11

Practical filtering for stochastic volatility models

Jonathan R. Stroud
Department of Statistics, Wharton School, University of Pennsylvania

Nicholas G. Polson
Graduate School of Business, University of Chicago

Peter Müller
M. D. Anderson Cancer Center, University of Texas

Abstract

This paper provides a simulation-based approach to filtering and sequential parameter learning for stochastic volatility models. We develop a fast simulation-based approach using the practical filter of Polson, Stroud and Müller (2002). We compare our approach to sequential parameter learning and filtering with an auxiliary particle filtering algorithm based on Storvik (2002). For simulated data, there is close agreement between the two methods. For data on the S&P 500 market stock index from 1984–90, our algorithm agrees closely with a full MCMC analysis, whereas the auxiliary particle filter degenerates.

State Space and Unobserved Component Models: Theory and Applications, eds. Andrew C. Harvey, Siem Jan Koopman and Neil Shephard. Published by Cambridge University Press.

11.1 Introduction

Filtering and sequential parameter learning for stochastic volatility (SV) have many applications in financial decision making. SV models are commonly used in financial applications as their dynamics are flexible enough to model observed asset and derivative prices. However, many applied financial decision making problems are sequential in nature such as portfolio selection (e.g. Johannes, Polson and Stroud (2002b)) and option pricing. These applications require filtered estimates of spot volatility and sequential parameter estimates to account for estimation risk. In this paper, we provide a simulation-based approach for volatility state filtering that also incorporates sequential parameter learning. The methodology is based on the practical filter of Polson, Stroud and Müller (2002). Unlike previous simulation-based filtering methods, for example Kim, Shephard and Chib (1998) in the SV context, our algorithm incorporates sequential parameter learning within Markov chain Monte Carlo (MCMC).

Many authors have considered the problem of simulation-based filtering with known static parameters. A common approach uses particle filtering methods; see for example Gordon, Salmond and Smith (1993), Liu and Chen (1998), Carpenter, Clifford and Fearnhead (1999), Pitt and Shephard (1999b), or Doucet, Godsill and Andrieu (2000) for an excellent review of the subject. Liu and West (2001) provided an algorithm for joint state and parameter learning. Storvik (2002) provided an auxiliary particle filtering method that deals with sequential state and parameter learning. Other approaches include Chopin (2002) for static models, and Gilks and Berzuini (2001) and Fearnhead (2002) who provided useful approaches using MCMC, sufficient statistics and particle filters. For comparison purposes in our applications we use an auxiliary particle filter (APF) based on Pitt and Shephard (1999b) and Storvik (2002).

The goal of our analysis is to provide inference in sequential fashion about an unobserved state $\mathbf{x}_t$ given data $\mathbf{y}_{1:t}$ at each time t, for $t = 1, \ldots, T$, in a nonlinear, non-Gaussian state space model. In this paper we use the generic notation $\mathbf{y}_{1:t}$ to denote the vector $(\mathbf{y}_1, \ldots, \mathbf{y}_t)$. Our approach takes advantage of the mixture representation of the joint distribution of the states $\mathbf{x}_t$ and parameters θ given $\mathbf{y}_{1:t}$,

$$p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t}) = \int p(\mathbf{x}_t, \theta|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t}) p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t}) \mathrm{d}\mathbf{x}_{0:t-k}.$$

The practical filter of Polson, Stroud and Müller (2002) is a simulation-based filtering and sequential parameter learning algorithm. This approach has a number of computational advantages when parameter learning is also

incorporated. The main advantage is that it does not degenerate as particle filtering methods can, see also Johannes, Polson and Stroud (2002a). Section 11.2 describes our approach to filtering and sequential parameter learning using the practical filter.

To illustrate our methodology, we use simulated data and daily data from the S&P 500. For the simulated data, we find that our methodology is comparable with the method of Storvik (2002), which builds on the APF, for both filtering and sequential parameter learning. However, for the S&P 500 data, we find that the APF degenerates for sequential parameter learning. This is partly due to the fact that the model is misspecified for these data. More specifically, the APF degenerates when sequentially estimating the volatility of volatility parameter. The practical filter avoids degeneracies and also compares favorably with a full MCMC analysis. However, for the volatility of volatility parameter, the practical filter does have some difficulties capturing the full posterior uncertainty.

