Particle Learning and Smoothing*

Carlos Carvalho, Michael Johannes, Hedibert Lopes and Nicholas Polson

This version: September 2009

First draft: December 2007

Abstract

In this paper we develop particle learning (PL) methods to perform state filtering, smoothing and sequential parameter learning for a general class of state space models. Our approach differs from existing particle methods in that it uses state sufficient statistics as particles. State smoothing with parameter uncertainty is solved in the process of particle learning. We show that particle learning provides significant improvements over existing particle filtering algorithms as well as the popular filter-forward-backwards-sample (FFBS) and MCMC methods in a number of applications.

Keywords: Particle Learning, Filtering, Smoothing, Mixture Kalman Filter, Bayesian Inference, Bayes factor, State Space Models.

*Carvalho, Lopes and Polson are at The University of Chicago, Booth School of Business, 5807 S. Woodlawn, Chicago IL 60637. Emails: `[carlos.carvalho, hlopes, ngp]@ChicagoBooth.edu`. Johannes is at the Graduate School of Business, Columbia University, 3022 Broadway, NY, NY, 10027. Email: `mj335@columbia.edu`.

1 Introduction

There are two general inference problems in state space models: first, state filtering and smoothing and secondly, sequential parameter learning. Typically, state filtering and smoothing are accomplished by Kalman filter-type methods and parameter estimation is accomplished by other estimation methods (Shumway and Stoffer, 2006). This paper presents a methodology that performs state filtering and smoothing while simultaneously estimating parameters. The parameter estimation entails sequential parameter learning. We obtain the full joint posterior distribution of states and parameters sequentially over time as new data arrives.

In linear Gaussian models with learning, our approach provides a computationally fast alternative to the popular filter-forward-backwards-sample (FFBS) algorithm developed in Carter and Kohn (1994) and Frühwirth-Schnatter (1994). For nonlinear non-Gaussian state space models, we also provide an alternative to standard MCMC methods developed in Carlin, Polson and Stoffer (1992). Our approach also includes the mixture Kalman filter models of Chen and Liu (2000). Specifically, we can seamlessly incorporate nonlinearity into the state evolution and provide full parameter learning. For mixture Kalman filter models, we can incorporate full parameter learning. The purpose of the paper is to extend particle learning methods to a wide class of models and to document the performance of our methodology against common competing approaches.

The central idea behind particle learning (PL) is the creation of a particle method that directly samples from the joint posterior distribution of state and parameter sufficient statistics. Our approach differs from existing particle methods in that it uses state sufficient statistics as particles. The parameters and states can then be generated offline given their respective sufficient statistics. This is achieved by updating the particles with a resample-propagate step as opposed to the standard propagate-resample approach. Moreover, we explicitly treat conditional sufficient statistics as states to improve efficiency. By re-sampling first it also provides a more efficient algorithm for pure filtering.

There are a number of advantages to using particle learning. First, it provides optimal filtering distributions in the presence of sequential parameter learning. For Gaussian DLMs and conditionally Gaussian DLMs (CDLM), particles are defined over both state and parameter sufficient statistics, such as the Kalman filter's mean and variance. Our approach extends Chen and Liu's (2000) mixture Kalman filter (MKF) method by allowing for parameter learning. We show how nonlinearity in the state evolution, but linearity in the observation equation, is straightforward to implement. This dramatically extends the class of models that MKF methods apply to. In the fully general case, we are no longer able to marginalize out the state variables and use state sufficient statistics, we can still utilize parameter sufficient statistics and generate exact samples from the particle posterior. In a comprehensive simulation study, we demonstrate that PL provides a significant improvement over the current benchmarks in the literature such as Liu and West (2001) for parameter learning and forward-filtering, backward-sampling (FFBS) for filtering.

Secondly, the full smoothing distribution of states can be achieved by a non-trivial extension of Godsill, Doucet and West's (2004) smoothing results for filtering with known parameters to all models considered above. In doing so, PL also extends Chen and Liu's (2000) MKF in computing the sequence of smoothed distributions, $p(x^t|y^t)$. Posterior inference is typically solved via Markov chain Monte Carlo (MCMC) methods as $p\left(x^t|y^t\right)$ is a difficult-to-compute marginal from $p\left(x^t,\theta|y^t\right)$. Our simulations provide strong evidence that PL dominates the standard MCMC strategies in computing time and delivers similar accuracy when computing $p\left(x^t|y^t\right)$. Gains are most remarkable in models with nonlinearities (CNDM), where the common alternative of MCMC with single-states updates are well known to be inefficient (Papaspiliopoulos and Roberts, 2008). Another advantage of PL over MCMC in these classes of models is the direct and recursive access to approximations of predictive distributions a key element in sequential model comparison. Finally, computational efficiency and speed. The use of state sufficient statistics dramatically reduces the Monte Carlo error inherent in these simulation methods.

We therefore build upon existing particle methods by allowing for parameter learning to accompany the filtering process. To be efficient, our algorithms utilize a resample-propagate algorithm together with a particle set that includes state sufficient statistics. Until now most of the literature focuses exclusively on state filtering conditional on parameters using particle filtering methods. Extending these algorithms to handle parameter learning is an active area of research with important developments appearing in Liu and West (2001), Storvik (2002), Fearnhead (2002) and Johannes and Polson (2007) to cite a few.

The paper is outlined as follows. Section 2 presents the general filtering and smoothing strategy for PL. Section 3 considers the class of conditional dynamic linear models. Section 4 discuss the implementation of our methods in nonlinear specifications. Sections 5 compares the performance of PL to MCMC and alternative particle methods.

2 Particle Learning and Smoothing

Sequential learning of states and parameter via their joint posterior distributions is computationally difficult due to their dimensionality and complicated probabilistic relationship between the parameters, states, and data. The classic example of recursive estimation is the Kalman filter (Kalman, 1960) in the case of linear Gaussian models with known parameters. For nonlinear and nonGaussian data, MCMC methods have been developed to solve the filtering, smoothing and learning problem (Carlin, Polson and Stoffer, 1992) but are too slow for the sequential problem, which requires on-line simulation based on a recursive or iterative structure.

Consider a general state space model defined by the observation and evolution equations

$$\begin{aligned} y_{t+1} &\sim p\left(y_{t+1}|x_{t+1},\theta\right) \\ x_{t+1} &\sim p\left(x_{t+1}|x_t,\theta\right), \end{aligned}$$

with initial state distribution $p\left(x_0|\theta\right)$ and prior $p(\theta)$. The sequential parameter learning and state filtering problem is characterized by the joint posterior distribution, $p\left(x_t,\theta|y^t\right)$

whereas smoothing is characterized by $p\left(x^T, \theta | y^T\right)$, with $y^t = (y_1, \ldots, y_t)$, $x^t = (x_1, \ldots, x_t)$ and T denoting the last observation time. In this paper, we use a particle approach to sequentially filter states and learn parameters via $p\left(x_t, \theta | y^t\right)$ and to obtain $p\left(x^T | y^T\right)$ by backwards propagation. Our approach solves the filtering, learning and smoothing problems for state space models.

2.1 State Filtering with Learning: Re-sample-Propagate

Generically, we assume that at time t, i.e. conditional on y^t, the particle approximation to $p(z_t|y^t)$ is given by the particle set $\{z_t^{(i)}, i = 1, \ldots, N\}$, where $z_t = (x_t, s_t, \theta)$. We differ from the current literature in that we track state sufficient statistics s_t^x as well as parameter sufficient statistics s_t^θ, such that $s_t = (s_t^x, s_t^\theta)$. Once y_{t+1} is observed, the algorithm has to update the particles in accordance with

$$\begin{aligned} p(z_t|y^{t+1}) &= p(y_{t+1}|z_t)p(z_t|y^t)/p(y_{t+1}|y^t) & (1) \\ p(x_{t+1}|y^{t+1}) &= \int p(x_{t+1}|z_t, y_{t+1})p(z_t|y^{t+1})dz_t. & (2) \end{aligned}$$

These are the classic Kalman-type propagation and marginalization rules, although in reverse order. This leads to a re-sample-propagate algorithm. Updating sufficient statistics s_{t+1} is deterministic and drawing a new state and parameter vector (x_{t+1}, θ) given s_t is given by the appropriate conditional posterior. A particle approximation to (1) is

$$p^N(z_t|y^{t+1}) = \sum_{i=1}^N w_t^{(i)} \delta_{z_t^{(i)}}(z_t)$$

where $\delta_z(\cdot)$ denotes the delta-Dirac mass located in z and normalized weights

$$w_t^{(i)} = \frac{p(y_{t+1}|z_t^{(i)})}{\sum_{j=1}^N p(y_{t+1}|z_t^{(j)})}. \tag{3}$$

This updated approximation of $p(z_t|y^{t+1})$ is used in (2) to generate propagated samples from the posterior $p(x_{t+1}|z_t, y_{t+1})$, which are, in turn, used to deterministically update s_{t+1}^x by the recursive map

$$s_{t+1}^x = \mathcal{S}(s_t^x, x_{t+1}, \theta, y_{t+1}) \tag{4}$$

where we allow a dependence on θ. Similarly for s^{θ}_{t+1}. Finally, parameter learning $p(\theta|y^{t+1})$ will then be performed "offline" by simulating particles from $p(\theta|s_{t+1})$. Therefore, after observing y_{t+1}, we have a particle approximation $p^N(z_{t+1}|y^{t+1})$ given by a set of particles $\{z^{(i)}_{t+1}, i = 1, \ldots, N\}$.

Particle Learning

1. Resample particles $z_t = (x_t, s_t, \theta)$ with weights

$$w_t^{(i)} \propto p(y_{t+1}|z_t^{(i)})$$

2. Propagate new states x_{t+1} from $p(x_{t+1}|z_t, y_{t+1})$

3. Update state and parameter sufficient statistics $s_{t+1} = \mathcal{S}(s_t, x_{t+1}, y_{t+1})$

4. Sample θ from $p(\theta|s_{t+1})$

Our approach departures from the standard particle filtering literature by inverting the order in which states are propagated and updated. By re-sampling first we reduce the compounding of approximation errors as the states are propagated after being "informed" by y_{t+1}. In addition, borrowing the terminology of Pitt and Shephard (1999), PL is a fully-adapted filter that starts from a particle approximation $p^N(z_t|y^t)$ and provides direct (or exact) draws from the particle approximation $p^N(z_{t+1}|y^t)$. This reduces the accumulation of Monte Carlo error as there is no need to re-weight or compute importance sampling weights at the end of the filter.

Since s_t and x_{t+1} are random variables, the conditional sufficient statistics s_{t+1} are also random and are replenished, essentially as a state, in the filtering step. This is the key insight for handling the learning of θ. The particles for s_{t+1} are sequentially updated and replenished as a function of x_{t+1} and updated samples from $p(\theta|s_{t+1})$ can be obtained at the end of the filtering step. We describe this in detail in the next section.

Finally, updated samples for s_{t+1} are obtained as a function of the samples of x_{t+1}, with weights $1/N$, which prevents particle degeneracies in the estimation of θ. This is a feature of the “re-sample-propagate” mechanism of PL. Conversely, any “propagate-re-sample” strategy will lead to decay in the particles of x_{t+1} with significant negative effects on the parameter particles. This strategy will only be possible whenever both $p(y_{t+1}|z_t)$ and $p(x_{t+1}|z_t, y^{t+1})$ are analytically tractable which is the case in the classes of models considered here. The above interpretation provides a clear intuition for the construction of effective filters as the goal is to come up with proposals that closely match these densities.

2.2 Parameter Learning

For all models considered, we assume that the posterior for the parameter vector θ admits a conditional sufficient statistics structure given the states and data (x^t, y^t), so that

$$p(\theta|x^t, y^t) = p(\theta|s_t),$$

where s_t is a recursively defined sufficient statistics, $s_{t+1} = \mathcal{S}(s_t, x_{t+1}, y_{t+1})$. The use of sufficient statistics is foreshadowed in Gilks and Berzuini (2001), Storvik (2002) and Fearnhead (2002). Previous approaches to parameter learning include the addition of θ in the particle set or the introduction of evolutionary noise, turning θ into an artificial state (Gordon *et al.*, 1993). The first alternative breaks down quickly as there is no mechanism for replenishing the particles of θ whereas the second overestimate the uncertainty about θ by working with a different model. Liu and West (2001) propose a learning scheme adapted to the auxiliary particle filter that approximates $p(\theta|y^t)$ by a kernel mixture of normals. While being a widely used and applicable strategy, it still degenerates fairly quickly because the propagation mechanism for x_{t+1} and θ does not take into account the current information y_{t+1}. This is highlighted in our simulation studies of Section 5. Another alternative is to consider sampling over (θ, x^t) with a Metropolis kernel as in Gilks and Berzuini (2001) and Polson, Stroud and Müller (2008). The complexity of this strategy grows with t and suffers from the curse of dimensionality as discussed in Bengtsson, Bickel, and Li (2008). Our approach is of fixed dimension as it targets the filtering distribution of (s_t, x_t).

