac{\sum_{j=1}^M \mathrm{s}_t^{(j)}}{M}$,
- $\hat{P}_t^n = \dfrac{\sum_{j=1}^M (\mathrm{s}_t^{(j)} - \hat{x}_t^n)^2}{M}$,
- $\hat{P}_{t,t-1}^n = \dfrac{\sum_{j=1}^M (\mathrm{s}_t^{(j)} - \hat{x}_t^n)(\mathrm{s}_t^{(j-1)} - \hat{x}_{t-1}^n)}{M}$,
- $\hat{\pi}_t^n = \dfrac{\sum_{j=1}^M \tilde{\mathrm{s}}_t^{(j)}}{M}$,
- $E[I_t(\widehat{y_t - x_t^n})|Y_n] = \dfrac{\sum_{j=1}^M \tilde{\mathrm{s}}_t^{(j)}(y_t - \mathrm{s}_t^{(j)})}{M}$,
- $E[I_t((\widehat{y_t - x_t})^2)|Y_n] = \dfrac{\sum_{j=1}^M \tilde{\mathrm{s}}_t^{(j)}(y_t - \mathrm{s}_t^{(j)})^2}{M}$,
- $E[(1 - I_t)(\widehat{(y_t} - x_t)^2|Y_n)] = \dfrac{\sum_{j=1}^M (1 - \tilde{\mathrm{s}}_t^{(j)})(y_t - \mathrm{s}_t^{(j)})^2}{M}$,

where $\{\mathrm{s}_0^{(j)}, \cdots, \mathrm{s}_n^{(j)}, \tilde{\mathrm{s}}_1^{(j)}, \cdots, \tilde{\mathrm{s}}_n^{(j)}\}$ is the random sample from $f(x_0, \cdots, x_n, I_1, \cdots, I_n|Y_n)$. The following is the algorithm used in the smoothing step:

Particle smoothing algorithm for Algorithm B

Suppose that equally weighted particles $\{(\mathrm{f}_t^{(j)}, \tilde{\mathrm{f}}_t^{(j)}) : j = 1, \cdots, M\}$ from $f(x_t, I_t|Y_t)$ are available for $t = 1, \cdots, n$.

1. Choose $[\mathrm{s}_n^{(j)}, \tilde{\mathrm{s}}_n^{(j)}] = [\mathrm{f}_n^{(i)}, \tilde{\mathrm{f}}_n^{(i)}]$ with probability $\frac{1}{M}$.
2. For $n - 1$ to 0,
   a. Calculate $\mathrm{w}_{t|t+1}^{(i)} \propto f(\mathrm{s}_{t+1}^{(j)}, \tilde{\mathrm{s}}_{t+1}^{(j)}|\mathrm{f}_t^{(i)}, \tilde{\mathrm{f}}_t^{(i)}) \propto \exp\left(-\dfrac{(\mathrm{s}_{t+1}^{(j)} - \phi \mathrm{f}_t^{(i)})^2}{2Q}\right) \pi^{\tilde{\mathrm{s}}_{t+1}^{(j)}}(1 - \pi)^{1 - \tilde{\mathrm{s}}_{t+1}^{(j)}}$ for each i.
   b. Choose $[\mathrm{s}_t^{(j)}, \tilde{\mathrm{s}}_t^{(j)}] = [\mathrm{f}_t^{(i)}, \tilde{\mathrm{f}}_t^{(i)}]$ with probability $\mathrm{w}_{t|t+1}^{(i)}$
3. $(\mathrm{s}_{0:n}^{(j)}, \tilde{\mathrm{s}}_{0:n}^{(j)}) = \{(\mathrm{s}_0^{(j)}, \cdots, \mathrm{s}_n^{(j)}), (\tilde{\mathrm{s}}_0^{(j)}, \cdots, \tilde{\mathrm{s}}_n^{(j)})\}$ is the random sample from $f(x_0, \cdots, x_n, I_0, \cdots, I_n|Y_n)$.
4. Repeat 1-3 for $j = 1, \cdots, M$, and calculate the quantities specified above.

3.3 OTHER ISSUES

3.3.1 Initial Parameter Selection

Again, the initial parameter selection is important because good starting points can save computing time. However, there are seven parameters which need initial values in the normal mixtures case. To apply the method of moments, as in Chapter 2, four more equations should be solved, and so it is a complex job. Therefore, here I have just chosen reasonable values for the initial parameters for the normal mixtures and have then applied the method of moments for three other parameters. In many cases involving univariate data, the choice of starting values will not be critical for estimating the parameters of normal mixtures (See Everitt and Hand (1981)). The following is my suggestion for the initial parameters.

Initial parameter selection for Algorithm B

1. Pick some initial parameters for $m_1 - m_0 = k_1,\ R_0^{(0)} = k_2, R_1^{(0)} = k_3, \pi^{(0)} = k_4$.
2. $$E(y_t) = \pi m_1 + (1-\pi)m_0 = \pi(m_1 - m_0) + m_0 \cong \bar{y}$$
$$\rightarrow m_0^{(0)} = \bar{y} - \pi(m_1 - m_0) = \bar{y} - k_4 k_1,\ m_1^{(0)} = k_1 + m_0.$$
3. $\phi^{(0)} = \text{sign}\left(\frac{\sum_{t=3}^{n}(y_t - \bar{y_t})(y_{t-2} - \bar{y_{t-2}})}{n-2}\right) \cdot \min\left(\left|\frac{\sum_{t=3}^{n}(y_t - \bar{y_t})(y_{t-2} - \bar{y_{t-2}})}{n-2}\right|, 0.99\right)$.
4. $Q^{(0)} = \max\left(\frac{\sum_{t=2}^{n}(y_t - \bar{y_t} - \phi(y_{t-1} - \bar{y_{t-1}}))^2}{n-1} - \widehat{\text{var}(v_t)} - \hat{\phi}^2\widehat{\text{var}(v_{t-1})}, 0.01\right)$.
5. $$\begin{aligned} \text{var}(v_t) &= \text{var}(I_t z_{t1} + (1-I_t)z_{t0}) \\ &= \text{var}(E(I_t z_{t1} + (1-I_t)z_{t0}|I_t)) + E(\text{var}(I_t z_{t1} + (1-I_t)z_{t0}|I_t)) \\ &= \mu^2\pi(1-\pi) + R_1\pi R_0(1-\pi) \\ \rightarrow \quad & \widehat{\text{var}(v_t)} = k_1^2 k_4(1-k_4) + k_3 k_4 + k_2(1-k_4). \end{aligned}$$

The k's I used are

$$k_1 = -3,\ k_2 = k_3 = 4,\ k_4 = 0.5.$$

4.0 MISSING DATA PROBLEM

In this chapter, I introduce one modification of the model structure, so that the estimation method presented in Chapter 3 can handle missing data cases. As I mentioned before, real data has several missing values, but there has been no serious consideration of this problem. In Section 4.1, I will introduce a modified SV model, which can analyze data sets with missing values. Details will be presented in Section 4.2.

4.1 GENERAL IDEA AND MODEL STRUCTURE

The fundamental idea in this thesis for the missing data case, is to make the data complete by filling the missing values in, and to use the method for the complete data case presented in Chapter 3. Model structures for the missing data case are

$$x_t = \phi x_{t-1} + w_t, \tag{4.1}$$

$$y_t = a_t x_t + v_t, \tag{4.2}$$

where

$$\begin{aligned} w_t &\sim N(0, Q), \\ v_t &\sim I_t N(m_1, R_1) + (1 - I_t) N(m_0, R_0), \\ I_t &\sim Bernoulli(\pi), \\ x_0 &\sim N(\mu_0, \sigma_0^2), \\ a_t &= \begin{cases} 0 & \text{if missing} \\ 1 & \text{if observed.} \end{cases} \end{aligned} \tag{4.3}$$

The model now has a_t, which is a missing indicator. This a_t enables us to use the same model structure whether a value is missing or not. This idea was presented by Shumway and Stoffer (2000) for the linear Gaussian state-space models. Hence, my work can be considered as an extension of their work for the SV models.

4.2 DETAILS OF THE PROCEDURE

4.2.1 Overview of the Estimation Algorithm

The only difference in the algorithm for the missing data case is the *data-completion step*. In the data-completion step, I fill missing values in with values generated from the model. Through this step, the complete $\{y_t\}$, which do not include any missing values, can be obtained. The following is the algorithm for the whole procedure.

Parameter estimation algorithm for missing data cases

Let the initial guess of parameters be $\theta^{(0)} = \{\phi^{(0)}, Q^{(0)}, m_0^{(0)}, m_1^{(0)}, R_0^{(0)}, R_1^{(0)}, \pi^{(0)}\}$.

For $i = 1$ to i_{max},

1. Data-completion step: fill the missing data in with the values generated from the true model with given parameters $\theta^{(i-1)}$.
2. Filtering step: get particle filters $(\mathrm{f}_t^{(j)}, \tilde{\mathrm{f}}_t^{(j)})$ from $f(x_t, I_t|Y_t, \theta^{(i-1)})$, $j = 1, \cdots, M$, M: number of particles.
3. Smoothing step: get particle smoothers $\{\mathrm{s}_0^{(j)}, \cdots, \mathrm{s}_n^{(j)}, \tilde{\mathrm{s}}_1^{(j)}, \cdots, \tilde{\mathrm{s}}_n^{(j)}\}$ from $f(x_0, \cdots, x_n, I_1, \cdots, I_n\ |Y_n, \theta^{(i-1)})$, $j = 1, \cdots, M$ and calculate quantities for the estimation step.
4. Estimation step: get updated parameter estimates $\theta^{(i)} = \{\phi^{(i)}, Q^{(i)}, m_0^{(i)}, m_1^{(i)}, R_0^{(i)}, R_1^{(i)}, \pi^{(i)}\}$ by maximizing the expected likelihood.

Repeat 1-4 until the process converges.

Details are given in the following sections.

4.2.2 Data-Completion Step

When $a_t = 0$ in (4.2), the observation equation can be simplified as

$$y_t = v_t,$$

which has no relation to the state variable, x_t. Therefore, when the data is not observed, y_t can be directly generated from the observation equation if the parameters are given. Hence, the data can be completed as follows:

Data-Completion Step

If y_t is missing $(a_t = 0)$

Generate a random sample y_t from the normal mixture distribution;

$$y_t \sim I_t N(m_1, R_1) + (1 - I_t)N(m_0, R_0).$$

Through this data-completion step, it is possible to possess a data set with no missing values, and the method proposed in the previous chapters can be applied with only slight modification.

4.2.3 Filtering Step

In this filtering step for missing data cases, the only difference is the calculation of the weight for resampling $\{\mathrm{f}_t, \tilde{\mathrm{f}}_t\}$ from $\{\mathrm{p}_t, \tilde{\mathrm{p}}_t\}$. The filtering step returns random samples from $f(x_t, I_t|Y_t)$ for $t = 0, 1, \cdots, n$.

Particle filtering algorithm for missing data cases

1. Generate $\mathrm{f}_0^{(j)} \sim N(\mu_0, \sigma_0^2)$.
2. a. Generate a random number $w_t^{(j)} \sim N(0, Q)$.

   Generate a random number $B_t^{(j)} \sim Bernoulli(\pi)$.

   b. Compute $\mathrm{p}_t^{(j)} = \phi \mathrm{f}_{t-1}^{(j)} + w_t^{(j)}$.

   Compute $\tilde{\mathrm{p}}_t^{(j)} = B_t^{(j)}$.

   c. Compute

$$\mathrm{w}_t^{(j)} \quad = \quad p(y_t|\mathrm{p}_t^{(j)}, \tilde{\mathrm{p}}_t^{(j)}) \propto \frac{1}{R_t^{*(j)}} \exp\left(\frac{(y_t - a_t \mathrm{p}_t^{(j)} - \mu_t^{*(j)})^2}{2R_t^{*(j)}}\right), \tag{4.4}$$

   where $\mu_t^{*(j)} = \tilde{\mathrm{p}}_t^{(j)} m_1 + (1 - \tilde{\mathrm{p}}_t^{(j)}) m_0$ and $R_t^{*(j)} = \tilde{\mathrm{p}}_t^{(j)} R_1 + (1 - \tilde{\mathrm{p}}_t^{(j)}) R_0$.

   d. Generate $[\mathrm{f}_t^{(j)}, \tilde{\mathrm{f}}_t^{(j)}]$ by resampling.

Note that (4.4) has the missing indicator, a_t, in it.

4.2.4 Smoothing Step

The smoothing step is exactly the same as the complete data case except for the return quantities. It returns the followings quantities:

- $\hat{x_t}^n = \dfrac{\sum_{j=1}^M \mathrm{s}_t^{(j)}}{M}$,
- $\hat{P_t}^n = \dfrac{\sum_{j=1}^M (\mathrm{s}_t^{(j)} - \hat{x_t}^n)^2}{M}$,
- $\hat{P}_{t,t-1}^n = \dfrac{\sum_{j=1}^M (\mathrm{s}_t^{(j)} - \hat{x}_t^n)(\mathrm{s}_{t-1}^{(j)} - \hat{x}_{t-1}^n)}{M}$,
- $\pi_t^n = \dfrac{\sum_{j=1}^M \tilde{\mathrm{s}}_t^{(j)}}{M}$,
- $E[I_t(\widehat{y_t - a_t x_t^n})|Y_n] = \dfrac{\sum_{j=1}^M \tilde{\mathrm{s}}_t^{(j)}(y_t - a_t \mathrm{s}_t^{(j)})}{M}$,
- $E[(I_t)((\widehat{y_t - a_t x_t})^2)|Y_n] = \dfrac{\sum_{j=1}^M \tilde{\mathrm{s}}_t^{(j)}(y_t - a_t \mathrm{s}_t^{(j)})^2}{M}$,
- $E[(1 - I_t)((\widehat{y_t - a_t x_t})^2|Y_n)] = \dfrac{\sum_{j=1}^M (1 - \tilde{\mathrm{s}}_t^{(j)})(y_t - a_t \mathrm{s}_t^{(j)})^2}{M}$,

where $\{\mathrm{s}_0^{(j)}, \cdots, \mathrm{s}_n^{(j)}, \tilde{\mathrm{s}}_1^{(j)}, \cdots, \tilde{\mathrm{s}}_n^{(j)}\}$ is the random sample jointly drawn from $f(x_1, \cdots, x_n, I_1, \cdots, I_n|Y_n)$. The following is the smoothing algorithm for the missing data case.

Particle smoothing algorithm for missing data cases

Suppose that there are equally weighted particles $\{(\mathrm{f}_t^{(i)}, \tilde{\mathrm{f}}_t^{(i)}) : i = 1, \cdots, M\}$ from $f(x_t, I_t|Y_t)$ for $t = 1, \cdots, n$.

1. Choose $[\mathrm{s}_n^{(j)}, \tilde{\mathrm{s}}_n^{(j)}] = [\mathrm{f}_n^{(i)}, \tilde{\mathrm{f}}_n^{(i)}]$ with probability $\frac{1}{M}$.
2. For $n-1$ to 0,
   a. Calculate $\mathrm{w}_{t|t+1}^{(i)} \propto f(\mathrm{s}_{t+1}^{(j)}, \tilde{\mathrm{s}}_{t+1}^{(j)}|\mathrm{f}_t^{(i)}, \tilde{\mathrm{f}}_t^{(i)}) \propto \exp\left(-\frac{(\mathrm{s}_{t+1}^{(j)} - \phi \mathrm{f}_t^{(i)})^2}{2Q}\right) \pi^{\tilde{\mathrm{s}}_{t+1}^{(j)}}(1-\pi)^{1-\tilde{\mathrm{s}}_{t+1}^{(j)}}$ for each i.
   b. Choose $[\mathrm{s}_t^{(j)}, \tilde{\mathrm{s}}_t^{(j)}] = [\mathrm{f}_t^{(i)}, \tilde{\mathrm{f}}_t^{(i)}]$ with probability $\mathrm{w}_{t|t+1}^{(i)}$.
3. $(\mathrm{s}_{0:n}^{(j)}, \tilde{\mathrm{s}}_{0:n}^{(j)}) = \{(\mathrm{s}_0^{(j)}, \cdots, \mathrm{s}_n^{(j)}), (\tilde{\mathrm{s}}_0^{(j)}, \cdots, \tilde{\mathrm{s}}_n^{(j)})\}$ is the random sample from $f(x_0, \cdots, x_n, I_0, \cdots, I_n\ |Y_n)$.
4. Repeat 1-3 for $j = 1, \cdots, M$, and calculate the quantities specified above.