The rest of the paper is outlined as follows. Section 11.2 describes the basic filtering problem and how the practical filter works. Section 11.3 describes the stochastic volatility model and shows how to implement the practical and particle filters for state and sequential parameter learning. An application to the S&P 500 is provided with the focus of providing a computationally fast algorithm. Finally, Section 11.4 concludes.

11.2 Practical filtering

The basic filtering problem for dynamic state space models requires that posterior distributions be recomputed at each time point. Let $\mathbf{y}_t$ denote the observation vector at time t, let $\mathbf{x}_t$ denote the state vector, θ the parameters, and $\mathbf{x}_{s:t} = (\mathbf{x}_s, \ldots, \mathbf{x}_t)$, $\mathbf{y}_{s:t} = (\mathbf{y}_s, \ldots, \mathbf{y}_t)$ the block of states and observations, respectively, from time s up to time t. We need to compute the joint conditional posterior distribution $p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t})$. Filtering and sequential parameter learning are achieved by calculating the marginal state filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ and the sequence of marginal parameter posteriors $p(\theta|\mathbf{y}_{1:t})$.

We now describe the practical filter which provides a sequential simulation-based approach to estimating these distributions. The key idea is to express the filtering distribution $p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t})$ as a mixture of lag-filtering distributions. More specifically, we can write

$$p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t}) = \int p(\mathbf{x}_t, \theta|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t}) p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t}) \mathrm{d}\mathbf{x}_{0:t-k}.$$

Here $p(\mathbf{x}_t, \theta|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t})$ is the lag-filtering distribution of the state and parameter vector given knowledge of $\mathbf{x}_{0:t-k}$ and data $\mathbf{y}_{1:t}$. The use of a block of k data points and updating has also been exploited in the particle filtering approach. Pitt and Shephard (2001) showed how to use it for the auxiliary particle filter. See also Clapp and Godsill (1999) and Godsill and Clapp (2001). Chopin (2002) provided a batch importance sampling procedure that extends to the parameter learning case and uses ideas in Gilks and Berzuini (2001).

Our algorithm proceeds in two stages. First, due to the sequential nature of the problem, we already have draws of the states $\mathbf{x}_{0:t-k}^{(g)}$ from $p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t})$. Second, we use these to provide samples from the current filtering distribution $p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t})$ by simulating from the filtering distribution

$$p\left(\mathbf{x}_{t-k+1:t}, \theta|\mathbf{x}_{0:t-k}^{(g)}, \mathbf{y}_{1:t}\right).$$

Samples from this distribution can be obtained by iteratively simulating from the complete conditionals

$$p\left(\mathbf{x}_{t-k+1:t}|\theta, \mathbf{x}_{t-k}^{(g)}, \mathbf{y}_{t-k+1:t}\right), \tag{11.1}$$

$$p\left(\theta|\mathbf{x}_{t-k+1:t}, \mathbf{x}_{0:t-k}^{(g)}, \mathbf{y}_{1:t}\right). \tag{11.2}$$

Notice that, due to the Markovian property of the states, the fixed-lag smoothing distribution, (11.1), depends on the history only through $\mathbf{x}_{t-k}^{(g)}$. Hence, we need only know the last stored state variable $\mathbf{x}_{t-k}^{(g)}$ to simulate from this conditional distribution. On the other hand, this is unfortunately not true for the parameter vector θ. Here we need the full history $\mathbf{x}_{0:t-k}^{(g)}$ to be able to draw from its conditional. However, a number of authors have pointed out that sufficient statistics, which are typically available, greatly simplify the problem (e.g. Storvik (2002), Fearnhead (2002) and Polson, Stroud and Müller (2002)). In such cases, one can exploit a sufficient statistic $\mathbf{s}_{t-k}^{(g)} = S\left(\mathbf{x}_{0:t-k}^{(g)}, \mathbf{y}_{1:t-k}^{(g)}\right)$ instead of saving the entire histories $\mathbf{x}_{0:t-k}^{(g)}$ and $\mathbf{y}_{1:t-k}^{(g)}$. This then simplifies the parameter draw (11.2) as

