Practical Filtering with Sequential Parameter Learning

Nicholas G. Polson [1]
Graduate School of Business, University of Chicago

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

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

In this paper we develop a general simulation-based approach to filtering and sequential parameter learning. We begin with an algorithm for filtering in a general dynamic state space model and then extend this to incorporate sequential parameter learning. The key idea is to express the filtering distribution as a mixture of lag-smoothing distributions and to implement this sequentially. Our approach has a number of advantages over current methodologies. First, it allows for sequential parameter learning where sequential importance sampling approaches have difficulties. Second, it avoids degeneracies that plague particle filtering methods. We illustrate these advantages with a number of applications and describe when our methodology has particular advantages. We perform filtering and sequential parameter learning for a stochastic volatility model and a three-dimensional stochastic Lorenz flow model.

Keywords: State-Space Models, Hidden Markov Models; State-Dependent Variance, Filtering, Sequential Parameter Learning, MCMC, Stochastic Volatility, Lorenz Attractor.

[1]We would like to thank Arnaud Doucet, Anne Gron, Michael Johannes, Michael Pitt, and the participicants at the Durbin Workshop on State Space and Unobserved Components Models. Nick Polson is Professor of Statistics and Econometrics at the Graduate School of Business, University of Chicago, email: *ngp@gsb.uchicago.edu.* Jonathan Stroud is Assistant Professor of Statistics, The Wharton School, University of Pennsylvania, email: *stroud@wharton.upenn.edu.* Peter Müller is Professor of Biostatistics, MD Anderson Cancer Center, Texas Medical Center, email: *pm@odin.mdacc.tmc.edu.*

1 Introduction

Filtering and sequential parameter learning in general dynamic state space models provides a number of computational challenges. In this paper we provide a simulation-based methodology for state filtering in state space models. Our approach avoids degeneracies that plague particle filtering methods and allows sequential parameter learning where sequential importance sampling approaches have difficulties. It also applies to a wide range of state space models and can incorporate nonlinear and nonnormal models (Carlin, Polson and Stoffer, 1992), state-dependent variance models (Stroud, Müller and Polson, 2002) or hidden Markov models (Künsch, 2001).

First, we describe an algorithm for state filtering in general dynamic models with fixed parameters. Suppose that the data $\mathbf{y}_t$ are observed sequentially in time and that $\mathbf{x}_t$ is an unobserved state vector where the measurement (observation) and system (evolution) equations are of the form:

$$\begin{aligned} \text{Measurement:} \quad \mathbf{y}_t &\sim p(\mathbf{y}_t|\mathbf{x}_t) \\ \text{System:} \quad \mathbf{x}_t &\sim p(\mathbf{x}_t|\mathbf{x}_{t-1}). \end{aligned} \tag{1}$$

The initial state distribution $\mathbf{x}_0 \sim p(\mathbf{x}_0)$ is assumed to be given. To implement inference we need the marginal filtering distribution for the states $\mathbf{x}_t$. Filtering occurs when the posterior distribution of the states is re-estimated as each new observation arrives. It requires the sequential computation of the posterior distributions $p(\mathbf{x}_t|\mathbf{y}_{1,t})$. Throughout we use the notation $\mathbf{x}_{s,t} = (\mathbf{x}_s, \ldots, \mathbf{x}_t)$ and $\mathbf{y}_{s,t} = (\mathbf{y}_s, \ldots, \mathbf{y}_t)$ to denote the block of states and observations from time s up to time t. Secondly, we extend our algorithm to incorporate sequential parameter learning which requires both the marginal state filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1,t})$ and with the sequential posterior distributions $p(\theta|\mathbf{y}_{1,t})$ for the parameters. We provide an algorithm for simulating from the sequential joint posteriors $p(\mathbf{x}_{1,t}, \theta|\mathbf{y}_{1,t})$.

The key idea is to express the filtering distribution as a mixture of lag-smoothing distributions. The lag-smoothing distribution of the state vector $\mathbf{x}_{t-k+1,t}$ is defined as the conditional distribution $p(\mathbf{x}_{t-k+1,t}|\ \mathbf{x}_{t-k}, \mathbf{y}_{t-k+1,t})$ and the marginal lag-smoothing distribution is defined as $p(\mathbf{x}_{t-k}|\mathbf{y}_{1,t})$. The speed and accuracy of our algorithm depends on a number of factors. First we need to choose the appropriate length of the lag-smoothing distribution. We propose a number of diagnostic approaches for this choice. Secondly, we need to sample sequentially from the lag smoothing distribution which we do using standard MCMC smoothing methods for state space models (Carlin et al., 1992) and where possible we rely on the fast forward-filtering backward-sampling (FFBS, see Carter and Kohn, 1994; Frühwirth-Schnatter, 1994) algorithm for conditionally linear models.

From a theoretical perspective, incorporating sequential parameter learning is straightforward. Simply sample iteratively from the filtering distribution for the states given the parameters and then from the parameters given the states. However due to the sequential nature of the algorithm this can dramatically increase the computational complexity. To account for this, we propose an extension of the algorithm for sequential parameter learning that exploits a sufficient statistics, when available, for the parameter update. We also describe a general purpose algorithm and one that incorporates refreshing the initial states which helps with the accuracy of the sequential parameter learning problem.

Many authors have used simulation-based methods in the case of known static parameters; see, for example, Gordon, Salmond and Smith (1993), Carpenter, Clifford and Fearnhead (1999), Pitt and Shephard (1999) and Doucet, Godsill and Andrieu (2000). The most common approaches are based on particle filtering methods and on sequential importance sampling (Liu and Chen, 1998; Clapp and Godsill, 1999). Unfortunately, the automatic use of particle filtering methods is hindered by the possibility of particle degeneracies. Doucet et al. (2000) provide a theoretical discussion of

the degeneracies that can occur in such methods. Moreover, current methodology has difficulty in dealing with sequential static parameter learning and hence state filtering that incorporates parameter uncertainty. For other simulation-based filtering methods, see Handschin and Mayne (1969); Handschin (1970) and, more recently, Müller (1991), West (1993), Kitagawa (1994), Kitagawa (1996) and Hürzeler and Künsch (1998). Unlike previous simulation-based filtering methods, our algorithm directly uses MCMC. Little has been done on the hard problem of sequential parameter learning and filtering, notable exceptions are Liu and West (2000) and Clapp and Godsill (1999) where the parameter is treated as a time-varying state parameter with a stochastic error. Storvik (2002) provides a particle filtering approach to sequential parameter learning.

To illustrate our methodology, we first consider a stochastic volatility model where the state represents an unobserved volatility and the parameters drive the evolution of this process. We describe how to implement our fixed-lag filtering algorithm, with and without sequential parameter learning. We use a simulation study to compare our methodology with particle filtering methods in the case of known static parameters. Here both methods have similar operating characteristics. For sequential parameter learning we use daily data from the S&P500 stock index and compare our approach with the sequential importance sampling algorithm of Storvik (2002). In this case, we find that sequential importance sampling methods break down, whereas the practical filter with periodic refreshing of the initial states provides reilable inference.

Secondly, we consider a three-dimensional stochastic Lorenz model. Recent research (Bengtsson and Nychka, 2001) has focused on using particle filtering methods in the static parameter case, see also (West, 1993). The practical filter is straightforward to implement as we find a fast lag-smoothing algorithm based on a block Gibbs fashion using forward-filtering backward-sampling. This approach works even though there are nonlinearities in the system equation. We show how our fixed-lag methods work well in the both the case of known static parameters and sequential parameter learning.

The rest of the paper is outlined as follows. Section 2 describes our approach to the filtering problem with fixed parameters. We also show how our sequential filtering approach can be converted using backward sampling to generate samples from the full smoothing distribution $p(\mathbf{x}_{1,t}|\mathbf{y}_{1,t})$. Section 3 extend our basic filtering algorithm and incorporates sequential parameter learning. Section 4 applies our methodology. Finally, Section 5 concludes.

