Practical Filtering with Sequential Parameter Learning

Nicholas G. Polson †

Graduate School of Business, University of Chicago, Chicago, IL 60637, U.S.A.

Jonathan R. Stroud

Department of Statistics, The Wharton School, University of Pennsylvania, Philadelphia, PA 19104, U.S.A.

Peter Müller

Department of Biostatistics, M. D. Anderson Cancer Center, Houston, TX 77030, U.S.A.

Summary. This paper develops a simulation-based approach to sequential parameter learning and filtering in general state-space models. Our methodology is based on a rolling-window Markov chain Monte Carlo (MCMC) approach and can be easily implemented by modifying state-space smoothing algorithms. Furthermore, the filter avoids the degeneracies that hinder particle filters and is robust to outliers. We illustrate the methodology on a benchmark autoregressive where we show the robustness to outliers. In a three-dimensional nonlinear stochastic Lorenz model, we exploit a conditionally Gaussian structure for the states to provide a fast algorithm for simultaneously performing sequential parameter learning and filtering.

Key Words: Filtering, Hidden Markov Models, Lorenz Attractor, Markov chain Monte Carlo, Particle Filtering, Sequential Parameter Learning, State-Space Models.

1. Introduction

Modern-day applications of filtering in general state-space models also require sequential parameter learning. Common examples include financial time series and spatio-temporal models. However, combined sequential parameter learning and state filtering provides many computational challenges. Standard particle filtering methods such as sampling-importance resampling (SIR, Gordon, Salmond and Smith, 1993; Liu and Chen, 1995; Kitagawa, 1996) and auxiliary particle filters (APF, Pitt and Shephard, 1999) can lead to unbalanced weights for the particles and degeneracy in the presence of outliers, model misspecification and high-dimensional systems. Furthermore, the degeneracy problem is typically exacerbated when sequential parameter estimation is also required (see, for example, Andrieu, Doucet and Tadić, 2005). Our approach avoids the degeneracy problem and provides a straightforward approach to the combined filtering and sequential parameter learning problem.

Sequential parameter learning algorithms have been proposed within the particle filter framework. In particular, Gordon, Salmond and Smith (1993) augment the state vector to include the static parameters; Berzuini, Best, Gilks and Larizza (1997) use Markov chain Monte Carlo (MCMC) approaches within the particle filter; Liu and West (2001) propose a kernel density approach, whereas Hürzeler and Künsch (2001) and Pitt (2002) use likelihood-based methods. Andrieu and Doucet (2003) and Andrieu, Doucet and Tadić (2005) consider recursive and batch maximum likelihood methods based on stochastic gradients and expectation-maximisation (EM) approaches. Our approach is most closely related to Storvik (2002) who proposes a Bayesian approach based on sufficient statistics.

Our approach uses a rolling-window MCMC algorithm. The basic assumption is that at each time t, there exists a lag k such that the joint distribution of the initial $t-k$ states is unaffected by all observations after time t, conditional on the intermediate observations. The advantage of our methodology is that it is straightforward to implement and effective in many situations, particularly for conditionally Gaussian models with unknown parameters where MCMC smoothing methods have been well developed. By recasting the filtering problem as a sequence of small smoothing problems we can use standard MCMC approaches (Carlin, Polson and Stoffer, 1992; Carter and Kohn, 1994) to provide a simulation-based solution. The algorithm

†*Address for correspondence:* Nicholas G. Polson, Graduate School of Business, University of Chicago, 5807 South Woodlawn Avenue, Chicago, IL 60637, U.S.A. E-mail: **ngp@gsb.uchicago.edu**

does not reweight or resample particles and hence it avoids the degeneracy problem. The key features of our approach are the inclusion of static parameters, a linear computational effort, and a clever mechanism to exploit sufficient statistics to update parameter inference.

Following Storvik (2002), we update sequential parameter posterior distributions by exploiting a sufficient statistic structure. While Storvik's APF algorithm works well for sequential parameter learning when there is no model misspecification, it can degenerate in the presence of outliers. Choosing good proposal distributions in these cases is necessary to alleviate these problems (see Künsch, 2005). We avoid these problems and our methodology also has the flexibility for learning parameters in the observation equation.

An alternative approach to reduce the effect of outliers is fixed-lag particle filtering (see Clapp and Godsill, 1999; Pitt and Shephard, 2001). The key difference here is that we do not reweight the initial samples. This also allows our approach to be more efficient for sequential parameter learning in higher-dimensional problems. On the other hand, by avoiding reweighting, certain parameters such as the evolution variance in autoregressive and stochastic volatility models are still hard hard to learn as with standard particle filtering methods (see Stroud, Polson and Müller, 2004).

Filter implementation requires a number of choices. First, the researcher needs to specify a fixed number of state paths N to be simulated. Secondly, we must choose the number of MCMC iterations G to be performed at each step. In many cases this number can be reduced by using standard MCMC blocking methods and the forward-filtering backward-sampling (FFBS) algorithm (Carter and Kohn, 1994; Frühwirth-Schnatter, 1994). We describe these methods in detail in Section 2.2. Finally, the length k of the block of state variables for fixed-lag smoothing must be chosen and we provide diagnostics to do this.

The methodology is illustrated using two applications. First, we consider a benchmark autoregressive plus noise model with sequential parameter learning. In a simulation study we find that there is little difference between our approach and Storvik's method for pure state filtering. However, when sequential parameter estimation is added, we show that Storvik's algorithm can degenerate and give poor inference in the presence of outliers. Secondly, we consider a three-dimensional stochastic Lorenz model from the atmospheric science literature. Recent research for known parameters such as Bengtsson, Snyder and Nychka (2003) uses mixture ensemble filtering for the Lorenz model. We use a block Gibbs sampling implementation of FFBS, which takes account of the nonlinearities in the system equation and works well for both known static parameters and sequential parameter learning.

The rest of the paper is outlined as follows. Section 2 discusses filtering using the practical filter. We describe diagnostic approaches for choosing the lag-length k. Section 3 shows how to incorporate sequential parameter learning into the algorithm. Section 4 applies our methodology to the autoregressive and Lorenz models. Finally, Section 5 concludes.

2. Filtering and Sequential Parameter Learning

The combined state and parameter estimation problem can be described as follows. Consider a dynamic state-space model with an observation $\mathbf{y}_t$, an unobserved state vector $\mathbf{x}_t$, and a parameter vector $\boldsymbol{\theta}$. The initial state distribution $\mathbf{x}_0 \sim p(\mathbf{x}_0)$ is assumed to be known. 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 to t.

The model is specified in terms of the densities

$$\begin{aligned} (Observation) \quad \mathbf{y}_t &\sim p(\mathbf{y}_t|\mathbf{x}_t) \\ (Evolution) \quad \mathbf{x}_t &\sim p(\mathbf{x}_t|\mathbf{x}_{t-1}, \boldsymbol{\theta}) \\ (Prior) \quad \boldsymbol{\theta} &\sim p(\boldsymbol{\theta}). \end{aligned}$$

The Bayesian approach to combined filtering and parameter learning requires calculation of the joint posterior distribution $p(\mathbf{x}_{1:t}, \boldsymbol{\theta}|\mathbf{y}_{1:t})$. From this, we can provide marginal state and parameter distributions, $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ and $p(\boldsymbol{\theta}|\mathbf{y}_{1:t})$. In most cases, the joint filtering distribution is unavailable in closed form, and must be approximated using Monte Carlo methods (see Doucet, de Freitas and Gordon, 2001, for recent methods).

2.1. Filtering with Fixed Parameters

First consider the filtering problem with fixed parameters. Now, we can exploit the representation of the filtering distribution as a mixture of lag-k smoothing distributions where the filtering distribution is a marginal of the form

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

In order to sequentially compute the filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ we can use the identity above and marginalise with respect to a constant window of lagged states. To do this we propose to re-use earlier simulations in an efficient manner. From an algorithmic perspective, this provides the equivalent of the popular FFBS algorithm for normal dynamic linear models (Anderson and Moore, 1979) for general state space models, replacing the single state lag of the FFBS algorithm by a sliding window, but still of fixed length k. At the heart of the proposed algorithm is the representation of the lag-k smoothing distribution as a mixture with respect to the distribution $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t})$, namely

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

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 require samples from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t})$. The basic assumption is that for each t, there exists a lag k such that joint distribution of the prior states is unaffected by all observations after time t.