$$p\left(\theta|\mathbf{x}_{t-k+1:t}, \mathbf{s}_{t-k}^{(g)}, \mathbf{y}_{t-k+1:t}\right),$$

which implies that the computational cost for the θ update is fixed over time. Now our key assumption is that sequential draws $\mathbf{x}_{t-k}^{(g)}$ from the marginal posterior $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t})$ can reasonably be approximated by draws from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t-1})$, hence our use of $\left(\mathbf{x}_{t-k}^{(g)}, \mathbf{s}_{t-k}^{(g)}\right)$ in the conditioning above.

Finally, to ensure a fast filtering algorithm, we need to efficiently simulate

$\mathbf{x}_{t-k+1:t}$ from the fixed-lag smoothing distribution (11.1). This can be dealt with using a number of implementations. First, we have standard MCMC smoothing methods for state space models, for example Carlin, Polson and Stoffer 1992b. Moreover, in many instances with a careful choice of additional latent variables these can be implemented using the fast *forward-filtering backward-sampling* (FFBS) algorithm; see Carter and Kohn 1994, Frühwirth-Schnatter 1994c and Shephard 1994 for conditionally Gaussian models, and Shephard and Pitt (1997) for non-Gaussian measurement models.

We now describe the practical filtering algorithm for state and sequential parameter learning for conditionally Gaussian models. Nonlinear or non-Gaussian models can be implemented by replacing the FFBS step with single-state updating or sub-blocking (see, Carlin, Polson and Stoffer 1992b and Shephard and Pitt (1997), respectively).

Algorithm Filtering with sequential parameter learning
Initialisation: For $g = 1, \ldots, G$: generate $\theta^{(g)} \sim p(\theta)$.
Burn-in: For $t = 1, \ldots, k$:

For $g = 1, \ldots, G$: initialise $\theta^0 = \theta^{(g)}$.

For $i = 1, \ldots, I$:

Generate $\mathbf{x}_{0:t}^i \sim p(\mathbf{x}_{0:t}|\theta^{i-1}, y_{1:t})$.
Generate $\theta^i \sim p(\theta|\mathbf{x}_{0:t}^i, y_{1:t})$.

Set $\left(\mathbf{x}_0^{(g)}, \theta^{(g)}\right) = \left(\mathbf{x}_0^I, \theta^I\right)$.

Sequential updating: For $t = k+1, \ldots, T$:

For $g = 1, \ldots, G$: initialise $\theta^0 = \theta^{(g)}$.

For $i = 1, \ldots, I$:

Generate $\mathbf{x}_{t-k+1:t}^i \sim p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}^{(g)}, \theta^{i-1}, \mathbf{y}_{t-k+1:t})$
Generate $\theta^i \sim p(\theta|\mathbf{x}_{0:t-k}^{(g)}, \mathbf{x}_{t-k+1:t}^i, \mathbf{y}_{1:t})$

Set $\left(\mathbf{x}_{t-k+1}^{(g)}, \theta^{(g)}\right) = (\mathbf{x}_{t-k+1}^I, \theta^I)$ and leave $\mathbf{x}_{0:t-k}^{(g)}$ unchanged.

The speed and accuracy of the algorithm depends on the choice of the three parameters (G, I, k). First, the Monte Carlo sample size G relates to the desired accuracy for state and parameter estimation. In our stochastic volatility model, we find that $G = 250$ is sufficient. The number of MCMC iterations for fixed-lag smoothing is specified by I. In the stochastic volatility example we can use FFBS for the states and we can get a direct draw of the states given the parameters and hence $I = 1$ in this case. However, we also need to update the parameters given the states. With a small number

of data points a larger value of I is necessary; however, as the data arrive this can be reduced as the previous parameter draw is already nearly a draw from the desired stationary distribution. The use of a reasonable burn-in period can also alleviate the problem with the choice of I. Finally, the choice of the lag length k depends on the memory of the process. For example, for a stationary AR(1) state process, the correlation decays geometrically in k. In many cases, a small value of k is sufficient; for example, we find that in the stochastic volatility model $k = 50$ usually suffices.