2 Filtering

2.1 Filtering with Fixed Parameters

We begin with the case of filtering with known static parameters. Consider a state-space model of the form $\mathbf{y}_t \sim p(\mathbf{y}_t|\mathbf{x}_t)$ and $\mathbf{x}_t \sim p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The state filtering distributions are given by $p(\mathbf{x}_t|\mathbf{y}_{1,t})$. Now suppose that we observe a new observation $\mathbf{y}_t$ and we wish to update the filtering distribution $p(\mathbf{x}_{t-1}|\mathbf{y}_{1,t-1})$ to $p(\mathbf{x}_t|\mathbf{y}_{1,t})$. We start by writing the desired filtering distribution as the marginal

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

At the heart of the proposed algorithm is the representation of this lag smoothing distribution as a mixture with respect to the smoothing distribution on $\mathbf{x}_{t-1}$.

$$\begin{aligned} p(\mathbf{x}_{t-k+1,t}|\mathbf{y}_{1,t}) &= \int p(\mathbf{x}_{t-k+1,t}|\mathbf{x}_{t-k},\mathbf{y}_{1,t})\ dp(\mathbf{x}_{t-k}|\mathbf{y}_{1,t}) \\ &= \int p(\mathbf{x}_{t-k+1,t}|\mathbf{x}_{t-k},\mathbf{y}_{t-k+1,t})\ dp(\mathbf{x}_{t-k}|\mathbf{y}_{1,t}) \end{aligned} \quad (2)$$

The second identity exploits the Markov property of the state space model. To resolve the integral with respect to $p(\mathbf{x}_{t-k}|\mathbf{y}_{1,t})$ we argue that as we are implementing the algorithm sequentially that samples from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1,t})$ are approximately samples from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1,t-1})$. The approximation is reasonable to the extent to which the predictive distribution $p(y_t|y_{t-k+1,t-1}, x_{t-k})$ is conditionally independent of $\mathbf{x}_{t-k}$ due to the identity

$$p(\mathbf{x}_{t-k}|\mathbf{y}_{1,t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1,t-1})}{p(\mathbf{y}_t|\mathbf{y}_{t-k+1,t-1})} p(\mathbf{x}_{t-k}|\mathbf{y}_{1,t-1}).$$

In many instances, the sensitivity of the predictive distribution to the initial state typically will decay quickly and many cases at an exponential rate; (see, for example, LeGland and Mevel, 1997; Künsch, 2001). Therefore, we can further simplify (2) using

$$p(\mathbf{x}_{t-k+1,t}|\mathbf{y}_{1,t}) = \int p(\mathbf{x}_{t-k+1,t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1,t})\ dp(\mathbf{x}_{t-k}|\mathbf{y}_{1,t}) \approx \int p(\mathbf{x}_{t-k+1,t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1,t})\ dp(\mathbf{x}_{t-k}|\mathbf{y}_{1,t-1}). \quad (3)$$

This identity motivates the following simulation-based approach for filtering $p(\mathbf{x}_t|\mathbf{y}_{1,t})$.

The algorithm is initialized with samples generated from the smoothing distribution $p(\mathbf{x}_{0,k}|\mathbf{y}_{0,k})$. Starting with $t = k+1$ we generate samples from the desired smoothing distributions $p(\mathbf{x}_t|\mathbf{y}_{1,t})$ by following (3). At each time t we generate new samples $\mathbf{x}_{t-k+1,t} \sim p(\mathbf{x}_{t-k+1,t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1,t})$, using imputed values for $\mathbf{x}_{t-k}$ saved from the previous iteration $t-1$. We save the simulated $\mathbf{x}_{t-k+1}$ for use in the next iteration with $t+1$. The algorithm reduces filtering to a lag smoothing problem, required to simulate from $p(\mathbf{x}_{t-k+1,t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1,t})$. We comment on possible approaches for the lag smoothing below, after the statement of the algorithm.

Algorithm 1: Filtering with Known Parameters

Burn-In: For $t = 1, \ldots, k$:

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

Generate $\mathbf{x}_{0,t} \sim p(\mathbf{x}_{0,t}|\mathbf{y}_{1,t})$ and save $\tilde{\mathbf{x}}_0^{(i)} = \mathbf{x}_0$.

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

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

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

Set $\tilde{\mathbf{x}}_{t-k+1}^{(i)}$ equal to $\mathbf{x}_{t-k+1}$ and leave $\tilde{\mathbf{x}}_{0,t-k}^{(i)}$ unchanged.

The algorithm replaces the hard problem of posterior inference for a general dynamic state space model by $(T-k)$ easy smoothing problems for short time series of length k. Posterior inference for a short length k state space model is typically easy. Implementing fixed-lag smoothing for the length k time series can be done efficiently in several ways. For an efficient filtering algorithm it is important that a fast algorithm for drawing from this lag-k smoothing distribution is available as this has to be repeated to converge at every time point. The computational advantage of the algorithm is that it reduces the original length T filtering problem to a smoothing problem for a series of length k that is moderately small. For a linear Gaussian model samples can be easily

generated using the forward-filtering backwards-sampling (FFBS) algorithm of Carter and Kohn (1994) and Frühwirth-Schnatter (1994). This algorithm exploits the factorization

$$p(\mathbf{x}_{t-k+1,t}|\mathbf{x}_{t-k},\mathbf{y}_{t-k+1,t}) \;=\; p(\mathbf{x}_t|\mathbf{x}_{t-k},\mathbf{y}_{t-k+1,t}) \prod_{j=t-k+1}^{t-1} p(\mathbf{x}_j|\mathbf{x}_{j+1},\mathbf{x}_{t-k},\mathbf{y}_{t-k+1,j}).$$

A useful approach to dealing with nonlinearities and nonnormality is to introduce an extra latent indicator variable $\mathbf{z}_t$ and implement the filtering algorithm in an augmented model. Examples of latent variables in state-space models include Carlin et al. (1992) for non-Gaussian errors, Carter and Kohn (1994); Frühwirth-Schnatter (1994); Shephard (1994) for conditionally Gaussian models, and Stroud et al. (2002) for state-dependent variance models. For filtering and particle filtering see (Anderson and Moore, 1979; Doucet, Godsill and Andrieu, 2000) and for hidden Markov models (Lindgren, 1978; Geman and Geman, 1984; Künsch, 2001). In many cases a latent state variable $\mathbf{z}_t$ can be used so that conditioning on it makes fixed-lag smoothing for $\mathbf{x}_t$ available in a block fashion, for example using FFBS. This helps with the computationally speed where a direct MCMC approach would be too slow.

2.2 Diagnostics for Fixed-Lag Filtering

An important choice in fixed-lag filtering is the lag length k. It needs to be chosen such that the samples $\mathbf{x}_{t-k}^{(i)} \sim p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t-1})$ from the previous step can be thought of as approximate samples from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t})$ in the next step. This depends on the sensitivity of the predictive distribution to the initial value. See LeGland and Mevel (1997); Künsch (2001) for a discussion from a theoretical perspective.

From a practical perspective, there are a number of ways the researcher can specify k. First, one can directly compute a diagnostic measure D_k comparing the distance between the two predictive distributions. Plotting as a function of k provides intuition into a reasonable choice of lag-length. Choosing for distance metrics D_k can be based on theoretical consideration; for example a χ^2 or total variation distance. Another possibility is to perform MCMC smoothing for the distributions $p(\mathbf{x}_{0,t-1}|\mathbf{y}_{1,t-1})$ and $p(\mathbf{x}_{0,t}|\mathbf{y}_{1,t})$ and compare the marginal distributions $p(\mathbf{x}_{t-k}|\mathbf{y}_{1,t-1})$ and $p(\mathbf{x}_{t-k}|\mathbf{y}_{1,t})$ for a particular value of k. If the densities are close, then $\mathbf{x}_{t-k}$ and $\mathbf{y}_t$ are approximately independent given $\mathbf{y}_{t-k+1,t-1}$.

For example, we can estimate the predictive distribution using the practical filter using

$$\hat{p}(\mathbf{y}_{t+1}|\mathbf{y}_{1,t}) = \frac{1}{M}\sum_{i=1}^{M} p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}^{(i)})$$

where $\mathbf{x}_{t+1}^{(i)}$ is generated from $p(\mathbf{x}_{t+1}|\mathbf{x}_t^{(i)})$ and $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{y}_{1,t})$. Then we can generate a predictive sample for as we increase the lag and find out when the sensitivity to k decays.