Under this assumption, we can use the samples from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t-1})$ as approximate samples from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t})$. This approach will be reasonable when the predictive distribution $p(\mathbf{y}_t|\mathbf{y}_{t-k+1:t-1}, \mathbf{x}_{t-k})$ is conditionally independent of $\mathbf{x}_{t-k}$. To see this, note the identity

$$p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{t-k+1:t-1})}\, p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t-1}). \quad (2)$$

In many instances, the sensitivity of the predictive distribution to the initial state will decay quickly and in many cases at an exponential rate (see LeGland and Mevel, 1997; Künsch, 2001; Cappe, Moulines and Ryden, 2005).

The filtering algorithm for fixed parameters proceeds as follows. During an initial warm-up period ($t = 1, \ldots, k-1$) a full MCMC smoothing algorithm is run on $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, and the filtering distribution is approximated by samples of $\mathbf{x}_t$. The algorithm then proceeds (for $t = k, \ldots, T$) by drawing joint samples from $p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1:t})$ using an MCMC scheme. Finally, samples of $\mathbf{x}_{t-k}$ are stored for use at the next stage of the algorithm, and the filtering distribution is approximated by the marginal draws of $\mathbf{x}_t$.

A description of the filtering algorithm with fixed parameters is given below.

Algorithm 1: Filtering with Fixed Parameters

At each time period $t = k, \ldots, T$:

For each sample path $i = 1, \ldots, N$:

Step 1: Run an MCMC with stationary distribution $p(\mathbf{x}_{t-k+1:t}|\mathbf{x}^{(i)}_{t-k}, \mathbf{y}_{t-k+1:t})$.

Step 2: Define $\mathbf{x}^{(i)}_{t-k+1:t}$ as the last iteration of $\mathbf{x}_{t-k+1:t}$ in the MCMC.

Step 3: Store $\mathbf{x}^{(i)}_{t-k+1}$ as a draw from $p(\mathbf{x}_{t-k+1}|\mathbf{y}_{1:t})$.

Step 4: Report $\mathbf{x}^{(i)}_t$ as a draw from $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

The parameter N is the number of saved histories. The histories are independent draws from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t})$ and as such can be used to compute filtered estimates or perform density estimation using standard Rao-Blackwellisation.

The parameter G is the number of MCMC iterations used to obtain samples from the smoothing distribution at each stage. Samples from $p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1:t})$ can be drawn in several ways, depending on

the model, as described below. When the MCMC algorithm mixes rapidly, we can choose G to be small. For linear Gaussian models, independent samples can be drawn directly without using MCMC, although when parameters are fixed, this is clearly unnecessary since the filtering density is available in closed form.

The parameter k is the width of the rolling window used for state updating. In models where there is weak posterior dependence between state variables over time, we can choose a relatively small value of k. In the next subsection, we suggest a number of diagnostic approaches to selecting k. In the case of sequential parameter learning, the common static parameter induces additional dependence, requiring a larger choice of k.

2.2. Methods for State Generation

Generating states in an efficient manner given the parameters is essential for the fast implementation of our filtering approach. In this section, we describe various Monte Carlo smoothing algorithms for common classes of state-space models that can be used within our algorithm.

For linear Gaussian models, samples can be easily generated using the efficient forward-filtering backward-sampling (FFBS) algorithm of Carter and Kohn (1994) and Frühwirth-Schnatter (1994). The FFBS algorithm exploits the factorisation of the joint distribution $p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1:t})$ used in Algorithm 1 as

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

This allows for a recursive approach to drawing $\mathbf{x}_{t-k+1:t}$. Specifically, we first sample from $p(\mathbf{x}_t|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1:t})$ and then sample iteratively from $p(\mathbf{x}_j|\mathbf{x}_{j+1}, \mathbf{x}_{t-k}, \mathbf{y}_{t-k+1:j})$. This provides direct draws of the state vector given the parameters without requiring MCMC.

In multivariate state-space models, the state vector can often be decomposed into sub-blocks $\mathbf{x}_{t,b}$, $b = 1, \ldots, B$, such that, conditional on $\mathbf{x}_{t,-b} = \{\mathbf{x}_{t,b'},\ b' \neq b\}$, $\mathbf{x}_{t,b}$ has a Gaussian state-space form

$$\mathbf{y}_{t,b} \;=\; \mathbf{H}_{t,b}\mathbf{x}_{t,b} + \mathbf{e}_{t,b}, \qquad \mathbf{e}_{t,b} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{t,b}) \tag{3}$$

$$\mathbf{x}_{t,b} \;=\; \mathbf{M}_{t,b}\mathbf{x}_{t-1,b} + \mathbf{a}_{t,b} + \mathbf{w}_{t,b}, \quad \mathbf{w}_{t,b} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{t,b}), \tag{4}$$

where $\mathbf{y}_{t,b}, \mathbf{H}_{t,b}, \mathbf{R}_{t,b}, \mathbf{M}_{t,b}, \mathbf{a}_{t,b}$, and $\mathbf{Q}_{t,b}$ are known functions of $\mathbf{y}_t$ and $\mathbf{x}_{t,-b}$. In such cases, MCMC sampling is performed by iterating through the full conditional distributions $\mathbf{x}_{s:t,b} \sim p(\mathbf{x}_{s:t,b}|\mathbf{x}_{s:t,-b}, \mathbf{y}_{s:t})$ for $b = 1, \ldots, B$, where each sub-block is sampled efficiently using FFBS. This method is illustrated in the Lorenz example of Section 4.2.

MCMC sampling schemes can often be improved by introducing an extra block of latent state variables. While this increases the dimensionality of the state vector it poses little extra computational burden on our approach assuming that state sampling can be done efficiently. Examples of latent variables in state-space models include Carlin, Polson and Stoffer (1992) for non-Gaussian errors, Carter and Kohn (1994); Frühwirth-Schnatter (1994); Shephard (1994) for conditionally Gaussian models, Kim, Shephard and Chib (1998) for stochastic volatility models, and Stroud, Müller and Polson (2003) for state-dependent variance models.

Monte Carlo smoothing algorithms for other types of state-space models have also been developed. For discrete state and hidden Markov models, see Lindgren (1978), Geman and Geman (1984) and Künsch (2001), and Scott (2001) provides efficient MCMC methods for these models. For nonlinear state-space models, Carlin, Polson and Stoffer (1992) and Geweke and Tanizaki (2001) provide single-state and block updating MCMC schemes. For log-concave observation densities, Shephard and Pitt (1997) use an efficient block-updating Metropolis algorithm, and Gamerman (1998) provides algorithms for dynamic generalised linear models.

2.3. Comparison to Fixed-Lag Particle Filtering

Our approach can be contrasted with fixed-lag particle filtering, as described in Clapp and Godsill (1999), Pitt and Shephard (2001), and Gilks and Berzuini (2001). The fixed-lag APF approach starts with particles $\mathbf{x}_{t-k}^{(i)} \sim p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t-k})$ and then selects an index $z^{(i)}$ with probability proportional to $p(z|\mathbf{y}_{t-k+1:t}) \propto$

$\prod_{j=t-k+1}^{t} p(\mathbf{y}_j|\boldsymbol{\mu}_j^z)$ where $\boldsymbol{\mu}_j^z$ are forecasts of the states $\mathbf{x}_j$. Then, given a sampled index $z^{(i)}$, we draw a path of new states $\mathbf{x}_{t-k+1}^{(i)}, \ldots, \mathbf{x}_{t-k}^{(i)}$, conditional on $\mathbf{x}_{t-k}^{(z^{(i)})}$. Finally, this path is re-sampled with probability

$$\prod_{j=t-k+1}^{t} \frac{p(\mathbf{y}_j|\mathbf{x}_j^{(i)})}{p(\mathbf{y}_j|\boldsymbol{\mu}_j^{(z^{(i)})})} .$$

This approach leads to efficiency gains in terms of reducing bias but still has the problem of particle degeneracy in higher dimensions. Our approach avoids degeneracies by not reweighting.

2.4. Diagnostics for Practical Filtering

An important choice for filter implementation is the lag length k. The basic assumption is that the samples $\mathbf{x}_{t-k}^{(i)}$ from the previous step can be thought of as approximate samples for the next step, specifically, that the samples $\mathbf{x}_{t-k}^{(i)} \sim p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t-1})$ are approximate samples from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t})$. Therefore, we have to choose k large enough for this to be a reasonable approximation in practice. As alternative way to think about this assumption is that the choice of k depends on the sensitivity of the predictive distribution to the initial value and how quickly the initial state is dissipated.

To choose k, we use a simple diagnostic measure D_k comparing the distance between the two predictive distributions plotted as a function of k. Distance metrics D_k can be chosen based on theoretical considerations; for example, a Kullback-Leibler divergence metric can be used as Kitagawa (1996).

