# Particle Filtering and Parameter Learning

Michael Johannes and Nicholas Polson*

First draft: April 2006
This draft: October 2008

**Abstract**

This paper provides a new approach for sequentially learning parameters and states in a wide class of state space models using particle filters. Our approach generates direct i.i.d. samples from a particle approximation to the joint posterior distribution of both parameters and latent states, avoiding the use of and the degeneracies inherent in sequential importance sampling. We illustrate the efficiency of our approach by sequentially learning parameters and filtering states in two models: a log-stochastic volatility model and robust version of the Kalman filter model with t-errors in both the observation and state equation. In both cases, we show using simulated data that our approach efficiently learns the parameters and states sequentially, generating higher effective sample sizes than existing algorithms. We use the approach for two real data examples, sequentially learning in a stochastic volatility model of Nasdaq stock returns and about predictable components in a model of core inflation.

*Johannes is at the Graduate School of Business, Columbia University, 3022 Broadway, NY, NY, 10027, mj335@columbia.edu. Polson is at the Graduate School of Business, University of Chicago, 5807 S. Woodlawn, Chicago IL 60637, ngp@gsb.uchicago.edu. We thank Seung Yae for extraordinary research assistance and seminar participants at University of Chicago, SAMSI 2008, the 2007 Summer Meetings of the Econometric Society, and the 2005 ICMS meeting on Parameter Estimation and Continuous Time Models.

# 1 Introduction

Sequential parameter learning is a central problem in econometrics. Most of the learning literature focuses on estimating latent states, assuming known static parameters. The Kalman filter and particle filters are classic examples. Known static parameters may be a reasonable assumption for many statistics and engineering settings, where parameters are often physical constants or are easy to estimate using laboratory data, but in economics and finance settings it is more realistic to assume that neither the state nor the parameters are known. The goal of this paper is to provide a new, computationally simple and accurate recursive method for sequentially learning unobserved state variables, $x_t$, and static parameters, $\theta$, given observed data $y^t = (y_1, ..., y_t)$.

Sequential methods are extremely important for practical empirical research, although they are rarely used in academic settings. Sequential methods provide a mechanism for learning about parameters, state variables, models, and hypothesis as new data arrives. This can be contrasted with the typical econometric research approach that learns about these quantities using a single, typically very long time series. Sequential methods provide the same information, at the end of the sample, but also provide the dynamics estimates in the interim, documenting how investors or policy makers would learn in real time.

On the theoretical side, our main contribution is a recursive algorithm that directly (or perfectly or exactly) samples from an $N-$particle approximation, $p^N(\theta, s_t, x_t|y^t)$, to the joint posterior distribution, $p(\theta, s_t, x_t|y^t)$, where $s_t$ is a vector of sufficient statistics. This can be contrasted with indirect or approximate sampling methods such as importance sampling. The approach applies to the entire class of partial non-Gaussian or conditionally Gaussian state space models, a widely used class (see Carlin, Polson, and Stoffer (1992), Frühwirth-Schnatter (1994), Carter and Kohn (1994, 1996) and Shephard (1994)). We demonstrate that the algorithm is more efficient than previous attempts.

Sampling directly from the particle approximation is important because particle filters generally suffer from two sources of errors. The first source, inherent in all particle filters, is the error generated by approximating the target distribution by a discrete 'particle' distribution. This can be mitigated by choosing a large number of particles, but disappears only in the limit. The second source is generated by the common use of importance sampling, which provides an *approximate* sample from the particle approximation. Our approach eliminates this latter source of error by direct or exact sampling. In the context of state filtering, Pitt and Shephard (1999) provide algorithms that directly sample from the par-

ticle approximation and document drastic improvements. Our approach can be viewed as an extension of their approach incorporating parameter learning.

To do this, we track recursively defined parameter sufficient statistics, $s_t$, instead of the entire state history, $x^t$, following Storvik (2002) and Fearnhead (2002), who previously used particle filters for parameter estimation. By targeting $p(\theta, s_t, x_t|y^t)$ instead of $p(\theta, x^t|y^t)$ (as is done in, for example, the resample-move algorithm of Gilks and Berzuini (2001)), the dimensionality of the target distribution does not increase over time. This reduces the computational storage demands, but also mitigates the curse of dimensionality problem inherent in high-dimensional sampling problems, as argued by Chopin (2004) and Klaas, et al. (2005). For pure state filtering, Klaas et al. (2005) document the reductions in Monte Carlo error generated by focussing on $p(x_t|\theta, y^t)$ instead of $p(x^t|\theta, y^t)$. In our case, the computational complexity of sampling from $p(\theta, s_t, x_t|y^t)$ is much less than $p(\theta, x^t|y^t)$. Unlike Fearnhead, our approach does not require MCMC moves.