4.2.5 Estimation Step

For the estimation step, everything is the same as before, except that there is the term a_t in $\hat{m}_0, \hat{m}_1, \hat{R}_0, \hat{R}_1$.

1. $[\phi, Q]$

$$\hat{\phi} = \frac{S_{10}}{S_{00}}, \tag{4.5}$$

$$\hat{Q} = \frac{1}{n}\left(S_{11} - \frac{S_{10}^2}{S_{00}}\right), \tag{4.6}$$

   where $S_{00} = \sum_{t=1}^n (x_{t-1}^n)^2 + P_{t-1}^n,\ S_{11} = \sum_{t=1}^n (x_t^n)^2 + P_t^n,\ S_{10} = \sum_{t-1}^n x_t^n x_{t-1}^n + P_{t,t-1}^n$.
2. $[\pi]$

$$\hat{\pi} = \frac{\sum_{t=1}^n \pi_t^n}{n}.$$

3. $[m_0, m_1, R_0, R_1]$

$$\hat{m_0} = \frac{\sum_{t=1}^{n} E[(1-I_t)(y_t - a_t x_t)|Y_n]}{n - \sum_{t=1}^{n} \pi_t^n}, \tag{4.7}$$

$$\hat{m_1} = \frac{\sum_{t=1}^{n} E[I_t(y_t - a_t x_t)|Y_n]}{\sum_{t=1}^{n} \pi_t^n}, \tag{4.8}$$

$$\hat{R_0} = \frac{\sum E[(1-I_t)((y_t - a_t x_t - \hat{m}_0)^2|Y_n)]}{n - \sum_{t=1}^{n} \pi_t^n}, \tag{4.9}$$

$$\hat{R_1} = \frac{\sum E[(I_t)((y_t - a_t x_t - \hat{m}_1)^2)]}{\sum_{t=1}^{n} \pi_t^n}. \tag{4.10}$$

For the estimates of expectations, I use the sample mean of the function of particles. For example,

$$E[I_t(y_t \widehat{-a_t} x_t^n)|Y_n] = \frac{\sum_{j=1}^{M} \tilde{\mathrm{s}}_t^{(j)}(y_t - a_t \mathrm{s}_t^{(j)})}{M},$$

which is provided from the smoothing step.

5.0 SIMULATION STUDY AND DATA ANALYSIS

In this chapter, I apply the proposed method to two simulated data sets and two real data sets. Data A is generated from the usual version of SV models, Model A, which has the logarithm of chi-square distribution as its observation noise. Data B is generated from the SV model proposed in Chapter 3, Model B, which has two normal mixtures as its observation noise. Also, I consider two types of pound/dollar exchange rates to see the performance of the proposed method for the real data in Sections 5.2 and 5.3.

5.1 SIMULATION STUDIES

I consider two simulated data sets in this section. Data A is generated from Model A:

$$\begin{aligned} x_t &= 0.9x_{t-1} + w_t, \\ y_t &= -3 + x_t + v_t, \end{aligned}$$

where $w_t \sim N(0,1)$, $v_t \sim \log(\chi_1^2) - E(\log(\chi_1^2))$ and $t = 1, \cdots, 1000$. Here, the true parameter set of (ϕ, Q, α) is $(0.9, 1, -3)$. To make this process stationary, I generate 11000 samples and discard the first 10000 values.

Data B is generated from Model B:

$$\begin{aligned} x_t &= 0.8x_{t-1} + w_t, \\ y_t &= -5.5 + x_t + v_t, \end{aligned}$$

where $w_t \sim N(0, 1.5)$, $v_t \sim I_t N(-3,5) + (1-I_t)N(0,3) + 1.5$ and $I_t \sim Bernoulli(0.5)$. Again, the length of the data, $\{y_t\}$, is 1000. The true parameter set of $(\phi, Q,\ m_0, m_1,\ R_0, R_1,\ \pi)$ is

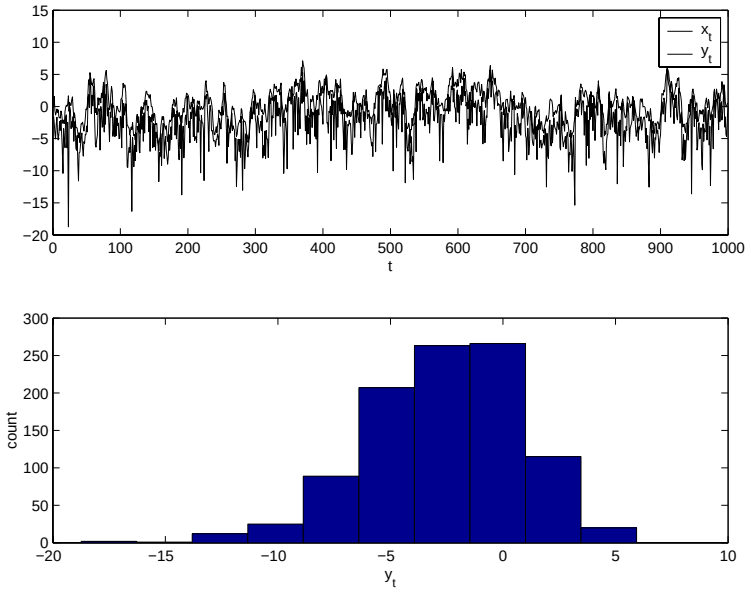

Figure 5.1: Time plot (top), and histogram (bottom) of data A.

(0.8,1.5,-4,-7,3,5,0.5). I use this data set to observe the behavior of the estimation procedure, when there is a departure from the log of chi-square assumption.

5.1.1 Data A

Figure 5.1 shows the plot of data A (top) and the histogram of data A (bottom). We can see that $\{y_t\}$ is skewed to the left since it is generated by using $\log(\chi^2)$. I will apply two methods presented in Chapter 2 and Chapter 3 to this data (Algorithms A and B). Furthermore, I randomly remove some parts of the data and apply the method in Chapter 4 to examine the performance of the proposed method for missing data cases.

5.1.1.1 Method in Chapter 2: With Algorithm A By the procedure described in Section 2.4.1, $(0.9656, 2.9862, -2.6034)$ is selected for the initial parameter for (ϕ, Q, α). Table 5.1 shows the results of estimation procedure. I started with 500 particles (M=500) and ϵ=0.1 and at 5th iteration, the process was stopped because the relative likelihood at the next iteration was less the 0.1. I increased M and decreased ϵ, and repeated the procedure until I achieved the tolerance 0.001.

Table 5.1: Estimation results for Data A: Algorithm A

iteration(i)	$\phi^{(i)}$	$Q^{(i)}$	$\alpha^{(i)}$	Relative Likelihood	M	ϵ
0	0.9656	2.9862	-2.6034	-	500	0.1
1	0.8258	2.3918	-2.6047	33.8690		
2	0.8236	2.2016	-2.6015	0.4544		
3	0.8322	2.0849	-2.5987	0.0985		
4	0.8390	1.9795	-2.5982	0.0308		
5	0.8453	1.8952	-2.5921	0.0545		
6	0.8507	1.8246	-2.5935	0.0487	500	0.01
7	0.8535	1.7500	-2.5927	0.0690		
8	0.8582	1.6936	-2.5862	0.0300		
9	0.8616	1.6373	-2.5858	0.0664		
10	0.8655	1.5990	-2.5932	0.0150	1000	0.001
15	0.8732	1.4950	-2.5801	0.0020		
20	0.8767	1.4316	-2.5706	0.0011		
23	0.8783	1.4059	-2.5659	0.0025		

Figure 5.2 shows the trajectory of the parameter estimates and the relative likelihoods. Matlab functions are given in Appendix D. The final estimates, along with their standard errors (in parentheses), were

$$\hat{\phi} = 0.8783 \ (0.0184), \quad \hat{Q} = 1.4059 \ (0.1425), \quad \hat{\alpha} = -2.5659 \ (0.1109).$$

See Appendix C for the details of the standard error evaluation. It can be said that the estimation procedure works well in the sense that the estimates are close to the true parameters, $(0.9, 1, -3)$.

5.1.1.2 Method in Chapter 3: With Algorithm B Based on the method in Section 3.3.1, I use (0.9656, 2.9862, -2.6034, -5.6034, 4, 4, 0.5) as the initial values for the parameters, $(\phi^{(0)}, Q^{(0)}, m_0^{(0)}, m_1^{(0)}, R_0^{(0)}, R_1^{(0)}, \pi^{(0)})$. Table 5.2 shows the results of the estimation procedure and Figure 5.3 presents the history of relative likelihood at each iteration. The process was stopped when the value of relative likelihood was less than 0.001. The final estimates, along with their standard deviations (in parentheses), were

$$\hat{\phi} = 0.9077 \ (0.0151), \ \hat{Q} = 1.0180 \ (0.0950), \ \hat{m}_0 = -1.3622 \ (0.1049), \ \hat{m}_1 = -4.1971 \ (0.3013), \ \hat{R}_0 = 1.8067 \ (0.2614), \ \hat{R}_1 = 9.1332(1.0564), \ \hat{\pi} = 0.3160 \ (0.0359).$$

Note that the parameter estimates from the method in Chapter 2 were (0.8783, 1.4059, -2.5659) for (ϕ, Q, α) and the true parameters are $(0.9, 1, -3)$. In this approach, $\hat{\alpha} = \hat{\pi}\hat{m}_1 + (1 - \hat{\pi})\hat{m}_0 = -2.2579$; $(\hat{\phi}, \hat{Q}, \hat{\alpha}) = (0.9077, 1.0180, -2.2579)$. This results show that the normal mixture model gives good estimates even if the true observation noise is not a normal mixture distribution.

5.1.1.3 Method in Chapter 4: Missing Data Case Now, to see the performance of the presented method for missing data cases, I randomly erased some parts of Data A, and applied the method. The estimation results are given in Table 5.3. (1) is the result for non-missing data, which is presented in the previous section. I randomly removed 10% of the data and applied the method presented in Chapter 4, and got the result in (2). (3) is the result for the case where 20% of the data is missing. Although, as the rate of missing data increases, it gets harder to achieve a certain tolerance and a bigger number of particles

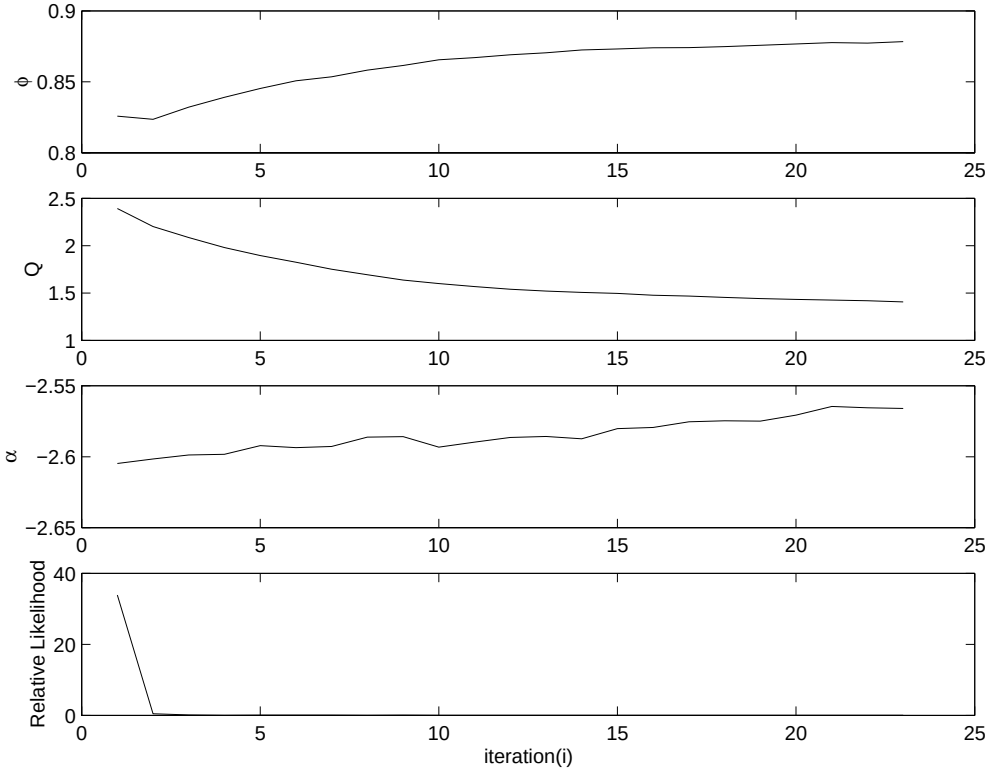

Figure 5.2: Parameter estimation results for (ϕ, Q, α) (1st to 3rd), and the relative likelihood (bottom) for data A: Algorithm A.

Table 5.2: Estimation results for Data A: Algorithm B

i	$\phi^{(i)}$	$Q^{(i)}$	$m_0^{(i)}$	$m_1^{(i)}$	$R_0^{(i)}$	$R_1^{(i)}$	$\pi^{(i)}$	Rel.Like.	M	ϵ
0	0.9656	2.9862	-2.6034	-5.6034	4	4	0.5	-		
1	0.9385	0.5904	-1.1494	-4.0508	2.9586	5.6761	0.4713	18.2485	500	0.1
2	0.9358	0.6083	-1.1710	-3.9735	2.4938	6.7706	0.4495	4.7136		
6	0.9304	0.6858	-1.2409	-3.9016	1.9047	8.6768	0.3940	0.1227		
7	0.9300	0.7075	-1.2421	-3.8966	1.8675	8.8125	0.3841	0.0423	500	0.01
9	0.9259	0.7554	-1.2564	-3.9349	1.8180	8.9206	0.3721	0.0294		
10	0.9247	0.7768	-1.2611	-3.9579	1.8052	8.9361	0.3680	0.0075	1000	0.001
20	0.9119	0.9631	-1.3412	-4.1303	1.7821	9.0800	0.3289	0.0002		
24	0.9077	1.0180	-1.3622	-4.1971	1.8067	9.1332	0.3160	0.0026		

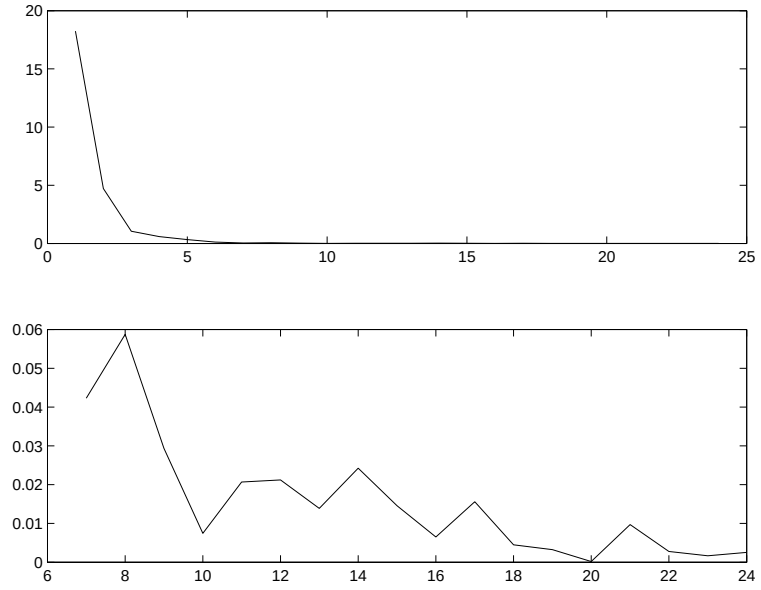

Figure 5.3: Relative likelihood of Data A: Algorithm B.

is needed, we can see that my proposed method handles missing data cases pretty well. Standard deviations of parameter estimates are also presented in Table 5.3.

5.1.2 Data B

Figure 5.4 shows the plot of data B (top) and the histogram of data B (bottom). I apply two methods presented in Chapter 2 and Chapter 3, Algorithms A and B, to this data. Furthermore, I randomly remove some parts of data and apply the method in Chapter 4 to see the performance of the proposed method for missing data cases. I consider this data especially to see the merit of normal mixture idea in Chapter 3.