Clearly, there are a number of cases where a fixed-lag filtering approach is inappropriate. For example, in some nonstationary environments or long memory processes a fixed choice of k leads to a poor approximation. Possible remedies include using a stochastic choice of k maybe even as a function of historical MCMC draws or the use of periodic refreshing which we discuss in the next section. Although these strategies increase the computational time in many situations the fixed-lag-filter approach is the only available methodology.

3 Sequential Parameter Learning and Filtering

One of the main advantages of our algorithm is the ability of the fixed-lag-filter to allow sequential parameter learning. State filtering and sequential parameter learning now requires the sequence of joint posterior distributions $p(\mathbf{x}_t, \theta|\mathbf{y}_{1,t})$ The implied marginal state filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1,t})$ and sequential parameter learning distribution $p(\theta|\mathbf{y}_{1,t})$ assess the uncertainty in $\mathbf{x}_t$ and θ, respectively.

Arguing as before in (2) and (3), we consider the lag-k smoothing distribution, now augmented with the static parameter θ:

$$p(\mathbf{x}_{t-k+1,t}, \theta|\mathbf{y}_{1,t}) = \int p(\mathbf{x}_{t-k+1,t}, \theta|\mathbf{x}_{0,t-k}, \mathbf{y}_{1,t})\ dp(\mathbf{x}_{0,t-k}|\mathbf{y}_{1,t})$$
$$\approx \int p(\mathbf{x}_{t-k+1,t}, \theta|\mathbf{x}_{0,t-k}, \mathbf{y}_{1,t})\ dp(\mathbf{x}_{0,t-k}|\mathbf{y}_{1,t-1}). \quad (4)$$

Again we use the approximation that draws from $p(\mathbf{x}_{0,t-k}|\mathbf{y}_{1,t-1})$ are approximate draws from $p(\mathbf{x}_{0,t-k}|\mathbf{y}_{1,t})$. In contrast to (2) we can not use the Markov property of the dynamic model to simplify further. The Markov property holds only conditional on θ. Marginally, after integrating out θ, the states $\mathbf{x}_{0,t}$ are dependent. We proceed in a slightly different fashion as follows. First, assume that $\tilde{\mathbf{x}}^{(i)}_{0,t-k}$ are samples from $p(\tilde{\mathbf{x}}_{0,t-k}|\mathbf{y}_{1,t-1})$ which are approximate samples from $p(\tilde{\mathbf{x}}_{0,t-k}|\mathbf{y}_{1,t})$, using the same argument as the fixed parameter case. Next, we generate from $p(\mathbf{x}_{t-k+1,t}, \theta|\tilde{\mathbf{x}}_{0,t-k}, \mathbf{y}_{1,t})$. This is achieved by repeated simulation from

$$p(\mathbf{x}_{t-k+1,t}|\tilde{\mathbf{x}}^{(i)}_{t-k}, \theta, \mathbf{y}_{t-k+1,t}) \text{ and } p(\theta|\tilde{\mathbf{x}}^{(i)}_{0,t-k}, \mathbf{x}_{t-k+1,t}, \mathbf{y}_{1,t}), \quad (5)$$

Here, we have exploited conditional independence of $\mathbf{x}_{t-k+1,t}$ and $(\mathbf{x}_{0,t-k-1}, \mathbf{y}_{1,t-k})$ given $(\mathbf{x}_{t-k}, \theta)$, that is, the Markov property of the dynamic model given the static parameter. Now, repeated sampling from (5) implements a two-step Gibbs sampler to simulate from $p(\mathbf{x}_{t-k+1,t}, \theta|\tilde{\mathbf{x}}_{0,t-k}, \mathbf{y}_{1,t})$. Save the last imputed value $\mathbf{x}_{t-k+1}$ as $\tilde{\mathbf{x}}^{(i)}_{t-k+1}$. And by (4), we can report $\mathbf{x}_t$ as a draw from the filter distribution, as desired. This is summarized in the following algorithm.

Algorithm 2: Filtering with Parameter Learning

Initialization: Set $\theta^{(i)} = \theta_0$, $i = 1, \ldots, M$. Use, for example, the prior mean.

Burn-In: For $t = 1, \ldots, k$:

For $i = 1, \ldots, M$: initialize $\theta = \theta^{(i)}$.

For $g = 1, \ldots, G$:

Generate $\mathbf{x}_{0,t} \sim p(\mathbf{x}_{0,t}|\theta, \mathbf{y}_{1,t})$

Generate $\theta \sim p(\theta|\mathbf{x}_{0,t}, \mathbf{y}_{1,t})$

Set $(\tilde{\mathbf{x}}^{(i)}_0, \theta^{(i)})$ equal to the last imputed $(\mathbf{x}_0, \theta)$.

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

For $i = 1, \ldots, M$: initialize $\theta = \theta^{(i)}$.

For $g = 1, \ldots, G$:

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

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

Set $(\tilde{\mathbf{x}}^{(i)}_{t-k+1}, \theta^{(i)})$ equal to the last imputed $(\mathbf{x}_{t-k+1}, \theta)$ and leave $\tilde{\mathbf{x}}^{(i)}_{0,t-k}$ unchanged.

The filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1,t})$ can then be approximated by the empirical distribution of imputed $\tilde{\mathbf{x}}^{(i)}_t$. Alternatively, depending on the nature of the lag smoother applied in the algorithm, we can use Rao-Blackwellization and evaluate the filtering distribution as an average across filtering distributions conditional on imputed histories $\tilde{\mathbf{x}}^{(i)}_{0,t-k}$:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1,t}) &= \int\int p(\mathbf{x}_t|\mathbf{x}_{1,t-k}, \theta, \mathbf{y}_{t-k+1,t})\; p(\theta|\mathbf{x}_{1,t-k}, \mathbf{y}_{1,t}) d\theta\; p(\mathbf{x}_{1,t-k}|\mathbf{y}_{1,t})\; d\mathbf{x}_{1,t-k} \\ &\approx \frac{1}{M}\sum_{i=1}^{M} \int p(\mathbf{x}_t|\tilde{\mathbf{x}}^{(i)}_{0,t-k}, \theta, \mathbf{y}_{t-k+1,t})\; p(\theta|\tilde{\mathbf{x}}^{(i)}_{1,t-k}, \mathbf{y}_{1,t}) d\theta \end{aligned} \tag{6}$$

Evaluation the predictive distribution

$$p(\mathbf{x}_t|\mathbf{y}_{1,t}, \tilde{\mathbf{x}}^{(i)}_{0,t-k}) = \int p(\mathbf{x}_t|\tilde{\mathbf{x}}^{(i)}_{0,t-k}, \theta, \mathbf{y}_{t-k+1,t})\; p(\theta|\tilde{\mathbf{x}}^{(i)}_{0,t-k}, \mathbf{y}_{1,t}) d\theta$$

is part of the lag smoother. The sequential parameter posterior distribution can similarly be approximated either as empirical distribution of the imputed $\theta^{(i)}$, or via the identity

$$\begin{aligned} p(\theta|\mathbf{y}_{1,t}) &= \int\int p(\theta|\tilde{\mathbf{x}}^{(i)}_{1,t-k}, \tilde{\mathbf{x}}^{(i)}_{t-k+1,t}, \mathbf{y}_{t-k+1,t})\; p(\tilde{\mathbf{x}}^{(i)}_{t-k+1,t}|\tilde{\mathbf{x}}^{(i)}_{t-k}, \mathbf{y}_{1,t}) d\tilde{\mathbf{x}}^{(i)}_{t-k+1,t} p(\mathbf{x}_{1,t-k}|\mathbf{y}_{1,t})\; d\mathbf{x}_{1,t-k} \\ &\approx \frac{1}{M}\sum_{i=1}^{M} \int p(\theta|\tilde{\mathbf{x}}^{(i)}_{1,t-k}, \tilde{\mathbf{x}}^{(i)}_{t-k+1,t}, \mathbf{y}_{t-k+1,t})\; p(\tilde{\mathbf{x}}^{(i)}_{t-k+1,t}|\tilde{\mathbf{x}}^{(i)}_{t-k}, \mathbf{y}_{1,t}) d\tilde{\mathbf{x}}^{(i)}_{t-k+1,t} \end{aligned} \tag{7}$$