The introduction of sufficient statistics provides the mechanism to break particle decay by sequentially targeting (s_t, x_t), where s_t is automatically replenished as a function of the current samples for x_t and observation y_t. One way to interpret this strategy is to think of s_t as an additional latent state in the model, with a deterministic evolution determined by (4). This can be seen by the following factorization of the joint conditional posterior

$$p(x_t, s_t, \theta|y^t) = p(\theta|s_t)p(x_t, s_t|y^t).$$

Our algorithm will be based on developing a particle approximation $p^N(x_t, s_t|y^t)$ to the joint distribution of states and sufficient statistics.

Example 1. First order DLM. *For illustration, let start considering the simple first order dynamic linear model, also known as the local level model (West and Harrison, 1997), where*

$$(y_{t+1}|x_{t+1}, \theta) \sim N(x_{t+1}, \sigma^2)$$
$$(x_{t+1}|x_t, \theta) \sim N(x_t, \tau^2)$$

with $\theta = (\sigma^2, \tau^2)$, $x_0 \sim N(m_0, C_0)$, $\sigma^2 \sim IG(a_0, b_0)$ *and* $\tau^2 \sim IG(c_0, d_0)$*. The initial hyperparameters* m_0, C_0, a_0, b_0, c_0 *and* d_0 *can be fixed or random. This model implies* $(y_{t+1}|x_t, \theta) \sim N(x_t, \sigma^2 + \tau^2)$, $(x_{t+1}|y_{t+1}, x_t, \theta) \sim N(\mu_t, \omega^2)$*, where* $\mu_t = \omega^2(\sigma^{-2}y_{t+1} + \tau^{-2}x_t)$ *and* $\omega^{-2} = \sigma^{-2} + \tau^{-2}$*. For the parameter conditionals,* $(\sigma^2|y^{t+1}, x^{t+1}) \sim IG(a_{t+1}, b_{t+1})$ *and* $(\tau^2|y^{t+1}, x^{t+1}) \sim IG(c_{t+1}, d_{t+1})$*, where* $a_{t+1} = a_t + 1/2$, $c_{t+1} = c_t + 1/2$, $b_{t+1} = b_t + 0.5(y_{t+1} - x_{t+1})^2$ *and* $d_{t+1} = d_t + 0.5(x_{t+1} - x_t)^2$*. Therefore, one can track the vector of conditional sufficient statistics* $s_{t+1} = (y_{t+1}^2, y_{t+1}x_{t+1}, x_{t+1}^2, x_t^2, x_{t+1}x_t)$.

2.3 State Sufficient Statistics

The use of state sufficient statistics will lead to an efficient approach in dynamic linear models. In the predictive, we marginalize states and just track conditional state sufficient statistics. Here we use the fact that

$$p(x_t|y^t) = \int p(x_t|s_t^x)p(s_t^x|y^t)ds_t^x = E_{p(s_t^x|y^t)}[p(x_t|s_t^x)].$$

and we track particles for $p(s_t^x|y^t)$. The filtering recursions linking $p(s_t^x|y^t)$ to $p(s_{t+1}^x|y^{t+1})$ are given by

$$\begin{aligned} p(s_{t+1}^x|y^{t+1}) &= \int p(s_{t+1}^x|s_t^x, x_{t+1}, y_{t+1})p(s_t^x, x_{t+1}|y^{t+1})ds_t^x dx_{t+1} \\ &\propto \int p(s_{t+1}^x|s_t^x, x_{t+1}, y_{t+1})p(y_{t+1}|s_t^x)p(x_{t+1}|s_t^x, y_{t+1})p(s_t^x|y^t)ds_t^x dx_{t+1}, \end{aligned}$$

where we use $p(y_{t+1}|s_t^x)$ for resampling weights. Instead of marginalizing x_t, you now marginalize over s_t^x and x_{t+1}. For this to be effective we need the following conditional posterior

$$p(x_{t+1}|s_t^x, y_{t+1}) = \int p(x_{t+1}|x_t, y_{t+1})p(x_t|s_t^x)dx_t.$$

We can then proceed with the particle learning algorithm. The weights $p(y_{t+1}|s_t^x)$ are flatter than those of $p(y_{t+1}|x_t)$, due to Rao-Blackwellisation, and this will add to the efficiency of the algorithm.

Example 1. (cont.) *Conditional on parameters* $(x_t|\theta) \sim N(m_t, C_t)$ *with* $s_t^x = (m_t, C_t)$. *It is then straightforward to derive* $(y_{t+1}|m_t, C_t, \theta) \sim N(m_t, C_t + \sigma^2 + \tau^2)$. *The recursions for the state sufficient statistics vector* s_t^x *are the well known Kalman recursions,* $m_{t+1} = (1 - A_{t+1})m_t + A_{t+1}y_{t+1}$ *and* $C_{t+1} = A_{t+1}\sigma^2$, *where* $A_{t+1} = (C_t + \tau^2)/(C_t + \tau^2 + \sigma^2)$ *is the Kalman gain.*

2.4 Smoothing

After one sequential pass through the data, our particle approach computes samples from $p^N(x_t, s_t|y^t)$ for all t. However, in many situations, we also require samples from the full smoothing distribution $p(x^T|y^T)$. This is typically carried out by a MCMC scheme. We now show that our particle learning strategy provides a direct backward sequential pass to also sample from the smoothing distribution. To compute the marginal smoothing distribution, we write the joint posterior of (x^T, θ) as

$$p(x^T, \theta|y^T) = \prod_{t=1}^{T-1} p(x_t|x_{t+1}, \theta, y^t)p(x_T, \theta|y^T).$$

By Bayes rule and conditional independence we have

$$p(x_t|x_{t+1}, \theta, y^t) \propto p(x_{t+1}|x_t, \theta, y^t)p(x_t|\theta, y^t). \tag{5}$$

We can now derive a recursive backward sampling algorithm to sample from $p(x^T, \theta|y^T)$ by sequentially sampling from filtered particles with weights proportional to $p(x_{t+1}|x_t, \theta, y^t)$. In detail, randomly choose at T, $(\tilde{x}_T, \tilde{s}_T)$ from the particle approximation $p^N(x_T, s_T|y^T)$ and sample $\tilde{\theta} \sim p(\theta|\tilde{s}_T)$. Going backwards, for $t = T-1, \ldots, 1$, we choose $\tilde{x}_t = x_t^{(i)}$ from the filtered particles $\{x_t^{(i)}, i = 1, \ldots, N\}$ with (unnormalized) weights $w_{t|t+1}^{(i)} = p(\tilde{x}_{t+1}|x_t^{(i)}, \tilde{\theta})$.

Particle Smoothing

1. Forward filtering $(x^T, \theta)^{(i)}$ via *Particle Learning*.
2. Backwards smoothing, for each pair $(x_T^{(i)}, \theta^{(i)})$, with weights

$$w_{t|t+1}^{(j)} \propto p(x_{t+1}^{(i)}|x_t^{(j)}, \theta^{(i)})$$

for $t = T-1, \ldots, 1$.

Our particle smoothing algorithm is an extension of Godsill, Doucet and West (2004) method to state space models with parameter learning. See also, Briers, Doucet and Maskell (2009) for an alternative SMC smoother. Both SMC smoothers are $O(TN^2)$, so the computational time to obtain draws from $p(x^T|y^T)$ is expected to be much larger than the computational time to obtain draws from $p(x_t|y^t)$, for $t = 1, \ldots, T$, from standard SMC filters. An $O(TN)$ smoothing algorithm has recently been introduced by Fearnhead, Wyncoll and Tawn (2008).

Example 1. (cont.) *For* $t = T-1, \ldots, 2, 1$, *we can derive* $(x_t|x_{t+1}, y^T, \theta) \sim N(a_t, D_t\tau^2)$ *and* $(x_t|y^T, \theta) \sim N(m_t^T, C_t^T)$, *where* $a_t = (1 - D_t)m_t + D_t x_{t+1}$ $m_t^T = (1 - D_t)m_t + D_t m_{t+1}^T$, $C_t^T = (1 - D_t)C_t + D_t^2 C_{t+1}^T$, *and* $D_t = C_t/(C_t + \tau^2)$ *with terminal conditions* $m_T^T = m_T$ *and* $C_T^T = C_T$.

2.5 Model Monitoring

The particle output of PL can also be used to for sequential predictive problems and model assessment in state space models. Specifically, the marginal predictive for a given model M can be approximated via

$$p^N(y_{t+1}|y^t, M) = \frac{1}{N}\sum_{i=1}^{N} p(y_{t+1}|(x_t, \theta)^{(i)}, M).$$

This then allows the computation of a particle approximation to the Bayes factor B_t or sequential likelihood ratios for competing models M_0 and M_1 (see, for example, West, 1986) defined by:

$$B_t = \frac{p(y^t|M_1)}{p(y^t|M_0)}$$

where $p(y^t|M_i) = \prod_{j=1}^{t} p(y_j|y^{j-1}, M_i)$, for $i = 0, 1$. We then simply sequentially approximate each predictive term using our particle method.

Model Monitoring

Calculate the marginal likelihood using

$$p^N(y_{t+1}|y^t) = \frac{1}{N}\sum_{i=1}^{N} p(y_{t+1}|(x_t,\theta)^{(i)}).$$

An important advantage of PL over MCMC schemes is that it directly provides the filtered joint posteriors $p(x_t,\theta|y^t)$ and hence $p(y_{t+1}|y^t)$ whereas MCMC would have to be repeated T times to make that available.

2.6 Discussion

Our approach relies on three main insights to deal with these problems: (i) conditional sufficient statistics are used to represent the posterior of θ. This allows us to think of the sufficient statistics as additional states that are sequentially updated. Whenever possible, sufficient statistics for the latent states are also introduced. This increases the efficiency of our algorithm by reducing the variance of sampling weights in what can be called a Rao-Blackwellized filter. (ii) We use a re-sample/propagate framework to provide exact samples from our particle approximation when moving from $p^N(x_t,\theta|y^t)$ to $p^N(x_{t+1},\theta|y^{t+1})$. This avoids sample importance re-sampling (SIR) (Gordon, Salmond and Smith, 1993) and the associated "decay" in the particle approximation. Finally, (iii) we extend the backward propagation smoothing algorithm of Godsill, Doucet and West (2004) to incorporate uncertainty about θ.

Another way to see why the algorithm performs well is as follows. Consider the pure filtering problem. Traditionally, this is performed via a prediction step and then an updating step, i.e.

$$p(x_{t+1}|y^t) = \int p(x_{t+1}|x_t)p(x_t|y^t)dx_t \tag{6}$$

$$p(x_{t+1}|y^{t+1}) = p(y_{t+1}|x_{t+1})p(x_{t+1}|y^t)/p(y_{t+1}|y^t). \tag{7}$$

(see Kalman, 1960). This has lead to numerous particle filtering algorithms that are first based on propagation (the prediction step) and then on re-sampling (the updating step). These algorithms have well known degeneracy problems, most clearly discussed in Pitt and Shephard (1999). Our strategy uses Bayes rule and reverses the logic with:

$$p(x_t|y^{t+1}) = p(y_{t+1}|x_t)p(x_t|y^t)/p(y_{t+1}|y^t) \tag{8}$$

$$p(x_{t+1}|y^{t+1}) = \int p(x_{t+1}|x_t, y_{t+1})p(x_t|y^{t+1})dx_t. \tag{9}$$

Therefore, our algorithm will first re-sample (smooth) and then propagate. Notice that this approach first solves a smoothing problem, re-sampling to characterize $p(x_t|y^{t+1})$, and then propagates these re-sampled states. Since information in y_{t+1} is used in both steps, the algorithm will be more efficient. Mathematically, the weights used for re-sampling are

$$p(y_{t+1}|x_t) = \int p(y_{t+1}|x_{t+1})p(x_{t+1}|x_t)dx_{t+1}. \tag{10}$$

Throughout, we note that efficiency improvements can be made by marginalizing out as many variables from the predictive distribution in the re-sampling step as possible. Most efficient will be re-sampling with predictive $p(y_{t+1}|s_t^x, s_t)$, that is, given just the conditional sufficient statistics for parameters and states (s_t^x) (see Section 3). This point can be casted in terms of the effective sample size (see Kong, Liu and Wong, 1994) and its relationship to the re-sampling weights in pure filtering context, conditionally on θ. The weights for standard approaches are based on blind propagation and are given by $w(x_{t+1}) = p(y_{t+1}|x_{t+1}, \theta)$. Our algorithm reverses this logic and first re-samples and then propagates. This has weights $w(x_t) = p(y_{t+1}|x_t, \theta)$. The most efficient approach is to marginalize,whenever possible, over the state vector and condition on s_t^x leading to weights $w(s_t^x)$. This is the case for the models presented in the next Section.