11.3 Filtering for stochastic volatility models

To illustrate our methodology we use a standard daily stock index return dataset from the Standard and Poor's S&P 500 market index from 1984–90. The stock market crash of October 1987 provides a negative return in excess of 20%. This can lead to filtered parameter estimates changing abruptly and provides a useful testing ground for the practical and particle filters. We now develop a fast practical filtering method and we compare this with the sequential particle filtering approach of Storvik (2002).

In the standard univariate log-stochastic volatility model (e.g. Jacquier, Polson and Rossi (1994), Ghysels, Harvey and Renault (1996)), the returns y_t and log-volatility states x_t follow a state space model of the form

$$\begin{aligned} y_t &= \exp(x_t/2)\epsilon_t, \\ x_t &= \alpha + \beta x_{t-1} + \sigma\eta_t \end{aligned}$$

with initial log-volatility $x_0 \sim \mathcal{N}(m_0, C_0)$. The parameter vector θ consists of the volatility mean reversion parameters $\psi = (\alpha, \beta)$ and the volatility of volatility σ. Here we assume the errors ϵ_t and η_t are independent $\mathcal{N}(0, 1)$ white noise sequences, although it is possible to include a cross correlation or leverage effect (e.g. Jacquier, Polson and Rossi (2003)).

We now describe how to implement the practical filter and the auxiliary particle filtering methods of Storvik (2002) for SV models. Previous work which has used particle filters on SV includes Kim, Shephard and Chib (1998), Pitt Shephard (1999b, 2001), but these papers only considered the problem of filtering the volatility with known parameters and not the case of sequential parameter learning.

11.3.1 Implementing the practical filter

In order to implement the practical filter we need to determine a fast algorithm for fixed-lag smoothing for the states. To do this, we transform the

observations to $y_t^* = \log y_t^2$ and use the approximating mixture model of Kim, Shephard and Chib (1998):

$$\begin{aligned} y_t^* &= x_t + \epsilon_t^*, \\ x_t &= \alpha + \beta x_{t-1} + \sigma\eta_t, \end{aligned}$$

where ϵ_t^* is a discrete mixture of seven normals with known parameters and weights. To perform fixed-lag smoothing for $\mathbf{x}_{s:t} = (x_s, \ldots, x_t)$, we introduce a set of discrete mixture indicators $\mathbf{z}_{s:t} = (z_s, \ldots, z_t)$ with $z_t \in \{1, \ldots, 7\}$, and update the states and indicators in a two-block Gibbs sampler. First, we generate the states $\mathbf{x}$ given the indicators $\mathbf{z}$ using the FFBS algorithm. Then we generate the indicators given the states using independent multinomial draws.

For the parameters $\theta = (\psi, \sigma^2)$, the sufficient statistic structure $S(\cdot)$ is straightforward to determine. We assume a conjugate prior, $p(\theta) = p(\psi|\sigma^2)\, p(\sigma^2)$, where $\psi|\sigma^2 \sim \mathcal{N}(\psi_0, A_0^{-1}\sigma^2)$ and $\sigma^2 \sim \mathcal{IG}(a_0, b_0)$, which leads to closed-form posterior distributions at each time t,

$$\psi|\sigma^2, \mathbf{x}_{0:t}, \mathbf{y}_{1:t} \sim \mathcal{N}\left(\psi_t, A_t^{-1}\sigma^2\right) \quad \text{and} \quad \sigma^2|\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \sim \mathcal{IG}(a_t, b_t).$$

The sufficient statistics for θ are $\mathbf{s}_t = \{\psi_t, A_t, a_t, b_t\}$, and the lag-$k$ updating recursions are given by $\psi_t = A_t^{-1}(A_{t-k}\psi_{t-k} + H'\mathbf{x})$, $A_t = A_{t-k} + H'H$, $a_t = a_{t-k} + 0.5k$, and $b_t = b_{t-k} + 0.5(\mathbf{x} - H\psi_{t-k})'(\mathbf{x} - H\psi_{t-k})$, where $\mathbf{x} = (x_{t-k+1}, \ldots, x_t)'$, $H = (H_{t-k+1}, \ldots, H_t)'$ and $H_t = (1, x_{t-1})'$.