5.1.2.1 Method in Chapter 2: With Algorithm A I estimate the parameters in the model with (1.1) and (1.3). Here, the assumption on the observation noise is violated. The initial parameter set (0.8798, 4.0601,-5.7376) was used for (ϕ, Q, α). Table 5.4 and Figure 5.5 show the results of parameter estimation procedure. The final estimates, along with their standard deviations (in parentheses), were

Table 5.3: Estimation results for Data A: Missing data case

	ϕ	Q	m_0	m_1	R_0	R_1	π	α
True	0.9	1						-3
(1)	0.9077	1.0180	-1.3622	-4.1971	1.8067	9.1332	0.3160	-2.2579
s.d	0.0151	0.0950	0.1049	0.3013	0.2614	1.0564	0.0359	
(2)	0.8730	0.9679	-1.7672	-4.9413	2.8191	10.8303	0.2705	-2.6259
s.d	0.0224	0.1511	0.2169	0.5323	0.3613	1.5125	0.0566	
(3)	0.8608	0.9064	-1.5988	-4.6787	2.8207	11.6589	0.3017	-2.5281
s.d	0.0272	0.1756	0.3191	0.4729	0.4806	1.5701	0.0538	

(1) No missing, $\epsilon = 0.001$, $M = 1000$

(2) 10 % missing, $\epsilon = 0.001$, $M = 4000$

(3) 20 % missing $\epsilon = 0.01$, $M = 2000$

ϵ is tolerance which assesses convergence and M is the number of particles.

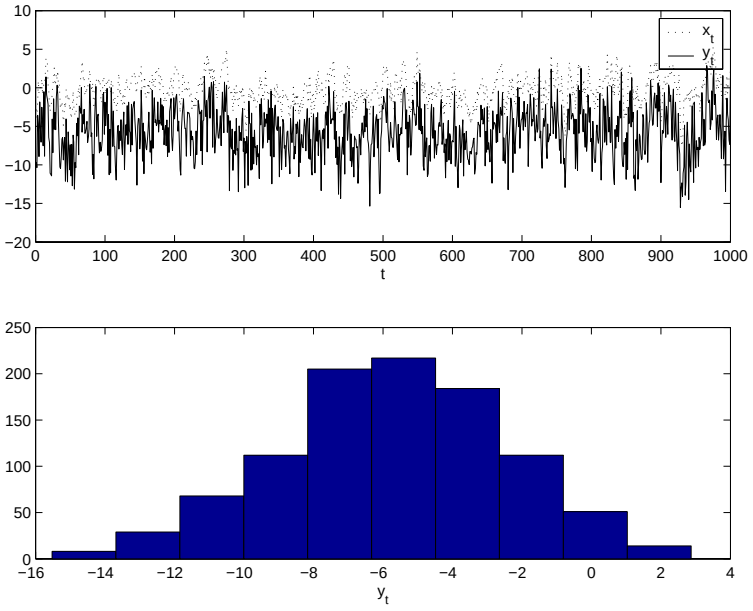

Figure 5.4: Time plot (top), and histogram (bottom) of data B.

Table 5.4: Estimation results for Data B: Algorithm A

iteration(i)	$\phi^{(i)}$	$Q^{(i)}$	$\alpha^{(i)}$	Relative Likelihood	M	ϵ
0	0.8798	4.0601	-5.7376	-		
1	0.6921	3.5925	-5.7327	29.9114	500	0.1
2	0.6534	3.6711	-5.7289	0.6513		
3	0.6348	3.7797	-5.7183	0.1602		
4	0.6241	3.8853	-5.7279	0.0349		
5	0.6167	3.9825	-5.7211	0.0462		
6	0.6117	4.0536	-5.7215	0.0247	500	0.01
7	0.6030	4.1181	-5.7191	0.0252		
8	0.6026	4.1798	-5.7213	0.0221		
9	0.5959	4.2129	-5.7262	0.0139	1000	0.001
10	0.5933	4.2683	-5.7338	0.0098		
15	0.5865	4.3589	-5.7446	0.0019		
20	0.5833	4.3719	-5.7427	0.0020		

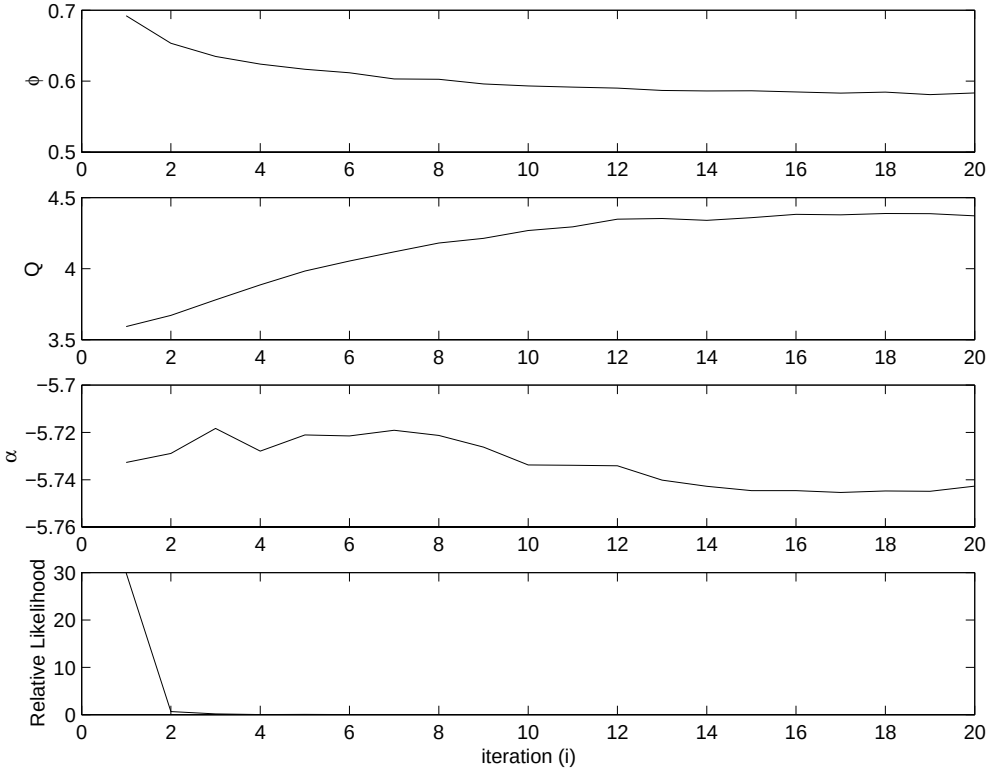

Figure 5.5: Parameter estimation results for (ϕ, Q, α) (1st to 3rd), and the relative likelihood (bottom) for data B: Algorithm A.

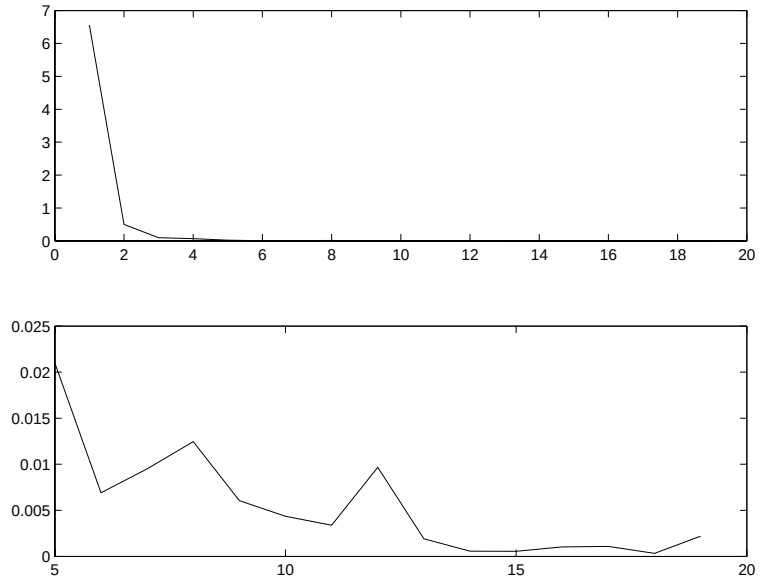

Figure 5.6: Relative likelihood: Data B.

$$\hat{\phi} = 0.5833\ (0.0347),\ \hat{Q} = 4.3719\ (0.3183),\ \hat{\alpha} = -5.7427\ (0.1156),$$

where the true parameters are $(0.8, 1.5, -5.5)$. This result shows that when the assumed model is far from the true model, it may result in wrong estimates if Algorithm A is applied. Therefore, extension of the SV model by adopting normal mixtures is meaningful.

5.1.2.2 Method in Chapter 3: With Algorithm B The following is the result, when I fit Data B with normal mixture observation error. Table 5.5 shows the results of the parameter estimation procedure; see Figure 5.6 for the history of the relative likelihood. (0.8798, 4.0601, -5.7376, -8.7376, 4, 4, 0.5) were used as initial parameters of $(\phi, Q, m_0, m_1, R_0, R_1, \pi)$. At 20th iteration, relative likelihood was less than 0.001, which is pre-determined tolerance, ϵ, and the process was considered converged. The final estimates, along with their standard deviations were

$$\hat{\phi} = 0.7654\ (0.0303),\ \hat{Q} = 1.8131\ (0.2188),\ \hat{m}_0 = -4.3240\ (0.1611),\ \hat{m}_1 = -7.2851\ (0.2361),\ \hat{R}_0 = 3.0971\ (0.4034), \hat{R}_1 = 4.9751\ (0.5950),\ \hat{\pi} = 0.4761\ (0.0408).$$

These estimates are pretty similar to the true parameters $(0.8, 1.5, -4, -7, 3, 5, 0.5)$, while Algorithm A returns (0.5804,4.4151,-5.7439) as $(\hat{\phi}, \hat{Q}, \hat{\alpha})$. Therefore, when the $\log(\chi^2)$ as-

Table 5.5: Estimation results for Data B: Algorithm B

i	$\phi^{(i)}$	$Q^{(i)}$	$m_0^{(i)}$	$m_1^{(i)}$	$R_0^{(i)}$	$R_1^{(i)}$	$\pi^{(i)}$	Rel.Like.	M	ϵ
0	0.8798	4.0601	-5.7376	-8.7376	4	4	0.5	-		
1	0.8051	1.7523	-4.2599	-7.2300	3.7679	4.2274	0.4952	6.5581	500	0.1
5	0.7726	1.7443	-4.2909	-7.2530	3.3299	4.8029	0.4824	0.0210		
6	0.7713	1.7547	-4.3006	-7.2627	3.2950	4.8571	0.4809	0.0069	500	0.05
10	0.7698	1.7683	-4.3056	-7.2799	3.1548	4.9449	0.4782	0.0044		
11	0.7677	1.7727	-4.3025	-7.2838	3.1393	4.9379	0.4774	0.0034	500	0.01
13	0.7680	1.7822	-4.3066	-7.2712	3.1222	4.9843	0.4771	0.0019		
14	0.7675	1.7897	-4.3106	-7.2788	3.1197	4.9778	0.4767	0.0006	2000	0.01
15	0.7672	1.7906	-4.3137	-7.2786	3.1154	4.9930	0.4760	0.0006		
16	0.7670	1.7946	-4.3165	-7.2780	3.1051	4.9808	0.4755	0.0010	2000	0.001
19	0.7654	1.8131	-4.3240	-7.2851	3.0971	4.9751	0.4761	0.0022		

sumption is not met, the result from the method based on that assumption (Algorithm A) is not reliable. However, the method based on the normal mixture idea (Algorithm B) works well in both cases.

5.1.2.3 Method in Chapter 4: Missing Data Case I randomly removed some parts of Data B, and applied the method for missing data cases. The estimation results are given in Table 5.6. (1) is the result for the non-missing data, which is presented in the previous section. I randomly removed 10% of the data and applied the method presented in Chapter 4, and got the result in (2). (3) is the result for the case where 20% of the data is missing. Although, as the rate of missing data increases, it gets harder to achieve a certain tolerance and a bigger number of particles are needed, we can see that my proposed method handles missing data cases pretty well. Standard deviations of parameter estimates are also presented in Table 5.6.

5.2 POUND AND DOLLAR DAILY EXCHANGE RATES

The data presented in this section is the pound-dollar daily exchange rates from October 1st, 1981, to June 28th, 1985, which have been used by Harvey et al. (1994). See Figure 5.2 for the plot of the log of r_t^2. I introduce this data to compare the performance of my method

Table 5.6: Estimation results for Data B: Missing data case

	ϕ	Q	m_0	m_1	R_0	R_1	π
True	0.8	1.5	-4	-7	3	5	0.5
(1)	0.7654	1.8131	-4.3240	-7.2851	3.0971	4.9751	0.4761
s.d	0.0303	0.2188	0.1611	0.2361	0.4034	0.5950	0.0408
(2)	0.7231	1.8238	-4.3254	-7.3412	3.3379	5.2886	0.4810
s.d	0.0355	0.2113	0.0903	0.2830	0.6774	0.8861	0.0117
(3)	0.7629	1.3329	-4.1378	-7.1377	3.2313	5.2576	0.4837
s.d	0.1824	1.6103	1.0013	0.5510	0.5597	2.6681	0.0559

(1) No missing, ϵ=0.001, M=2000

(2) 10 % missing, ϵ=0.01, M=2000

(3) 20 % missing, ϵ=0.05, M=2000

to that of previously suggested methods by other authors for non-missing data.

5.2.1 Estimates in the References

Doucet and Tadic (2003) and Durbin and Koopman (2000) used slightly different model structures but they can be easily converted to my model:

$$X_{t+1} = \phi X_t + \sigma V_{t+1}, \tag{5.1}$$

$$Y_t = \beta \exp(X_t/2) W_t, \tag{5.2}$$

where $V_t \sim N(0,1)$, $W_t \sim N(0,1)$ and $X_0 \sim N\left(0, \frac{\sigma^2}{1-\phi^2}\right)$. Equation (5.2) can be re-expressed as

$$\log(Y_t^2) = \log \beta^2 + X_t + \log W_t^2 \tag{5.3}$$

$$\log(Y_t^2) = (\log \beta^2 - 1.2749) + X_t + (\log W_t^2 - E(\log W_t^2)) \tag{5.4}$$

Equation (5.2) and (5.4) are identical to the state equation and the observation equation of my model (Model A), respectively. This makes it possible to compare the estimates from the reference and the estimates from the proposed method.

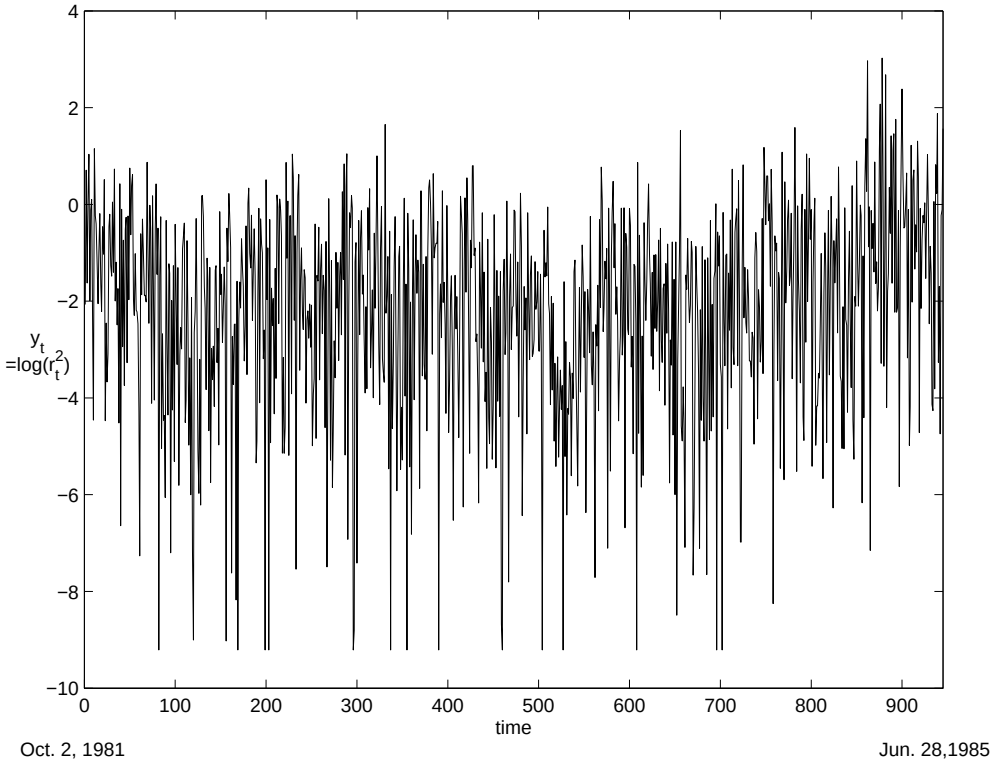

Figure 5.7: The logarithm of squares of the log return of pound/dollar exchange rates, $y_t = \log r_t^2$.