Convergence properties of the algorithm are straightforward to establish. The choice of particle size N to achieve a desired level of accuracy depends on the speed of Monte Carlo accumulation of error. In many cases this will be uniformly bounded. As with MCMC the larger the signal to noise ratio the more errors propagate. This should not be confused with the nature of particle approximations to diffuse distributions which can also lead to larger particle sizes. At its simplest level the algorithm only requires samples $\theta^{(i)}$ from the prior $p(\theta)$. However, a natural class are mixtures of the form $p(\theta) = \int p(\theta|z_0)p(z_0)dz_0$. The conditional $p(\theta|z_0)$ is chosen to be naturally conjugate so as to facilitate conditional sufficient statistics. If z_0 is fixed, then we start all particles out with the same z_0 value. More commonly, we will start with a sample $z_0^{(i)} \sim p(Z_0)$. PL will first resample these draws with the marginal likelihood $p(y_1|z_0^{(i)})$ providing large efficiency gains over blindly samples. Mixtures allow for a range of non-conjugate priors such as vague "uninformative" priors induced by scale mixtures of normals.

3 Conditional Dynamic Linear Models

We now explicitly derive our PL algorithm in a class of conditional dynamic linear models which are an extension of the models considered in West and Harrison (1997). This consists of a vast class of models that embeds many of the commonly used dynamic models. MCMC via Forward-filtering Backwards-sampling or mixture Kalman filtering (MKF) (Liu and Chen, 2000) are the current methods of use for the estimation of these models. As an approach for filtering, PL has a number of advantages over MKF. First, our algorithm is more efficient as it is a perfectly-adapted filter. Second, we extend MKF by including learning about fixed parameters and smoothing for states.

The conditional DLM defined by the observation and evolution equations takes the form of a linear system conditional on an auxiliary state λ_{t+1}

$$(y_{t+1}|x_{t+1}, \lambda_{t+1}, \theta) \sim N(F_{\lambda_{t+1}} x_{t+1}, V_{\lambda_{t+1}})$$
$$(x_{t+1}|x_t, \lambda_{t+1}, \theta) \sim N(G_{\lambda_{t+1}} x_t, W_{\lambda_{t+1}})$$

with θ containing Fs, Gs, Vs and Ws. The marginal distribution of observation error and state shock distribution are any combination of normal, scale mixture of normals, or discrete mixture of normals depending on the specification of the distribution on the auxiliary state variable $p(\lambda_{t+1}|\theta)$, so that,

$$p(y_{t+1}|x_{t+1},\theta) = \int f_N(y_{t+1}; F_{\lambda_{t+1}}x_{t+1}, V_{\lambda_{t+1}})p(\lambda_{t+1}|\theta)d\lambda_{t+1}.$$

Extensions to hidden Markov specifications where λ_{t+1} evolves according to $p(\lambda_{t+1}|\lambda_t,\theta)$ are straightforward and are discussed in Example 3.1 below.

3.1 Particle Learning in CDLM

In CDLMs the state filtering and parameter learning problem is equivalent to a filtering problem for the joint distribution of their respective sufficient statistics. This is a direct factorization of the full joint as

$$p\left(x_{t+1},\theta,\lambda_{t+1},s_{t+1},s^x_{t+1}|y^{t+1}\right) = p(\theta|s_{t+1})p(x_{t+1}|s^x_{t+1},\lambda_{t+1})p\left(\lambda_{t+1},s_{t+1},s^x_{t+1}|y^{t+1}\right),$$

The conditional sufficient statistics for states (s^x_t) and parameters (s^θ_t) satisfy the deterministic updating rules

$$s^x_{t+1} = \mathcal{K}\left(s^x_t,\theta,\lambda_{t+1},y_{t+1}\right) \tag{11}$$

$$s^\theta_{t+1} = \mathcal{S}\left(s^\theta_t,x_{t+1},\lambda_{t+1},y_{t+1}\right) \tag{12}$$

where $\mathcal{K}(\cdot)$ denotes the Kalman filter recursions and $\mathcal{S}(\cdot)$ the recursive update of the parameter sufficient statistics. With $s^x_t = (m_t, C_t)$ as Kalman filter first and second moments at time t we have, conditional on θ,

$$p\left(x_{t+1}|s^x_{t+1},\theta,\lambda_{t+1}\right) \sim N(a_{t+1},R_{t+1})$$

where $a_{t+1} = G_{\lambda_{t+1}}m_t$ and $R_{t+1} = G_{\lambda_{t+1}}C_tG'_{\lambda_{t+1}} + W_{\lambda_{t+1}}$ Updating state sufficient statistics (m_{t+1}, C_{t+1}) is achieved by

$$m_{t+1} = G_{\lambda_{t+1}}m_t + A_{t+1}\left(y_{t+1} - F_{\lambda_{t+1}}G_{\lambda_{t+1}}m_t\right) \tag{13}$$

$$C^{-1}_{t+1} = R^{-1}_{t+1} + F'_{\lambda_{t+1}}F_{\lambda_{t+1}}V^{-1}_{\lambda_{t+1}} \tag{14}$$

where the Kalman gain matrix is $A_{t+1} = R_{t+1} F_{\lambda_{t+1}} Q_{t+1}^{-1}$ and $Q_{t+1} = F_{\lambda_{t+1}} R_{t+1} F_{\lambda_{t+1}} + V_{\lambda_{t+1}}$.

We are now ready to define the PL scheme for the CDLM. First, without loss of generality, assume that the auxiliary state variable is discrete with $\lambda_{t+1} \sim p(\lambda_{t+1}|\lambda_t, \theta)$. We start, at time t, with a particle approximation for the joint posterior

$$p^N(x_t, \theta, \lambda_t, s_t, s_t^x|y^t) = \frac{1}{N}\sum_{i=1}^{N} \delta_{(x_t,\theta,\lambda_t,s_t,s_t^x)^{(i)}}(x_t, \theta, \lambda_t, s_t, s_t^x)$$

that are then propagated to $t+1$ by first re-sampling the current particles with weights proportional to the predictive $p(y_{t+1}|(\theta, s_t^x))$. This provides a particle approximation to $p(x_t, \theta, \lambda_t, s_t, s_t^x|y^{t+1})$, the smoothing distribution. New states λ_{t+1} and x_{t+1} are then propagated through the conditional posterior distributions $p(\lambda_{t+1}|\lambda_t, \theta, y_{t+1})$ and $p(x_{t+1}|\lambda_{t+1}, x_t, \theta, y_{t+1})$. Finally the conditional sufficient statistics are updated according to (11) and (12) and new samples for θ are obtained from $p(\theta|s_{t+1})$. Notice that in the conditional dynamic linear models all the above densities are available for evaluation and sampling. For instance, the predictive is computed via

$$p(y_{t+1}|(\lambda_t, s_t^x, \theta)^{(i)}) = \sum_{\lambda_{t+1}} p(y_{t+1}|\lambda_{t+1}, (s_t^x, \theta)^{(i)}) p(\lambda_{t+1}|\lambda_t, \theta)$$

where the inner predictive distribution is given by

$$p\left(y_{t+1}|\lambda_{t+1}, s_t^x, \theta\right) = \int p\left(y_{t+1}|x_{t+1}, \lambda_{t+1}, \theta\right) p(x_{t+1}|s_t^x, \theta) dx_{t+1}.$$

Starting with particle set $\{(x_0, \theta, \lambda_0, s_0, s_0^x)^{(i)}, i = 1, \ldots, N\}$ at time $t = 0$, the above discussion can be summarized in the following PL algorithm:

```
                    ***Algorithm 1: CDLM***
For $t = 0, \dots, T-1$
For $i = 1, \dots, N$
  *Step 1* (Re-sample): Generate an index $k^i \sim \text{Multinomial}(w_{t+1}^{(1)}, \dots, w_{t+1}^{(N)})$ where
      $w_{t+1}^{(j)} \propto p(y_{t+1}|(\lambda_t, s_t^x, \theta)^{(j)}) \qquad j = 1, \dots, N$
  *Step 2* (Propagate): States
      $\lambda_{t+1}^{(i)} \sim p(\lambda_{t+1}|(\lambda_t, \theta)^{(k^i)}, y_{t+1})$
      $x_{t+1}^{(i)} \sim p(x_{t+1}|(x_t, \theta)^{(k^i)}, \lambda_{t+1}^{(i)}, y_{t+1})$
  *Step 3* (Propagate): Sufficient Statistics
      $s_{t+1}^{x(i)} = \mathcal{K}((s_t^x, \theta)^{(k^i)}, \lambda_{t+1}^{(i)}, y_{t+1})$
      $s_{t+1}^{(i)} = \mathcal{S}(s_t^{k^i}, x_{t+1}^{(i)}, \lambda_{t+1}^{(i)}, y_{t+1})$
  *Step 4* (Propagate): Parameters
      $\theta^{(i)} \sim p(\theta|s_{t+1}^{(i)})$
```

In the general case where the auxiliary state variable λ_t is continuous it might not be possible to integrate out λ_{t+1} form the predictive in step 1. We extend the above scheme by adding to the current particle set a propagated particle $\lambda_{t+1} \sim p(\lambda_{t+1}|(\lambda_t, \theta)^{(i)})$ and define the following PL algorithm:

```
Algorithm 2: Auxiliary State CDLM
For $t = 0, \dots, T-1$
  Step 0 (Propagate) State $\lambda_{t+1}$
      $\lambda_{t+1}^{(i)} \sim p(\lambda_{t+1}|(\lambda_t, \theta)^{(i)}) \qquad i = 1, \dots, N$
  For $i = 1, \dots, N$
      Step 1 (Re-sample): Generate an index $k^i \sim$ Multinomial$(w_{t+1}^{(1)}, \dots, w_{t+1}^{(N)})$
          where
          $w_{t+1}^{(j)} \propto p(y_{t+1}|(\lambda_{t+1}, s_t^x, \theta)^{(j)}) \qquad j = 1, \dots, N$
      Step 2 (Propagate): State $x_{t+1}$
          $x_{t+1}^{(i)} \sim p(x_{t+1}|(x_t, \lambda_{t+1}, \theta)^{(k^i)}, y_{t+1})$
      Step 3 (Propagate): Sufficient Statistics (same as in Algorithm 1)
      Step 4 (Propagate): Parameters (same as in Algorithm 1)
```

Both of the above algorithms can be combined with the backwards propagation scheme of Section 2.4 to provide a full draw from the marginal posterior distribution for all the states given the data, namely $p(x^T|y^T)$.