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 $\mathbf{y}_{t-k+1:t-1}$. We can estimate the predictive distribution using

$$p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}) \approx \frac{1}{N}\sum_{i=1}^{N} p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}^{(i)})$$

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. We can also look at differences in smoothed means, and in the case of parameter learning, we can look at the sensitivity of these measures to different parameter values. We use such an approach on the autoregressive example in Section 4.1.

Clearly, there are a number of cases where our approach is inappropriate. For example, in some non-stationary 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, where the entire state trajectories are updated using MCMC moves. Although these strategies increase the computational time, in many situations the practical filter is the only available methodology.

3. Sequential Parameter Learning and Filtering

One of the main advantages of our algorithm is the ability to incorporate sequential parameter learning. Combined state filtering and sequential parameter learning requires the computation of the joint distribution of the states and parameters, $p(\mathbf{x}_t, \boldsymbol{\theta}|\mathbf{y}_{1:t})$, for each value of t. We need to generate samples $\{\mathbf{x}_t^{(i)}, \boldsymbol{\theta}^{(i)}\}_{i=1}^N$ from these joint distributions. The natural extension of the algorithm in Section 2 is the following algorithm for sequential parameter learning.

3.1. Sequential Parameter Learning with Saved Histories

In the combined state and parameter learning case, we will view $p(\mathbf{x}_t, \boldsymbol{\theta}|\mathbf{y}_{1:t})$ as a marginal from the joint smoothing distribution, $p(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta}|\mathbf{y}_{1:t})$, which can be decomposed as

$$\begin{aligned} p(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta}|\mathbf{y}_{1:t}) &= \int p(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta}|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t})\ dp(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t}) \\ &\approx \int p(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta}|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t})\ dp(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t-1}). \end{aligned} \tag{5}$$

Notice that, in contrast to the fixed parameter case, we now average over the joint distribution of the initial $t-k$ states, $\mathbf{x}_{0:t-k}$, whereas in the pure filtering case, due to the Markov property we only needed to average over the marginal draws of $\mathbf{x}_{t-k}$. We make the assumption that the samples $p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t-1})$ are approximate draws from $p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t})$, i.e., that the joint distribution of $\mathbf{x}_{0:t-k}$ is unaffected by $\mathbf{y}_t$, given the values of $\mathbf{y}_{t-k+1:t-1}$.

The algorithm is as follows. First, assume that $\mathbf{x}_{0:t-k}^{(i)}$ are samples from $p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t-1})$ which are approximately samples from $p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t})$. Next, we generate samples from the joint distribution

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

This is achieved by using MCMC and iterating between draws from the following distributions:

$$p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}^{(i)}, \boldsymbol{\theta}, \mathbf{y}_{t-k+1:t}) \quad \text{and} \quad p(\boldsymbol{\theta}|\mathbf{x}_{0:t-k}^{(i)}, \mathbf{x}_{t-k+1:t}, \mathbf{y}_{1:t}). \tag{6}$$

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 $(\mathbf{x}_{t-k}, \boldsymbol{\theta})$, that is, the Markov property of the dynamic model given the static parameters. Repeated sampling from (6) provides a two-block Gibbs sampler to simulate from $p(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta}|\mathbf{x}_{0:t-k}^{(i)}, \mathbf{y}_{1:t})$. We then take the last imputed values $\mathbf{x}_t$ as a sample from the filtering density, and store the last imputed value $\mathbf{x}_{t-k+1}$ as $\mathbf{x}_{t-k+1}^{(i)}$ for use in the next step of the algorithm.

The combined state and parameter filtering algorithm can summarised as follows.

Algorithm 2: Filtering with Sequential Parameter Learning

At each time period $t = k, \ldots, T$:

For each sample path $i = 1, \ldots, N$:

Step 1: Run an MCMC with stationary distribution $p(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta}|\mathbf{x}_{0:t-k}^{(i)}, \mathbf{y}_{1:t})$.

Step 2: Define $(\mathbf{x}_{t-k+1:t}^{(i)}, \boldsymbol{\theta}^{(i)})$ as the last iteration of $(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta})$ in the MCMC.

Step 3: Store $\mathbf{x}_{t-k+1}^{(i)}$ as a draw from $p(\mathbf{x}_{t-k+1}|\mathbf{y}_{1:t})$.

Step 4: Report $(\mathbf{x}_t^{(i)}, \boldsymbol{\theta}^{(i)})$ as a draw from $p(\mathbf{x}_t, \boldsymbol{\theta}|\mathbf{y}_{1:t})$.

For each sample $\mathbf{x}_{0:t-k}^{(i)}$, we will typically require multiple MCMC iterations $G > 1$ to obtain a joint draw of $(\mathbf{x}_t, \boldsymbol{\theta})$. The required number of MCMC iterations G will depend on the dependence between $\mathbf{x}_{t-k+1:t}$ and $\boldsymbol{\theta}$ as in the MCMC smoothing case. Block updating should be used wherever possible to reduce the number of required MCMC iterations.

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 k is related to how the parameter $\boldsymbol{\theta}$ introduces dependence into the state equation. If $\boldsymbol{\theta}$ enters the model only in the evolution equation then the choice of appropriate lag k could be highly dependent on the posterior range of $\boldsymbol{\theta}$. In other words, in trying to find a value of k such that $\mathbf{x}_{t-k}$ and $\mathbf{y}_t$ are approximately conditionally independent given $\mathbf{y}_{t-k+1:t-1}$ could critically depend on the distribution of $\boldsymbol{\theta}$. If necessary we suggest an adaptive choice of k, for example, periodic refreshing with $k = t$, to reduce the approximation error.

The filtering distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ can then be approximated by the empirical distribution of the imputed $\mathbf{x}_t^{(i)}$. Alternatively, depending on the nature of the lag smoother applied in Step 1 of Algorithm 2, we can use Rao-Blackwellisation and evaluate the filtering distribution as an average across filtering distributions given the imputed histories $\mathbf{x}_{0:t-k}^{(i)}$:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{1:t}) &= \iint p(\mathbf{x}_t|\mathbf{x}_{0:t-k}, \boldsymbol{\theta}, \mathbf{y}_{t-k+1:t})\, p(\boldsymbol{\theta}|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t})\, d\boldsymbol{\theta}\, p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t})\, d\mathbf{x}_{0:t-k} \\ &\approx \frac{1}{N}\sum_{i=1}^{N} \int p(\mathbf{x}_t|\mathbf{x}_{0:t-k}^{(i)}, \boldsymbol{\theta}, \mathbf{y}_{t-k+1:t})\, p(\boldsymbol{\theta}|\mathbf{x}_{1:t-k}^{(i)}, \mathbf{y}_{1:t})\, d\boldsymbol{\theta}. \end{aligned} \tag{7}$$

where the second line is the Monte Carlo approximation.

The sequential parameter posterior distribution can similarly be approximated either by the empirical distribution of the imputed $\boldsymbol{\theta}^{(i)}$, or Rao-Blackwellisation. Now we describe simulation methods for generating parameters given states.

3.2. Methods for Parameter Generation

Generating parameters in an efficient manner (see Step 1 of Algorithm 2) requires sampling from the full conditional distributions of the parameter vector conditional on the states. This can often be achieved as follows. Many state-space models, conditional on the states, can be written in the following form.

$$\tilde{\mathbf{y}}_t = \mathbf{H}_t'\,\boldsymbol{\alpha} + \mathbf{e}_t, \quad \mathbf{e}_t \sim \mathcal{N}(0, \tau^2\mathbf{I}) \tag{8}$$

$$\tilde{\mathbf{x}}_t = \mathbf{F}_t'\,\boldsymbol{\beta} + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(0, \sigma^2\mathbf{I}). \tag{9}$$

Here the parameter vector is $\boldsymbol{\theta} = (\alpha, \beta, \sigma, \tau)$, and the observations and states can enter in a possibly nonlinear way through $\tilde{\mathbf{y}}_t = Y(\mathbf{y}_t, \mathbf{x}_t)$, $\tilde{\mathbf{x}}_t = X(\mathbf{x}_t, \mathbf{x}_{t-1})$, $\mathbf{H}_t' = H(\mathbf{x}_t)$, and $\mathbf{F}_t' = F(\mathbf{x}_{t-1})$. We assume these are given functions of the observations and states.