Table 5.7: Parameter estimates in the reference papers

Ref.	ϕ	σ	β	$\log\beta^2$-1.2749 $=\alpha$	Q=σ^2
Doucet and Tadic	0.968	0.188	0.638	-2.1737	0.0353
Durbin and Koopman	0.973	0.173	0.634	-2.1863	0.0299

See Table 5.7 for the estimates. Doucet and Tadic (2003) used a batch ML algorithm for 1000 iterations with 10000 particles. Durbin and Koopman (2000) estimated parameters by approximating the likelihood to the linear Gaussian model.

5.2.2 Estimates Using Algorithm A

By the procedure described in Section 2.4.1, I got (-2.2194, 0.99, 0.01) as initial parameters for (α, ϕ, Q). Figure 5.8 shows the estimation results: the final estimates at 200th iteration, along with their standard deviations, were

$$\hat{\phi} = 0.9757\ (0.0083),\ \hat{Q} = 0.0255\ (0.0032),\ \hat{\alpha} = -2.2320\ (0.1005).$$

5.2.3 Estimates Using Algorithm B

When the normal mixture error model was applied, the final estimates, along with their standard deviations, were

$$\hat{\phi} = 0.9783\ (0.0079),\ \hat{Q} = 0.0228\ (0.0023), \hat{m}_0 = -1.2918\ (0.1071),\ \hat{m}_1 = -4.0999\ (0.2252),\ \hat{R}_0 = 1.3464\ (0.1202), \hat{R}_1 = 4.8946\ (0.5242),\ \hat{\pi} = 0.3374(0.0349).$$

See Figure 5.9 for the whole history of the parameter estimation. Table 5.2.3 makes it possible to compare the results from the four different approaches. Since all four results are quite similar, it can be said that my two methods work as well as the other two methods.

5.3 POUND AND DOLLAR DAILY EXCHANGE RATES WITH MISSING VALUES

I consider other Pound-dollar daily exchange rates to validate the performance of the proposed method for missing data cases. This data set is British pound and dollar exchange rate data from Franses and van Dijk (2000), expressed as the number of units of foreign currency per US dollar. This data is collected for the period of December 31, 1979 to December 31, 1998. [1]

[1] Original source: Federal Reserve Bank of New York.

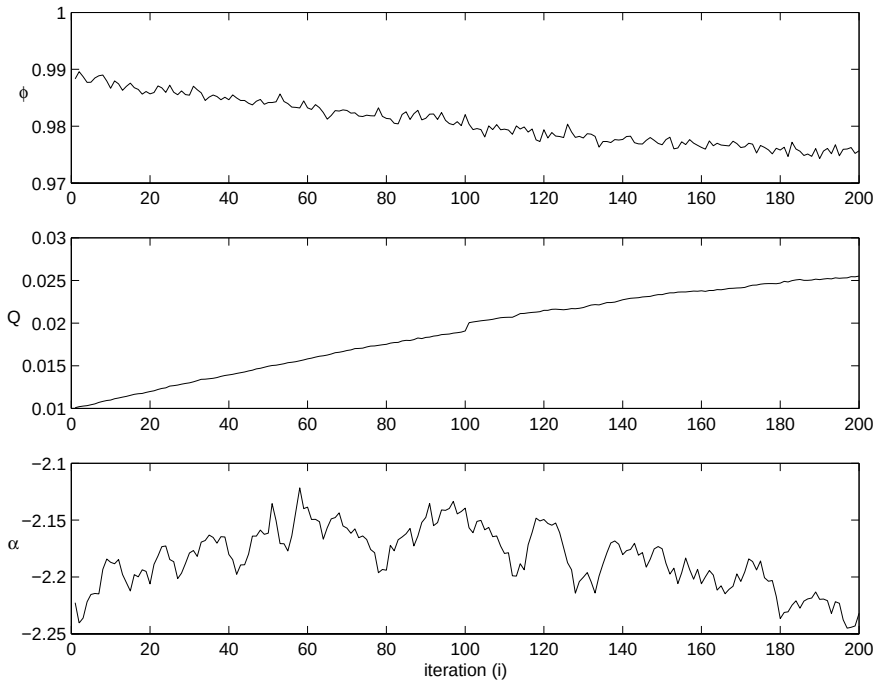

Figure 5.8: Estimation results for the pound/dollar exchange rates: Algorithm A.

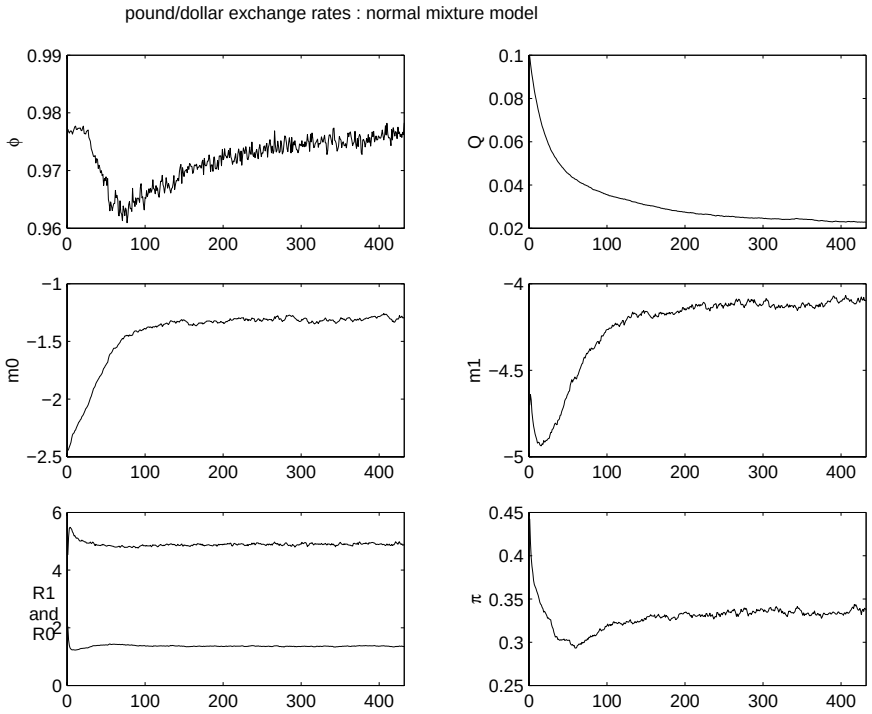

Figure 5.9: Estimation results for the pound/dollar exchange rates: Algorithm B.

Table 5.8: Estimation results for the pound/dollar exchange rates

Ref.	ϕ	α	Q
Doucet and Tadic	0.968	-2.1737	0.0353
Durbin and Koopman	0.973	-2.1863	0.0299
Algorithm A	0.9757	-2.2320	0.0255
Algorithm B	0.9783	-2.2393	0.0228

Here, I use only part of the data, (Jan. 1 1996 – December 31, 1998). There are 784 observations and 29 of them are missing. In order to apply the proposed method, the following transformations are necessary.

$$e_t \quad : \quad \text{pound-dollar exchange rate, } t = 1, \cdots, 784. \tag{5.5}$$

$$r_t \quad = \quad \log(e_{t+1}) - \log(e_t) : \text{ log return of exchange rate, } t = 1, \cdots, 783. \tag{5.6}$$

$$y_t \quad = \quad \log((r_t - \bar{r_t})^2), \ t = 1, \cdots, 783. \tag{5.7}$$

Raw data, e_t, can be transformed to the log return of the data, r_t, and y_t can be obtained by squaring r_t and taking the logarithm of it. For y_t, I subtract $\bar{r_t}$ from r_t to make $(r_t - \bar{r_t})^2$ non-zero value ($\bar{r_t} = -0.0001558$). The number of missing values in $\{y_t\}$ is 57. This is bigger than 29 because one missing exchange rate, e_t, makes two log returns, r_t and r_{t+1}, missing. Figure 5.10 shows the plots of the data. The method presented in Chapter 4 is used in order to analyze this data.

Table 5.9 presents the results of parameter estimation procedure. I started with 500 particles and increased them until I achieved the tolerance 0.01. Figure 5.11 shows the trajectory of parameter estimates and the history of the relative likelihood. The final estimates, along with their standard deviations, were

$$\hat{\phi} = 0.8963 \ (0.0269), \ \hat{Q} = 0.0697 \ (0.0102), \ \hat{m}_0 = -11.5022 \ (0.0985), \ \hat{m}_1 = -14.0647 \ (0.2665), \ \hat{R}_0 = 1.8165 \ (0.1612), \ \hat{R}_1 = 5.4195 \ (0.6948), \ \hat{\pi} = 0.3195 \ (0.0339).$$

This result shows that my proposed method can be used to estimate parameters in the SV model with missing data. It is not easy to compare this results with those in Franses and van Dijk (2000) because they used weekly data, which use every Wednesday and if Wednesday's data is missing use Tuesday or Thursday if Tuesday's data is also missing, and they applied GARCH model to the weekly data set. Their final estimates for GARCH model[2], along with their standard deviations[3], were

$$\hat{\omega} = 0.171 \ (1.105), \ \hat{\alpha}_1 = 0.071 \ (0.028), \ \hat{\beta}_1 = 0.856 \ (0.062).$$

[2]GARCH(1,1) model: $h_t = \omega + \alpha_1 \epsilon_{t-1}^2 + \beta_1 h_{t-1}, \ \epsilon_t = z_t \sqrt{h_t}$, where z_t is independent and identically distributed with $N(0, 1)$.

[3]Standard error based on the Hessian matrix, see Franses and van Dijk (2000) for details

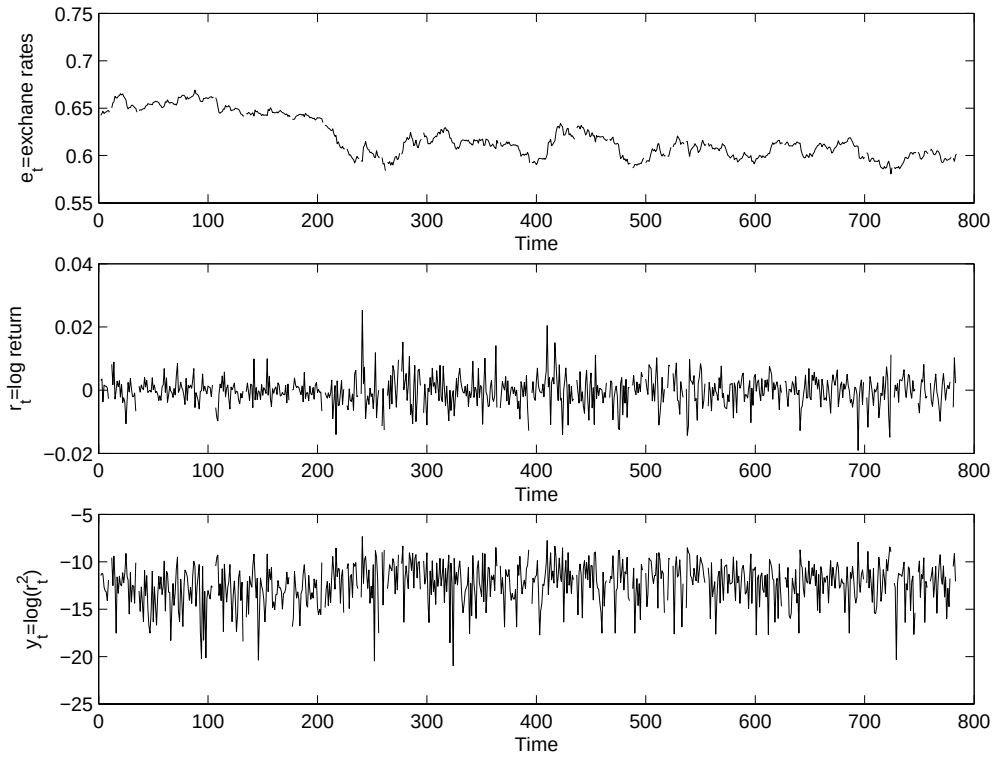

Figure 5.10: Pound/dollar daily exchange rates with missing values; the exchange rates, e_t (top), the log returns, r_t (middle) and the transformed log returns, y_t (bottom).

Table 5.9: Estimation results for the pound/dollar exchange rates with missing values

i	$\phi^{(i)}$	$Q^{(i)}$	$m_0^{(i)}$	$m_1^{(i)}$	$R_0^{(i)}$	$R_1^{(i)}$	$\pi^{(i)}$	Rel.Like.	M	ϵ
0	0.9500	0.1000	-10.8424	-13.8424	4.0000	4.0000	0.5000	-		
1	0.9232	0.0969	-11.1812	-13.4800	2.3380	4.2875	0.4763	27.6039	500	0.1
2	0.9090	0.0947	-11.2122	-13.4460	1.8826	4.8810	0.4586	4.0416		
6	0.8840	0.0888	-11.3082	-13.5552	1.5619	5.3957	0.4324	0.1049		
7	0.8773	0.0879	-11.3307	-13.5816	1.5350	5.3550	0.4261	0.0709	500	0.05
12	0.8906	0.0868	-11.3803	-13.6935	1.5734	5.7232	0.4037	0.0592		
13	0.8890	0.0860	-11.3886	-13.7054	1.5973	5.5871	0.4000	0.0488	1000	0.01
21	0.8865	0.0815	-11.4265	-13.8347	1.6855	5.5782	0.3753	0.0360		*
22	0.8886	0.0810	-11.4234	-13.8402	1.7338	5.6475	0.3720	0.0477	2000	0.01
30	0.8975	0.0770	-11.4732	-13.9622	1.6981	5.5026	0.3546	0.0214		*
31	0.8921	0.0763	-11.4696	-13.9759	1.7441	5.5955	0.3522	0.0818	3000	0.01
39	0.8896	0.0725	-11.4839	-14.0073	1.8220	5.4217	0.3322	0.1187		*
40	0.8874	0.0720	-11.4808	-14.0423	1.8101	5.5347	0.3322	0.0534	4000	0.01
46	0.8963	0.0697	-11.5022	-14.0647	1.8165	5.4195	0.3195	0.2010		

*: Process didn't converge for 10 iterations, so I increased M.

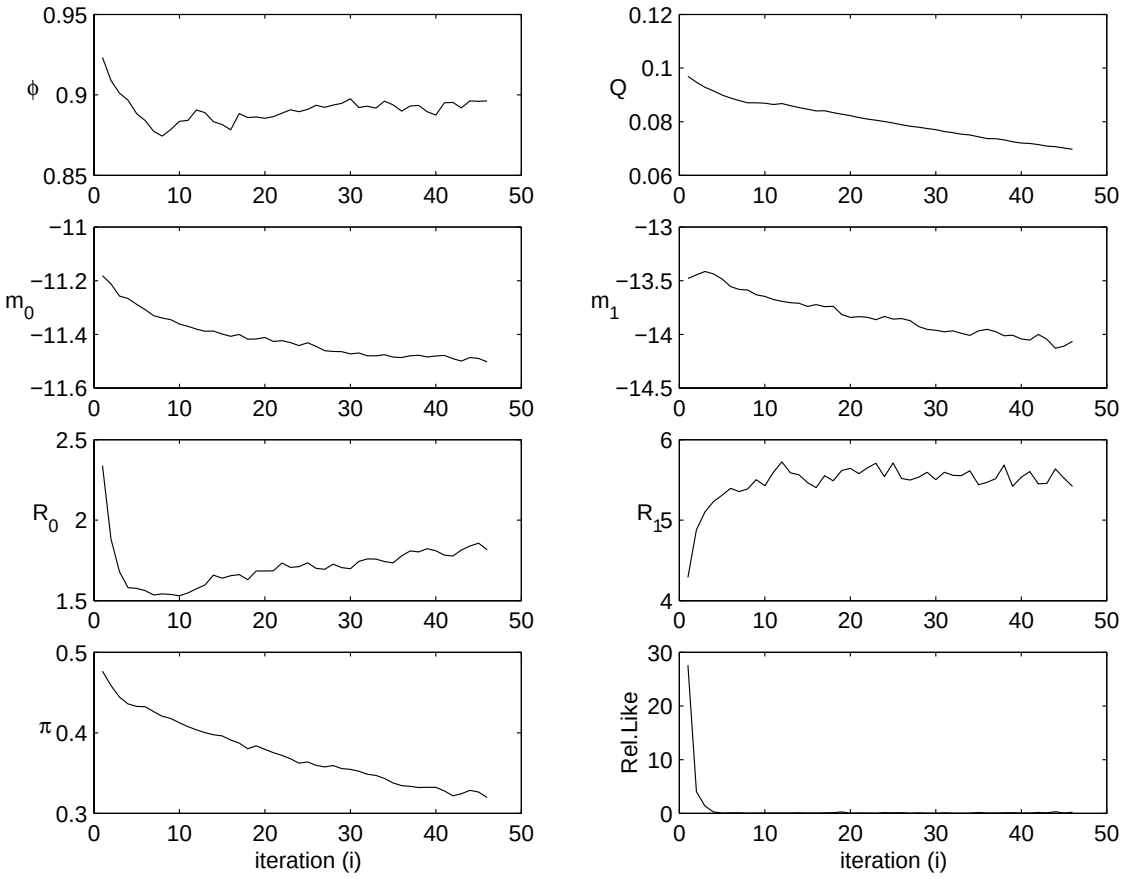

Figure 5.11: Estimation results for the pound/dollar exchange rates with missing values.