The speed and accuracy of the algorithm depends on a number of choices. In particular the researcher has to specify the three parameters (M, G, k). The parameter M corresponds to the number of saved histories that will be used to perform Monte Carlo estimation. As these samples are independent a relatively small number is required. Typically values in applications are $M = 1000$. The parameter G corresponds to the number of MCMC iterations required to perform fixed-lag filtering. In many case we can choose $G = 1$ and use FFBS. However, in sequential parameter learning problems some MCMC iteration will be required. Due to the sequential nature of the problem, the initial starting values will already be close to the desired stationary distribution and again G will be reasonably small. Finally, the choice of k has to be determined to ensure a good approximation to the underlying filtering distributions. For static parameter this was discussed this in detail in Section 2.2. For the parameter learning case, the choice of the lag-length length k is related to how the parameter θ introduces dependence into the state equation. If θ enters the model only in the evolution equation then the choice of appropriate lag k could be highly dependent on the posterior range of θ. In other words, in trying to find a value of k such that $\mathbf{x}_{t-k}$ and $\mathbf{x}_{t+1}$ are approximately conditionally independent given $\mathbf{y}_{t-k,t}$ could critically depend on plausible values of θ. If necessary we suggest an adaptive choice of k. During an initial burn in $t < t_0$ one could use a larger window k_0 paralleling the common practice of adjusting MCMC implementations over some finite initial burn-in period.

Finally, one might be able to explit a sufficient structure for the θ update from $p(\theta|\tilde{\mathbf{x}}^{(i)}_{0,t-k}, \mathbf{x}_{t-k+1}, y_{1,t})$. More formally, supposed that there exists a sufficient statistic $s(\mathbf{x}_{0,t-k})$ such that $\theta \sim p(\theta|s(\mathbf{x}^{(i)}_{0,t-k}, \mathbf{x}_{t-k+1,t}, \mathbf{y}_{1,t})$.

Here we let the sufficient statistic be a function of past states namely $s(\mathbf{x}_{0,t-k})$ and to draw iteratively from the joint posterior $p(s(\mathbf{x}_{0,t-k}), \theta, \mathbf{x}_{t-k+1,t}|\mathbf{y}_{1,t})$. This has the added advantage of refreshing the initial values of the sufficient statistic and hence improving the approximation but has the disadvantage that we need to be able to simulate from the further conditional $p(s(\mathbf{x}_{0,t-k}), \theta, \mathbf{x}_{t-k+1,t}|\mathbf{y}_{1,t})$.

4 Applications

To illustrate our methodology we consider two applications. First, we use a simulated dataset from a stochastic volatility model to compare our algorithm for filtering with known parameters (Algorithm 1) with standard particle filtering methods. Then we use daily data from the Standard and Poor's S&P500 index and compare our method for sequential parameter learning (Algorithm 2) with the particle filtering approach of Storvik (2002). Secondly, we show how to perform sequential parameter learning for a three-dimensional stochastic Lorenz flow model.

4.1 Stochastic Volatility

Consider a standard univariate log-stochastic volatility model (Jacquier, Polson and Rossi, 1994) from financial time series. The returns y_t and log-volatility states x_t follow a non-Gaussian state space model of the form

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

with $x_0 \sim \mathcal{N}(m_0, C_0)$ and ϵ_t and η_t are iid $\mathcal{N}(0,1)$. Here the parameter vector is $\theta = (\alpha, \beta, \sigma_v^2)$. For our initial analysis, we use a simulated datset of length $T = 500$ with parameter values $(\alpha, \beta, \sigma_v^2) = (-.0084, .979, .04)$ which are typical of many financial series.

Algorithm 1 requires the choice of a fixed-lag-filter. Depending on the application the sensitivity of the approximation to the choice of k should be carefully studied. For these applications we found that by studying the quantiles for the filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1,t})$ using QQ-plots across a wide range of choices we found that $k = 25$ is reasonable in these SV applications. For the choice of parameter G we use the block updating scheme such as the using mixture-of-normals and FFBS approach of Carter and Kohn (1994). In the static parameter case this allows us to use $G = 1$ and for the parameter learning case we can use a relatively small value of G.

Particle filtering methods are straightforward to apply. Here the current filtering distribution is summarized by large number of particles, typically $N = 100,000$, and particle weights that are sequentially resampled according to the observation likelihood. Figure 1 plots the filtered states using our methodology and the particle filter for the simulated dataset. In this univariate example with no parameter learning there is no advantage to using the practical filter. The particle filter does not degenerate and the number of particles N is typically less that the $M \times G \times k$ iteration required for fixed-lag-filtering.

The main advantage of our approach is its ability to handle to case of sequential parameter learning. Here we use daily data on the S&P500 stock index from 1980-2000 shown in Figure 2. The most noticeable feature of data is the negative 20% drop in the stock market crash of October 29, 1987. Our sequential learning algorithm (Algorithm 2) is straightforward to apply. We use the same parameter triple (M, G, k) as used above and we also exploited a sufficient statistic structure for updating θ so as to avoid storing past histories of states.

Specifically, the sufficient statistic structure is determined as follows. Let $\psi = (\alpha, \beta)'$ denote the parameter vector. We assume a conjugate prior on $\theta = (\psi, \sigma_v^2)$ of the form $\sigma_v^2 \sim \mathcal{IG}(n_0/2, d_0/2)$ and $\psi|\sigma_v^2 \sim \mathcal{N}\left(\psi_0, A_0^{-1}\sigma_v^2\right)$. This leads to the full conditionals at time t:

$$\sigma_v^2|\mathbf{x}_{0,t}, \mathbf{y}_{1,t} \sim \mathcal{IG}(n_t/2, d_t/2) \text{ and } \psi|\sigma_v^2, \mathbf{x}_{0,t}, \mathbf{y}_{1,t} \sim \mathcal{N}\left(\psi_t, A_t^{-1}\sigma_v^2\right).$$

The parameters of this conditional posterior define our sufficient statistics $\mathbf{s}_t = \{n_t, d_t, \psi_t, A_t\}$ and the lag-k updating recursions are given by $n_t = n_{t-k} + k$, $d_t = d_{t-k} + (\mathbf{x} - H\psi_{t-k})'(\mathbf{x} - H\psi_{t-k})$, $\psi_t = A_t\left(A_{t-k}^{-1}\psi_{t-k} + H'\mathbf{x}\right)$, and $A_t = A_{t-k} + H'H$. Here $\mathbf{x} = \mathbf{x}_{t-k+1,t}$, $\tilde{\mathbf{x}} = \mathbf{x}_{t-k,t-1}$, $H = (H_{t-k+1}, \ldots, H_t)'$, and $H_t = (1, x_{t-1})'$. For the results reported below, we choose diffuse priors centered at the true values: $n_0 = 2$, $d_0 = .08$, $\psi_0 = (-.084, .979)'$ and $A_0^{-1} = \text{diag}(100, 100)$.

To compare our methodology to sequential particle filtering methods we used the algorithm of Storvik (2002). This is a sequential importance sampling procedure that assumes an initial set of N particles $(\mathbf{x}_t, \theta) \sim p(\mathbf{x}_t, \theta|\mathbf{y}_{1,t})$. Letting s_t denote the posterior sufficient statistics that are updated at each iteration, the algorithm then draws

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

and reweights $(\mathbf{x}_{t+1}, \theta)$ with weights proportional to the observation likelihood $p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}, \theta)$.

Figure 1 uses a simulated dataset $y(t)$ and plots the filtered states and sequential parameter estimates for the auxiliary particle filter (APF) of Storvik (2002) versus the practical filter. Specifically, for both models we compare the 2.5,50,97.5% quantiles versus the estimates obtain via a full MCMC sequential analysis. The practical filter uses $(M = 250, G = 25, k = 25)$ and the APF uses a large number of initial particles $N = 100,000$. The running times for both algorithms are similar with the practical filter slightly faster. The large number of inital particles should ensure a good approximation. For state estimation both methods recover the underlying states with a high degree of accuracy. It is in the case of sequential parameter learning where the benefits of the practical filter become apparent. Notice that the APF performs poorly on recovering the quantiles of the posterior distributions for α and is particularly bad in sequentially estimting the persistence parameter β. On the other hand, the practical filter delivers a similar accuracy to that of state estimation. To study the performance of both algorithms further we used daily data on the S&P500 stock index from 19..- 19.. This included the carsh of 1987 on Oct ... when the index fell to -20%. This provides a good test case to see how both methods perform with an outlying observation. and the use of a possibly misspecified model. The bottom panel compare the APF with the practical filter. Notice that the sequential parameter estimates form the APF virtually degenerate even with the large number of initial particles. The practical filter, on tbe other hand, does a good job of uncovering the posterior mean and quantiles. The only place of difference with the full MCMC analysis is in the days after the crash of 1987 which is to be expected. It is here where the approximation used by the practical filter is most likely to be poor.