Example 2. Dynamic Factor Model with Time-Varying Loadings. *Consider data* $y_t = (y_{t1}, y_{t2})'$, $t = 1, \dots, T$, *following a dynamic factor model with time-varying loadings driven by a discrete latent state* λ_t *with possible values* $\{1, 2\}$. *Specifically, we have*

$$(y_{t+1}|x_{t+1}, \lambda_{t+1}, \theta) \sim N(\boldsymbol{\beta}_{t+1} x_{t+1}, \sigma^2 I_2)$$
$$(x_{t+1}|x_t, \lambda_{t+1}, \theta) \sim N(x_t, \tau^2)$$

with time-varying loadings $\boldsymbol{\beta}_{t+1} = (1, \beta_{\lambda_{t+1}})'$ *and initial state distribution* $x_0 \sim N(m_0, C_0)$. *The jumps in the factor loadings are driven by a Markov switching process* $(\lambda_{t+1}|\lambda_t, \theta)$, *whose transition matrix Π has diagonal elements $Pr(\lambda_{t+1} = 1|\lambda_t = 1, \theta) = p$ and $Pr(\lambda_{t+1} = 2|\lambda_t = 2, \theta) = q$. The parameters are $\theta = (\beta_1, \beta_2, \sigma^2, \tau^2, p, q)'$. See Lopes and Carvalho (2007) and Carvalho and Lopes (2007) for related Markov switching models.*

We are able to develop the algorithm marginalizing over both (x_{t+1}, λ_{t+1}) by using state sufficient statistics $s_t^x = (m_t, C_t)$ as particles. From the Kalman filter recursions we know that $p(x_t|\lambda^t, \theta, y^t) \sim N(m_t, C_t)$. The mapping for state sufficient statistics $(m_{t+1}, C_{t+1}) = \mathcal{K}(m_t, C_t, \lambda_{t+1}, \theta, y_{t+1})$ is given by the one-step Kalman update as in (13) and (14). The prior distributions are conditionally conjugate where $(\beta_i|\sigma^2) \sim N(b_{i0}, \sigma^2 B_{i0})$ for $i = 1, 2$, $\sigma^2 \sim IG(\nu_{00}/2, d_{00}/2)$ and $\tau^2 \sim IG(\nu_{10}/2, d_{10}/2)$. For the transition probabilities, we assume that $p \sim Beta(p_1, p_2)$ and $q \sim Beta(q_1, q_2)$. Assume that at time t, we have particles $\{(x_t, \theta, \lambda_t, s_t^x, s_t)^{(i)}, i = 1, \ldots, N\}$ approximating $p(x_t, \theta, \lambda_t, s_t^x, s_t|y^t)$. The PL algorithm can be described through the following steps:

1. Re-sampling: *Draw an index $k^i \sim$ Multinomial$(w_t^{(1)}, \ldots, w_t^{(N)})$ with weights $w_t^{(i)} \propto p(y_{t+1}|(m_t, C_t, \lambda_t, \theta)^{(k^i)})$ where*

   $$p(y_{t+1}|m_t, C_t, \lambda_t, \theta) = \sum_{\lambda_{t+1}=1}^{2} f_N(y_{t+1}; \boldsymbol{\beta}_{t+1} m_t, (C_t + \tau^2)\boldsymbol{\beta}_{t+1}\boldsymbol{\beta}'_{t+1} + \sigma^2 I_2) p(\lambda_{t+1}|\lambda_t, \theta),$$

   where $f_N(x; a, b)$ denotes the density of the normal distribution with mean a and variance b and evaluation at the point x.

2. Propagating λ: *Draw auxiliary state $\lambda_{t+1}^{(i)}$ from $Pr(\lambda_{t+1}|(s_t^x, \lambda_t, \theta)^{(k^i)}, y_{t+1})$, where*

   $$Pr(\lambda_{t+1}|s_t^x, \lambda_t, \theta, y_{t+1}) \propto f_N(y_{t+1}; \boldsymbol{\beta}_{t+1} m_t, (C_t + \tau^2)\boldsymbol{\beta}_{t+1}\boldsymbol{\beta}'_{t+1} + \sigma^2 I_2) Pr(\lambda_{t+1}|\lambda_t, \theta).$$

3. Propagating x: *Draw states $x_{t+1}^{(i)} \sim p(x_{t+1}|\lambda_{t+1}^{(i)}, (s_t^x, \theta)^{(k^i)}, y_{t+1})$*

4. Updating sufficient statistics for states: *The Kalman filter recursions yield*

   $$m_{t+1} = m_t + A_{t+1}(y_{t+1} - \boldsymbol{\beta}_{t+1} m_t)$$
   $$C_{t+1} = C_t + \tau^2 - A_{t+1} Q_{t+1}^{-1} A'_{t+1}$$

where $Q_{t+1} = (C_t + \tau^2)\boldsymbol{\beta}_{t+1}\boldsymbol{\beta}_{t+1} + \sigma^2 I_2$ *and* $A_{t+1} = (C_t + \tau^2)Q_{t+1}^{-1}\boldsymbol{\beta}_{t+1}$.

5. Updating sufficient statistics for parameters: *The conditional posterior* $p(\theta|s_t)$ *is decomposed into*

$$\begin{aligned} p(\beta_i|\sigma^2, s_{t+1}) &\sim N(b_{i,t+1}, \sigma^2 B_{i,t+1}) \qquad i = 1, 2 \\ p(\sigma^2|s_{t+1}) &\sim IG(\nu_{0,t+1}/2, d_{0,t+1}/2) \\ p(\tau^2|s_{t+1}) &\sim IG(\nu_{1,t+1}/2, d_{1,t+1}/2) \\ p(p|s_{t+1}) &\sim Beta(p_{1,t+1}, p_{2,t+1}) \\ p(q|s_{t+1}) &\sim Beta(q_{1,t+1}, q_{2,t+1}) \end{aligned}$$

with $B_{i,t+1}^{-1} = B_{it}^{-1} + x_{t+1}^2 \mathbb{I}_{\lambda_{t+1}=i}$, $b_{i,t+1} = B_{i,t+1}\left(B_{it}^{-1} b_{it} + x_t y_{t2} \mathbb{I}_{\lambda_{t+1}=i}\right)$ *and* $\nu_{i,t+1} = \nu_{i,t} + 1$, *for* $i = 1, 2$, $d_{1,t+1} = d_{1t} + (x_{t+1} - x_t)^2$, $p_{1,t+1} = p_{1t} + \mathbb{I}_{\lambda_t=1,\lambda_{t+1}=1}$, $p_{2,t+1} = p_{2t} + \mathbb{I}_{\lambda_t=1,\lambda_{t+1}=2}$, $q_{1,t+1} = q_{1t} + \mathbb{I}_{\lambda_t=2,\lambda_{t+1}=2}$ $q_{2,t+1} = q_{2t} + \mathbb{I}_{\lambda_t=2,\lambda_{t+1}=1}$ *and* $d_{0,t+1} = d_{0t} + \sum_{j=1}^{2}\left[(y_{t+1,2} - b_{j,t+1}x_{t+1})\, y_{t+1,2} + b_{j,t+1}B_{j0}^{-1} + (y_{t+1,1} - x_{t+1})^2\right]\mathbb{I}_{\lambda_{t+1}=j}$,

Figure 1 and 2 illustrates the performance of the PL algorithm. The first panel of Figure 1 displays the true underlying λ *process along with filtered and smoothed estimates whereas the second panel presents the same information for the common factor. Whilst filtering doesn't uncover the true state due to sequential uncertainty the posterior bands for the smoothed posterior are very close to the truth. Figure 2 provides the sequential parameter learning plots. The plots clearly illustrate how learning is achieved in these models. After the regime switches the parameter posteriors learn and converge on the true values quickly.*

4 Nonlinear Filtering and Learning

We now extend our PL filter to a general class of non-linear state space models, namely conditional Gaussian dynamic model (CGDM). This class generalizes conditional dynamic linear models by allowing non-linear evolution equations. In this context we take advantage of most efficiency gains of PL as we are still able follow the re-sample/propagate logic and filter sufficient statistics for θ. Consider a conditional Gaussian state space model with non-linear evolution equation,

$$(y_{t+1}|x_{t+1},\lambda_{t+1},\theta) \sim N(F_{\lambda_{t+1}}x_{t+1}, V_{\lambda_{t+1}}) \tag{15}$$

$$(x_{t+1}|x_t,\lambda_{t+1},\theta) \sim N(G_{\lambda_{t+1}}h(x_t), W_{\lambda_{t+1}}) \tag{16}$$

where $h(\cdot)$ is a given non-linear function and, again, θ contains Fs, Gs, Vs and Ws. Due to the non-linearity in the evolution we are no longer able to work with state sufficient statistics s_t^x but we are still able to evaluate the predictive $p\left(y_{t+1}|x_t,\lambda_t,\theta\right)$. In general, take as the particle set: $\left\{\left(x_t,\theta,\lambda_t,s_t\right)^{(i)}, i=1,\ldots,N\right\}$. For discrete λ we can define the following algorithm:

Algorithm 3: CGDM

```
For $t = 0, \ldots, T-1$
For $i = 1, \ldots, N$

  *Step 1* (Re-sample): Generate an index $k^i \sim$ Multinomial$(w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N)})$ where

      $w_t^{(j)} \propto p(y_{t+1}|(x_t,\lambda_t,\theta)^{(j)})$        $j = 1, \ldots, N$

  *Step 2* (Propagate): States

      $\lambda_{t+1}^{(i)} \sim p(\lambda_{t+1}|(\lambda_t,\theta)^{(k^i)}, y_{t+1})$
      $x_{t+1}^{(i)} \sim p(x_{t+1}|(x_t,\theta)^{(k^i)}, \lambda_{t+1}^{(i)}, y_{t+1})$

  *Step 3* (Propagate): Parameter Sufficient Statistics (same as in Algorithm 1)

  *Step 4* (Propagate): Parameters (same as in Algorithm 1)
```

When λ is continuous we first propagate $\lambda_{t+1}^{(i)} \sim p(\lambda_{t+1}|(\lambda_t,\theta)^{(i)})$, for $i = 1,\dots,N$, then we re-sample the particle $(x_t,\lambda_{t+1},\theta,s_t)^{(i)}$ with the appropriate predictive distribution $p(y_{t+1}|(x_t,\lambda_{t+1},\theta)^{(i)})$ as in Algorithm 2. Finally it is straightforward to extend the backward smoothing strategy of Section 2.4 to obtain samples from $p(x^T|y^T)$.

Example 3. Heavy-tailed nonlinear state space model. *Consider the following non-Gaussian and nonlinear state space model*

$$(y_{t+1}|x_{t+1},\lambda_{t+1},\theta) \sim N(x_{t+1},\lambda_{t+1}\sigma^2)$$
$$(x_{t+1}|x_t,\lambda_{t+1},\theta) \sim N(\beta h(x_t),\tau^2)$$

where $\theta = (\beta,\sigma^2,\tau^2)$, $h(x_t) = x_t/(1+x_t^2)$ and $\lambda_{t+1} \sim IG(\nu/2,\nu/2)$, for known ν. Therefore, the distribution of $(y_{t+1}|x_{t+1},\theta) \sim t_\nu(x_{t+1},\sigma^2)$, a t-Student with ν degrees of freedom. Filtering strategies for this model without parameter learning are described in DeJong et al. *(2007).*

The PL algorithm works as follows. First, start with particle set $\{(x_t,\theta,\lambda_{t+1},s_t)^{(i)}, i = 1,\dots,N\}$ for $p(x_t,\theta,\lambda_{t+1},s_t|y^t)$. Then, at any given time $t = 0,\dots,T-1$ and $i = 1,\dots,N$, we first draw an index $k^i \sim$ Multinomial$(w_t^{(1)},\dots,w_t^{(N)})$, with weights $w_t^{(j)} \propto p(y_{t+1}|(x_t,\lambda_{t+1},\theta)^{(j)})$, $j = 1,\dots,N$, and $p(y_{t+1}|x_t,\lambda_{t+1},\theta) = f_N(y_{t+1};\beta h(x_t),\lambda_{t+1}\sigma^2+\tau^2)$. Then, we draw a new state $x_{t+1}^{(i)} \sim p(x_{t+1}|(\lambda_{t+1},x_t,\theta)^{(k^i)},y_{t+1}) \equiv f_N(x_{t+1};\mu_{t+1}^{(i)},V_{t+1}^{(i)})$, where $\mu_{t+1} = V_{t+1}(\lambda_{t+1}^{-1}\sigma^{-2}y_{t+1}+\tau^{-2}\beta h(x_t))$ and $V_{t+1}^{-1} = \lambda_{t+1}^{-1}\sigma^{-2}+\tau^{-2}$.

Finally, similar to example 1, posterior parameter learning for $\theta = (\beta,\sigma^2,\tau^2)$ follow directly from conditionally normal-inverse gamma update. Figure 3 illustrates the above PL algorithm in a simulated example where $\beta = 0.9$, $\sigma^2 = 0.04$ and $\tau^2 = 0.01$. The algorithm uncovers the true parameters very efficiently in a sequential fashion. In Section 5 we revisit this example to compare the performances of PL, MCMC (Carlin, Polson and Stoffer, 1992) and the benchmark particle filter with parameter learning (Liu and West, 2001).