6.0 CONCLUSIONS AND FUTURE WORK

6.1 SUMMARY AND CONTRIBUTIONS

This thesis makes several original contributions. The first contribution is to combine the EM algorithm and particle methods in order to provide a new strategy for the parameter estimation problem of SV models. Although the main objective of my thesis is to propose a parameter estimation method for the SV models when some parts of data are missing, I start with developing a method for non-missing data cases and modify the method in order to be able to handle missing data cases.

The EM algorithm makes it possible to get maximum likelihood estimators without struggling with the likelihood of the data, which can't be expressed in a closed form for the SV model. In Chapter 2, I complete the estimation algorithm by applying the particle smoothing algorithm of Godsill et al. (2004), to the SV model with (1.1) and (1.3) as an observation equation and a state equation.

In Chapter 5, I validate this suggested algorithm by applying it to simulated data sets; the estimation result in Section 5.1.1.1 shows that the method presented in Chapter 2 gives a satisfactory result for the data generated from the assumed log of χ^2 distribution. However, in Section 5.1.2.1, it is observed that when the normal assumption in (1.2) is violated (when the data is not generated from the assumed log of χ^2 distribution), the proposed method in Chapter 2 may result in poor estimates, and this is a problem common to all likelihood or posterior based estimation methods which adhere to the normal assumption in (1.2), such as Jacquier et al. (1994). My effort to solve this problem leads to minor modification of the SV model, which is the next contribution of my thesis.

The second contribution of my thesis is to expand the scope of application of the SV

models by adopting normal mixtures for the observation noise in (1.3) and estimating related parameters from the given data. In Chapter 3, I present the modified SV model and the algorithm for this case. Simply speaking, I change the observation error from the log of χ^2 distribution to a more comprehensive distribution, a mixture distribution with two normal components.

In Chapter 5, I apply this algorithm to two simulated data sets. The estimation results show that this modified algorithm yields acceptable results when the normal assumption in (1.2) is violated (See Section 5.1.2.2), as well as when the normal assumption holds (See Section 5.1.1.2). Hence, this modification solves the problem addressed previously.

The idea to use normal mixtures for the observation noise has already been introduced by Kim et al. (1998). However, they use this idea for the sake of convenience in the sampling procedure in MCMC setting, and so they fix all the parameters related to the normal mixture distributions. They use 7 normal mixtures which approximate the logarithm of χ^2_1. The normal mixture idea presented in this thesis is different from their idea since it allows related parameters to be estimated from the data, enabling application of the proposed algorithm to cases where the observation noise in (1.3) is not $\log(\chi^2_1)$.

Last but not least, this thesis presents a parameter estimation method for the SV model with missing data. In Chapter 4, the missing data problem has been addressed and solved by using the property of the state-space models and imputation. Furthermore, the algorithm for the SV model with missing data, which is a slight modification of the algorithm in Chapter 3, is presented. It is well known that state-space models have strength when dealing with missing data.

This missing data problem can also be taken care of by Bayesian methods which sample from conditional distribution of missing data r_t^*, given all other parameters and state variables. In a Bayesian setting, missing data are considered as unknown parameters, so if there is a lot of missing data their methods are less effective. The method proposed in this thesis is easy to understand and to apply, because estimation results can be obtained by a slight modification of the previously suggested algorithm for non-missing data cases.

In Chapter 5, this modified algorithm is tested for simulated and real data sets. For the simulated data sets, I randomly erase 10% or 20% of the data sets and apply the algorithm

to them. As you can see in Section 5.1.1.3 and Section 5.1.2.3, this method gives successful results when some parts of data sets are missing. Also, the method works well without any problem for the real data set.

6.2 FUTURE WORK

One drawback of this suggested method is computational speed. Performing one iteration of the parameter estimation procedure involves generating 2×M samples for particle filters and smoothers and calculating several quantities to obtain the expected likelihood. Also, a common criticism of the EM algorithm is that its convergence can be quite slow.

In order to speed up the process, I try to start with good initial values by using the method of moments idea in Sections 2.4.1 and 3.3.1. Also, I use a small number of particles at first and increase particle size later. In spite of these efforts, it takes a long time to achieve convergence in practice. I think that speed and accuracy are two goals of process, which somewhat conflict with each other; still I want to study how to speed up my algorithm in a near future.

Throughout this thesis, I only considered univariate SV models. Now, the natural next step is to extend the method for univariate SV models to multivariate SV models. In a real world, multivariate data is available in many cases, and so the extension will be useful.

In my thesis work, I focus only on SV models. However, the proposed technique can be used for general non-linear non-Gaussian state-space models, whose scope is very wide. They have been used to explain data from many areas, such as economics and medicine. Hence, I will apply the proposed method to other state-space models.

Last, I want to consider one minor thing. When I consider real data sets, I find that one missing exchange rate yields two missing log returns. In this thesis, I consider both log returns as missing values, but in this case, there is loss of information; for example, when the exchange rate at time t, e_t, is missing, r_t and r_{t+1} is missing. Here, the information that $r_{t+1} + r_t = \log e_{t+1} - \log e_{t-1}$ is not used. So, I want to consider the model which can make use of this information.

APPENDIX A

NOTATION

I list the notation and meanings.

Data

- e_t: exchange rate (raw data)
- r_t: log returns of raw data
- σ_t^2: variance of r_t
- y_t: transformed data, $y_t = \log(r_t^2)$
- x_t: unknown state variable (volatility) at time t
- I_t: indicator variable which has the information about which normal y_t comes from
- v_t: observation noise (or observation error) in state-space models
- w_t: state noise (or state error) in state-space models
- a_t: missing indicator (1 if observed, 0 otherwise)
- n: length of data

Filters and Smoothers

- $\mathrm{f}_t^{(j)}$: jth particle filter of x_t
- $\mathrm{p}_t^{(j)}$: jth particle predictor of x_t
- $\mathrm{s}_t^{(j)}$: jth particle smoother of x_t
- $\tilde{\mathrm{f}}_t^{(j)}$: jth particle filter of I_t
- $\tilde{\mathrm{p}}_t^{(j)}$: jth particle predictor of I_t
- $\tilde{\mathrm{s}}_t^{(j)}$: jth particle smoother of I_t

- $\mathrm{w}_t^{(j)}$: a weight associated with $\mathrm{f}_t^{(j)}$ (Chapter 2) or $(\mathrm{f}_t^{(j)}, \tilde{\mathrm{f}}_t^{(j)})$ (Chapter 3 and 4) in filtering step
- $\mathrm{w}_{t|t+1}^{(j)}$: a weight associated with $\mathrm{s}_t^{(j)}$ (Chapter 2) or $(\mathrm{s}_t^{(j)}, \tilde{\mathrm{s}}_t^{(j)})$ (Chapter 3 and 4) in smoothing step
- x_t^s: $= E(x_t|Y_s)$
- $P_{t1,t2}^s$: $= E\{(x_{t1} - x_{t1}^s)(x_{t2} - x_{t2}^s)|Y_s\}$
- P_t^s: $= P_{t,t}^s$

Model structures and Algorithms

- Model A: linearized SV model

$$\begin{aligned} x_t &= \phi x_{t-1} + w_t, \\ y_t &= \alpha + x_t + v_t, \end{aligned}$$

  where $w_t \sim N(0, Q)$ and $v_t \sim \log(\chi_1^2) - E(\log(\chi_1^2))$.
- Model B: SV model with normal mixtures as its observation error

$$\begin{aligned} x_t &= \phi x_{t-1} + w_t, \\ y_t &= x_t + v_t, \end{aligned}$$

  where $w_t \sim N(0, Q)$, $v_t = I_t z_{t1} + (1 - I_t) z_{t0},\ I_t \sim Bernoulli(\pi)$ and $z_{ti} \sim N(m_i, R_i)$.
- SV model for missing data case:

$$\begin{aligned} x_t &= \phi x_{t-1} + w_t, \\ y_t &= a_t x_t + v_t, \end{aligned}$$

  where $w_t \sim N(0, Q)$, $v_t = I_t z_{t1} + (1 - I_t) z_{t0},\ I_t \sim Bernoulli(\pi)$, $z_{ti} \sim N(m_i, R_i)$ and a_t is missing indicator.
- Algorithm A: Parameter estimation algorithm based on the assumptions of Model A, algorithm in Chapter 2
- Algorithm B: Parameter estimation algorithm based on the assumptions of Model B, algorithm in Chapter 3

- Algorithm for missing data cases: modified version of algorithm B which is presented in Chapter 4

Others

- Y_t: $= \{y_1, \cdots, y_t\}$
- $f(\cdot), f(\cdot|\cdot)$: density function, conditional density function
- $p(\cdot), p(\cdot|\cdot)$: density function, conditional density function
- $q(\cdot|\cdot)$: importance density
- M: number of particles
- γ^*: $=E(\log(\chi_1^2)) \approx 1.2749$

APPENDIX B

CALCULATIONS FOR THE STANDARD DEVIATION

B.1 ALGORITHM A FOR CHAPTER 2

The complete log-likelihood is

$$\begin{aligned}\log f(X,Y) &= -\frac{1}{2}\left\{\log\sigma_0^2+\frac{(x_0-\mu_0)^2}{\sigma_0^2}\right\}-\frac{1}{2}\left\{n\log Q+\sum_{t=1}^{n}\frac{(x_t-\phi x_{t-1})^2}{Q}\right\}\\ &\quad -\frac{1}{2}\left[\sum_{t=1}^{n}\{\exp(y_t-x_t-\alpha-\gamma^*)-(y_t-x_t-\alpha-\gamma^*)\}\right]+C_2,\end{aligned}$$

where C_2 is constant and $\gamma^* = E(\log(\chi_1^2)) \approx 1.2729$. So the first partial derivatives of the complete log-likelihood are

$$\frac{\partial \log f}{\partial \phi} = \sum_{t=1}^{n}\frac{x_{t-1}(x_t-\phi x_{t-1})}{Q}, \tag{B.1}$$

$$\frac{\partial \log f}{\partial Q} = -\frac{1}{2}\sum_{t=1}^{n}\left(\frac{1}{Q}-\frac{(x_t-\phi x_{t-1})^2}{Q^2}\right), \tag{B.2}$$

$$\frac{\partial \log f}{\partial \alpha} = \frac{1}{2}\sum_{t=1}^{n}\{\exp(y_t-x_t-\alpha-\gamma^*)-1\}. \tag{B.3}$$

Now, the second partial derivatives of the complete log-likelihood are

$$\frac{\partial^2 \log f}{\partial \phi^2} = -\frac{\sum_{t=1}^{n} x_{t-1}^2}{Q}, \tag{B.4}$$

$$\frac{\partial^2 \log f}{\partial Q^2} = -\frac{\sum_{t=1}^{n} (x_t - \phi x_{t-1})^2}{Q^3} + \frac{n}{2Q^2}, \tag{B.5}$$

$$\frac{\partial^2 \log f}{\partial \phi \partial Q} = -\sum_{t=1}^{n} \frac{x_{t-1}(x_t - \phi x_{t-1})}{Q^2}, \tag{B.6}$$

$$\frac{\partial^2 \log f}{\partial \alpha^2} = -\frac{1}{2}\sum_{t=1}^{n} \{\exp(y_t - x_t - \alpha - \gamma^*)\}, \tag{B.7}$$

$$\frac{\partial^2 \log f}{\partial \phi \partial \alpha} = 0, \tag{B.8}$$

$$\frac{\partial^2 \log f}{\partial Q \partial \alpha} = 0. \tag{B.9}$$

Hence,

$$\frac{\partial^2 \log f}{\partial \theta^2} = \begin{pmatrix} \frac{\partial^2 \log f}{\partial \phi^2} & \frac{\partial^2 \log f}{\partial \phi \partial Q} & \frac{\partial^2 \log f}{\partial \phi \partial \alpha} \\ symm. & \frac{\partial^2 \log f}{\partial Q^2} & \frac{\partial^2 \log f}{\partial Q \partial \alpha} \\ & & \frac{\partial^2 \log f}{\partial \alpha^2} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 \log f}{\partial \phi^2} & \frac{\partial^2 \log f}{\partial \phi \partial Q} & 0 \\ & \frac{\partial^2 \log f}{\partial Q^2} & 0 \\ & & \frac{\partial^2 \log f}{\partial \alpha^2} \end{pmatrix}.$$

So, the information matrix can be derived by (2.36) and the variance-covariance matrix can be obtained by taking the inverse of it.

B.2 ALGORITHM B FOR CHAPTER 3

The complete log-likelihood is

$$\begin{aligned} \log f(X, Y|\theta) \propto & -\frac{1}{2}\sum_{t=1}^{n}\left[\log Q + \frac{(x_t - \phi x_{t-1})^2}{Q}\right] \\ & + \sum_{t=1}^{n}[I_t \log \pi + (1 - I_t)\log(1 - \pi)] \\ & -\frac{1}{2}\sum_{t=1}^{n}\left[(1 - I_t)\left\{\log R_0 + \frac{(y_t - x_t - m_0)^2}{R_0}\right\}\right. \\ & \left. + I_t\left\{\log R_1 + \frac{(y_t - x_t - m_1)^2}{R_1}\right\}\right]. \end{aligned} \tag{B.10}$$

So, the first partial derivatives of complete log-likelihood are

$$\frac{\partial \log f}{\partial \phi} = \sum_{t=1}^{n} \frac{x_{t-1}(x_t - \phi x_{t-1})}{Q}, \tag{B.11}$$

$$\frac{\partial \log f}{\partial Q} = -\frac{1}{2}\sum_{t=1}^{n}\left(\frac{1}{Q} - \frac{(x_t - \phi x_{t-1})^2}{Q^2}\right), \tag{B.12}$$

$$\frac{\partial \log f}{\partial m_0} = \sum_{t=1}^{n} \frac{(1-I_t)(y_t - x_t - m_0)}{R_0}, \tag{B.13}$$

$$\frac{\partial \log f}{\partial m_1} = \sum_{t=1}^{n} \frac{I_t(y_t - x_t - m_1)}{R_1}, \tag{B.14}$$

$$\frac{\partial \log f}{\partial R_0} = -\frac{1}{2}\sum_{t=1}^{n}(1-I_t)\left\{\frac{1}{R_0} - \frac{(y_t - x_t - m_0)^2}{R_0^2}\right\}, \tag{B.15}$$

$$\frac{\partial \log f}{\partial R_1} = -\frac{1}{2}\sum_{t=1}^{n} I_t\left\{\frac{1}{R_1} - \frac{(y_t - x_t - m_1)^2}{R_1^2}\right\}, \tag{B.16}$$

$$\frac{\partial \log f}{\partial \pi} = \frac{\sum_{t=1}^{n} I_t}{\pi} - \frac{\sum_{t=1}^{n}(1-I_t)}{1-\pi}. \tag{B.17}$$