The SV practical filtering algorithm then iterates between three blocks:

$$\begin{aligned} \text{States:} \quad & p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}, \mathbf{z}_{t-k+1:t}, \theta, \mathbf{y}_{t-k+1:t}), \\ \text{Indicators:} \quad & p(\mathbf{z}_{t-k+1:t}|\mathbf{x}_{t-k+1:t}, \mathbf{y}_{t-k+1:t}), \\ \text{Parameters:} \quad & p(\theta|\mathbf{s}_{t-k}, \mathbf{x}_{t-k+1:t}). \end{aligned}$$

This cycle is repeated I times for each sample path, $g = 1, \ldots, G$.

Our filtering approach also allows us to periodically refresh the states and parameters, if needed. More specifically, a full update of the state trajectories and parameters $(\mathbf{x}_{0:t}^{(g)}, \theta^{(g)}), g = 1, \ldots, G$, can be performed every T^* time steps. This improves the parameter learning problem dramatically although it slows down the algorithm. However, typically a few iterations are sufficient to obtain convergence to the smoothing distribution $p(\mathbf{x}_{0:t}, \mathbf{z}_{1:t}, \theta|\mathbf{y}_{1:t})$ due to our fast MCMC smoother. A similar approach has been used to improve particle filters using periodic MCMC moves (Doucet, Godsill and Andrieu 2000). However, this approach is more costly for particle filters since the number of particles, N, is typically much larger than the G paths required for our approach.

11.3.2 Auxiliary particle filtering with parameter learning

To compare our methodology to the particle filter we also implement the parameter learning algorithm of Storvik (2002). This is a sequential importance sampling procedure that assumes an initial set of N particles $(\mathbf{x}_{0:t-1}, \theta) \sim p(\mathbf{x}_{0:t-1}, \theta|\mathbf{y}_{1:t-1})$. Letting $\mathbf{s}_{t-1}$ denote the posterior sufficient statistics that are updated at each time step, the algorithm then draws

$$\theta \sim p(\theta|\mathbf{s}_{t-1}) \text{ and } \mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}, \theta)$$

and reweights $(\mathbf{x}_{0:t}, \theta)$ proportionally to the likelihood $p(\mathbf{y}_t|\mathbf{x}_t, \theta)$.

For the SV model, we implement an APF version of the Storvik algorithm using the sampling/importance resampling-based auxiliary proposal of Pitt and Shephard (1999b), Section 3.2). Given an initial particle set, the APF first selects particles with high 'likelihood', and propagates those forward to the next time step. In our implementation, we define the selection probabilities as $\lambda_t^j \propto p(y_t|\mu_t^j)$, where $\mu_t^j = \alpha^j + \beta^j x_{t-1}^j$ is the estimate of x_t for particle j. For each selected particle k_j, we sample (x_t^j, θ^j) from the dynamic prior $p(x_t, \theta|x_{0:t-1}^{k_j}, y_{1:t})$. We then reweight the particles by the likelihood $p(y_t|x_t^j)$ to obtain a sample from the posterior. The APF algorithm for joint state and parameter learning is given below.

Algorithm Auxiliary particle filter with parameter learning
Initialisation: For $j = 1, \ldots, N$:

Generate $x_0^j \sim p(x_0)$, and set $\mathbf{s}_0^j = \mathbf{s}_0$.
Initialise weights to $w_0^j = N^{-1}$.