This leads to an important class of algorithms where sequential parameter learning can be efficiently achieved by exploiting a sufficient statistic structure. More explicitly, if we assume the standard normal-inverse gamma priors for $(\boldsymbol{\alpha}, \tau^2)$ and $(\boldsymbol{\beta}, \sigma^2)$ for the parameters we obtain full conditional posteriors that depend on a sufficient statistic structure, namely

$$p(\boldsymbol{\alpha}|\tau^2, \mathbf{x}_{0:t}, \mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{a}_t, \tau^2\mathbf{A}_t^{-1}), \qquad p(\tau^2|\mathbf{x}_{0:t}, \mathbf{y}_{1:t}) = \mathcal{IG}(\xi_t, e_t), \tag{10}$$

$$p(\boldsymbol{\beta}|\sigma^2, \mathbf{x}_{0:t}, \mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{b}_t, \sigma^2\mathbf{B}_t^{-1}), \qquad p(\sigma^2|\mathbf{x}_{0:t}, \mathbf{y}_{1:t}) = \mathcal{IG}(\nu_t, d_t). \tag{11}$$

where $\mathbf{T}_t = (\mathbf{a}_t, \mathbf{b}_t, \mathbf{A}_t, \mathbf{B}_t, \xi_t, \nu_t, e_t, d_t)$ is the vector of sufficient statistics. To reduce computational cost notice that we can recursively compute the sufficient statistics $\mathbf{T}_t$ via

$$\mathbf{A}_t = \mathbf{A}_{t-k} + \mathbf{H}'\mathbf{H}, \qquad \mathbf{a}_t = \mathbf{A}_t^{-1}(\mathbf{A}_{t-k}\mathbf{a}_{t-k} + \mathbf{H}'\tilde{\mathbf{y}}), \tag{12}$$

$$\mathbf{B}_t = \mathbf{B}_{t-k} + \mathbf{F}'\mathbf{F}, \qquad \mathbf{b}_t = \mathbf{B}_t^{-1}(\mathbf{B}_{t-k}\mathbf{b}_{t-k} + \mathbf{F}'\tilde{\mathbf{x}}), \tag{13}$$

$$\xi_t = \xi_{t-k} + kn/2, \qquad e_t = e_{t-k} + (\mathbf{a}_{t-k}'\mathbf{A}_{t-k}\mathbf{a}_{t-k} + \tilde{\mathbf{y}}'\tilde{\mathbf{y}} - \mathbf{a}_t'\mathbf{A}_t\mathbf{a}_t)/2, \tag{14}$$

$$\nu_t = \nu_{t-k} + kp/2, \qquad d_t = d_{t-k} + (\mathbf{b}_{t-k}'\mathbf{B}_{t-k}\mathbf{b}_{t-k} + \tilde{\mathbf{x}}'\tilde{\mathbf{x}} - \mathbf{b}_t'\mathbf{B}_t\mathbf{b}_t)/2, \tag{15}$$

where $\mathbf{H}$ is the horizontal catenation of $\mathbf{H}_{t-k+1}, \ldots, \mathbf{H}_t$ and likewise for $\mathbf{F}$, $\tilde{\mathbf{x}}$, and $\tilde{\mathbf{y}}$, and where n and p denote the dimension of $\tilde{\mathbf{y}}_t$ and $\tilde{\mathbf{x}}_t$, respectively. In the next subsection we explicitly describe our algorithm that exploits this sufficient statistic structure.

Many other types of state-space models also lead to natural sufficient statistic structures that can lead to fast convergence, for example, change-point models (Chib, 1998) and dynamic categorical time series models (Carlin and Polson, 1992). Again this is the key to obtaining a fast filtering algorithm with a linear computational cost.

We now describe the sequential parameter learning algorithm with sufficient statistics.

3.3. Sequential Parameter Learning with Sufficient Statistics

Sequential parameter learning with sufficient statistics will now track the state and sufficient statistic pair $(\mathbf{x}_{t-k}^{(i)}, \mathbf{T}_{t-k}^{(i)})$ as a draw from its probability distribution. This alleviates the problem of having to store the full state trajectory $\mathbf{x}_{0:t-k}^{(i)}$, and leads to a computationally efficient algorithm. Conditional on $(\mathbf{x}_{t-k}^{(i)}, \mathbf{T}_{t-k}^{(i)})$, we run an MCMC algorithm to sample from

$$p(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta} | \mathbf{x}_{t-k}^{(i)}, \mathbf{T}_{t-k}^{(i)}, \mathbf{y}_{t-k+1:t}).$$

This is achieved by iterating between the following conditional distributions

$$p(\mathbf{x}_{t-k+1:t} | \mathbf{x}_{t-k}^{(i)}, \boldsymbol{\theta}, \mathbf{y}_{t-k+1:t}) \quad \text{and} \quad p(\boldsymbol{\theta} | \mathbf{T}_{t-k}^{(i)}, \mathbf{x}_{t-k+1:t}, \mathbf{y}_{t-k+1:t}).$$

After convergence, the sufficient statistics are updated through their recursion

$$\mathbf{T}_{t-k+1}^{(i)} = f(\mathbf{T}_{t-k}^{(i)}, \mathbf{x}_{t-k+1}^{(i)}, \mathbf{y}_{t-k+1})$$

such as those given in (12)–(15), and where $\mathbf{x}_{t-k+1}^{(i)}$ is the newly imputed state. Then we store $(\mathbf{x}_{t-k+1}^{(i)}, \mathbf{T}_{t-k+1}^{(i)})$ as a draw from $p(\mathbf{x}_{t-k+1}, \mathbf{T}_{t-k+1} | \mathbf{y}_{1:t})$. Finally, we take $(\mathbf{x}_t^{(i)}, \boldsymbol{\theta}^{(i)})$ as a draw from the desired joint filtering distribution $p(\mathbf{x}_t, \boldsymbol{\theta} | \mathbf{y}_{1:t})$.

The sequential parameter learning algorithm with sufficient statistics is described below.

Algorithm 2a: Filtering with Sufficient Statistics

At each time period $t = k, \ldots, T$:

For each sample path $i = 1, \ldots, N$:

Step 1: Run an MCMC with stationary distribution $p(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta} | \mathbf{x}_{t-k}^{(i)}, \mathbf{T}_{t-k}^{(i)}, \mathbf{y}_{t-k+1:t})$.

Step 2: Define $(\mathbf{x}_{t-k+1:t}^{(i)}, \boldsymbol{\theta}^{(i)})$ as the last iteration of $(\mathbf{x}_{t-k+1:t}, \boldsymbol{\theta})$ in the MCMC.

Step 3: Compute $\mathbf{T}_{t-k+1}^{(i)} = f(\mathbf{T}_{t-k}^{(i)}, \mathbf{x}_{t-k+1}^{(i)}, \mathbf{y}_{t-k+1})$.

Step 4: Store $(\mathbf{x}_{t-k+1}^{(i)}, \mathbf{T}_{t-k+1}^{(i)})$ as a draw from $p(\mathbf{x}_{t-k+1}, \mathbf{T}_{t-k+1} | \mathbf{y}_{1:t})$.

Step 5: Report $(\mathbf{x}_t^{(i)}, \boldsymbol{\theta}^{(i)})$ as a draw from $p(\mathbf{x}_t, \boldsymbol{\theta} | \mathbf{y}_{1:t})$.

This algorithm has a number of key advantages. Primarily, it provides a fast algorithm for sequential parameter learning, as it only has to track the state and sufficient statistics instead of the full state trajectory. While algorithms such as Storvik (2002) and Fearnhead (2002) exploit sufficient statistic structure, our approach does not reweight particles, so it does not degenerate, and it also allows for straightforward estimation of parameters in the observation equation. SIR particle filters require additional MCMC steps to generate these parameters, making them far less computationally efficient.

4. Applications

In this section, we analyze the empirical performance of our algorithm and compare the results to the SIR algorithm of Storvik (2002). First, we consider a benchmark autoregressive model with parameter learning. Here we show that Storvik's algorithm and our approach handle both state and sequential parameter learning efficiently on simulated data. However, the SIR algorithm degenerates with the addition of a large outlier whereas the practical filter does not. Secondly, we consider a three-dimensional stochastic Lorenz model, which illustrates the application of practical filtering to nonlinear and higher-dimensional models.

4.1. Autoregressive AR(1) with Noise

The benchmark model for studying filtering methods in the presence of outliers and sequential parameter learning is an autoregressive process with noise (see, for example, Pitt and Shephard, 1999; Storvik, 2002).