5 Comparing Particle Learning to Existing Methods

PL combined with the backward smoothing algorithm (as in Section 2.4) is an alternative to MCMC methods for state space models. In general MCMC methods (see Gamerman and Lopes, 2006) use Markov chains designed to explore the posterior distribution $p\left(x^T, \theta | y^T\right)$ of states and parameters conditional on all the information available, $y^T = (y_1, \ldots, y_T)$. For example, an MCMC strategy would have to iterate through

$$p(\theta|x^T, y^T) \text{ and } p(x^T|\theta, y^T).$$

However, MCMC relies on the convergence of very high-dimensional Markov chains. In the purely conditional Gaussian linear models or when states are discrete, $p(x^T|\theta, y^T)$ can be sampled in block using FFBS. Even in these ideal cases, achieving convergence is far from a easy task and the computational complexity is enormous as at each iteration one would have to filter forward and backward sample for the full state vector x^T. The particle learning algorithm presented here has two advantages: (i) it requires only one forward/backward pass through the data for all N particles and (ii) the approximation accuracy does not rely on convergence results that are virtually impossible to access in practice (see Papaspiliopoulos and Roberts, 2008).

In the presence of non-linearities, MCMC methods will suffer even further as no FFBS scheme is available for the full state vector x^T. One would have to resort to univariate updates of $p(x_t|x_{(-t)}, \theta, y^T)$ as in Carlin, Polson and Stoffer (1992). It is well known that these methods generate very "sticky" Markov chains, increasing computational complexity and slowing down convergence. The computational complexity of our approach is only $O(NT)$ where N is the number of particles. PL is also attractive given the simple nature of its implementation (especially if compared to more novel hybrid methods). We now present three experiments comparing the performance of PL against MCMC alternatives.

Example 4. State Sufficient Statistics. *In this first simulation exercise we revisit the local level model of example 1 in order to compare PL to its version that takes advantage* *of state sufficient statistics, i.e. by marginalizing the latent state. The main goal is to study the Monte Carlo error of the two filters. We simulated a time series of length* $T = 100$ *with* $\sigma^2 = 1$, $\tau^2 = 0.1$ *and* $x_0 = 0_p$. *The prior distributions are* $\sigma^2 \sim IG(5,4)$, $\tau^2 \sim IG(5,0.4)$ *and* $x_0 \sim N(0,10)$. *We run two filters: one with sequential learning for* x_t, σ^2 *and* τ^2 *(we call it simply* PL*), and the other with sequential learning for state sufficient statistics,* σ^2 *and* τ^2 *(we call it* PLsuff*). In both cases, the particle filters are based on either one long particle set of size* $N = 100000$ *(we can it* Long*) or 20 short particle sets of size* $N = 5000$ *(we call it* Short*). The results are in Figures 4 to 6. Figure 4 shows that the differences between* PL *and* PLsuff *dissipate for fairly large* N. *However, when* N *is small* PLsuff *has smaller Monte Carlo error and is less biased than* PL, *particularly when estimating* σ^2 *and* τ^2 *(see Figure 5). Similar findings appear in Figure 6 where the mean square errors of the quantiles from the 20* Short *runs are compared to those from the* Long PLsuff *run.*

Example 5. Resample-propagate or propagate-resample? *In this second simulation exercise we revisit the local level model once again in order to compare PL to three other particle filters. They are the Bayesian bootstrap filter (BBF) of Gordon* et. al. *(1993), its fully adapted version (FABBF), and the auxiliary particle filter (APF) of Pitt and Shephard (1999). BBF and FABBF are propagate-resample filters, while PL and APF are resample-propagate filters. The main goal is to study the Monte Carlo error of the four filters.*

We start with the pure case scenario, i.e. with fixed parameters. We simulated 20 time series of length $T = 100$ *from the local level model with parameters* $\tau^2 = 0.013$, $\sigma^2 = 0.13$ *and* $x_0 = 0$. *Therefore, the signal to noise ratio* σ_x/σ *equals* 0.32. *Other combinations were also tried and similar results found. The prior distribution of the initial state* x_0 *was set at* $N(0,10)$. *For each time series, we run 20 times each on of the four filters, all based on* $N = 1000$ *particles. We use five quantiles to compare the various filters. Let* q_α^t *be such that* $Pr(x_t < q_t^\alpha | y^t) = \alpha$, *for* $\alpha = (0.05, 0.25, 0.5, 0.75, 0.95)$. *Then, the mean square* *error (MSE) for filter f, at time t and quantile α is*

$$MSE_{t,f}^{\alpha} = \frac{1}{400}\sum_{d=1}^{20}\sum_{r=1}^{20}(q_{t,d}^{\alpha} - \hat{q}_{t,d,f,r}^{\alpha})^2$$

where d and r index the data set and the particle filter run, respectively. We compare PL, APF and FABBF via logarithm relative MSE (LRMSE), relative to the benchmark BBF. Results are summarized in Figure 7. PL is uniformly better than all three alternatives. Notice that the only algorithmic different between PL and FABBF is that PL reverses the propagate-resample steps.

We now move to the parameter learning scenario, where σ^2 is still kept fixed but learning of τ^2 is performed. Three time series of length $T = 1000$ were simulated from the local level model with $x_0 = 0$ and (σ^2, τ^2) in $\{(0.1, 0.01), (0.01, 0.01), (0.01, 0.1)\}$. The independent prior distributions for x_0 and τ^2 are $x_0 \sim N(0, 1)$ and $\tau^2 \sim IG(10, 9\tau_0^2)$, where τ_0^2 is the true value of τ^2 for a given time series. In all filters τ^2 is sampled offline from $p(\tau^2|S_t)$ where S_t is the vector of conditional sufficient statistics. We run the filters 100 times, all with the same seed within run, for each one of the three simulated data sets. Finally, the number of particles was set at $N = 5000$, with similar results found for smaller N, ranging from 250 to 2000 particles. Mean absolute errors (MAE) over the 100 replications are constructed by comparing quantiles of the true sequential distributions $p(x_t|y^t)$ and $p(\tau^2|y^t)$ to quantiles of the estimated sequential distributions $p^N(x_t|y^t)$ and $p^N(\tau^2|y^t)$. More specifically, for time t, a in $\{x, \tau^2\}$, α in $\{0.01, 0.50, 0.99\}$, true quantiles $q_{t,a}^{\alpha}$ and PL quantiles $\hat{q}_{t,a,r}^{\alpha}$,

$$MAE_{t,a}^{\alpha} = \frac{1}{100}\sum_{r=1}^{100}|q_{t,a}^{\alpha} - \hat{q}_{t,a,r}^{\alpha}|.$$

Across different quantiles and combinations of error variances, PL is at least as good as FABBF and in many cases significantly better than BBF.

Example 6. PL versus FFBS. *We revisit the first order dynamic linear model introduced in example 1 in order to compare our PL smoother and the forward-filtering, backward-*

sampling (FFBS) smoother. Under the pure filter scenario, i.e. assuming θ is known, Figure 9 compares the true smoothed distributions $p(x_t|y^T)$ to approximations based on PL and on FFBS. Now, when parameter learning is introduced, PL performance is comparable to that of the FFBS when approximating $p(\sigma^2, \tau^2|y^T)$, as shown in Figure 10. We argue that, based on these empirical findings, PL and FFBS comparable results, as far as smoothed distributions is of interest. We now turn to the issue of computational cost, measured here by the running time in seconds of both schemes. Data was simulated based on $(\sigma^2, \tau^2, x_0) = (1.0, 0.5, 0.0)$. The prior distribution of x_0 is $N(0, 100)$, while σ^2 and τ^2 are kept fixed throughout this exercise. PL was based on N particles and FFBS based on $2N$ iterations, with the first M discarded. Table 1 summarizes the results. For fixed N, the (computational) costs of both PL and FFBS increase linearly with T, with FFBS twice as fast as PL. For fixed T, the cost of FFBS increases linearly with N, while the cost of PL increases exponentially with N. These findings were anticipated in Section 2.4.

Table 1: Computing time of PL and FFBS for smoothing.

$N = 500$			$T = 100$		
T	PL	FFBS	N	PL	FFBS
200	18.8	9.1	500	9.3	4.7
500	47.7	23.4	1000	32.8	9.6
1000	93.9	46.1	2000	127.7	21.7

The previous exercise provides evidence of the ability of PL to effectively solve the problem of learning and smoothing. We now compare the performance of PL against the most commonly used filter for situations where θ is unknown, i.e, the filter proposed by Liu and West (2001) (LW). This is a filtering strategy that adapts the auxiliary particle filter of Pitt and Shephard (1999) and solves the learning of θ by incorporating a kernel density approximation for $p(\theta|y^t)$. We also use this section to discuss the robustness of PL when the sample size T grows. In the following two experiments, our focus is to assess the performance of each filter in properly estimating the sequence of $p(\theta|y^t)$ for all t.

Example 7. PL versus LW. *We revisit a variation of the first order dynamic linear model introduced in example 1 in order to compare our PL algorithm to Liu and West's filter for a situation where θ is unknown. More precisely, we simulate data from*

$$\begin{aligned}(y_{t+1}|x_{t+1},\beta) &\sim N(x_t,\sigma^2)\\ (x_{t+1}|x_t,\beta) &\sim N(\beta x_t,\tau^2)\end{aligned}$$

for $t = 1,\ldots,T = 100$, $\sigma^2 = 1$, $x_1 = 0.0$ and three possible values for the parameter $\tau^2 = (0.01, 0.25, 1.00)$. So, the signal to noise ratio $\tau/\sigma = 0.1, 0.5, 1.0$. Only β and x_t are sequentially estimated and their independent prior distributions are $N(1.0, 1.0)$ and $N(0.0, 1.0)$, respectively. The particle set has length $N = 2000$ and both filters were run 50 times to study the size of the Monte Carlo error. The shrinkage/smoothing parameter δ of Liu and West's filter was set at $\delta = 0.95$, but fairly similar results were found for δ ranging from 0.8 to 0.99. Our findings, summarized in Figure 11, favor PL over LW uniformly across all scenarios. The discrepancy is higher when σ_x/σ is small, which is usually the case in state space applications.

6 Final Remarks

In this paper we provide particle learning tools (PL) for a large class of state space models. Our methodology incorporates sequential parameter learning, state filtering and smoothing. This provides an alternative to the popular FFBS/MCMC (Carter and Kohn, 1994) approach for conditional dynamic linear models (DLMs) and also to MCMC approaches to nonlinear non-Gaussian models. It is also a generalization of the mixture Kalman filter (MKF) approach of Liu and Chen (2000) that includes parameter learning and smoothing.

The key assumption is the existence of a conditional sufficient statistic structure for the parameters which is commonly available in many commonly used models.

We provide extensive simulation evidence and theoretical Monte Carlo convergence bounds to address the efficiency of PL versus standard methods. Computational time and accuracy are used to assess the performance. Our approach compares very favorably with these existing strategies and is robust to particle degeneracies as the sample size grows. Finally PL has the additional advantage of being an intuitive and easy-to-implement computational scheme and should, therefore, become a default choice for posterior inference in this very general class of models.

References

Bengtsson, T., Bickel, P. and Li, B. (2008) Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. *Probability and Statistics: Essays in Honor of David A. Freedman*. Institute of Mathematical Statistics Collections, 2, 316-334.

Briers, M., Doucet, A. and Maskell, S. (2009) Smoothing Algorithms for State-Space Models. *IEEE Transactions on Signal Processing* (to appear).

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

Carter, C. and Kohn, R. (1994) On Gibbs sampling for state space models. *Biometrika*, 82, 339-350.

Carvalho, C.M. and Lopes, H.F. (2007) Simulation-based sequential analysis of Markov switching stochastic volatility models. *Computational Statistics and Data Analysis*, 51, 4526-4542.

Doucet, A., de Freitas, J. and Gordon, N. (Eds) (2001) *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York.

Fearnhead, P. (2002) Markov chain Monte Carlo, sufficient statistics, and particle filters. *Journal of Computational and Graphical Statistics*, 11, 848-862.

Fearnhead, P. and Clifford, P. (2003) On-line inference for hidden Markov models via particle filters. *Journal of the Royal Statistical Society, Series B*, 65, 887-899.

Fearnhead, P., Wyncoll, D. and Tawn, J. (2008) A sequential smoothing algorithm with linear computational cost. Technical Report. Department of Mathematics and Statistics, Lancaster University.

Frühwirth-Schnatter, S. (1994) Applied state space modelling of non-Gaussian time series using integration-based Kalman filtering. *Statistics and Computing*, 4, 259-269.

Gamerman, D. and Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman and Hall / CRC.