Sequential updating: For $t = 1, \ldots, T$:

Compute first-stage weights, $\lambda_t^j \propto p(y_t|\mu_t^j)$, for $j = 1, \ldots, N$.
Sample the index k_j with probabilities λ_t^j, for $j = 1, \ldots, N$.
For $j = 1, \ldots, N$:

Generate $\theta^j \sim p(\theta|\mathbf{s}_{t-1}^{k_j})$.
Generate $x_t^j \sim p(x_t|x_{t-1}^{k_j}, \theta^j)$.
Update $\mathbf{s}_t^j = S(x_t^j, \mathbf{s}_{t-1}^{k_j})$, and set $\tilde{\mathbf{x}}_t^j = (\theta^j, x_t^j, \mathbf{s}_t^j)$.
Compute the second-stage weights, $w_t^j = w_{t-1}^{k_j} p(y_t|x_t^j)/\lambda_t^{k_j}$.

Normalise weights, $w_t^j = w_t^j / \sum_{i=1}^N w_t^i$.
Compute effective sample size $N_{\text{eff}} = 1/\sum_{i=1}^N \left(w_t^i\right)^2$.
If $N_{\text{eff}} < 0.8N$, resample $\tilde{\mathbf{x}}_t^j$ with probability w_t^j, and set $w_t^j = N^{-1}$.

To improve efficiency of the algorithm, we use stratified sampling for the multinomial and resampling steps above (see, for example, Carpenter, Clifford and Fearnhead (1999)).

We now turn to our two applications.

11.3.3 Applications

We use the two filtering and sequential parameter learning algorithms, and compare the methods with estimates determined via full MCMC estimation. The two algorithms use the following specifications, chosen to make the running times similar:

(i) Storvik's algorithm with $N = 100\,000$ particles.
(ii) Practical filter with $G = 250, I = 50$ and $k = 50$.

Figure 11.1 displays the filtered volatility and sequential parameter learning plots for both algorithms on a simulated dataset. A time series of length 500 is generated using the parameter values

$$\theta = (\alpha, \beta, \sigma^2) = (-0.0084, 0.98, 0.04).$$

For both filtering algorithms, an initial prior of $x_0 \sim \mathcal{N}(0, 1)$ is used along with the conjugate prior $p(\theta)$ described in Section 11.3.1 with hyperparameters $\psi_0 = (0, 0.95)'$, $A_0 = \text{diag}(10, 100)$, $a_0 = 2.25$ and $b_0 = 0.0625$. This implies a mildly informative prior which should give no advantage to either method. To implement the practical filter, refreshing is used every $T^* = 125$ time steps. For comparison purposes, we also run a full MCMC smoothing algorithm at each time point. The thick lines denote the full MCMC quantiles and the thin lines denote those from the two filters. As may be expected, there is close agreement between the practical filter and the true MCMC results for both the states x_t and the parameters θ in this simulated example. The auxiliary particle filter also performs quite well.

For daily data on the S&P 500, a different picture emerges. Figure 11.2 displays the filtered volatility and sequential parameter learning plots for both algorithms on the S&P 500 dataset. The priors used are the same as those in the simulation study. The 5, 50 and 95 percentiles are plotted along with the 'true' estimates obtained from a full MCMC analysis. For the practical filter, refreshing is used every $T^* = 250$ time steps. The main difference in the results comes in estimating the volatility of volatility parameter σ sequentially, and the differences are more pronounced at earlier time points. The advantage of practical filtering is that there are no degeneracies and at the end of the sample there is close agreement with the