We consider the following AR(1) plus noise model

$$\begin{aligned} y_t &= x_t + \epsilon_t, & \epsilon_t \sim N(0, \tau^2), \\ x_t &= \alpha + \beta x_{t-1} + \omega_t, & \omega_t \sim N(0, \sigma^2), \end{aligned}$$

with initial distribution $x_0 \sim \mathcal{N}(0, .1)$ and $\tau^2 = .1$. Data are simulated from the model using parameters $\alpha = 0$, $\beta = 0.9$ and $\sigma^2 = .04$, corresponding to a fairly persistent time series.

We first consider generating the states conditional on parameters. The full conditional distribution $p(\mathbf{x}_{t-k+1:t}|\boldsymbol{\theta}, x_{t-k}, \mathbf{y}_{1:t})$ is multivariate normal where the prior on x_{t-k+1} is given by the transition density, $\mathcal{N}(\alpha + \beta x_{t-k}, \sigma^2)$. Since this is a linear state-space model, the states $\mathbf{x}_{t-k+1:t}$ can be efficiently generated using the FFBS algorithm.

Next we consider generating the parameters. Following Section 3.2, we define $\boldsymbol{\beta} = (\alpha, \beta)$, $\tilde{\mathbf{x}}_t = x_t$ and $\mathbf{F}_t = (1, x_{t-1})'$, and so the evolution equation is conditionally Gaussian of the form (9). A normal inverse-gamma prior is selected with hyperparameters $\mathbf{b}_0 = (0, 0.9)$, $\mathbf{B}_0 = \text{diag}(10, 5)$, $\nu_0 = 10$ and $d_0 = .4$, and direct simulation of $(\boldsymbol{\beta}, \sigma^2)$ is possible from the full conditional distributions given in (11).

In order to provide some guidelines for the choice of lag-length, k, Figure 1 compares the smoothed means $E(x_{99-k}|\mathbf{y}_{1:100})$ and $E(x_{99-k}|\mathbf{y}_{1:99})$, and the densities $p(x_{99-k}|\mathbf{y}_{1:100})$ and $p(x_{99-k}|\mathbf{y}_{1:99})$, as a function of k. The data are simulated using $(\alpha, \beta, \sigma^2) = (0, .9, .04)$, and the differences in means and densities are computed over a range of plausible parameter values: $\beta \in (0.7, 0.9, 1.1)$ and $\sigma^2 \in (0.01, 0.04, 0.07)$. The results are obtained analytically through the Kalman filter and smoother. We see that a choice of $k = 25$ is ample for the problem at hand.

To test the accuracy of our approach, we compare the filtered moments with the true values obtained from a full MCMC. Table 1 compares the filtered mean and standard deviations for the states x_t and parameters (α, β) at times $t = 0, 200, \ldots, 1000$. The evolution variance is held fixed at its true value of $\sigma^2 = .04$. Here the practical filter was run using $(N, G, k) = (10000, 10, 25)$ while the MCMC was run using 10000 iterations. Notice that in the case of known evolution variance, the filtered moments for the states and parameters for the two algorithms are all within Monte Carlo sampling error.

However, when the evolution variance σ^2 is treated as unknown, the algorithm performs poorly. In Table 2, we see that while the posterior mean for full MCMC starts at 0.5, decreases to 0.43 and returns to 0.5 at the end of the sample, the practical filter has a posterior mean that is approximately constant at 0.57 and has difficultly tracking the movement in the true posterior mean. This is due to the fact that learning the sufficient statistic d_t of Section 3.2 is hard given the choice of k. This is consistent with a similar finding about the evolution variance in a stochastic volatility model (see Stroud, Polson and Müller, 2004).

Addressing the sensitivity of the algorithms to outliers is important. In this case, the practical filter has an advantage over standard particle filtering methods, particularly for sequential parameter learning. Figure 2 includes an outlying data point $y_{50} = 6$ and tracks the same sequential parameter distributions and state filtering distributions. Now we see the difference in parameter estimation after the outlier. The practical filter with $(N, G, k) = (2000, 5, 25)$ matches full MCMC whereas Storvik's SIR algorithm with $N = 10000$ particles underestimates the shift in $p(\alpha|\mathbf{y}_{1:50})$ and hence overestimates $p(\beta|\mathbf{y}_{1:50})$ after the outlier.

Figure 3 illustrates the problem more dramatically by plotting histograms of the state and parameter posteriors at time $t = 50$. We can now see that the main problem with the SIR algorithm is that it suffers from particle degeneracies in the state filtering distribution, which in turn leads to poor estimation of the posterior distributions for the parameters.

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:

$$\begin{aligned} \dot{x} &= s(y - x) \\ \dot{y} &= rx - y - xz \\ \dot{z} &= xy - bz \end{aligned}$$

where the dot denotes a time derivative. Here the state vector $\mathbf{x} = (x, y, z)'$ represents a particle's position in phase space. For our analysis, we take a first-order Euler discretisation of the model with a time step

of Δ and add an evolution noise term. For each $t = 0, 1, \ldots, T$, let $\mathbf{x}_t$ denote the state vector at time Δt, and $\mathbf{x}_t^\Delta = \Delta^{-1}(\mathbf{x}_t - \mathbf{x}_{t-1})$ denote the finite difference, and define similar quantities for x, y and z. The discretised evolution equation is given by

$$x_{t+1}^\Delta = s(y_t - x_t) + w_t^x \tag{16}$$
$$y_{t+1}^\Delta = rx_t - y_t - x_t z_t + w_t^y \tag{17}$$
$$z_{t+1}^\Delta = x_t y_t - bz_t + w_t^z \tag{18}$$

where $\mathbf{w}_t = (w_t^x, w_t^y, w_t^z)' \sim \mathcal{N}(0, \Delta^{-1}\sigma^2\mathbf{I})$ is a Gaussian white noise process. At each time step $t = 1, \ldots, T$, we collect noisy observations, $\mathbf{y}_t = (x_t^\dagger, y_t^\dagger, z_t^\dagger)'$, which are generated through the observation equation

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{e}_t, \quad \mathbf{e}_t \sim \mathcal{N}(\mathbf{0}, \tau^2\mathbf{I})$$

where $\mathbf{I}$ is the identity matrix, and the errors $\mathbf{e}_t$ are uncorrelated in time.

Currently, most filtering approaches for this model are based on ensemble filters (see, for example, Bengtsson, Snyder and Nychka, 2003). Our combined state and parameter filtering algorithm with sufficient statistics (Algorithm 2a) can also provide sequential posterior parameter distributions. We provide sequential estimates of the system parameters as well as the observation and evolution variances. Furthermore, we show that a natural block sampling approach applies for state generation in our filtering algorithm. This provides a computationally efficient approach for implementing our method.

We first need to design a fast MCMC smoothing algorithm for the states. The evolution equation is nonlinear, but if we define the sub-blocks $\mathbf{x}_{t,1} = x_t$ and $\mathbf{x}_{t,2} = (y_t, z_t)$ as in Section 2.2, the model is conditionally Gaussian with two sub-blocks (the model matrices are defined in Appendix A). An efficient two-block Gibbs sampler based on FFBS can then be used to generate the states.

We now turn to the problem of sequential parameter learning. Defining the parameter vector $\boldsymbol{\beta} = (s, r, b)'$, we can write the evolution equation as in (9), where

$$\tilde{\mathbf{x}}_{t+1} = \begin{pmatrix} x_{t+1}^\Delta \\ y_{t+1}^\Delta + y_t + x_t z_t \\ z_{t+1}^\Delta - x_t y_t \end{pmatrix}, \quad \mathbf{F}_{t+1} = \begin{pmatrix} y_t - x_t & 0 & 0 \\ 0 & x_t & 0 \\ 0 & 0 & -z_t \end{pmatrix},$$

which leads to normal-inverse gamma posteriors for $(\boldsymbol{\beta}, \sigma^2)$ as given in (11). For the unknown observation variance, τ^2, we assume an inverse-gamma prior, leading to the full conditional $p(\tau^2|\mathbf{x}_{1:t}, \mathbf{y}_{1:t}) = \mathcal{IG}(\xi_t, e_t)$. The lag-$k$ updating recursions for the hyperparameters are given by $\xi_t = \xi_{t-k} + 3k/2$ and $e_t = e_{t-k} + \sum_{j=t-k+1}^{t}(\mathbf{y}_j - \mathbf{x}_j)'(\mathbf{y}_j - \mathbf{x}_j)/2$.