Godsill, S.J., Doucet, A. and West, M. (2004) Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99, 156-168.

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

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

Johannes, M., Polson, N.G. (2007) Exact Particle Filtering and Learning. *Working Paper*. The University of Chicago Booth School of Business.

Kalman, R.E. (1960) A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, 82, 35-45.

Kitagawa, G. (1987) Non-Gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association*, 82, 1032-1041.

Liu, J. and Chen, R. (2000). Mixture Kalman filters. *Journal of Royal Statistical Society, Series B*, 62, 493-508.

Liu, J. and West, M. (2001) Combined parameters and state estimation in simulation-based filtering. In *Sequential Monte Carlo Methods in Practice* (Eds. A. Doucet, N. de Freitas and N. Gordon). Springer-Verlag, New York.

Papaspiliopoulos, O., Roberts, G. (2008) Stability of the Gibbs sampler for Bayesian hierarchical models. *Annals of Statistics*, 36, 95-117.

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

Polson, N. G. (1991) Comment on “Practical Markov Chain Monte Carlo” by C. Geyer. *Statistical Science*, 7, 490-491.

Polson, N.G., Stroud, J. and Müller, P. (2008) Practical filtering with sequential parameter learning. *Journal of the Royal Statistical Society, Series B*, 70, 413-428.

Shumway, R.H. and Stoffer, D.S. (2006). *Time Series Analysis and its Applications* (2nd Edition). Springer-Verlag.

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

West, M. (1986) Bayesian model monitoring. *Journal of the Royal Statistical Society, Series B*, 48, 70-78.

West, M. and Harrison, J. (1997) *Bayesian forecasting and dynamic models (2nd Edition)*, Springer-Verlag, New York.

Scott, S. (2002) Bayesian methods for hidden Markov Models: recursive computing in the 21st century. *Journal of the American Statistical Association*, 97, 335-351.

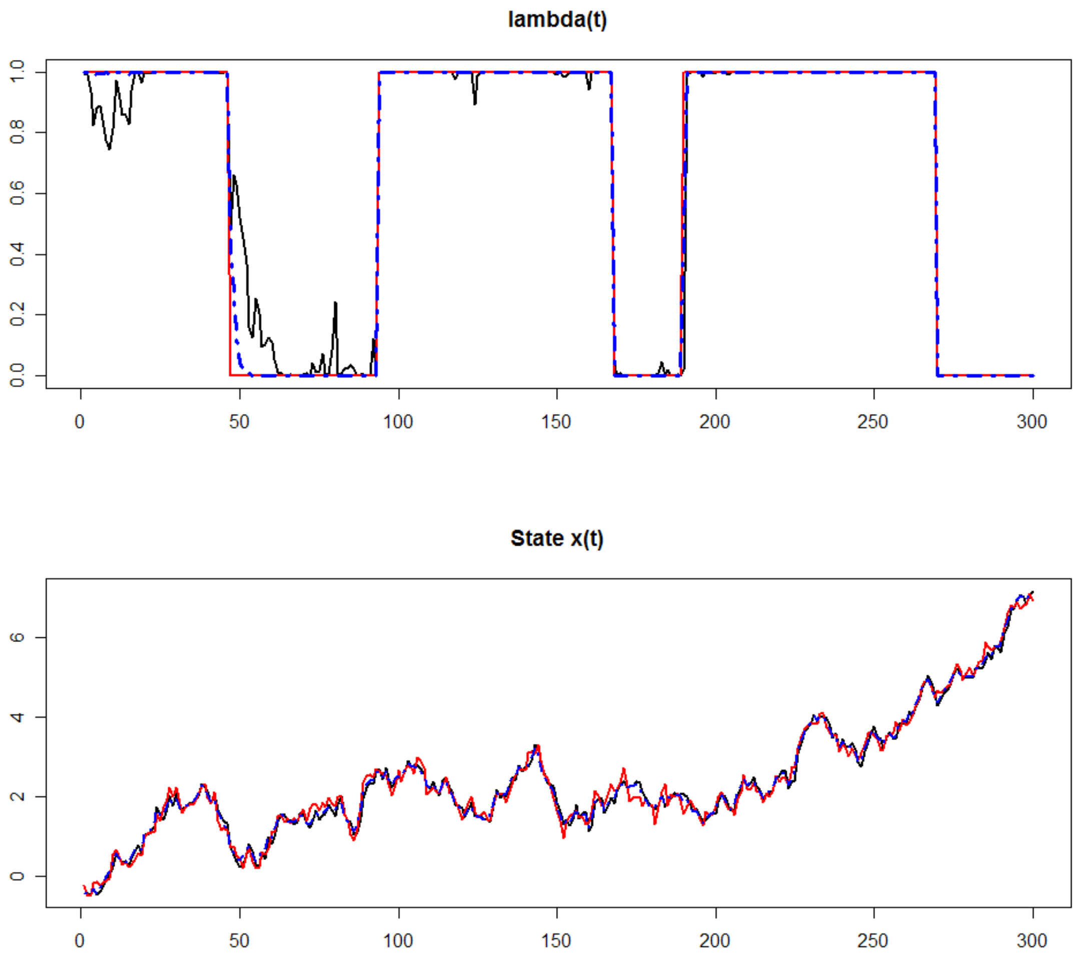

Figure 1: *Dynamic factor model (state learning).* Top panel: True value of λ_t (red line), $Pr(\lambda_t = 1|y^t)$ (black line) and $Pr(\lambda_t = 1|y^T)$ (blue line) Bottom panel: True value of x_t (red line), $E(x_t|y^t)$ (black line) and $E(x_t|y^T)$ (blue line).

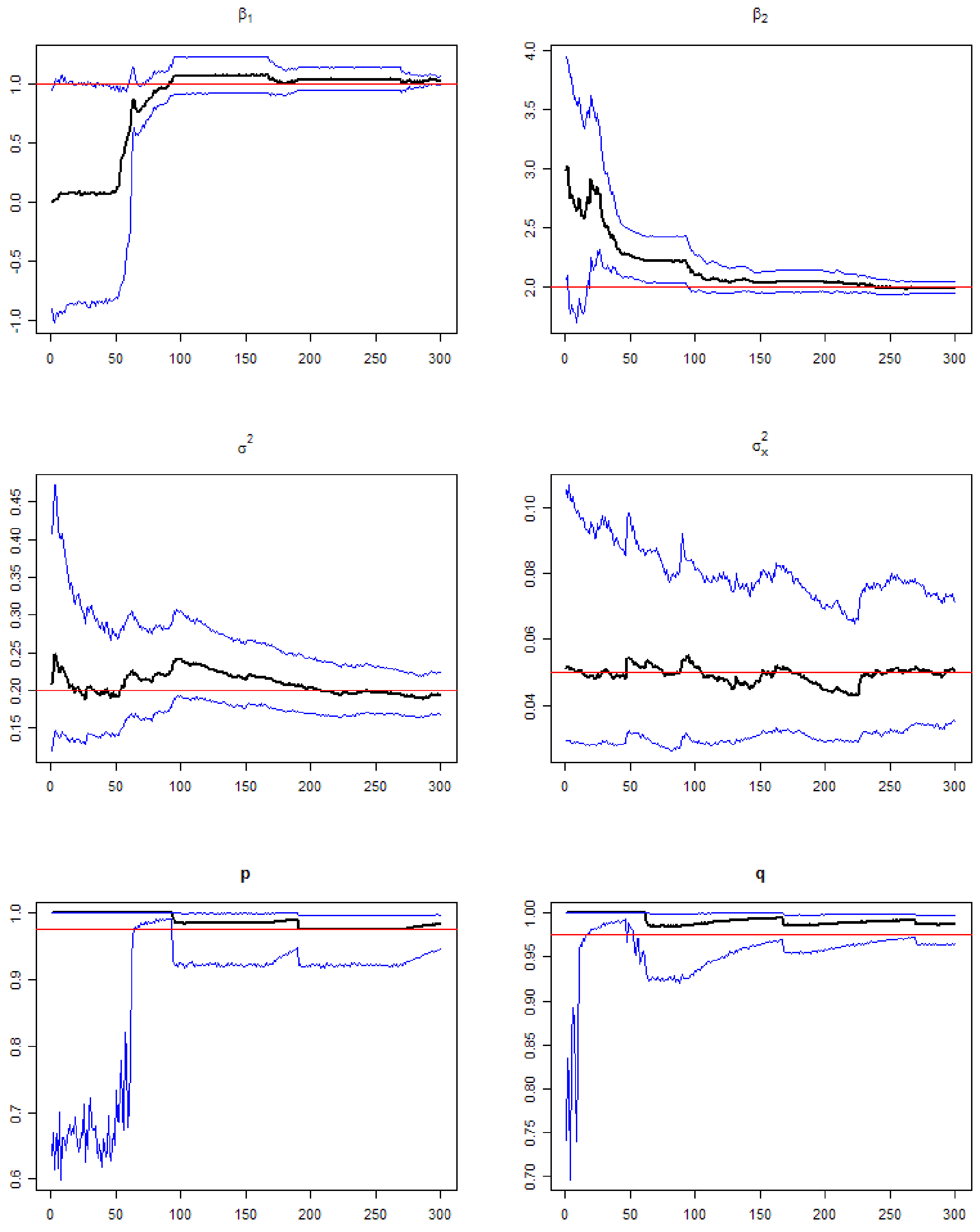

Figure 2: *Dynamic factor model (parameter learning).* Sequential posterior median (black line) and posterior 95% credibility intervals (blue lines) for model parameters β_1, β_2, σ^2, τ^2, p and q. True values are the red lines.

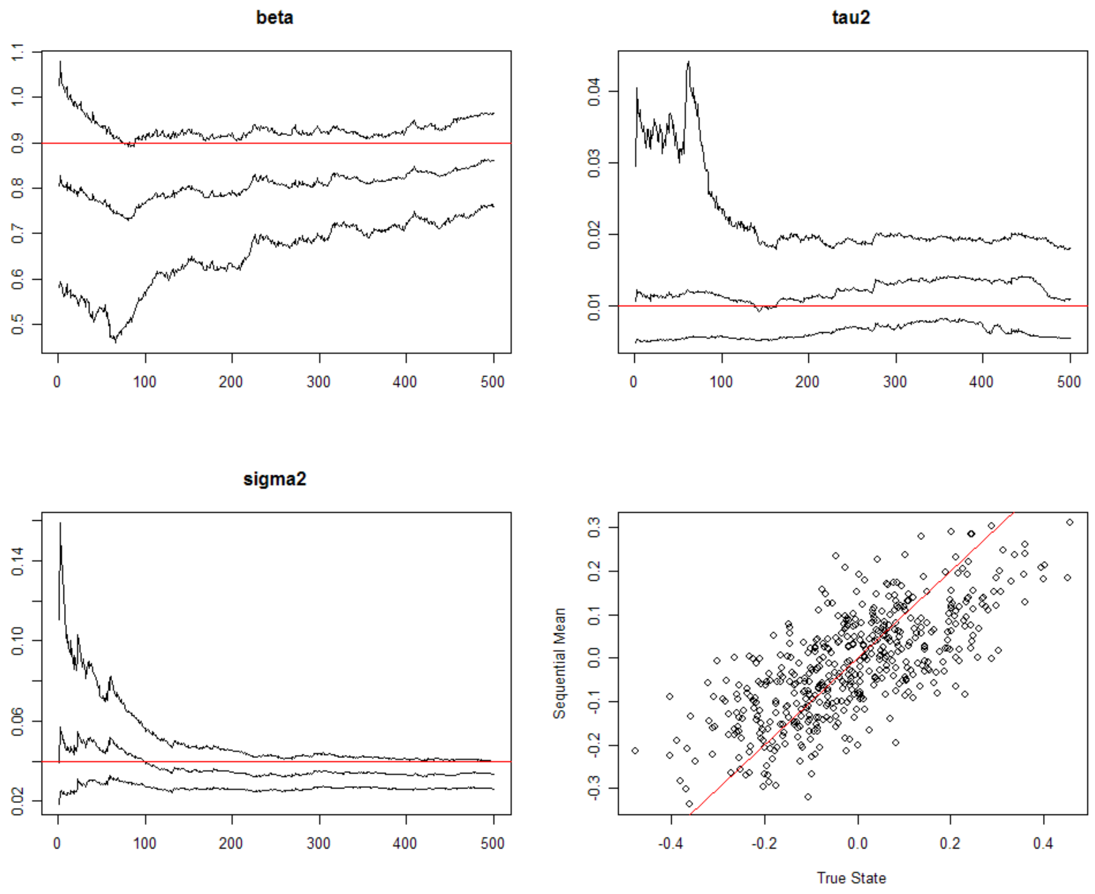

Figure 3: *Heavy-tailed non-Gaussian, nonlinear model.* Sequential posterior median and posterior 95% credibility intervals (black lines) for model parameters β, σ^2 and τ^2. True values are the red lines. The bottom right panel the true value of x_t against $E(x_t|y^t)$.