Finally, we provide an analysis of the root mean squared error (RMSE) for both methods on the two examples considered above. Table 1 provides the summary statistics. Notice that in both cases over a range of possible algorithmic setting that the RMSE of the practical filter is above twice as good as that for the APF where estimating the states and the parameter α. Not surprisingly given the previous results, there is a larger increase for estimating β where the RMSE can increase by a factor of five.

To study the parameter learning further, Tables 1 and 2 show the root mean squared errors for Storvik's algorithm and for Practical filtering (Algorithm 2).

Table 1: RMSE's for simulated SV dataset.

Algorithm	params	v_t	α	β
Storvik	N=1000	0.0650	0.0022	0.0017
Storvik	N=5000	0.0313	0.0014	0.0009
Storvik	N=10,000	0.0709	0.0038	0.0050
Storvik	N=25,000	0.0635	0.0033	0.0042
Storvik	N=50,000	0.0729	0.0039	0.0052
Storvik	N=100,000	0.0727	0.0043	0.0053
Practical	M=250, G=10, k=10	0.0590	0.0053	0.0087
Practical	M=250, G=10, k=25	0.0837	0.0046	0.0095
Practical	M=250, G=10, k=50	0.0864	0.0044	0.0098
Practical	M=250, G=25, k=10	0.1153	0.0037	0.0027
Practical	M=250, G=25, k=25	0.0417	0.0022	0.0011
Practical	M=250, G=25, k=50	0.0408	0.0021	0.0010

Table 2: RMSE's for S&P 500 dataset.

Algorithm	params	v_t	α	β
Storvik	N=1,000	0.0915	0.0051	0.0087
Storvik	N=5,000	0.0747	0.0052	0.0073
Storvik	N=10,000	0.0857	0.0148	0.0139
Storvik	N=25,000	0.0656	0.0090	0.0098
Storvik	N=50,000	0.0712	0.0105	0.0104
Storvik	N=100,000	0.0660	0.0098	0.0102
Practical	M=250, G=10, k=10	0.0599	0.0039	0.0048
Practical	M=250, G=10, k=25	0.0434	0.0024	0.0025
Practical	M=250, G=10, k=50	0.0432	0.0024	0.0024
Practical	M=250, G=25, k=10	0.0641	0.0042	0.0051
Practical	M=250, G=25, k=25	0.0432	0.0024	0.0025
Practical	M=250, G=25, k=50	0.0424	0.0024	0.0024

Polson et al. (2002) consider the case of learning sequentially about the volatility of volatility parameter σ_v. In this case the assumption that the information contained in $\mathbf{y}_{t+1}$ decays exponentially fast is not valid. The APF method degenerates very quickly after the crash and the practical filter provides good point estimation but has posterior quantiles that are too smooth relative to the full MCMC analysis. The algorithm with refreshing only alleviates this problem marginally and the issues of learning parameters that have long memory in terms of the unobserved states is an open problem.

4.2 Stochastic Lorenz Model

The Lorenz model is a three-dimensional coupled system of nonlinear differential equations to explain dynamic flow. Specifically, Lorenz (1963) proposed a deterministic system:

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= \rho x - y - xz, \\ \dot{z} &= xy - \beta z, \end{aligned}$$

where the dot denotes a time derivative; $\mathbf{x}(t) = (x(t), y(t), z(t))$ is the state vector, and $\psi = (\sigma, \rho, \beta)$ is the parameter vector.

We observe the system in discrete time with a stochastic evolution error; hence we refer to this as a stochastic Lorenz model. For our time discretization, we use a simple Euler scheme with time step Δ:

$$x_{(t+1)\Delta} = x_{t\Delta} + \sigma(y_{t\Delta} - x_{t\Delta})\Delta + w\sqrt{\Delta}\,\eta^x_{t\Delta} \quad (8)$$

$$y_{(t+1)\Delta} = y_{t\Delta} + (\rho x_{t\Delta} - y_{t\Delta} - x_{t\Delta}z_{t\Delta})\Delta + w\sqrt{\Delta}\,\eta^y_{t\Delta} \quad (9)$$

$$z_{(t+1)\Delta} = z_{t\Delta} + (x_{t\Delta}y_{t\Delta} - \beta z_{t\Delta})\Delta + w\sqrt{\Delta}\,\eta^z_{t\Delta}. \quad (10)$$

At each time t we observe a set of noisy measurements, $\mathbf{y}_t = (\tilde{x}_t, \tilde{y}_t, \tilde{z}_t)'$, which are related to the state vector through the measurement equation:

$$\mathbf{y}_t = \mathbf{x}_t + v\,\epsilon_t.$$

The practical filter can be used with a fixed static parameter and compared to the particle filtering approach proposed by Evensen (1994) and Bengtsson and Nychka (2001). Moreover, Algorithm 2 can account for sequential parameter learning and we will provide sequential posterior parameter distributions for the underlying parameter governing the state evolution. We show that a natural Gibbs sampling approach applies to simulate from the lag-smoothing distribution in our filtering algorithm. This provides a computationally fast approach for implementing our method.

To study our first filtering algorithm, we generate a path of $T = 200$ observations, $\mathbf{y}_1, \ldots, \mathbf{y}_{200}$ from the stochastic Lorenz model above with parameter values $\psi = (\sigma, \rho, \beta) = (10, 28, 2.67)$, and variance parameters $w^2 = 0.5$, $v^2 = 2.0$, with an initial state vector $\mathbf{x}_0 = (-5.9, -5.5, 24.6)$. Our choice of parameters ψ corresponds to the original values used by Lorenz (1963). Figure 3 shows the true state trajectory along with the filtered means $E(\mathbf{x}_t|\mathbf{y}_{1,t})$.

The practical filter (Algorithm 1) for filtering with fixed parameters requires a fast lag-smoothing algorithm to update the states from the distribution $p(\mathbf{x}_{t-k,t}|\theta, \mathbf{y}_{1,t})$. Even though there are non-linearities induced in the mean functions of $y_{t+\Delta}$ and $z_{t+\Delta}$ we now show that a Gibbs blocking algorithm that uses the fast FFBS algorithm can be implemented that is generating in a Gibbs block fashion from $p(\mathbf{y}, \mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{y}, \mathbf{z})$.

This two-block Gibbs sampling algorithm exploits the multiplicative structure of the evolution mean function. This allows us to obtain a linear evolution for a subset of the states if we condition on the rest. Specifically, let $x = (x_{t-k+1}, \ldots, x_t)'$, $y = (y_{t-k+1}, \ldots, y_t)'$ and $z = (z_{t-k+1}, \ldots, z_t)'$. To generate the states $\mathbf{x} = (x, y, z)$, we first generate $x \sim p(x|y, z, \theta)$ and then $(y, z) \sim p(y, z|x, \theta)$. Both of these full conditionals can be recognized as the smoothing distribution in a linear state-space model, and hence we can sample each block efficiently using the FFBS algorithm.

For the x update, consider a state space model with observation equation given by:

$$\begin{pmatrix} \tilde{x}_t \\ \dot{y}_{t+1} + y_t \\ \dot{z}_{t+1} + \beta z_t \end{pmatrix} = \begin{pmatrix} 1 \\ \rho - z_t \\ y_t \end{pmatrix} x_t + \begin{pmatrix} \epsilon^x_t \\ \eta^y_t \\ \eta^z_t \end{pmatrix},$$

where $(\epsilon^x_t, \eta^y_t, \eta^z_t)' \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(v^2, \Delta w^2, \Delta w^2))$ and $\dot{}$ denotes the first-order difference operator defined by $\dot{w}_t = \Delta^{-1}(w_t - w_{t-1})$. We the combine this augmented observation equation with the system equation (1) to obtain a linear state-space model for x. Now, conditioning on (y, z), we can generate the vector x as a block using FFBS.