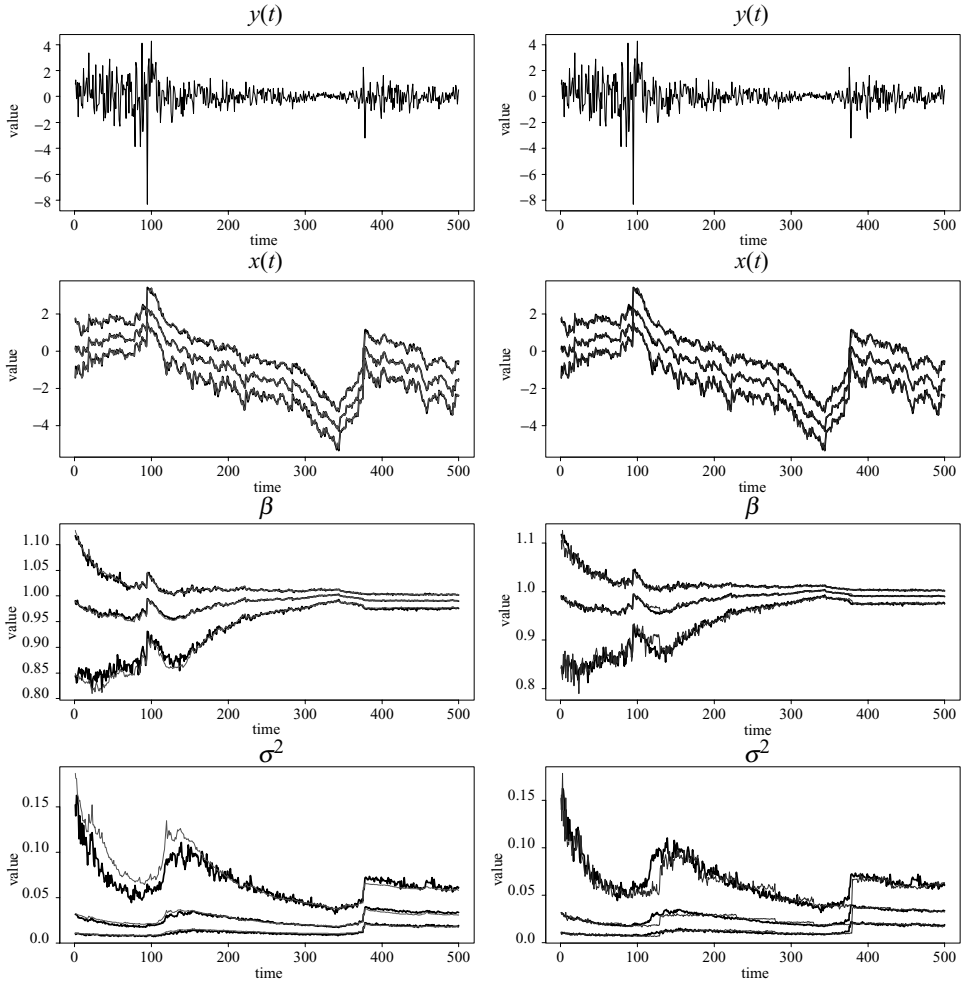

Fig. 11.1. SV model, simulated data. Filtered 5, 50, 95 percentiles. Left: Storvik's algorithm with $N = 100\,000$ particles. Right: Practical filter with $G = 250, I = 50, k = 50$. Thin lines are used to denote the filtering quantiles, and thick lines denote the quantiles from a full MCMC analysis.

full MCMC estimates after the Crash of 1987. The practical filter does a reasonably good job of tracking the shift in the filtered estimates of σ and eventually the quantiles match those obtained from the full analysis.

While the particle filter provides reliable inference about the state volatility vector similar to the practical filter, the difference lies in the sequential parameter estimates. The main problem with the particle filter is that it has great difficulty in handling the jump in σ during the stock market crash of October 1987. The sequential parameter posterior for σ nearly degenerates