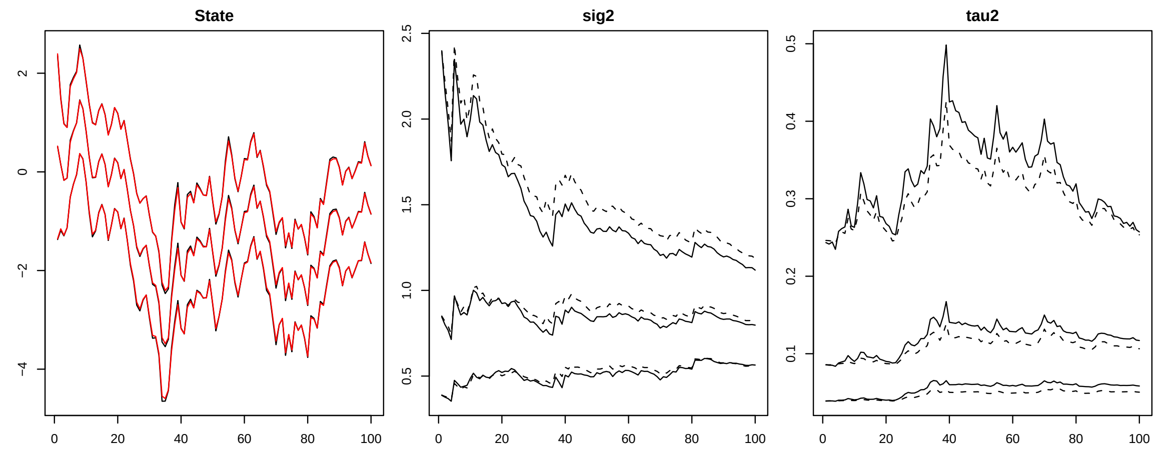

Figure 4: *PL and PL with state sufficient statistics (Long runs).* Left panel - $p(x_t|y^t)$ - PL (black), PLsuff (red); Middle panel - $p(\sigma^2|y^t)$ - PL (solid line), PLsuff (dotted line); Right panel - $p(\tau^2|y^t)$ - PL (solid line), PLsuff (dotted line).

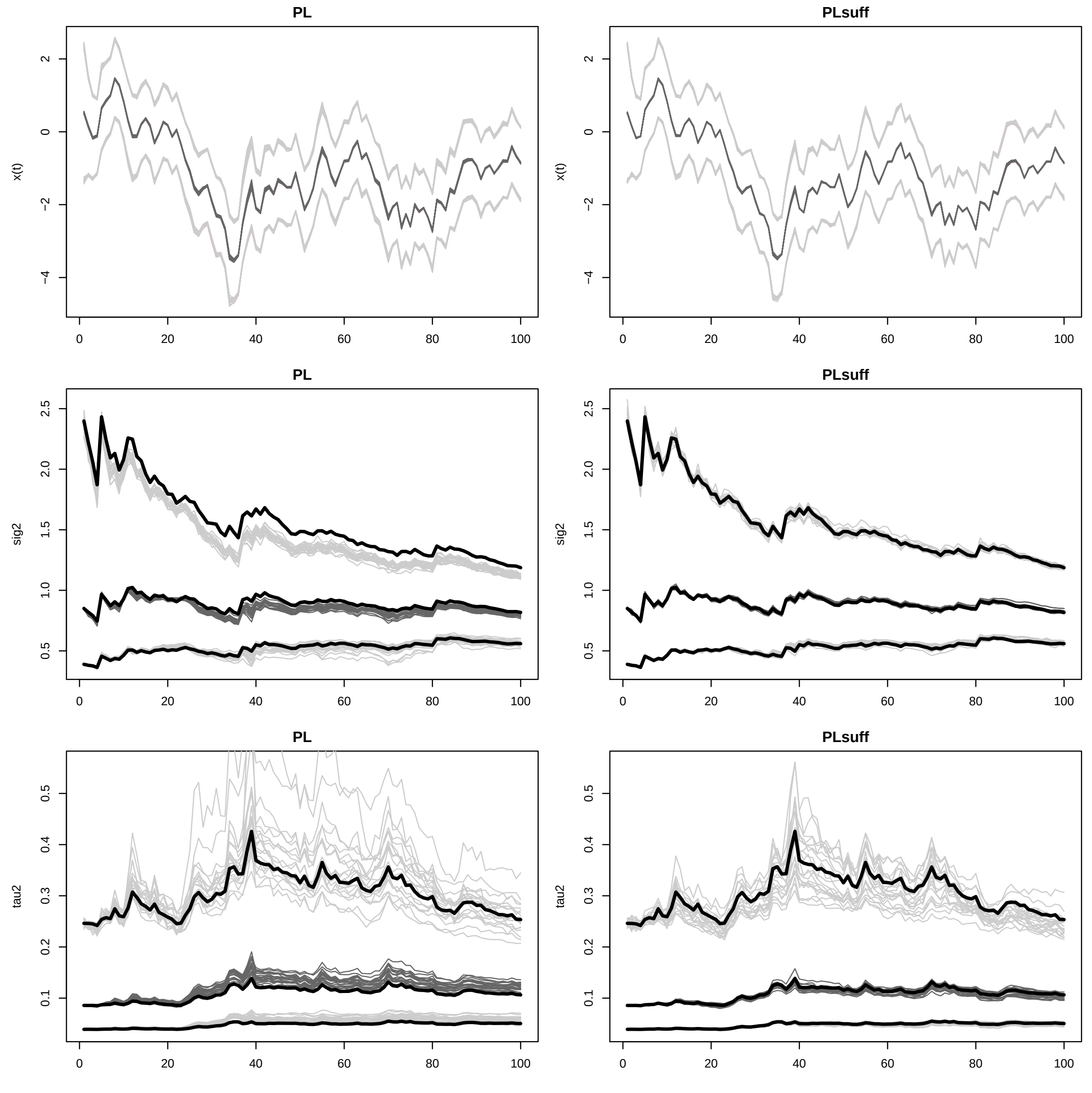

Figure 5: *PL and PL with state sufficient statistics (20 short runs).* PL runs (left columns) and PLsuff runs (right columns). One long run (black) and 20 short runs (gray); $p(x_t|y^t)$ (top row), $p(\sigma^2|y^t)$ (middle row) and $p(\tau^2|y^t)$ (bottom row).

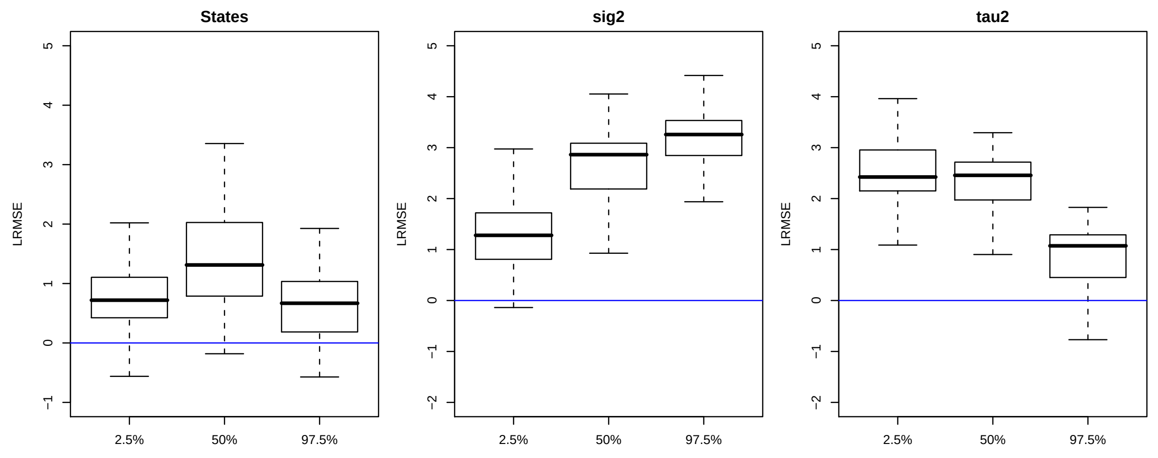

Figure 6: *PL and PL with state sufficient statistics (mean square errors).* Logarithm of the relative mean square error for three quantiles of $p^N(x_t|y^t)$, $p^N(\sigma^2|y^t)$ and $p^N(\tau^2|y^t)$, averaged across the 20 $N = 5000$ runs. PL relative to PLsuff.

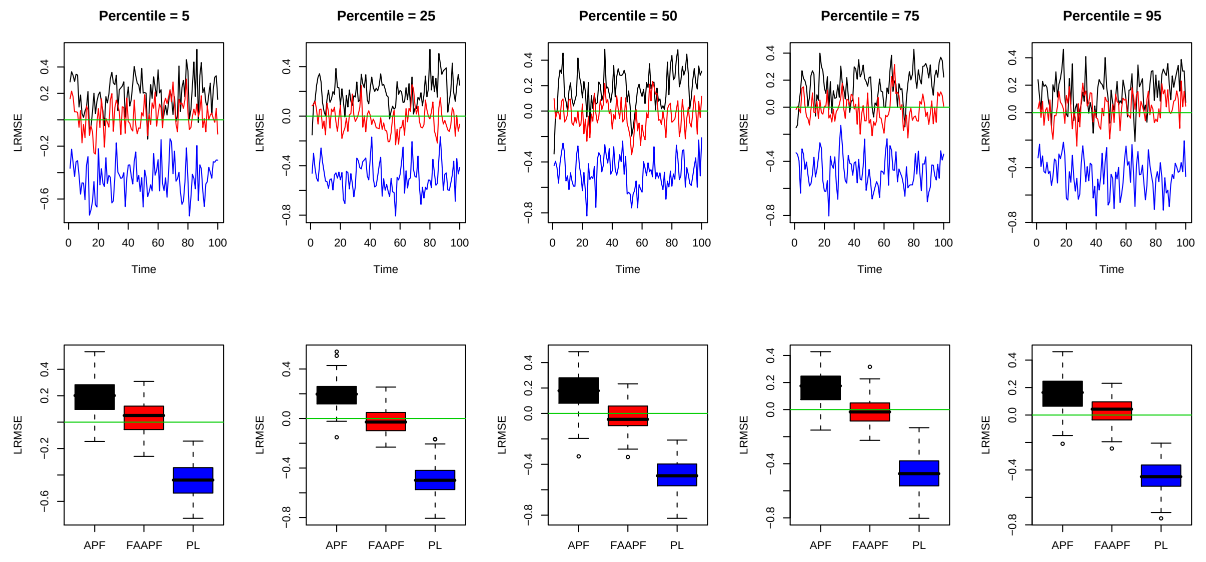

Figure 7: *PL, APF and FABBF pure filter.* Logarithm of the relative mean square error for five quantiles of $p^N(x_t|y^t)$. Boxplots on the second row are based on the time series plots on the first row.