Our approach begins with a simple factorization: by the definition sufficiency,

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

This decomposes the joint learning problem into one of computing the marginal filtering distribution of sufficient statistics and states and updating parameters conditional on sufficient statistics. Notice the importance of using sufficient statistics: the filtering distribution is defined over $s_t$ and $x_t$, whose dimensionality is fixed for all time periods. To generate an exact or direct i.i.d. sample from the particle approximation, we express $p(s_{t+1}, x_{t+1}|y^{t+1})$ as a marginal distribution against the previous period's target distribution, $p(\theta, s_t, x_t|y^t)$. Using this new expression, $p^N(s_{t+1}, x_{t+1}|y^{t+1})$ is a standard mixture distribution that can be directly sampled by selecting the mixture index (resampling), propagating $x_t$, updating and $s_t$, and then drawing parameters from $p(\theta|s_{t+1})$.

This initial resampling is crucial. It insures that high-likelihood state, sufficient statistics, and parameter particles are propagated forward. In the case of pure state filtering, Pitt and Shephard (1999) introduced the idea of resampling first, and they find drastic improvements. For the class of models we consider, our algorithm is fully-adapted, in the terminology of Pitt and Shephard (1999), and therefore provides exact or i.i.d. samples from $p^N(\theta, s_t, x_t|y^t)$. The main competitor method, Storvik (2002), propagates first and then resamples, resulting in poor samples from particle approximation.

One of the key advantages of our approach is the ability to compute marginal likelihoods, integrating out both states and parameters. These marginal likelihoods can by

used for sequential model specification and hypothesis tests using Bayes factors and also for computing standardized residuals. Computing Bayes factors in state space models is often difficult, even with MCMC methods. Marginal likelihoods are easy to compute using our particle filtering approach, providing a complete toolkit for estimation and inference as researchers can now learn about parameters, states, models and hypotheses sequentially through time. To our knowledge, this has never been done previously integrating out all of the parameter uncertainty.

To demonstrate our approach, we consider a range of empirical applications with both simulated and real data examples using a stochastic volatility model and an linear autoregressive model with t-distributed errors. Using simulated data, we show that our algorithm outperforms existing particle filtering algorithms such as Storvik's, accurately and efficiently learning the parameters and state variables for sample sizes commonly encountered in practice. In an extension of this paper, Carvalho, Johannes, Lopes, and Polson (2008) provide comparisons with MCMC algorithms and shows for the class of models considered here, that particle based algorithms provide equal or better accuracy than MCMC methods at a fraction of the computational cost. They also compare additional comparisons to the approach of Liu and West (2001).

We consider two real data examples: modeling the stochastic volatility of Nasdaq 100 stock index returns and a model of PCE core inflation with persistent and transitory components, allowing for potential unit roots and non-normal shocks. We consider sequential parameter learning, sequential model choice and hypothesis testing. The stochastic volatility results are interesting in part because we consider a sample period beginning in 1999 when volatility was particularly high and continuing through 2004, 2005, and 2006, when volatility was particularly low. The sequential parameter estimates highlight how investor's views about the structural parameters and volatility change over time. The model choice results provide strong evidence in support of a stationary stochastic volatility process.

The core inflation example generates even more interesting results. In addition to substantial variation in the parameter estimates over time, there is strong evidence for t-distributed shocks to the state equation, but little evidence for fat-tailed observation errors. This is indicative of outliers in the shocks to the predictable component. Combined with the high persistence of the persistent component, this creates problems for standard macroeconomic approaches to optimal monetary policy which often use quadratic objective functions. This is clearly problematic and points to the importance of explicitly modeling

the policy makers preferences over tail behavior. Moreover, in all specifications there a evidence against a unit root, although the evidence is not overwhelming.