For the (y, z) update, consider a model with observation equation given by

$$\begin{pmatrix} \dot{x}_{t+1} + x_t \\ \tilde{y}_t \\ \tilde{z}_t \end{pmatrix} = \begin{pmatrix} \sigma & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} y_t \\ z_t \end{pmatrix} + \begin{pmatrix} \eta^x_t \\ \epsilon^y_t \\ \epsilon^z_t \end{pmatrix},$$

where $(\eta_t^x, \epsilon_t^y, \epsilon_t^z)' \sim \mathcal{N}(\mathbf{0}, \text{diag}(\Delta w^2, v^2, v^2))$. This can be combined with the evolution equation in (2)–(3) to define a linear state-space model for (y, z). And hence we can simulate (y, z) as a block using the FFBS algorithm.

We now turn to the problem of sequential parameter learning.

4.3 Sequential Parameter Learning

For sequential parameter learning, we need to specify the distribution of the parameters $\theta = (v^2, w^2, \psi)$. Conjugate priors are available for all parameters: namely, $v^2 \sim \mathcal{IG}(n_0/2, d_0/2)$, and $w^2 \sim \mathcal{IG}(m_0/2, e_0/2)$ and $\psi|w^2 \sim \mathcal{N}(\psi_0, A_0^{-1}w^2)$, where $\{n_0, d_0, m_0, e_0, \psi_0, A_0\}$ are hyperparameters specified by the user. We assume that v^2 and (w^2, ψ) are independent *a priorí* so that $\pi(\theta) = \pi(v^2)\pi(w^2)\pi(\psi|w^2)$. In order to implement our Algorithm 2, we need the joint posterior $p(v^2, w^2, \psi|\mathbf{x}_{0,t}, \mathbf{y}_{1,t})$.

The conjugate prior leads to a closed-form posterior of the form

$$\begin{aligned} v^2|\mathbf{x}_{0,t}, \mathbf{y}_{1,t} &\sim \mathcal{IG}(n_t/2, d_t/2) \\ w^2|\mathbf{x}_{0,t}, \mathbf{y}_{1,t} &\sim \mathcal{IG}(m_t/2, e_t/2) \\ \psi|w^2, \mathbf{x}_{0,t}, \mathbf{y}_{1,t} &\sim \mathcal{N}\left(\psi_t, A_t^{-1}\sigma_v^2\right). \end{aligned}$$

So the sufficient statistics are $s_t = \{n_t, d_t, m_t, e_t, \psi_t, A_t\}$ and the lag-k updating recursions are given by $n_t = n_{t-k} + 3k$, $d_t = d_{t-k} + (\mathbf{y} - \mathbf{x})'(\mathbf{y} - \mathbf{x})$, $m_t = m_{t-k} + 3k\Delta$, $e_t = e_{t-k} + (\mathbf{x} - H\tilde{\mathbf{x}})'(\mathbf{x} - H\tilde{\mathbf{x}})$, $\psi_t = A_t(A_{t-k}^{-1}\psi_{t-k} + H\mathbf{x})$, and $A_t = A_{t-k} + H'H$. Here we have defined $\mathbf{y} = \mathbf{y}_{t-k+1,t}$, $\mathbf{x} = \hat{\mathbf{x}}_{t-k+1,t}$, $\tilde{\mathbf{x}} = \hat{\mathbf{x}}_{t-k,t-1}$, $\hat{\mathbf{x}} = (\dot{x}_t, \dot{y}_t + x_{t-1}z_{t-1}, \dot{z}_t - x_{t-1}y_{t-1})'$, $H = (H_{t-k+1}, \ldots, H_t)'$, and $H_t = \text{Diag}\ (y_{t-1} - x_{t-1}, x_{t-1}, -z_{t-1})$.

For our analysis, we choose diffuse priors defined by $n_0 = m_0 = 2$, $d_0 = e_0 = 2$, $\psi_0 = (10, 28, 2.67)'$ and $A_0 = \text{Diag}\ (100, 100, 100)$, and a diffuse prior for the intial state, $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \text{Diag}\ (100, 100, 100))$. Figure 4 shows the filtered paths for the state variables $(x(t), y(t), z(t))$ together with the sequential parameter plots for the parameters. First, notice the close agreement with the true simulated values for the state estimation. There is little error with the practical filter. Moreover, as we learn the true parameter values over time, we see that the mean level parameters are the most straightfoward to learn, namely (σ, ρ, β). The observation and evolution variances take longer to converge to the true values but again after $T = 500$ time steps, our marginal posteriors are in agreement with the underlying parameter values.

5 Conclusions

In this paper we show how to use a fixed-lag-filtering method for state filtering and sequential parameter learning in general state space models. Our approach requires the choice of a fixed-lag-smoothing algorithm and a fast MCMC algorithm to draw the states. Our methodology is particularly attractive as it is straightforward to incorporate sequential parameter learning. Parameter uncertainty can then be incorporated into state filtering. Our methodology also does not suffer from particle degeneracies which hinder the automatic use of particle filtering methods and it can also be applied in high dimensions.

Sequential parameter learning still poses a number of computational challenges. First, parameters that are in the evolution equation are notoriously hard to estimate sequentially. This is partly due to the long memory in the process. Little research has been done on long-memory processes and how to provide filtered states and parameters. There are a number of recent applications to continuous time models and jump models; for example, see Pitt and Shephard (1999) and Johannes,

Polson and Stroud (2002a). For an application to sequential portfolio allocation see Johannes, Polson and Stroud (2002b).

References

Anderson, B. and Moore, J. (1979) *Optimal Filtering*. Englewood Cliffs: Prentice Hall.

Bengtsson, T. and Nychka, D. (2001) Adaptive methods in numerical weather prediction. *Tech. rep.*, Geophysical Statistics Project, National Center for Atmospheric Research.

Carlin, B., Polson, N. and Stoffer, D. (1992) A Monte Carlo approach to nonnormal and nonlinear state-space modeling. *Journal of the American Statistical Association*, **87**, 493–500.

Carpenter, J., Clifford, P. and Fearnhead, P. (1999) An improved particle filter for nonlinear problems. *IEE Proceedings – Radar, Sonar and Navigation*, **146**, 2–7.

Carter, C. and Kohn, R. (1994) On Gibbs sampling for state space models. *Biometrika*, **81**, 541–553.

Clapp, T. and Godsill, S. (1999) Fixed-lag smoothing using sequential importance sampling. In *Bayesian Statistics 6* (eds. J. Bernardo, J. Berger, A. Dawid and A. Smith). Oxford: Oxford University Press.

Doucet, A., Godsill, S. and Andrieu, C. (2000) On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, **10**, 197–208.

Evensen, G. (1994) Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, **99**, 10,143–10,162.

Frühwirth-Schnatter, S. (1994) Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, **15**, 183–202.

Geman, S. and Geman, D. (1984) Stochastic relatation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721–741.

Gordon, N., Salmond, D. and Smith, A. (1993) Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings*, vol. F-140, 107–113. IEE.

Handschin, J. (1970) Monte Carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica*, **6**, 555–563.

Handschin, J. and Mayne, D. (1969) Monte Carlo techniques to estimate the condtional expectation in multi-stage non-linear filtering. *International Journal of Control*, **9**, 547–559.

Hürzeler, M. and Künsch, H. (1998) Monte Carlo approximations for general state-space models. *Journal of Computational and Graphical Statistics*, **7**, 175–193.

Jacquier, E., Polson, N. and Rossi, P. (1994) Bayesian analysis of stochastic volatility (with discussion). *Journal of Business and Economic Statistics*, **12**, 371–417.

Johannes, M., Polson, N. and Stroud, J. (2002a) Nonlinear filtering of stochastic differential equations with jumps. *Tech. rep.*, Graduate School of Business, University of Chicago.

— (2002b) Sequential optimal portfolio performance: Market and volatility timing. *Tech. rep.*, Graduate School of Business, University of Chicago.