Now, the second partial derivatives of the complete log-likelihood are

$$\frac{\partial^2 \log f}{\partial \phi^2} = -\frac{\sum_{t=1}^{n} x_{t-1}^2}{Q}, \tag{B.18}$$

$$\frac{\partial^2 \log f}{\partial Q^2} = -\frac{\sum_{t=1}^{n} (x_t - \phi x_{t-1})^2}{Q^3} + \frac{n}{2Q^2}, \tag{B.19}$$

$$\frac{\partial^2 \log f}{\partial \phi \partial Q} = -\sum_{t=1}^{n} \frac{x_{t-1}(x_t - \phi x_{t-1})}{Q^2}, \tag{B.20}$$

$$\frac{\partial^2 \log f}{\partial m_0^2} = -\sum_{t=1}^{n} \frac{(1 - I_t)}{R_0}, \tag{B.21}$$

$$\frac{\partial^2 \log f}{\partial m_1^2} = -\sum_{t=1}^{n} \frac{I_t}{R_1}, \tag{B.22}$$

$$\frac{\partial^2 \log f}{\partial R_0^2} = \frac{1}{2} \sum_{t=1}^{n} (1 - I_t) \left\{ \frac{1}{R_0^2} - \frac{2(y_t - x_t - m_0)^2}{R_0^3} \right\}, \tag{B.23}$$

$$\frac{\partial^2 \log f}{\partial R_1^2} = \frac{1}{2} \sum_{t=1}^{n} I_t \left\{ \frac{1}{R_1^2} - \frac{2(y_t - x_t - m_1)^2}{R_1^3} \right\}, \tag{B.24}$$

$$\frac{\partial^2 \log f}{\partial m_0 \partial R_0} = -\sum_{t=1}^{n} \frac{(1 - I_t)(y_t - x_t - m_0)}{R_0^2}, \tag{B.25}$$

$$\frac{\partial^2 \log f}{\partial m_1 \partial R_1} = -\sum_{t=1}^{n} \frac{I_t(y_t - x_t - m_1)}{R_1^2}, \tag{B.26}$$

$$\frac{\partial^2 \log f}{\partial \pi^2} = -\frac{\sum_{t=1}^{n} I_t}{\pi^2} - \frac{\sum_{t=1}^{n} (1 - I_t)}{(1 - \pi)^2}. \tag{B.27}$$

Hence,

$$\frac{\partial^2 \log f}{\partial \theta^2} = \begin{pmatrix} \frac{\partial^2 \log f}{\partial \phi^2} & \frac{\partial^2 \log f}{\partial \phi \partial Q} & 0 & 0 & 0 & 0 & 0 \\ & \frac{\partial^2 \log f}{\partial Q^2} & 0 & 0 & 0 & 0 & 0 \\ & & \frac{\partial^2 \log f}{\partial m_0^2} & 0 & \frac{\partial^2 \log f}{\partial m_0 \partial R_0} & 0 & 0 \\ & & & \frac{\partial^2 \log f}{\partial m_1^2} & 0 & \frac{\partial^2 \log f}{\partial m_1 \partial R_1} & 0 \\ symm. & & & & \frac{\partial^2 \log f}{\partial R_0^2} & 0 & 0 \\ & & & & & \frac{\partial^2 \log f}{\partial R_1^2} & 0 \\ & & & & & & \frac{\partial^2 \log f}{\partial \pi^2} \end{pmatrix}.$$

Therefore, the information matrix can be derived by (2.36) and the variance-covariance matrix can be obtained by taking the inverse of it.

APPENDIX C

PRACTICAL PROBLEM IN CALCULATION OF STANDARD DEVIATIONS

C.1 PRACTICAL PROBLEMS AND A POSSIBLE SOLUTION

In principle, standard deviations of parameter estimates can be obtained by using (2.36). However, when I applied (2.36) to simulated data and real data sets, I encountered a practical problem; the information matrices are not positive definite. Because the expression (2.36) is the summation of three terms, the property of positive definite matrices can not be guaranteed. Table C.1 contains the variance-covariance matrices at θ from 2 different tries for Data A[1]. As you can see, several diagonal elements of variance-covariance matrices are negative. When I examined the results term by term, it seemed that the second term in (2.36) might be the source of problem [2]. It is observed that the second term in (2.36) varies more than the other two terms when I simulated particles and calcuated each terms given

[1] $\theta = (0.9077, 1.0180, -1.3622, -4.1971, 1.8067, 9.1332, 0.3160)$, $M = 1000$.

[2]

$$\begin{aligned}
\text{Term1} &= \frac{1}{M}\sum_{i=1}^{M} -\frac{\partial^2}{\partial\theta\partial\theta'}\log f(X_i, Y|\theta) \\
\text{Term2} &= -\frac{1}{M}\sum_{i=1}^{M}\left(\frac{\partial}{\partial\theta}\log f(X_i, Y|\theta)\right)\left(\frac{\partial}{\partial\theta'}\log f(X_i, Y|\theta)\right) \\
\text{Term3} &= \left(\frac{1}{M}\sum_{i=1}^{M}\frac{\partial}{\partial\theta}\log f(X_i, Y|\theta)\right)
\end{aligned}$$

same parameters repeatedly. Figure C.1 contains the histograms of some elements of $D^{(i)}$, where $D^{(i)} = \left(\frac{\partial}{\partial\theta}\log f(X_i, Y|\theta)\right)\left(\frac{\partial}{\partial\theta'}\log f(X_i, Y|\theta)\right)$, which are calculated from 1000 paths of particle smoothers (Note that the second term of (2.36) is the sample mean of $\{D^{(i)}\}_{i=1}^{M}$). There exist several outliers and those outliers have big influence on the sample means, which are the elements of the second term in (2.36). Hence, I suggest to use the trimmed mean instead of the sample mean for the elements of the second term to solve this problem; use $\tilde{D}_{jk} = \frac{1}{M(1-\gamma)}\sum_{i=[M\gamma/2]}^{[M(1-\gamma/2)]} D_{jk}^{(i*)}$, where $D_{jk}^{(i*)}$ is ith smallest in $\{D_{jk}^{(i)}\}_{i=1}^{M}$ and γ is the proportion of trimming ($0 \le \gamma \le 1$). By eliminating 5% of the $D_{jk}^{(i)}$ (2.5% at each side), I could get samples without outliers and a positive definite variance-covariance matrix in the end. Table C.2 shows the variance-covariance matrices calculated by using trimming idea based on the same particles used to get Table C.1 ; they are positive definite matrices.

C.2 EXAMPLE : SIMPLE STATE-SPACE MODEL

To justify my trimming idea, I applied this idea to a simpler model where the result for the variance-covariance matrix is known. I sampled time series of size 1000 from the following simple state-space model:

$$x_t = \phi x_{t-1} + w_t, \tag{C.1}$$

$$y_t = x_t + v_t, \tag{C.2}$$

where, $w_t \sim N(0, Q)$ and $v_t \sim N(0, R)$ and $(\phi, Q, R) = (0.8, 1, 1.5)$.

Maximum likelihood estimate was obtained by the Newton–Rhapson method and it was (0.8420, 0.7005, 1.4883). The variance-covariance matrix at MLE, which is the inverse of the Hessian matrix, was

$$\hat{\mathrm{Var}}(\hat{\theta}) = \begin{pmatrix} 0.0004 & -0.0013 & 0.0009 \\ -0.0013 & 0.0081 & -0.0056 \\ 0.0009 & -0.0056 & 0.0084 \end{pmatrix}^{3}, \tag{C.3}$$

[3] See Section 4.3 of Shumway and Stoffer (2000).

Table C.1: Variance-covariance matrices for Data A (2 tries)

1	Var-cov	$\begin{pmatrix} 0.0002 & -0.0004 & 0.0003 & 0.0006 & 0.0020 & -0.0015 & -0.0016 \\ -0.0004 & 0.0093 & -0.0049 & -0.0045 & -0.0226 & 0.0184 & 0.0203 \\ 0.0003 & -0.0049 & -0.0004 & -0.0005 & 0.0022 & 0.0123 & 0.0209 \\ 0.0006 & -0.0045 & -0.0005 & 0.0221 & 0.0225 & 0.0121 & 0.0114 \\ 0.0020 & -0.0226 & 0.0022 & 0.0225 & 0.1475 & 0.0548 & 0.2367 \\ -0.0015 & 0.0184 & 0.0123 & 0.0121 & 0.0548 & -0.0597 & -0.1127 \\ -0.0016 & 0.0203 & 0.0209 & 0.0114 & 0.2367 & -0.1127 & 1.0528 \end{pmatrix}$
	term1	$1.0e+003 * \begin{pmatrix} -5.7275 & 0.0025 & 0 & 0 & 0 & 0 & 0 \\ 0.0025 & -0.4940 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4.6066 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.3800 & 0 & 0.0022 & 0 \\ 0 & 0 & 0 & 0 & -0.0343 & 0 & 0.0001 \\ 0 & 0 & 0 & 0.0022 & 0 & -0.1043 & 0 \\ 0 & 0 & 0 & 0 & 0.0001 & 0 & -0.0019 \end{pmatrix}$
	term2	$1.0e+003 * \begin{pmatrix} 0.8057 & -0.3054 & -0.1280 & 0.0511 & 0.0128 & 0.0215 & 0.0004 \\ -0.3054 & 0.4405 & -0.1305 & -0.0233 & -0.0024 & -0.0318 & -0.0007 \\ -0.1280 & -0.1305 & 3.7740 & 0.0133 & 0.0896 & -0.1207 & -0.0118 \\ 0.0511 & -0.0233 & 0.0133 & 0.3384 & 0.0173 & -0.0026 & -0.0023 \\ 0.0128 & -0.0024 & 0.0896 & 0.0173 & 0.0215 & 0.0044 & 0.0016 \\ 0.0215 & -0.0318 & -0.1207 & -0.0026 & 0.0044 & 0.0877 & 0.0004 \\ 0.0004 & -0.0007 & -0.0118 & -0.0023 & 0.0016 & 0.0004 & 0.0009 \end{pmatrix}$
	term3	$\begin{pmatrix} -2.4981 & 5.8551 & -11.8632 & -4.0242 & -0.9577 & -0.7992 & 0.0769 \end{pmatrix}$
2	Var-cov	$\begin{pmatrix} 0.0003 & -0.0016 & -0.0001 & -0.0005 & -0.0009 & 0.0012 & 0.0027 \\ -0.0016 & 0.0257 & -0.0001 & 0.0097 & 0.0117 & -0.0109 & -0.0260 \\ -0.0001 & -0.0001 & -0.0010 & -0.0075 & -0.0156 & 0.0088 & -0.0070 \\ -0.0005 & 0.0097 & -0.0075 & 0.0093 & -0.0384 & 0.0300 & -0.0169 \\ -0.0009 & 0.0117 & -0.0156 & -0.0384 & -0.0405 & 0.1032 & 0.1069 \\ 0.0012 & -0.0109 & 0.0088 & 0.0300 & 0.1032 & 0.0003 & 0.0441 \\ 0.0027 & -0.0260 & -0.0070 & -0.0169 & 0.1069 & 0.0441 & 1.2811 \end{pmatrix}$
	term1	$1.0e+003 * \begin{pmatrix} -5.7216 & 0.0032 & 0 & 0 & 0 & 0 & 0 \\ 0.0032 & -0.4943 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4.6037 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.3802 & 0 & 0.0006 & 0 \\ 0 & 0 & 0 & 0 & -0.0343 & 0 & 0.0000 \\ 0 & 0 & 0 & 0.0006 & 0 & -0.1047 & 0 \\ 0 & 0 & 0 & 0 & 0.0000 & 0 & -0.0019 \end{pmatrix}$
	term2	$1.0e+003 * \begin{pmatrix} 0.8960 & -0.3544 & -0.0645 & 0.0545 & 0.0137 & 0.0017 & 0.0025 \\ -0.3544 & 0.4625 & -0.1808 & -0.0015 & 0.0019 & -0.0081 & -0.0010 \\ -0.0645 & -0.1808 & 3.7585 & -0.0133 & 0.0976 & -0.1007 & -0.0125 \\ 0.0545 & -0.0015 & -0.0133 & 0.3427 & 0.0146 & -0.0117 & -0.0015 \\ 0.0137 & 0.0019 & 0.0976 & 0.0146 & 0.0219 & 0.0027 & 0.0016 \\ 0.0017 & -0.0081 & -0.1007 & -0.0117 & 0.0027 & 0.0865 & -0.0004 \\ 0.0025 & -0.0010 & -0.0125 & -0.0015 & 0.0016 & -0.0004 & 0.0009 \end{pmatrix}$
	term3	$\begin{pmatrix} -3.2987 & 5.9995 & -13.5751 & -1.0588 & -0.3848 & -0.4627 & 0.0487 \end{pmatrix}$

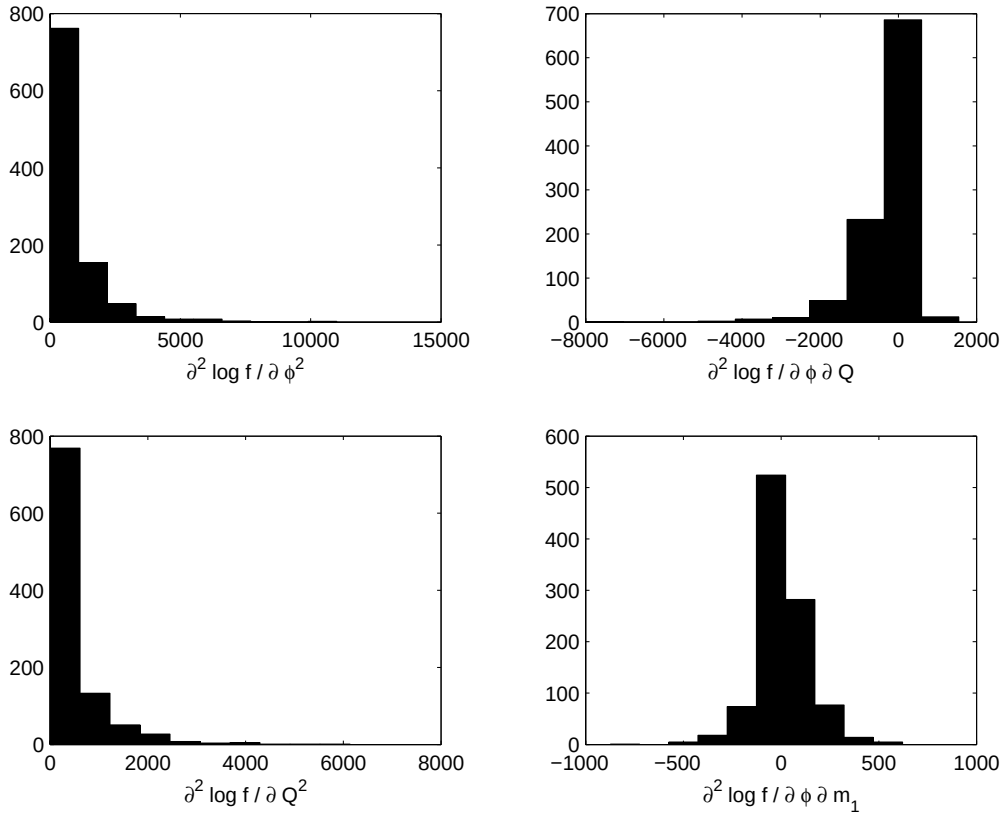

Figure C.1: Histogram of Term2: M=1000.

Table C.2: Variance-covariance matrices calculated using trimmed mean

1	$\begin{pmatrix} 0.0002 & -0.0005 & -0.0001 & 0.0001 & 0.0002 & 0.0009 & 0.0016 \\ -0.0005 & 0.0090 & 0.0007 & -0.0001 & 0.0001 & -0.0110 & -0.0185 \\ -0.0001 & 0.0007 & 0.0013 & -0.0001 & 0.0035 & -0.0054 & -0.0076 \\ 0.0001 & -0.0001 & -0.0001 & 0.0110 & 0.0068 & 0.0003 & -0.0072 \\ 0.0002 & 0.0001 & 0.0035 & 0.0068 & 0.0908 & -0.0018 & 0.0865 \\ 0.0009 & -0.0110 & -0.0054 & 0.0003 & -0.0018 & 0.0683 & 0.0718 \\ 0.0016 & -0.0185 & -0.0076 & -0.0072 & 0.0865 & 0.0718 & 1.1160 \end{pmatrix}$
2	$\begin{pmatrix} 0.0002 & -0.0005 & 0.0000 & 0.0001 & 0.0002 & -0.0000 & 0.0011 \\ -0.0005 & 0.0089 & -0.0003 & 0.0008 & 0.0008 & -0.0003 & -0.0071 \\ 0.0000 & -0.0003 & 0.0011 & 0.0009 & 0.0054 & -0.0033 & -0.0035 \\ 0.0001 & 0.0008 & 0.0009 & 0.0151 & 0.0152 & -0.0068 & -0.0037 \\ 0.0002 & 0.0008 & 0.0054 & 0.0152 & 0.1091 & -0.0147 & 0.0731 \\ -0.0000 & -0.0003 & -0.0033 & -0.0068 & -0.0147 & 0.0463 & 0.0079 \\ 0.0011 & -0.0071 & -0.0035 & -0.0037 & 0.0731 & 0.0079 & 1.0132 \end{pmatrix}$

Table C.3: Variance-covariance matrix for Example in Section C.2

no trimming:	5% trimming:
$\begin{pmatrix} 0.0001 & 0.0032 & -0.0020 \\ 0.0032 & -0.0260 & 0.0156 \\ -0.0020 & 0.0156 & -0.0022 \end{pmatrix}$	$\begin{pmatrix} 0.0005 & -0.0008 & 0.0002 \\ -0.0008 & 0.0082 & -0.0024 \\ 0.0002 & -0.0024 & 0.0065 \end{pmatrix}$

where $\theta = (\phi, Q, R)$. I applied my estimation method, which is modified to fit the simple state-space model, to the same data set, and got particle smoothers. Table C.3 contains the variance-covariance matrices calculated by using those particle smoothers. When (2.36) was directly used, I got non-positive definite matrix as you can see in the left side in Table C.3. The result from trimming was a positive definite matrix and close to the in verse of Hessian matrix in (C.3), justifying my trimming idea.