To study our filtering algorithm, we generate $T = 500$ observations from the model using a time step of $\Delta = .02$, variances of $(\sigma^2, \tau^2) = (0.5, 2)$ and an initial state vector of $\mathbf{x}_0 = (-5.9, -5.5, 24.6)'$. The model parameters are set to $\boldsymbol{\beta} = (s, r, b) = (10, 28, 2.67)$, corresponding to the original parameters used by Lorenz (1963), and producing the butterfly-shaped phase-space trajectory shown in Figure 4. For filter implementation, we choose diffuse but proper priors for the parameter and the initial state. For the parameters, we set $\mathbf{b}_0 = (10, 28, 2.67)'$, $\mathbf{B}_0 = \text{diag}(.01, .01, .01)$, $\nu_0 = 2$, $d_0 = .5$, $\xi_0 = 2$ and $e_0 = 1$, while for the initial states, we set $\mathbf{x}_0 \sim \mathcal{N}((0, 0, 0)', \text{diag}(100, 100, 100))$.

Figure 5 shows the filtered paths for the state variables along with sequential learning 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 Lorenz parameters, $\boldsymbol{\beta}$, are the easiest to learn. 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 provide a filtering algorithm that also incorporates sequential parameter learning. Our approach requires the choice of a fixed-lag and a fast MCMC algorithm to draw the states, for example, in a conditionally Gaussian model. The algorithm does not suffer from particle degeneracies which hinder particle

filtering methods. Moreover it is robust to outliers and can be applied to higher dimensional problems. The requirement for a real-time filtering algorithm is that it runs in $O(t)$ time and doesn't degenerate. We achieve this by exploiting a lag-k mixture representation of the filtering distribution and rely on a sufficient statistic structure as in Storvik (2002) for updating parameters conditional on states.

From an empirical perspective, these filtering methods have achieved recent success in financial applications, including sequential portfolio allocation and stochastic volatility jump diffusion models (Johannes, Polson and Stroud, 2002, 2005a,b). However, sequential parameter learning still poses a number of computational challenges. Evolution parameters are notoriously hard to estimate sequentially and more research is required to understand this problem. Another avenue for future research is filtering in continuous-time models with discretely sampled data, where filling-in-the-missing-data estimators have been proposed in an MCMC smoothing framework (see Eraker, 2001; Elerian, Shephard and Chib, 2001). Such approaches can be extended to the filtering context using the methodology proposed here.

References

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

Andrieu, C. and Doucet, A. (2003) On-line expectation-maximization type algorithms for parameter estimation in general state space models. *Proc. IEEE ICASSP*.

Andrieu, C., Doucet, A. and Tadić, V. (2005) Online simulation-based methods for parameter estimation in nonlinear non Gaussian state-space models. *Proc. IEEE CDC*.

Bengtsson, T., Snyder, C. and Nychka, D. (2003) Toward a nonlinear ensemble filter for high-dimensional systems. *Journal of Geophysical Research*, **108**, 8775.

Berzuini, C., Best, N., Gilks, W. and Larizza, C. (1997) Dynamic conditional independence models and Markov chain Monte Carlo methods. *Journal of the American Statistical Association*, **92**, 1403–1412.

Cappe, O., Moulines, E. and Ryden, T. (2005) *Inference in Hidden Markov Models*. New York: Springer.

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.

Carlin, B. P. and Polson, N. G. (1992) Monte Carlo Bayesian methods for discrete regression models and categorical time series. In *Bayesian Statistics 4* (eds. J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith). Oxford: Oxford University Press.

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

Chib, S. (1998) Estimation and comparison of multiple change point models. *Joural of Econometrics*, **86**, 221–241.

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), 743–752. Oxford: Oxford University Press.

Doucet, A., de Freitas, J. and Gordon, N. (eds.) (2001) *Sequential Monte Carlo Methods in Practice*. Kluwer.

Elerian, O., Shephard, N. and Chib, S. (2001) Likelihood inference for discretely observed non-linear diffusions. *Econometrica*, **69**, 959–993.

Eraker, B. (2001) MCMC analysis of diffusion models with application to finance. *Journal of Business and Economic Statistics*, **19**, 177–191.

Fearnhead, P. (2002) MCMC, sufficient statistics and particle filter. *Journal of Computational and Graphical Statistics*, **11**, 848–862.

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

Gamerman, D. (1998) Monte Carlo Markov chains for dynamic generalised linear models. *Biometrika*, **85**, 215–227.

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

Geweke, J. and Tanizaki, H. (2001) Bayesian estimation of state-space model using the Metropolis-Hastings algrorithm within Gibbs sampling. *Computational Statistics and Data Analysis*, **37**, 151–170.

Gilks, W. and Berzuini, C. (2001) Following a moving target – Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society, Series B*, **63**, 127–146.

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.

Hürzeler, M. and Künsch, H. (2001) Approximating and maximizing the likelihood for general SSM. In *Sequential Monte Carlo Methods in Practice* (eds. A. Doucet, J. de Freitas and N. Gordon). Springer.

Johannes, M., Polson, N. and Stroud, J. (2002) Sequential optimal portfolio performance: Market and volatility timing. *Tech. rep.*, Graduate School of Business, University of Chicago.

— (2005a) Optimal filtering of jump diffusions: Extracting latent states from asset prices. *Tech. rep.*, Graduate School of Business, University of Chicago.

— (2005b) Sequential parameter estimation in stochastic volatility models with jumps. *Tech. rep.*, Graduate School of Business, University of Chicago.

Kim, S., Shephard, N. and Chib, S. (1998) Stochastic volatility: Likelihood inference and comparison with ARCH models. *Review of Economic Studies*, **65**, 361–393.

Kitagawa, G. (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.

— (2005) Recursive Monte Carlo filters: Algorithms and theoretical analysis. *Annals of Statistics*.

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. (1995) Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, **93**, 1032–1044.

Liu, J. and West, M. (2001) 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.

Pitt, M. (2002) Smooth particle filters for likelihood evaluation and maximization. *Tech. rep.*, Department of Economics, University of Warwick.

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

— (2001) Auxiliary variable based particle filters. In *Sequential Monte Carlo Methods in Practice* (eds. A. Doucet, J. de Freitas and N. Gordon), 273–293. Springer.

Scott, S. (2001) Bayesian methods for hidden Markov models. *Journal of the American Statistical Association*, **97**, 337–351.

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

Shephard, N. and Pitt, M. (1997) Likelihood analysis of non-Gaussian measurement time series. *Biometrika*, **84**, 653–667.

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. (2003) Nonlinear state-space models with state-dependent variances. *Journal of the American Statistical Association*, **98**, 377–386.

Stroud, J., Polson, N. and Müller, P. (2004) Practical filtering for stochastic volatility models. In *State Space and Unobserved Component Models* (eds. A. Harvey, S. Koopmans and N. Shephard), 236–247. Oxford University Press.

Appendix A: State Generation in the Lorenz Model

Here, we define the Gibbs sampling algorithm for state generation in the Lorenz (1963) model. We partition the state vector into two sub-blocks, $\mathbf{x}_{t,1} = x_t$ and $\mathbf{x}_{t,2} = (y_t, z_t)'$. Given the initial state $\mathbf{x}_{t-k}$ and the parameter $\boldsymbol{\theta}$, the state update iterates between the two blocks:

- Generate $\mathbf{x}_{t-k+1:t,1} \sim p(\mathbf{x}_{t-k+1:t,1}|\mathbf{x}_{t-k,1}, \mathbf{x}_{t-k:t,2}, \boldsymbol{\theta}, \mathbf{y}_{t-k+1:t})$.
- Generate $\mathbf{x}_{t-k+1:t,2} \sim p(\mathbf{x}_{t-k+1:t,2}|\mathbf{x}_{t-k,2}, \mathbf{x}_{t-k:t,1}, \boldsymbol{\theta}, \mathbf{y}_{t-k+1:t})$.

Each block has a conditionally Gaussian state-space form (3)–(4), with model matrices defined below.