Kitagawa, G. (1994) The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, **46**, 605–623.

— (1996) Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, **5**, 1–25.

Künsch, H. (2001) State space and hidden Markov models. In *Complex Stochastic Systems* (eds. D. C. O.E. Barndorff-Nielsen and C. Klüppelberg). Boca Raton: Chapman and Hall.

LeGland, F. and Mevel, L. (1997) Exponential forgetting and geometric ergodicity in hidden Markov models. In *36th IEEE Conference on Decision and Control (CDC)*, 537–542.

Lindgren, G. (1978) Markov regime models for mixed distributions and switching regressions. *Scandinavian Journal of Statistics*, **5**, 81–89.

Liu, J. and Chen, R. (1998) Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, **93**, 1032–1044.

Liu, J. and West, M. (2000) Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo Methods in Practice* (eds. A. Doucet, J. de Freitas and N. Gordon). Springer.

Lorenz, E. (1963) Deterministic non-periodic flow. *Journal of Atmospheric Science*, **20**, 130–141.

Müller, P. (1991) Monte Carlo integration in general dynamic models. *Contemporary Mathematics*, **115**, 145–163.

Pitt, M. and Shephard, N. (1999) Filtering via simulation: Auxiliary particle filter. *Journal of the American Statistical Association*, **94**, 590–599.

Polson, N., Stroud, J. and Müller, P. (2002) Practical filtering for stochastic volatility models. *Tech. rep.*, Graduate School of Business, University of Chicago.

Shephard, N. (1994) Partial non-Gaussian state space. *Biometrika*, **81**, 115–131.

Storvik, G. (2002) Particle filters in state space models with the presence of unknown static parameters. *IEEE Trans. on Signal Processing*, **50**, 281–289.

Stroud, J., Müller, P. and Polson, N. (2002) Nonlinear state-space models with state-dependent variances. *Tech. rep.*, Institute of Statistics and Decision Sciences, Duke University.

West, M. (1993) Mixture models, Monte Carlo, Bayesian updating, and dynamic models. In *Computing Science and Statistics. Proceedings of the 24rd Symposium on the Interface*, vol. 24, 325–333.

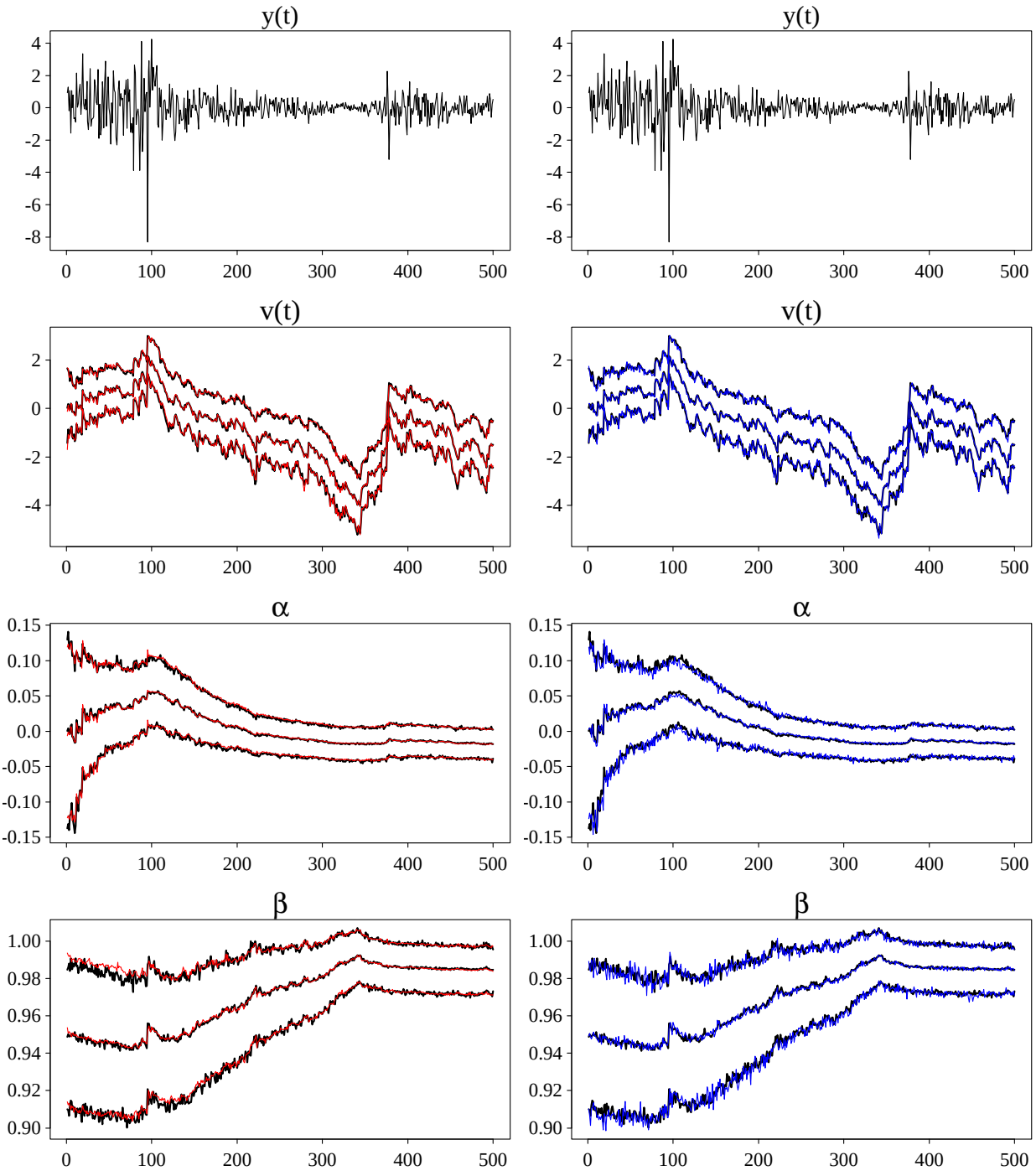

Figure 1: SV model, Simulated data. Filtered 2.5, 50, 97.5 percentiles. Left: Storvik's algorithm ($N = 100,000$ particles). Right: Practical filter ($M = 250, G = 25, k = 25$).