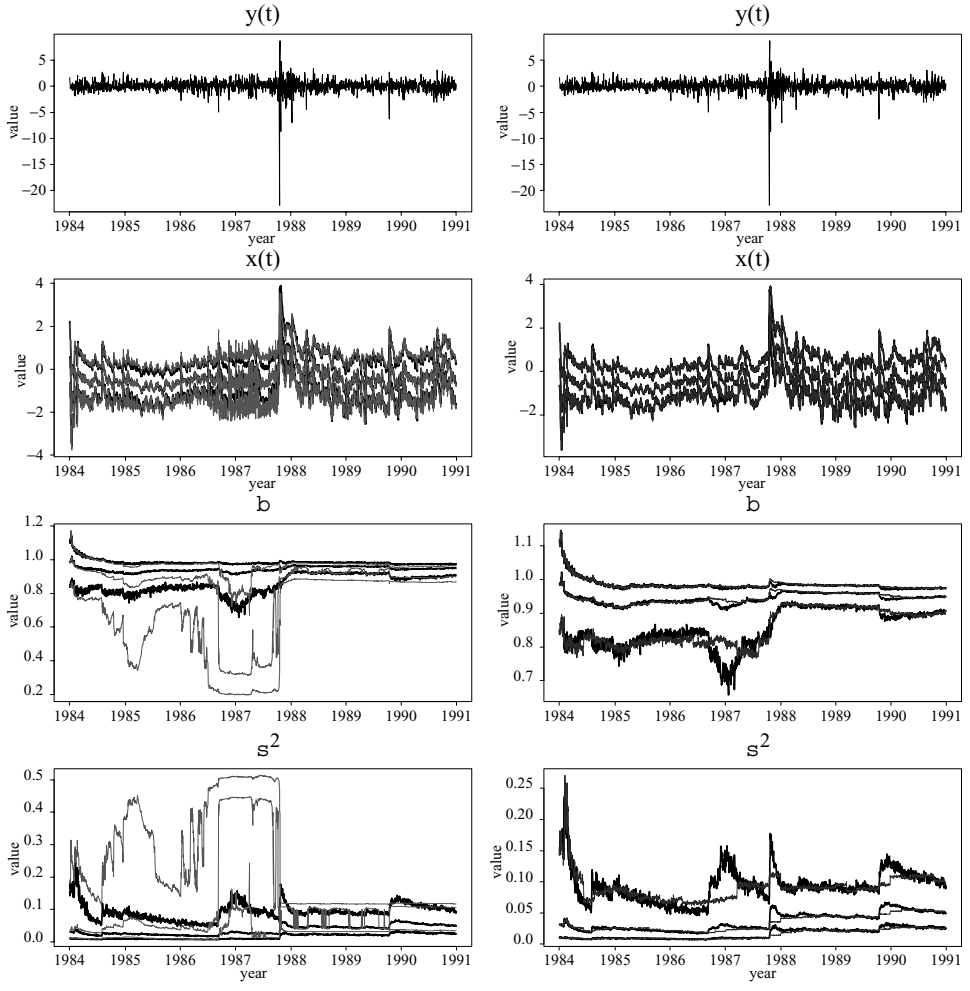

Fig. 11.2. SV model, S&P 500 data. Filtered 5, 50, 95 percentiles. Left: Storvik's algorithm with $N = 100\,000$ particles. Right: Practical filter with $G = 250, I = 50, k = 50$. Thin lines are used to denote the filtering quantiles, and thick lines denote the quantiles from a full MCMC analysis.

immediately after the crash as there are not enough particles to capture the appropriate posterior uncertainty in σ after the crash. Eventually it degenerates onto a value near the 'true' value learned from the full MCMC analysis but it totally misrepresents the appropriate posterior uncertainty.

11.4 Conclusions

In this paper we develop a sequential simulation-based methodology for filtering and sequential parameter learning in SV models. The method is easy to implement as it requires only simple modification of the smoothing algorithm. While filtering methods such as particle filtering can uncover states with known static parameters they have difficulty in also providing sequential parameter estimates. It is also not uncommon for these models to degenerate whereas our approach will not.

On a simulated dataset, we found that the filter of Storvik (2002) performed on a par with our methodology. For the S&P 500 daily stock return data, however, the auxiliary particle filter degenerates and has particular difficulty in estimating the volatility of volatility. Our procedure also avoids degeneracies but in dealing with the outlying crash it tends to oversmooth the uncertainty in our estimates for σ relative to a brute-force MCMC approach. This is also partly due to the fact that the learning parameters break the simple Markov property of updating and long-range dependence can reduce the accuracy of the particle filter approximation. We hope to study this further and provide other updating schemes.