APPENDIX D

MATLAB FUNCTIONS

D.1 SIMULATION OF DATA FROM THE SV MODELS

DATA A

```
function [y,x]=sim_model1(phi,Q,alpha,n)
% randsp : give random sample from the sv model
% y(t)=alpha+a(t)*x(t)+v(t)
% x(t)=phi*x(t-1)+w(t)
% w(t) Gaussian (0,Q)
% v(t) centered log of chisquares(1)
% n=1000; number of observation

w=randn(1,n+10000)*sqrt(Q);
v1=chi2rnd(1,1,n+10000);
cv=log(v1)+1.2749;
x0=randn(1);
x(1)=phi*x0+w(1);
for t=2:n+10000
    x(t)=phi*x(t-1)+w(t);
end
y=alpha+x+cv;
y=y(10001:n+10000);
x=x(10001:n+10000);
```

$[y\_A, x\_A] = sim\_model1(0.9, 1, -3, 1000);$

DATA B

```
function [y,x]=sim_model2(phi,Q,alpha,mu1,R0,R1,p,n)
% sim_modle2 : give stationary random sample from the sv model(2): normal mixtures
% y(t)=alpha+x(t)+v(t)
% x(t)=phi*x(t-1)+w(t)
% w(t) Gaussian (0,Q)
% v(t) mixtures of two normals N(mu1,R1),N(mu0,R0) mixing prob=p

w=randn(1,n+10000)*sqrt(Q);
```

```
Ind=(rand(1,n+10000)<p);
nor1=randn(1,n+10000);
v1=Ind.*(nor1*sqrt(R1)+mu1)+(1-Ind).*(nor1*sqrt(R0));

x0=randn(1);
x(1)=phi*x0+w(1);
for t=2:n+10000
    x(t)=phi*x(t-1)+w(t);
end
y=alpha+x+v1;
y=y(10001:n+10000);
x=x(10001:n+10000);
```

$[y\_B, x\_B] = sim\_model2(0.8, 1.5, -4, -3, 3, 5, 0.5, 1000);$

D.2 ALGORITHM A: PARAMETER ESTIMATION FOR CHAPTER 2

Initial parameter selection

```
function Iparm=Iparm_c(y)

%parm=[phi, Q,alpha];
n=max(size(y));

parm(3)=mean(y);
y2=y(1:n-2);
y1=y(2:n-1);
y0=y(3:n);
A=cov(y0,y2);
B=cov(y1,y2);
parm(1)= min(A(1,2)/B(1,2),0.95);
parm(2)=max(mean(( (y0-mean(y0))-parm(1)*(y1-mean(y1))).^2) - 5 -5*parm(1)^2,0.1);

Iparm=parm
```

Main function

```
function [Eparm,Vparm,RelLike,Tcal,Niter]=mainchis(y,a,Iparm,Tol,Miter,Npar,n,tt)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% main fuction for the estimation of parameters for regular sv model with missing values
% model: y(t)=alpha+a(t)x(t)+v(t) v(t):centered log of chisquares
%           x(t)=phi*x(t-1)+w(t) w(t):normal(0, Q) x(0):normal(mu0,Sig0)
%parameters: theta=[phi, Q, alpha]
%input values -- y: log(r^2) where r is returns,
%                a: missing indicator ( 1= observed, 0=missing),
%                Tol: tolerance, Miter=maximum iteration, Npar=number of particles.
%output values -- Eparm : History of parameter estimates [phi,Q, alpha],
%                 Vparm: var-cov matrix of parameter estimates
%                 RelLike: relative likelihood
%                   : loglikelihood at iteration i- log likelihood at iteration i-1
%                 Tcal: time for the whole calculation
```

```
%                  Niter: number of iteration until convergence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tic  %to get the time to elapse
% declaration of variables--------------
Eparm=zeros(Miter, 3);
RelLike=zeros(Miter,1);
Tcal=0;

%estimation--------------------
for i=1:Miter
      y_full=y;
      %1.filtering step
      f=PfilterChis(Iparm, y_full, a, n, Npar);
      %2. smoothing step
      s=PsmootherChis(y_full, a, f, Iparm, n, Npar);
      %3. estimation step
      Rparm=EstChis(s, y_full, n, Npar)
      Eparm(i,:)=Rparm;

      %4. Relative likelihood
      if i>1,
           RelLike(i)=RelLikeChis(s,n,y,Eparm(i-1,:),Rparm,Npar);
       end
       if i==1
           RelLike(i)=RelLikeChis(s,n,y,Iparm,Rparm,Npar);
       end

       %5. Check convergence
      Con=0;
      if i>tt,
      Con=ConChis(RelLike,i ,Tol );
      end
      if Con==1, Niter=i, break, end
      Iparm=Rparm;

  end

  %6. Variance of parameter estimates
  Vparmm=VarChis(s,y,n,Eparm(i-1,:),Npar)
  Eparm=Eparm(1:i-1,:);
  RelLike=RelLike(1:i-1,:);

  Tcal=toc; %to get the time to elapse
```

1. Filtering function

```
function f=PfilterChis(Iparm, y, a, n, Npar)
% returns filtered values
% Iparm=theta=[ phi, Q, alpha]
%assign parameters
phi=Iparm(1);
Q=Iparm(2);
alpha=Iparm(3);
```

```
mu0=0;
Sig0=Q/(1-phi^2);


f(:,1)=randn(Npar, 1)*sqrt(Sig0)+mu0;
wt=randn(Npar, n)*sqrt(Q);
for t=1:n
    p=phi*f(:,t)+wt(:,t);
    w=exp((y(t)-a(t)*p-1.2749-alpha)/2).*exp(-1*exp(y(t)-a(t)*p-1.2749-alpha)/2);
    f(:,t+1)=Rsample1(p,w,Npar);
end
```

2. Smoothing function

```
function s=PsmootherChis(y, a, f, Iparm, n, Npar)
%particle smoother :
%Iparm=theta=[ phi, Q, alpha]
phi=Iparm(1);
Q=Iparm(2);
alpha=Iparm(3);
mu0=0;
Sig0=Q/(1-phi^2);

%for t=n
st=unidrnd(Npar,1,Npar);
s(:,n+1)=f(st,n+1);

for j=1:Npar
    for t=n:-1:1
        w=exp(-1*(s(j,t+1)-phi*f(:,t)).^2/(2*Q));
        s(j,t)=RSample1(f(:,t),w,1);
    end
end
```

3. Estimation function

```
 function Rparm=EstChis(s, y, n,Npar)

 %theta=[ phi, Q, alpha]

 phi=sum(mean(s(:,1:n).*s(:,2:n+1)))/sum(mean(s(:,1:n).^2));
 Q=mean(mean((s(:,2:n+1)-phi*s(:,1:n)).^2));

 y_ext=repmat(y, Npar, 1);
 alpha=log(mean(mean(exp(y_ext-s(:,2:n+1)-1.2749))));

 Rparm=[phi,Q,alpha];
```

4. Relative likelihood

```
function Rel_Like=RelLikeChis(s,n,y,oldp,newp,m )

temp=0;
```

```
for j=1:m
    loglike1=comp_loglike(s(j,:),y,oldp,n);
    loglike2=comp_loglike(s(j,:),y,newp,n);
    m_log=(loglike1+loglike2)/2;
    temp=temp+exp(loglike1-m_log)/exp(loglike2-m_log);
end

Rel_Like=-1*log(temp/m)
```

5. Assessing convergence

```
function Con=ConChis(RelLike,i ,Tol )
Con=0;
if abs(RelLike(i))<Tol,
    Con=1;
end
```

6. Variance of parameter estimates

```
function [vparmm,vparmt,term1,term2,term2_trim,term3a]=VarChis_trim(s,y,theta,ptrim)
% theta=[phi, Q, alpha]
phi=theta(1);
Q=theta(2);
alpha=theta(3);
[Npar,n]=size(s);
n=n-1;


term1=zeros(3,3);
term2=zeros(3,3);
term3a=zeros(3,1);
ddl=zeros(3,3);
dl=zeros(3,1);
term2t=zeros(3,Npar);

for i=1:Npar
    x1=s(i,:);
    dl(1)=sum(x1(1:n).*(x1(2:n+1)-phi*x1(1:n)))/Q;
    dl(2)=-1*n/(2*Q)+sum(((x1(2:n+1)-phi*x1(1:n)).^2))/(2*Q^2);
    dl(3)=0.5*sum(exp(y-x1(2:n+1)-alpha-1.2749)-1);

    term2=term2+dl*dl'/Npar;
    term3a=term3a+dl/Npar;
    term2t(:,i)=dl;

    ddl(1,1)=-1*sum(x1(1:n).^2)/Q;
    dd1(1,2)=-1*sum(x1(1:n).*(x1(2:n+1)-phi*x1(1:n)))/Q^2;
    ddl(2,1)=ddl(1,2);
    ddl(2,2)=-1*sum((x1(2:n+1)-phi*x1(1:n)).^2)/Q^3 +n/(2*Q^2);
    ddl(3,3)=-0.5*sum(exp(y-x1(2:n+1)-alpha-1.2749));

    term1=term1+ddl/Npar;
end
term2_trim=zeros(3,3);
```

```
for i=1:Npar
    kk=term2t(:,i)*term2t(:,i)';
   for j=1:9
        forterm2(i,j)=kk(fix((j-1)/3)+1,rem(j-1,3)+1);
    end
end
term2_temp=trimmean(forterm2,ptrim)

for i=1:3
    for j=1:3
        term2_trim(i,j)=term2_temp((i-1)*3+j);
    end
end

obInfo=-1*term1 -term2 + (term3a)*(term3a)';
obsInfot=-1*term1-term2_trim+(term3a)*(term3a)';
vparmt=inv(obsInfot);
vparmm=inv(obInfo)
```

Miscellaneous functions

Resampling function

```
function Newdata=Rsample1(data,weight,NofSample)
n=max(size(data));
re_ind=rand(1,NofSample);
cmwt=cumsum(weight)/sum(weight);
for k=1:NofSample
    st=(re_ind(k)>cmwt(1:n-1));
    Newdata(k)=data(sum(st)+1);
end
Newdata=Newdata';
```

Complete likelihood

```
function loglike=comp_loglike(x,y,theta,n)

phi=theta(1);
Q=theta(2);
alpha=theta(3);
mu0=0;
Sig0=Q/(1-phi^2);

loglike=-1*(log(Sig0)+(x(1)-mu0)^2/Sig0+n*log(Q)+sum((x(2:n+1)-...
 phi*x(1:n)).^2)/Q+sum(exp(y-x(2:n+1)-alpha-1.2749)-(y-x(2:n+1)-alpha-1.2749)))/2;
```

Data completion function

```
function y_full=completey_c(y,a,init)
% complete y with missing by taking random sample

n=max(size(y));
```

```
for t=1:n
    if a(t)==0
        rdn=randn(1,1)
        y_full(t)=init(3)+log(rdn^2)-1.2749;
    else
        y_full(t)=y(t);
    end
end
```

D.3 ALGORITHM B: PARAMETER ESTIMATION FOR CHAPTER 3 AND CHAPTER 4

Initial parameter selection

```
function parm=Iparmnorm(y)
%parm=[phi, Q,m0,m1,R0,R1,pi];
%m0>m1;
k1=3;
k2=4;
k3=4;
k4=0.5;
n=max(size(y));
y2=y(1:n-2);
y1=y(2:n-1);
y0=y(3:n);
A=cov(y0,y2);
B=cov(y1,y2);
parm(1)= min(A(1,2)/B(1,2),0.95);
parm(3)=mean(y)+k4*k1;
parm(4)=parm(3)-k1;
parm(5)=k2;
parm(6)=k3;
parm(7)=k4;
varofv=k1^2*k4*(1-k4)+k3*k4+k2*(1-k4);
parm(2)=max(mean(( (y0-mean(y0))-parm(1)*(y1-mean(y1))).^2) -...
             varofv-varofv*parm(1)^2,0.1);
```

Main function

```
function [Eparm,Vparmm,Vparmt,RelLike,Tcal,Niter]=...
        mainnorm(y,a,Iparm,Tol,Miter,Npar,n,tt)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% main fuction for the estimation of parameters for regular sv model        %
%                       with missing values with stationary assumption      %
% use auxiliary PF                                                          %
% model: y(t)=alpha+a(t)x(t)+v(t)                                           %
%        v(t):two normal mixtures (1-I(t))*N(m0,R0)+I(t)*N(m1,R1),          %
%        I(t)~bernoulli(pi)                                                 %
%        x(t)=phi*x(t-1)+w(t) w(t):normal(0, Q) x(0):normal(mu0,Sig0)       %
% From stationary assumption: mu0=0, Sig0=Q/(1-phi^2)                       %
%parameters: theta=[phi, Q,m0,m1,R0,R1,pi];                                 %
```

```
%input values -- y: log(r^2) where r is returns,                              %
%                a: missing indicator ( 1= observed, 0=missing),              %
%                Iparm: initial parameter                                     %
%                Tol: tolerance to assess convergence,                        %
%                Miter=maximum number of iteration,                           %
%                Npar=number of particles.                                    %
%output values -- Eparm : Parameter estimates [phi,Q, m0,m1,R0,R1,pi],        %
%                  VParm: var-cov matrix of parmeter estimates                %
%                  RelLike: relative likelihood                               %
%              :loglikelihood at iteration i- loglikelihood at iteration i-1% 
%                  Tcal: time for the whole calculation                       %
%                  Niter:number of iterations until convergence               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic %to get the time to elapse: tic-toc

% declaration of variables--------------
Eparm=zeros(Miter, 7);
RelLike=zeros(Miter,1);
Tcal=0;
Niter=0;
%---------------------------------

%estimation--------------------
for i=1:Miter
      %1.data completion step to fill missing data in.
      if sum(a)<n %if there are missing data
          y_full=DataCompNorm(y,a,Iparm,n);
                %DataCompNorm:function to fill missing data in.
      else
          y_full=y;
      end

      %2.filtering step
      [f_x,f_I]=PfilterNormaux(Iparm, y_full, a, n, Npar);
        %PfilterNormaux: function for auxiliary PF
        %ref:Pitt and Shephard (2001); Sequential MC methods in Practice,
        % Edt'd Doucet et al

      %3. smoothing step
      [s_x,s_I]=PsmootherNorm(y_full, a, f_x,f_I, Iparm, n, Npar);
        %PsmootherNorm: function for Particle smoothing
        %ref: Godsill et al (2004) JASA  vol 99 pp 156-

      %4. estimation step
      Rparm=EstNorm(s_x,s_I, y_full, n, Npar)
        %EstNorm: function for parameter estimation
      Eparm(i,:)=Rparm;

      %5. Relative likelihood
      if i>1,
           RelLike(i)=RelLikeNorm(s_x,s_I,n,y_full,Eparm(i-1,:),Rparm,Npar);
       end
       if i==1
           RelLike(i)=RelLikeNorm(s_x,s_I,n,y_full,Iparm,Rparm,Npar);
```

```
       end

       %6. Check convergence
      Con=0;
      if i>tt,
      [Con,neg]=ConNorm(RelLike,i ,Tol );
      end
      if Con==1, Niter=i, break, end
      Iparm=Rparm;
  end

  %7. variance calculation
  [Vparmm,Vparmt,term1,term2,term2_trim,term3a]=v2_trim(s_x,s_I,Eparm(i-1,:),y_full,5)

  Eparm=Eparm(1:i-1,:);
  RelLike=RelLike(1:i-1,:);

  Tcal=toc; %to get the time to elapse
```