- The observation and evolution matrices for Block 1 are given by

$$\mathbf{y}_{t,1} = \begin{pmatrix} x_t^{\dagger} \\ y_t^{\Delta} + r y_{t-1} \\ z_t^{\Delta} + b z_{t-1} \end{pmatrix}, \quad \mathbf{H}_{t,1} = \begin{pmatrix} 1 \\ r - z_{t-1} \\ y_{t-1} \end{pmatrix}, \quad \mathbf{R}_{t,1} = \begin{pmatrix} \tau^2 & 0 & 0 \\ 0 & \sigma^2/\Delta & 0 \\ 0 & 0 & \sigma^2/\Delta \end{pmatrix},$$

  and

$$\mathbf{M}_{t,1} = 1 - \Delta s, \quad \mathbf{a}_{t,1} = \Delta s y_{t-1}, \quad \mathbf{Q}_{t,1} = \sigma^2/\Delta,$$

  and the initial prior for $\mathbf{x}_{t-k+1,1}$ is

$$\mathcal{N}(x_{t-k} + \Delta s(y_{t-k} - x_{t-k}), \Delta\sigma^2).$$

- The model matrices for Block 2 are given by

$$\mathbf{y}_{t,2} = \begin{pmatrix} \mathbf{x}_t^{\Delta} + s x_{t-1} \\ y_t^{\dagger} \\ z_t^{\dagger} \end{pmatrix}, \quad \mathbf{H}_{t,2} = \begin{pmatrix} s & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{R}_{t,2} = \begin{pmatrix} \sigma^2/\Delta & 0 & 0 \\ 0 & \tau^2 & 0 \\ 0 & 0 & \tau^2 \end{pmatrix},$$

  and

$$\mathbf{M}_{t,2} = \begin{pmatrix} 1 - \Delta & -\Delta x_{t-1} \\ \Delta x_{t-1} & 1 - \Delta b \end{pmatrix} \quad \mathbf{a}_{t,2} = \begin{pmatrix} \Delta r x_{t-1} \\ 0 \end{pmatrix}, \quad \mathbf{Q}_{t,2} = \begin{pmatrix} \sigma^2/\Delta & 0 \\ 0 & \sigma^2/\Delta \end{pmatrix},$$

  and the initial prior for $\mathbf{x}_{t-k+1,2}$ is

$$\mathcal{N}\left[\begin{pmatrix} y_{t-k} \\ z_{t-k} \end{pmatrix} + \Delta \begin{pmatrix} r x_{t-k} - y_{t-k} - x_{t-k} x_{t-k} \\ x_{t-k} y_{t-k} - b z_{t-k} \end{pmatrix}, \begin{pmatrix} \Delta\sigma^2 & 0 \\ 0 & \Delta\sigma^2 \end{pmatrix}\right].$$

Table 1. AR(1) plus noise model. Filtered mean (m) and standard deviation (s) for (x_t, α, β) at times $t = 0, 200, \ldots, 1000$. Here $\sigma^2 = .04$ is held fixed, and we use the priors given in Section 4.1. Results are obtained using MCMC with 10000 iterations, and practical filter with $(N, G, k) = (10000, 10, 25)$. The true parameter values are $(\alpha, \beta, \sigma^2) = (0, .9, .04)$.

	x_t				α				β			
t	MCMC		Practical		MCMC		Practical		MCMC		Practical	
	m	s	m	s	m	s	m	s	m	s	m	s
0	-.003	.317	.002	.313	.000	.064	.001	.063	.900	.090	.899	.090
200	.463	.205	.467	.207	.006	.014	.006	.014	.864	.036	.867	.036
400	.340	.206	.338	.205	.007	.010	.007	.010	.873	.025	.874	.026
600	-.703	.206	-.697	.206	-.001	.008	-.001	.008	.894	.019	.892	.020
800	-.481	.204	-.480	.205	.002	.007	.003	.007	.893	.017	.891	.017
1000	.076	.206	.076	.208	.002	.006	.002	.006	.887	.015	.886	.016

Table 2. AR(1) plus noise model. Filtered mean (m) and standard deviation (s) for $(\alpha, \beta, \sigma^2)$ at times $t = 0, 200, \ldots, 1000$. We use the priors given in Section 4.1. Results are obtained using MCMC with 10000 iterations, and practical filter with $(N, G, k) = (10000, 10, 25)$. The true parameter values are $(\alpha, \beta, \sigma^2) = (0, .9, .04)$.

	α				β				σ^2			
t	MCMC		Practical		MCMC		Practical		MCMC		Practical	
	m	s	m	s	m	s	m	s	m	s	m	s
0	.001	.071	.001	.075	.900	.101	.898	.108	.050	.032	.057	.035
200	.006	.015	.009	.018	.862	.045	.828	.065	.041	.012	.059	.024
400	.007	.011	.010	.013	.868	.032	.837	.053	.043	.009	.057	.021
600	-.001	.008	.000	.010	.888	.023	.861	.044	.043	.007	.057	.019
800	.003	.008	.004	.009	.880	.021	.861	.041	.047	.006	.057	.018
1000	.003	.007	.003	.008	.869	.019	.855	.040	.050	.006	.057	.018

Fig. 1. AR(1) plus noise model. *Left:* Difference between smoothed means $E(x_{99-k}|\mathbf{y}_{1:100})$ and $E(x_{99-k}|\mathbf{y}_{1:99})$ under various fixed parameter values. *Right:* Kullback-Leibler divergence between the smoothing densities $p(x_{99-k}|\mathbf{y}_{1:100})$ and $p(x_{99-k}|\mathbf{y}_{1:99})$. Results are obtained analytically through the Kalman filter and smoother.

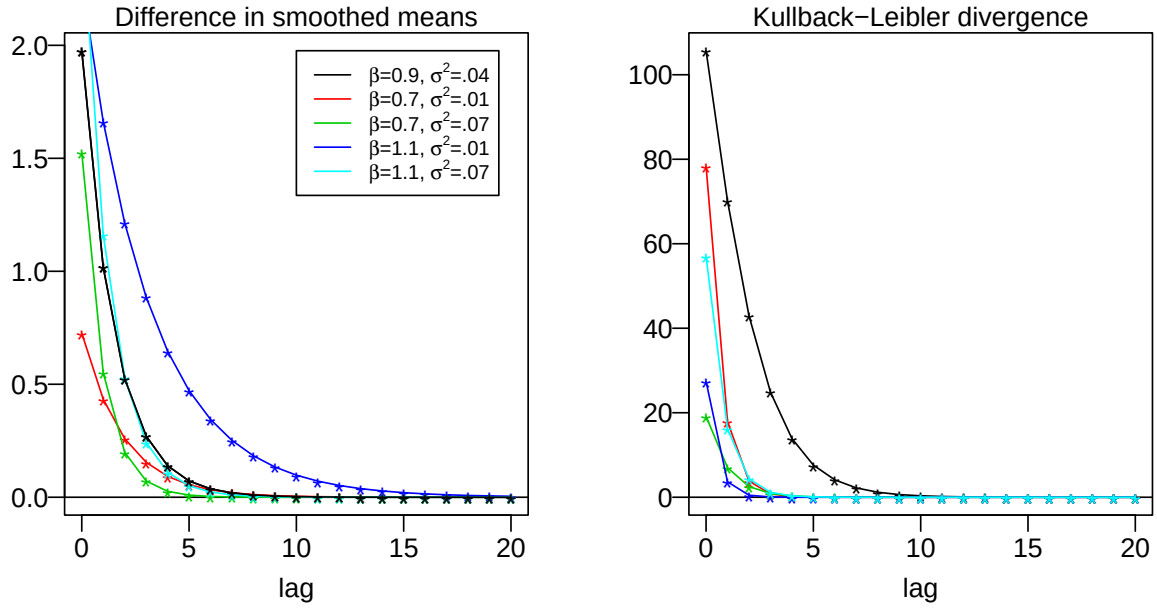

Fig. 2. Results for the AR(1) plus noise model with unknown parameters (α, β). An outlier of $y_t = 6$ is included at time $t = 50$. For each parameter, we plot the filtered mean and 95% intervals (mean plus minus 2 SD). *Left:* Practical filter with $(N, G, k) = (2000, 5, 25)$. *Right:* Storvik's algorithm with $N = 10000$.