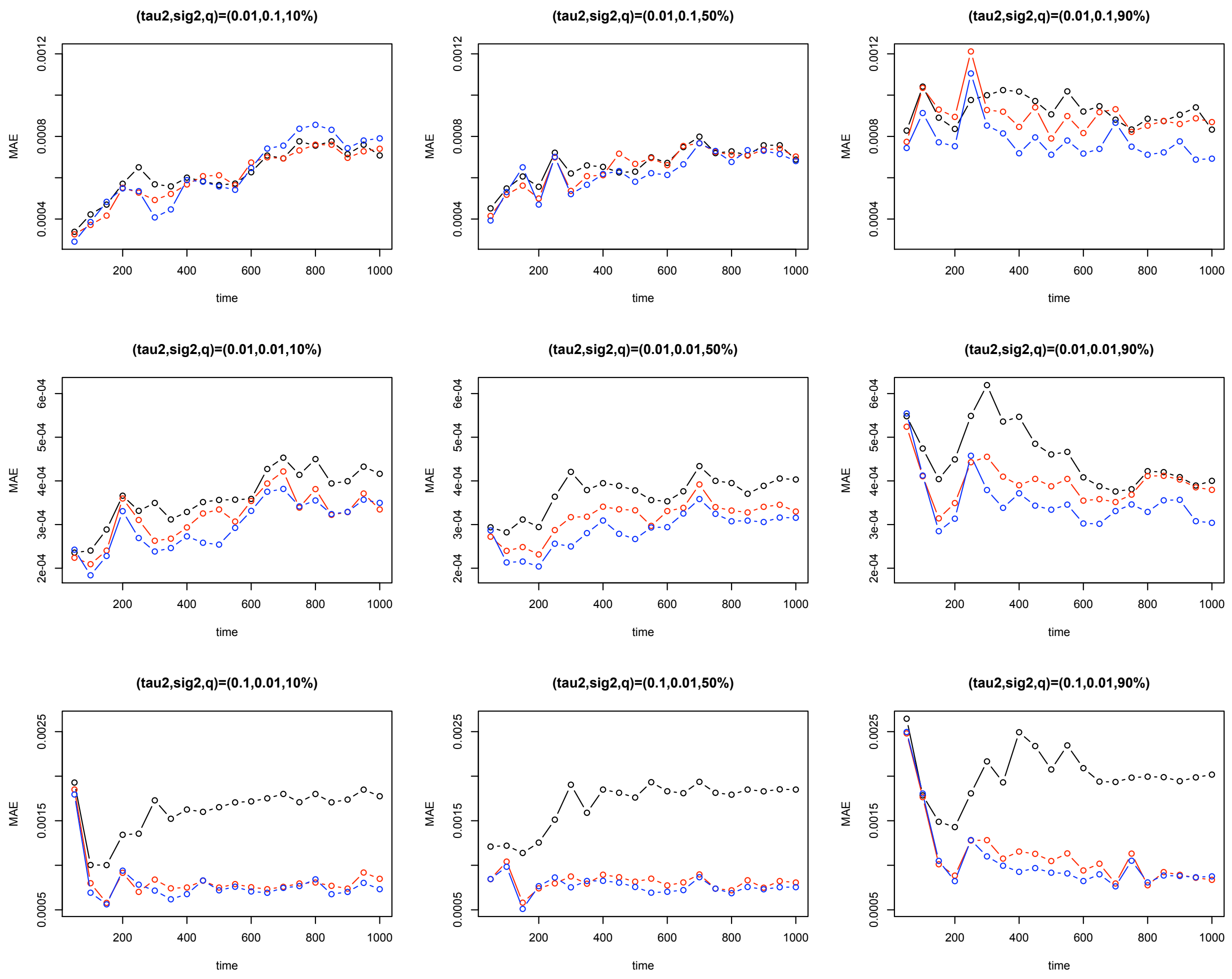

Figure 8: *PL, BBF and FABBF learning* τ^2. Mean absolute errors. PL (blue), FABBF (red) and BBF (black).

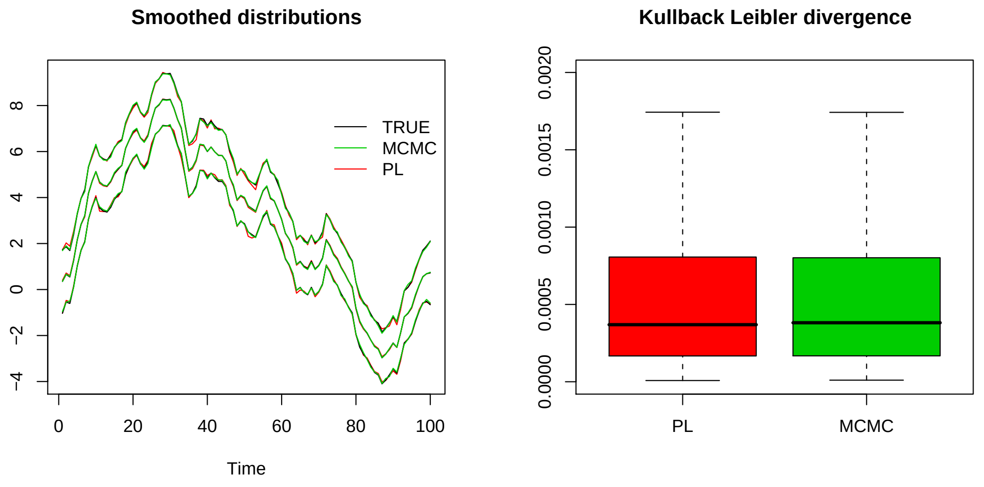

Figure 9: *PL and FFBS (smoothed distributions).* $T = 100$ simulated from a local level model with $\sigma^2 = 1$, $\tau^2 = 0.5$, $x_0 = 0$ and $x_0 \sim N(0, 100)$. PL is based on $N = 1000$ particles, while FFBS is based on $2N$ draws with the first N discarded.

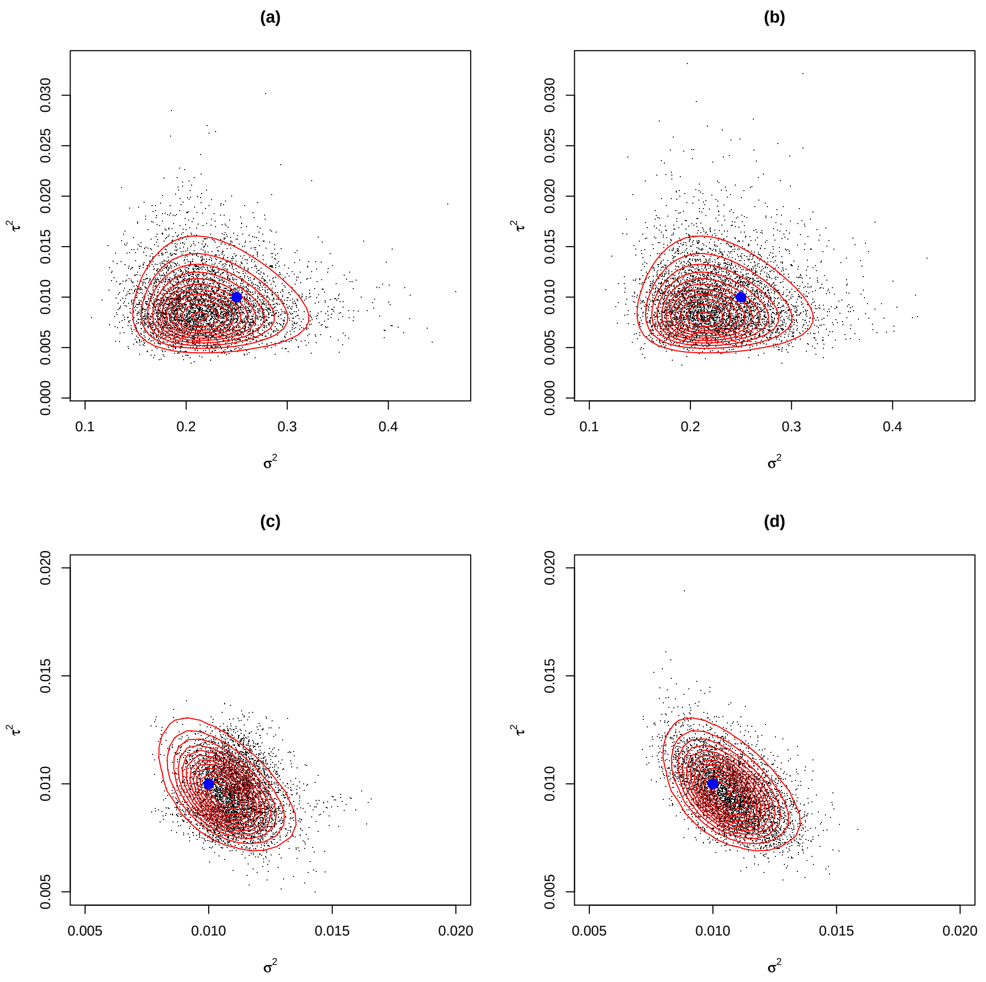

Figure 10: *PL and FFBS (parameter learning).* Contour plots for the true posterior $p(\sigma^2, \tau^2|y^T)$ (red contours) and posterior draws form PL, panels (a) and (c), and FFBS, panels (b) and (d). The blue dots represent the true value of the pair (σ^2, τ^2). The sample size is $T = 50$ (top row) and $T = 500$ (bottom row).

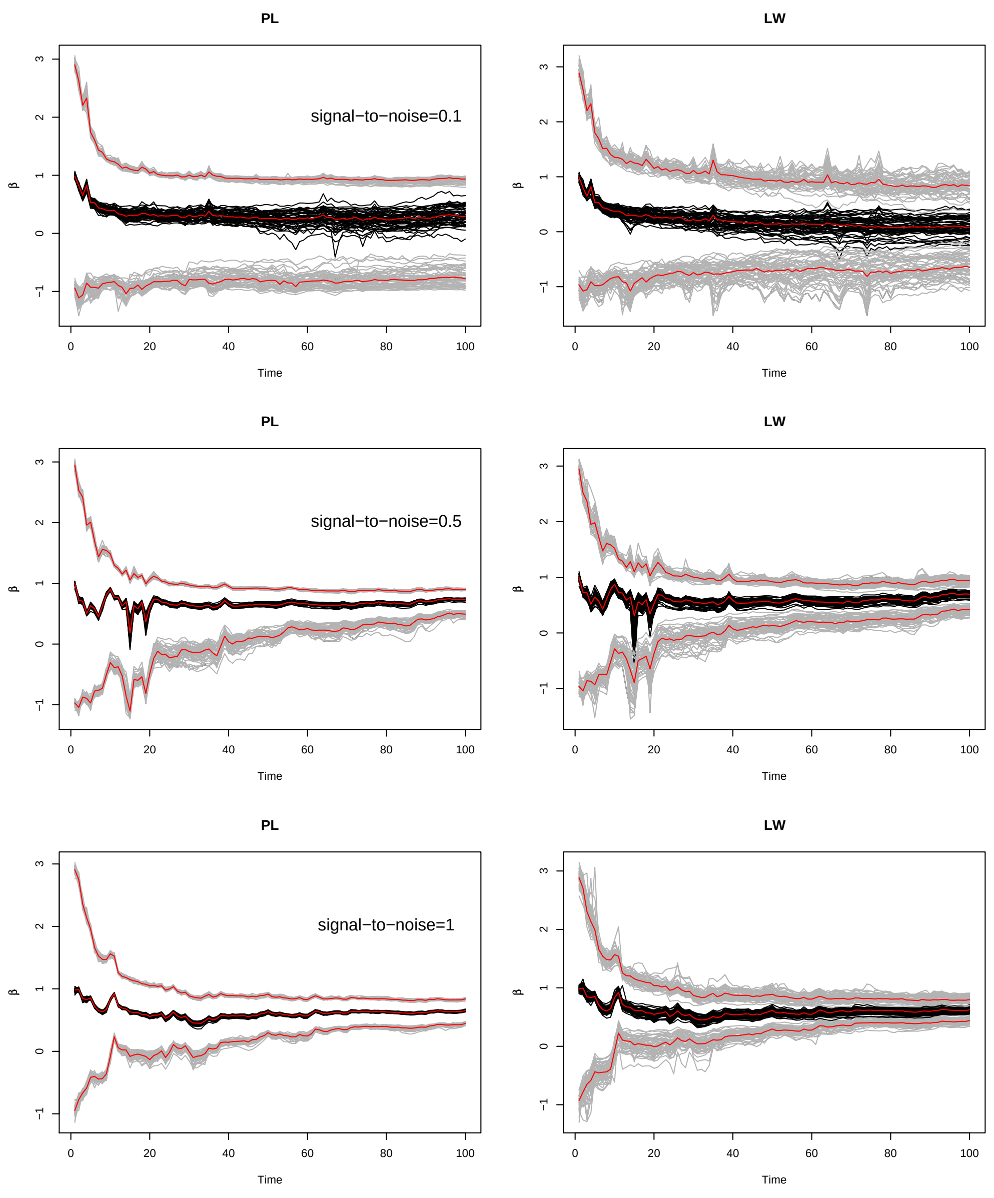

Figure 11: *PL and LW (parameter learning).* Posterior mean and 95% credibility interval from $p(\beta|y^t)$. Medians across the 50 runs appear in red. $N = 2000$ particles. `signal-to-noise` stands for σ_x/σ. In all cases, $\sigma = 1$.