1. Data completion function

```
function y_full=DataCompChis(y,a,Iparm,n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Data completion step for 2 normal mixtures sv model
% parm=[ phi,Q, m0,m1,R0,R1,pi],
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

phi=Iparm(1);
Q=Iparm(2);
m0=Iparm(3);
m1=Iparm(4);
R0=Iparm(5);
R1=Iparm(6);
pi=Iparm(7);

Ind=(rand(1,n)<pi);
rdn=randn(1,n);
y_miss=Ind.*(rdn*sqrt(R1)+m1)+(1-Ind).*(rdn*sqrt(R0)+m0);
y(a==0)=0;
y_full=a.*y+(1-a).*y_miss;
```

2. Filtering function

```
function [f_x,f_I]=PfilterNormaux(Iparm, y, a, n, Npar)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PfilterNormaux returns filtered values based on auxiliary PF: fully adapted.
%Iparm=theta=[phi, Q, m0,m1,R0,R1,pi]
%%ref:Pitt and Shephard (2001); Sequential Monte Carlo methods in Practice,
%        Edt'd Doucet et al
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%assign parameters---
phi=Iparm(1);
Q=Iparm(2);
```

```
m0=Iparm(3);
m1=Iparm(4);
R0=Iparm(5);
R1=Iparm(6);
prob=Iparm(7);
mu0=0;
Sig0=Q/(1-phi^2);
%---------------------

%for t=0
f_x(:,1)=randn(Npar, 1)*sqrt(Sig0)+mu0;
f_I(:,1)=(rand(Npar,1)<prob);
  tttt=repmat(1,1,Npar);
  temp_mat=[tttt,1-tttt;1-tttt,tttt];
  temp_pi=[1-prob,prob];
  temp_pi=temp_pi*temp_mat;
  temp_R=[R0,R1];
  temp_R=temp_R*temp_mat;
  temp_m=[m0,m1];
  temp_m=temp_m*temp_mat;
for t=1:n
    %draw auxiliary variable and normal index(k,j)
    temp_x=repmat(f_x(:,t)',1,2);
    weight1=temp_pi.*exp(-1*(y(t)-temp_m-phi*temp_x).^2./...
              (2*(temp_R+Q)))./sqrt(temp_R+Q);
    index_temp=1:Npar*2;
    index_sampled=Rsample1(index_temp,weight1,Npar);
    aux_sampled=temp_x(index_sampled);
    I_sampled=(index_sampled>Npar)';
    R_f=R0*(1-I_sampled)+R1*I_sampled;
    R_star=(R_f*Q)./(R_f+Q);
    m_star=R_star.*((y(t)-m0*(1-I_sampled)-m1*I_sampled)./R_f+...
           phi*aux_sampled/Q);
    f_x(:,t+1)=(randn(1,Npar).*sqrt(R_star)+m_star)';
    f_I(:,t+1)=I_sampled';
end
```

3. Smoothing function

```
function [s_x,s_I]=PsmootherNorm(y_full, a, f_x,f_I, Iparm, n, Npar)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%particle smoother : returns particle smoother
%Iparm=theta=[phi, Q,m0,m1,R0,R1,pi];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
phi=Iparm(1);
Q=Iparm(2);
m0=Iparm(3);
m1=Iparm(4);
R0=Iparm(5);
R1=Iparm(6);
pi=Iparm(7);
%-----------------------------------
%for t=n
st=unidrnd(Npar,1,Npar);
```

```
s_x(:,n+1)=f_x(st,n+1);
s_I(:,n+1)=f_I(st,n+1);

for j=1:Npar
    for t=n:-1:1
        w=exp(-1*(s_x(j,t+1)-phi*f_x(:,t)).^2/(2*Q)).*(pi*s_I(j,t+1)+...
                        (1-pi)*(1-s_I(j,t+1)));
        [s_x(j,t),s_I(j,t)]=RSample2(f_x(:,t),f_I(:,t),w,1);
    end
end
```

4. Estimation function

```
function Rparm=EstNorm(s_x,s_I, y, n, Npar)
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % Parameter estimation
 %theta=[ phi, Q, m0,m1,R0,R1,pi]
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 phi=sum(mean(s_x(:,1:n).*s_x(:,2:n+1)))/sum(mean(s_x(:,1:n).^2));
 Q=mean(mean((s_x(:,2:n+1)-phi*s_x(:,1:n)).^2));

 y_ext=repmat(y, Npar, 1);
 m0=sum(mean((1-s_I(:,2:n+1)).*(y_ext-s_x(:,2:n+1))))/(n-sum(mean(s_I(:,2:n+1))));
 m1=sum(mean(s_I(:,2:n+1).*(y_ext-s_x(:,2:n+1))))/(sum(mean(s_I(:,2:n+1))));

 R0=sum(mean((1-s_I(:,2:n+1)).*((y_ext-s_x(:,2:n+1)-m0).^2)))/(n-...
                     sum(mean(s_I(:,2:n+1))));
 R1=sum(mean(s_I(:,2:n+1).*((y_ext-s_x(:,2:n+1)-m1).^2)))/...
                     (sum(mean(s_I(:,2:n+1))));

 pi=mean(mean(s_I(:,2:n+1)));

 Rparm=[phi,Q,m0,m1,R0,R1,pi];
```

5. Relative likelihood

```
function Rel_Like=RelLikeNorm(s_x,s_I,n,y,oldp,newp,m )

temp=0;
for j=1:m
    loglike1=comp_logliken(s_x(j,:),s_I(j,:),y,oldp,n);
    loglike2=comp_logliken(s_x(j,:),s_I(j,:),y,newp,n);
    m_log=(loglike1+loglike2)/2;
    temp=temp+exp(loglike1-m_log)/exp(loglike2-m_log);
end

Rel_Like=-1*log(temp/m)
```

6. Assessing convergence

```
function [Con,neg]=ConNorm(RelLike,i ,Tol )
Con=0;
if RelLike(i)<Tol,
```

```
    Con=1;
end
neg=0;
if RelLike(i)<0, disp('RELLIKE NEGATIVE'), neg=1
end
```

7. Variace estimates

```
function [vparmm,vparmt,term1,term2,term2_trim,term3a]=v2_trim(sx,sI,theta,y,ptrim)
phi=theta(1);
Q=theta(2);
m0=theta(3);
m1=theta(4);
R0=theta(5);
R1=theta(6);
prob=theta(7);
[Npar,n]=size(sx);
n=n-1;

term1=zeros(7,7);
term2=zeros(7,7);
term3a=zeros(7,1);
ddl=zeros(7,7);
dl=zeros(7,1);
term2t=zeros(7,Npar);

for i=1:Npar
    x1=sx(i,:);
    I1=sI(i,2:n+1);
    sumI=sum(I1);
    dl(1)=sum(x1(1:n).*(x1(2:n+1)-phi*x1(1:n)))/Q;
    dl(2)=-0.5*(n/Q - sum((x1(2:n+1)-phi*x1(1:n)).^2)/Q^2);
    dl(3)=sumI/prob-(n-sumI)/(1-prob);
    dl(4)=sum( (1-I1).*(y-x1(2:n+1)-m0))/R0;
    dl(5)=sum(I1.*(y-x1(2:n+1)-m1))/R1;
    dl(6)=-0.5*((n-sumI)/R0 - sum((1-I1).*(y-x1(2:n+1)-m0).^2)/R0^2);
    dl(7)=-0.5*(sumI/R1-sum(I1.*(y-x1(2:n+1)-m1).^2)/R1^2);

    term2=term2+dl*dl'/Npar;
    term3a=term3a+dl/Npar;
    term2t(:,i)=dl;

    ddl(1,1)=-sum(x1(1:n).^2)/Q;
    ddl(1,2)=-1*dl(1)/Q;
    ddl(2,1)=ddl(1,2);
    ddl(2,2)=n/(2*Q^2)-sum((x1(2:n+1)-phi*x1(1:n)).^2)/Q^3;
    ddl(3,3)=-1*sumI/prob^2-(n-sumI)/(1-prob)^2;
    ddl(4,4)=-1*(n-sumI)/R0;
    ddl(5,5)=-1*sumI/R1;
    ddl(6,6)=(n-sumI)/(2*R0^2)-sum((1-I1).*((y-x1(2:n+1)-m0).^2))/R0^3;
    ddl(7,7)=sumI/(2*R1^2)-sum(I1.*((y-x1(2:n+1)-m1).^2))/R1^3;
    ddl(4,6)=-1*dl(4)/R0;
    ddl(6,4)=ddl(4,6);
    ddl(5,7)=-1*dl(5)/R1;
```

```
    ddl(7,5)=ddl(5,7);

    term1=term1+ddl/Npar;
end

term2_trim=zeros(7,7);
for i=1:Npar
    kk=term2t(:,i)*term2t(:,i)';
   for j=1:49
        forterm2(i,j)=kk(fix((j-1)/7)+1,rem(j-1,7)+1);
    end
end
term2_temp=trimmean(forterm2,ptrim)
for i=1:7
    for j=1:7
        term2_trim(i,j)=term2_temp((i-1)*7+j);
    end
end


obInfo=-1*term1 -term2 + (term3a)*(term3a)';
obsInfot=-1*term1-term2_trim+(term3a)*(term3a)';
vparmt=inv(obsInfot);
vparmm=inv(obInfo)
```

8. Miscellaneous functions

Random sampling (1)

```
function Newdata=Rsample1(data,weight,NofSample)
n=max(size(data));
re_ind=rand(1,NofSample);
cmwt=cumsum(weight)/sum(weight);
for k=1:NofSample
    st=(re_ind(k)>cmwt(1:n-1));
    Newdata(k)=data(sum(st)+1);
end
Newdata=Newdata';
```

Random sampling (2)

```
function [Newdata1,Newdata2]=Rsample2(data1,data2, weight,NofSample)
n=max(size(data1));
re_ind=rand(1,NofSample);
cmwt=cumsum(weight)/sum(weight);
for k=1:NofSample
    st=(re_ind(k)>cmwt(1:n-1));
    Newdata1(k)=data1(sum(st)+1);
    Newdata2(k)=data2(sum(st)+1);
end
Newdata1=Newdata1';
Newdata2=Newdata2';
```

Complete likelihood

```
function loglike=comp_logliken(x,I,y,theta,n)

phi=theta(1);
Q=theta(2);
m0=theta(3);
m1=theta(4);
R0=theta(5);
R1=theta(6);
pi=theta(7);
mu0=0;
Sig0=Q/(1-phi^2);

loglike=-0.5*(log(Sig0)+(x(1)-mu0)^2/Sig0) -0.5*(n*log(Q)+...
        (sum(x(2:n+1).^2)+(phi^2)*sum(x(1:n).^2) -2*phi*sum(x(1:n).*x(2:n+1)))/Q) +...
         sum(I(2:n+1))*log(pi)+(n-sum(I(2:n+1)))*log(1-pi) -...
         0.5* sum( I(2:n+1).* (log(R1)+((y-x(2:n+1)-m1).^2)/R1)+...
         (1-I(2:n+1)).*(log(R0)+((y-x(2:n+1)-m0).^2)/R0));
```

Sample code for Data B : 10 % missing

```
size_y=size(y);
if size_y(1)>size_y(2)
    y=y';
end

% --------a and n
n=max(size(y)); %n:length of the data
a=~isnan(y);

%---------set initial parameter; Use mme or give some numbers
y_I=y(a>0);
Iparm=Iparmnorm(y_I)

Tol=0.1;
Miter=30;
Npar=500;

[Eparm1a,Vparm1a,RelLike1a,Tcal1a,Niter1a]=mainnormaux(y,a,Iparm,Tol,Miter,Npar,n,5);

Tol=0.05;
Miter=30;
Npar=500;

Iparm2a=Eparm1a(end,:)
[Eparm2a,Vparm2a,RelLike2a,Tcal2a,Niter2a]=mainnormaux(y,a,Iparm2a,Tol,Miter,Npar,n,5);

Tol=0.01;
Miter=20;
Npar=500;
Iparm3a=Eparm2a(end,:)
```

```
[Eparm3a,Vparm3a,RelLike3a,Tcal3a,Niter3a]=mainnormaux(y,a,Iparm3a,Tol,Miter,Npar,n,3);

Tol=0.01;
Miter=25;
Npar=2000;

Iparm4a=Eparm3a(end,:)
[Eparm4a,Vparm4a,RelLike4a,Tcal4a,Niter4a]=mainnormaux(y,a,Iparm4a,Tol,Miter,Npar,n,2);
```

BIBLIOGRAPHY

[1] Anderson, W. N., Kleindorfer, G. B., Kleindorfer, P. R., and Woodroofe, M. B. (1969). Consistent estimates of the parameters of a linear system. *The Annals of Mathematical Statistics*, 40(6):2064–2075.

[2] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–187.

[3] Bollerslev, T. (1986). Generalized autoregressive conditional heteroscedasticity. *J. Econ.*, 31:307–327.

[4] Celeux, G. and Diebolt, J. (1985). The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quarterly*, 2:73–82.

[5] Chan, K. S. and Ledolter, J. (1995). Monte Carlo EM estimation for time series models involving counts. *Journal of the American Statistical Association*, 90(429):242–252.

[6] Chib, S., Nardari, F., and Shephard, N. (2002). Markov Chain Monte Carlo methods for stochastic volatility models. *Journal of Econometrics*, 108:281–316.

[7] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, 39:1–38.

[8] Doucet, A., de Freitas, N., and Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer, N.Y.

[9] Doucet, A. and Tadic, B. B. (2003). Parameter estimation in general state-space models using particle methods. *Annals of Institute of statistical mathematics*, 55(2):409–422.

[10] Durbin, J. and Koopman, S. J. (2000). Time series analysis of non-Gaussian observations based on state space models form both classical and Bayesian perspectives. *J. R. Statist. Soc. B*, 62:3–56.

[11] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. 50:987–1007.

[12] Everitt, B. S. and Hand, D. J. (1981). *Finite Mixture Distributions*. Chapman and Hall, N.Y.

[13] Franses, P. H. and van Dijk, D. (2000). *Nonlinear Time Series Models in Empirical Finance*. Cambrige University Press.

[14] Godsill, S., Doucet, A., and West, M. (2004). Monte Carlo smoothing for non-linear time series. *JASA*, 199:156–168.

[15] Harvey, A. C., Ruiz, E., and Shephard, N. (1994). Multivariate stochastic variance models. *Review of Economic Studies*, 61:247–264.

[16] Jacquier, E., Polson, N. G., and Rossi, P. E. (1994). Bayesian analysis of stochastic volatility models. *Journal of Business and Economic Statistics*, 12(4):371–417.

[17] Kim, S., Shephard, N., and Chib, S. (1998). Stochastic volatility: Likelihood inference and comparison with ARCH models. *The review of economic studies*, 65(3):361–393.

[18] Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state-space models. *J. Comput. Graph. Statist.*, 5:1–25.

[19] Kitagawa, G. and Sato, S. (2001). Monte Carlo smoothing and self-organising state-space model. In Doucet, A., de Freitas, N., and Gordon, N., editors, *Sequential Monte Carlo Methods in Practice*, pages 177–195. Springer-Verlag, NY.

[20] Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of Royal Statistical Society, Series B*, 44:226–233.

[21] Maskell, S. and Gordon, N. (2002). A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking. IEE Colloquium on Tracking.

[22] Mclachlan, G. J. and Krishnan, T. (1997). *The EM algorithm and extensions*. John Wiley and Sons, Inc., NY.

[23] Mellino, A. and Turnbull, S. (1990). Pricing foreign currency options with stochastic volatility. *Journal of Econometrics*, 45:7–39.

[24] Pulgarin, L. (2001). Applied comparison between GARCH and stochastic volatility models. Master's thesis, University of Pittsburgh.

[25] Shephard, N. (1996). Statistical aspects of ARCH and stochastic volatility. In Cox, Hinkley, and Barndorff-Nielson, editors, *Time Series Models: In econometrics, finance and other fields*. Chapman and Hall, London.

[26] Shumway, R. H. and Stoffer, D. S. (2000). *Time Series Analysis and its Applications*. Springer, N.Y.

[27] Tanner, M. A. (1996). *Tools for Statistical Inference*. Springer, N.Y.

[28] Wei, G. and Tanner, M. A. (1990). A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85:699–704.