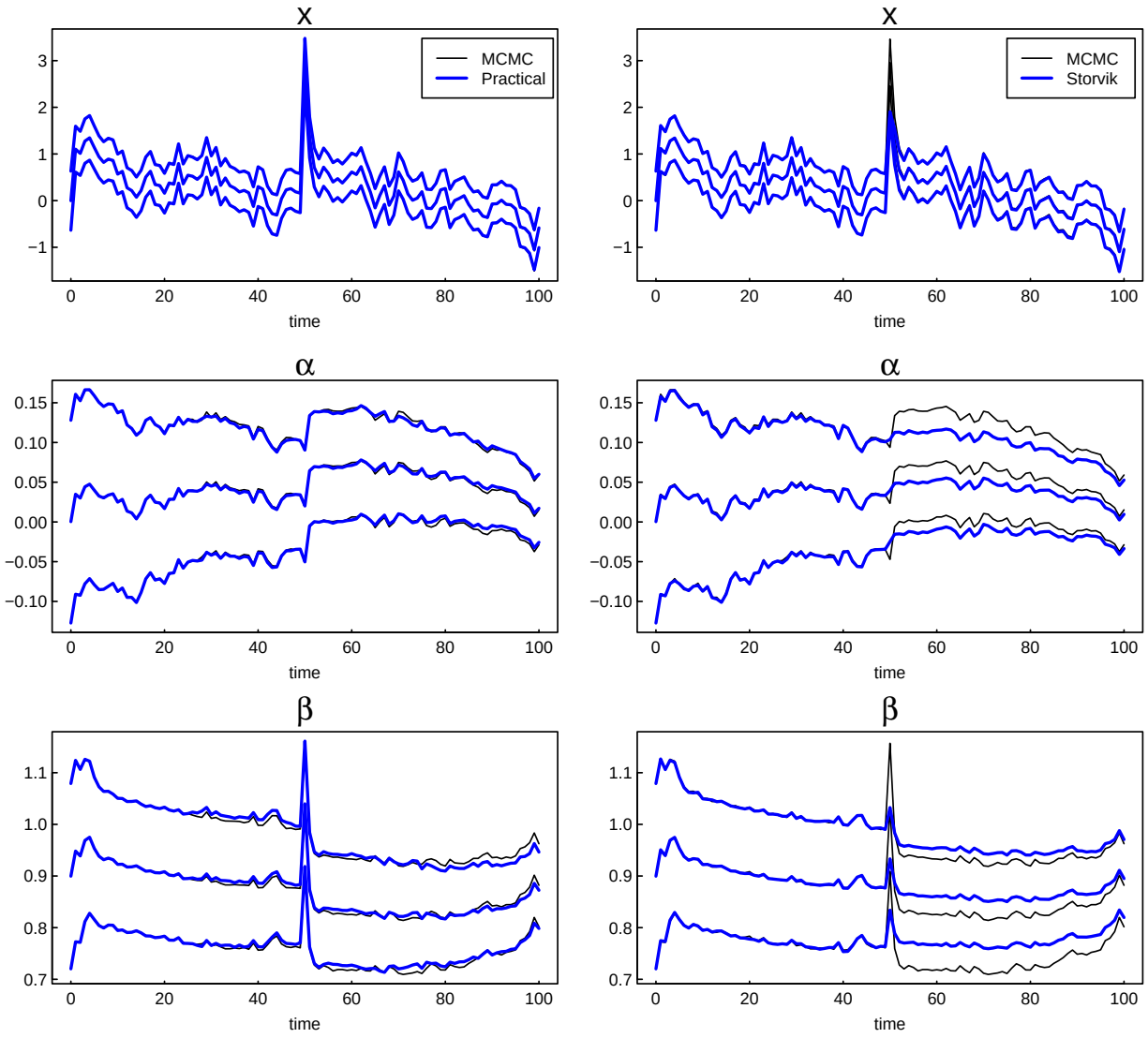

Fig. 3. Results for the AR(1) plus noise model with unknown parameters (α, β). An outlier of $y_t = 6$ is added at time $t = 50$. The posterior histogram is plotted for x_t, α, β at time $t = 50$, along with the true posterior density curves obtained by numerical integration using the Kalman filter. *Top:* Practical filter with $(N, G, k) = (2000, 5, 25)$. *Bottom:* Storvik's algorithm with $N = 10000$.

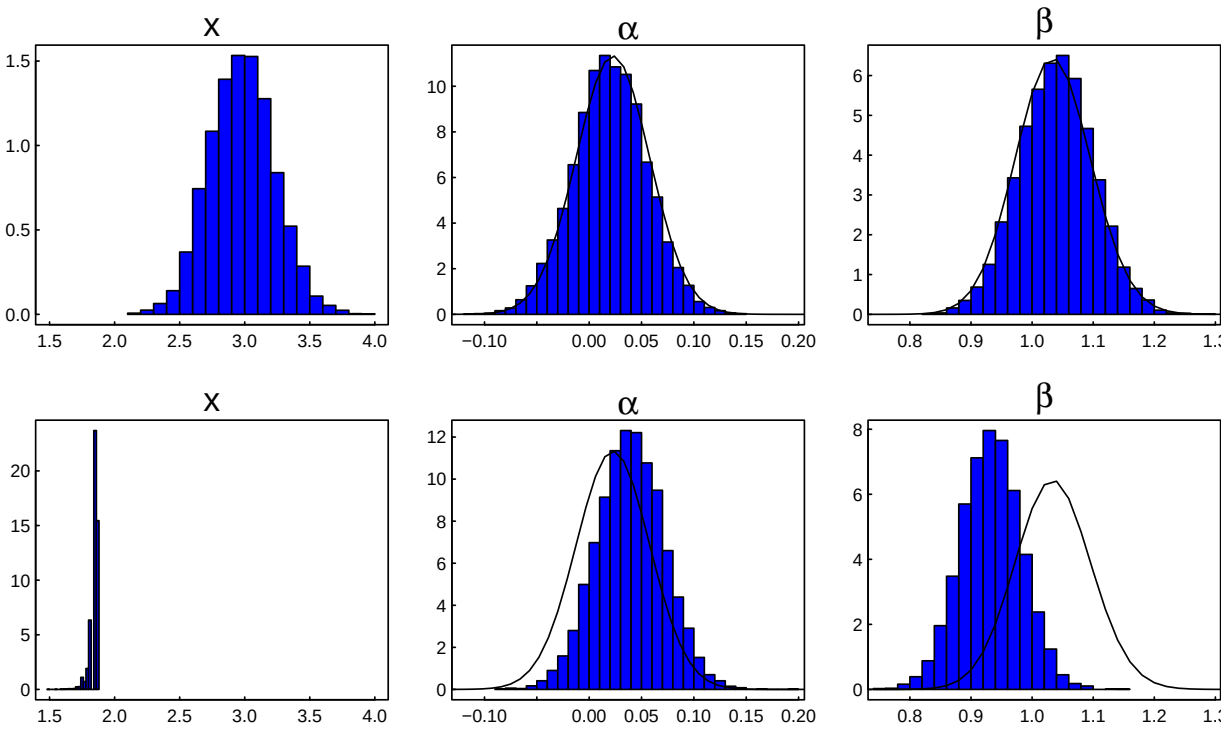

Fig. 4. Lorenz model: phase-space trajectory. Simulated path of state vector (black squares), and filtered mean (blue circles & line).

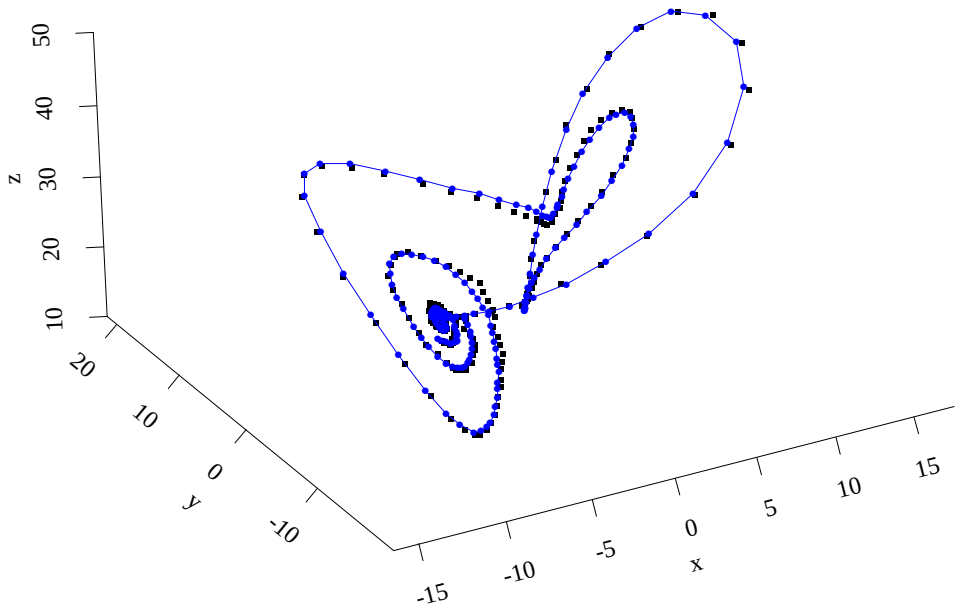

Fig. 5. Lorenz model. Filtered (5, 50, 95) percentiles for states and static parameters, using the practical filter with $(N, G, k) = (2500, 1, 25)$. Blue dots represent the true states. Green horizontal lines denote the true parameter values.