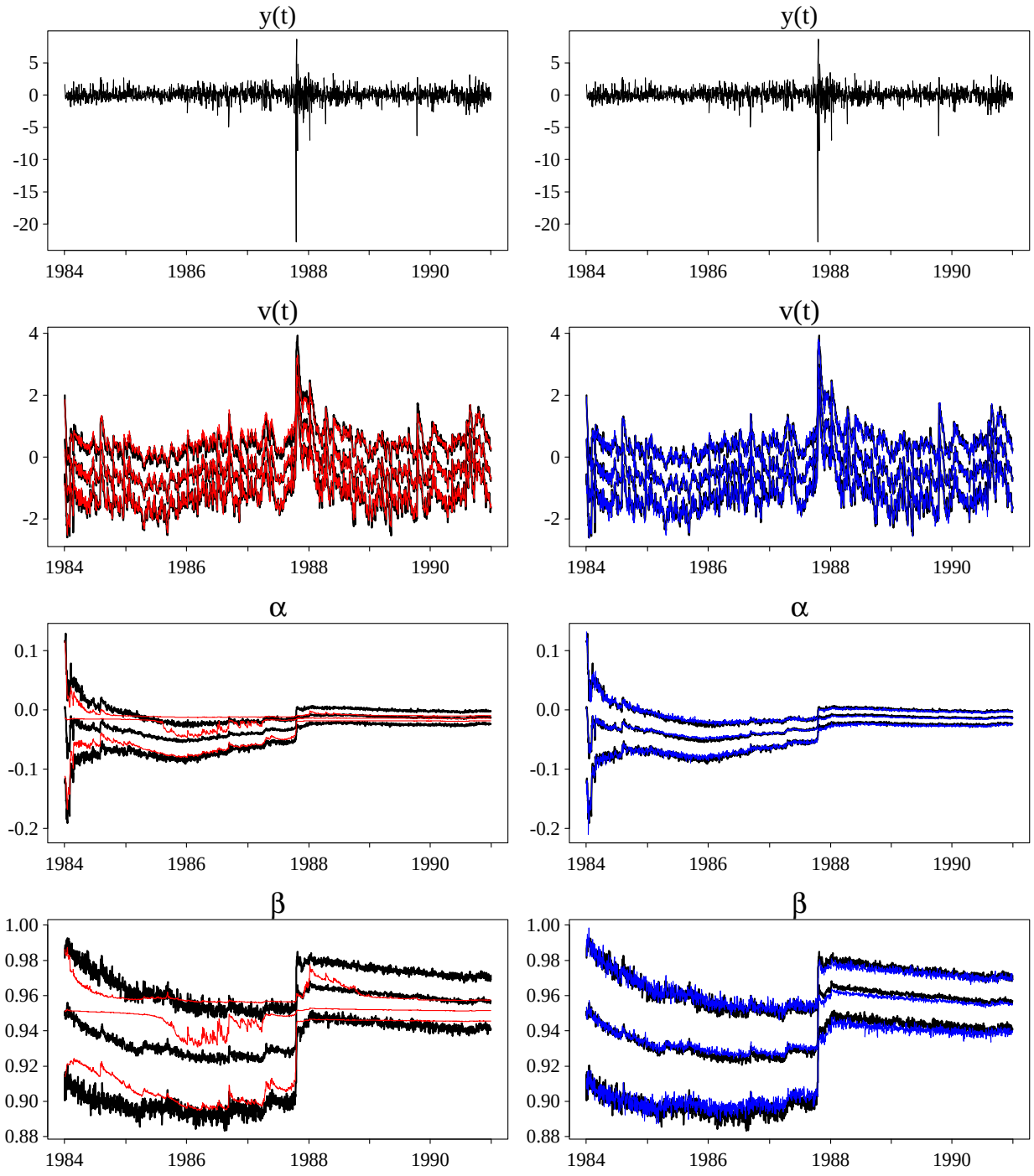

Figure 2: SV model, S&P 500 data. Filtered 2.5, 50, 97.5 percentiles. Left: Storvik's algorithm ($N = 100,000$ particles). Right: Practical filter ($M = 250, G = 25, k = 25$).

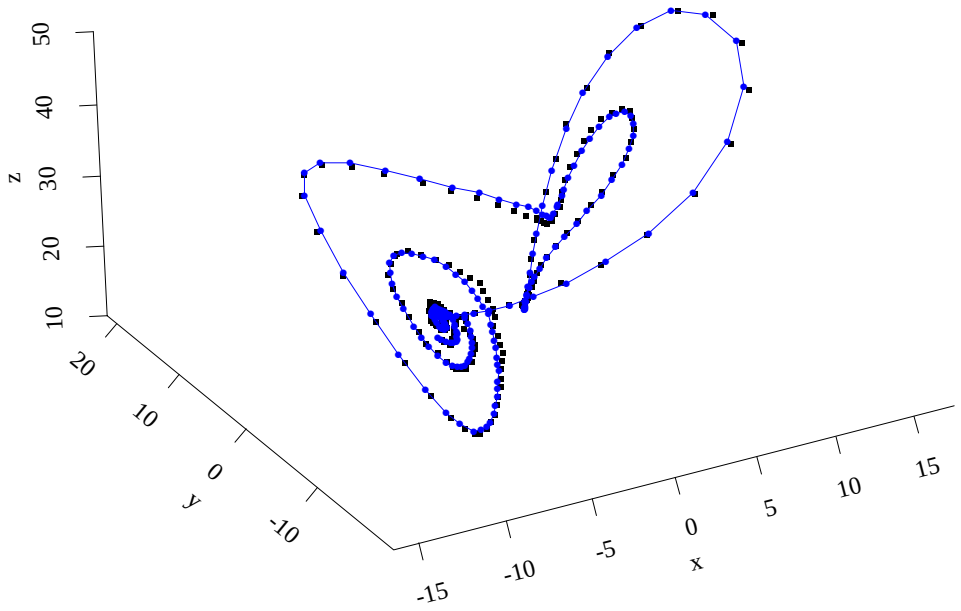

Figure 3: Lorenz model: phase-space trajectory. Simulated path of state vector (black squares), and filtered mean (blue line).

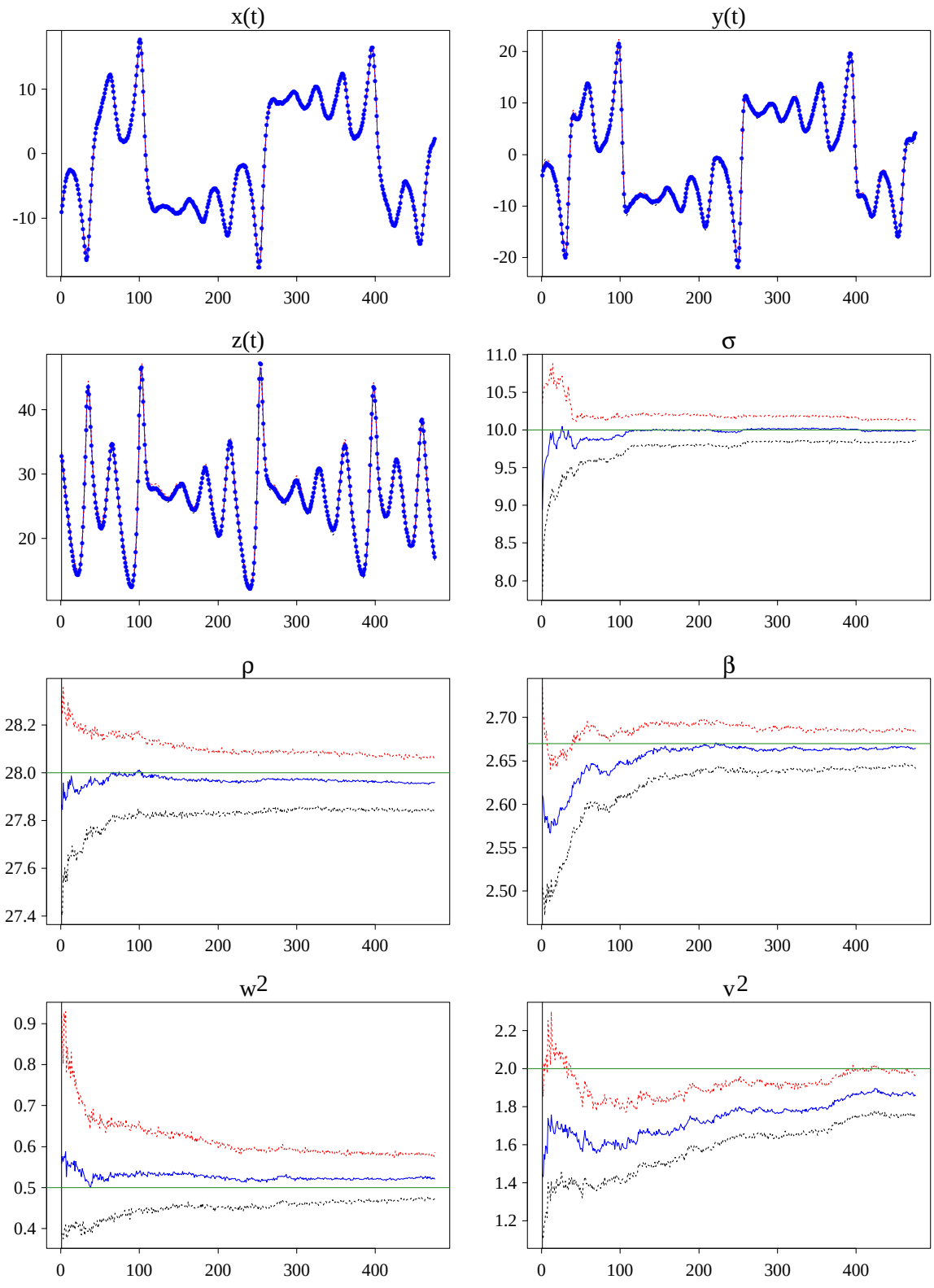

Figure 4: Lorenz model. Filtered 5, 50, 95 percentiles for states and static parameters, using the practical filter with $(M = 2500, G = 1, k = 25)$. Blue dots represent the true states. Green horizontal lines denote the true parameter values.