Finally, we discuss the issue of particle learning in large $T$ samples. One potential criticism of sequential parameter learning using particle filtering methods is that some particle filtering algorithms degenerate for large sample sizes (Andrieu, Doucet, and Tadic (2005)). We first show that our algorithms do not appear degenerate even for very long time series samples with $T = 2500$ monthly observations. The degeneracy occurs in algorithms that rely on learning the entire vector $x^t$, which we avoid by the use of sufficient statistics. We also discuss the role of Monte Carlo errors, and the asymptotics (in both $T$ and $N$) associated with these algorithms.

The rest of the paper is outlined as follows. Section 2 provides the overall framework, algorithms, comparisons with existing algorithms, and detailed examples. Section 3 summarizes the simulated and real data examples for the stochastic volatility model and a model with t-distributed errors. Section 4 concludes.

# 2 State filtering and parameter learning

Consider a Markov state space model specified via the observation equation, $p(y_t|x_t, \theta)$, state evolution, $p(x_{t+1}|x_t, \theta)$, initial state distribution, $p(x_0|\theta)$, and prior parameter distribution, $p(\theta)$. We consider the broad class of models in which $p(y_t|x_t, \theta)$ and $p(x_{t+1}|x_t, \theta)$ are discrete or scale mixtures of normal distributions. We also assume that the parameter posteriors must admit a conditional sufficient statistics, which are recursively defined via

$$s_{t+1} = \mathcal{S}(s_t, x_{t+1}, y_{t+1}).$$

This implies that $p(\theta|s_t)$ is a known, easy to simulate distribution. Later, in the context of specific conditionally Gaussian models, we will modify these sufficient statistics with additional auxiliary variables $\lambda_{t+1}$ that modulate the non-normality of the errors.

We use particle methods to characterize $p(\theta, s_t, x_t|y^t)$ via a particle approximation:

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

where $N$ is the number of particles and $(\theta, s_t, x_t)^{(i)}$ denotes the particle triplet of parameters, sufficient statistics, and states. A particle filtering algorithm consists of a recursive

algorithm for generating new particles, $(\theta, s_{t+1}, x_{t+1})^{(i)}$, given existing particles and a new observation, $y_{t+1}$.

A generic alternative to particle methods is to use Markov Chain Monte Carlo (MCMC) methods. MCMC methods can generate approximate samples from $p(\theta, x^t|y^t)$ and the output can be used to estimate the marginal distribution, $p(\theta, x_t|y^t)$. In practice, this is not computationally feasible for real time applications or for very large datasets. This would require running MCMC algorithms t-times, which is computationally prohibitive. Moreover, even in simple state space models, MCMC methods can suffer from well known problems such as slow convergence and can also be highly sensitive to parameterizations. Due to this, we consider only particle based algorithms.

## 2.1 General approach

Our general approach relies on a factorization of the joint posterior distribution:

$$p\left(\theta, s_{t+1}, x_{t+1}|y^{t+1}\right) = p(\theta|s_{t+1})\, p\left(s_{t+1}, x_{t+1}|y^{t+1}\right), \tag{2}$$

by the definition of sufficient statistics. Thus, the joint learning problem decomposes into a filtering step, computing the marginal distribution of sufficient statistics and state variables, and a parameter updating step. This decomposing, to our knowledge, is new. Before discussing any sampling algorithms, there are a number of immediate implications.

First, the dimensionality of the problem is fixed, as we track $x_t$ and $s_t$. This can be contrasted with traditional sequential importance sampling algorithms and the resample-move algorithm of Gilks and Berzuini (2001) that track the entire history $x^t$. Conditional on parameters, the problem of tracking $x^t$ by sampling from $p(x^t|\theta, y^t)$ is much more difficult than $p(x_t|\theta, y^t)$, since $t$ is much larger than the dimension of $x_t$. Formally, while particle approximations to both $p(x^t|\theta, y^t)$ and $p(x_t|\theta, y^t)$ converge, the errors in particle approximations to $p(x_t|\theta, y^t)$ are exponentially smaller than those generated by $p(x^t|\theta, y^t)$. Chopin (2004) discusses this issue formally and Klaas et al. (2005) provide additional evidence.[1] By introducing sufficient statistics, the dimensionality of the target distribution

[1]This appears to be widely understood in the filtering literature. For example, in the context of MLE estimation, Poyiadis, Doucet, and Singh (2005) note that previous sequential estimates were based on $p^N(x^t|\theta, y^t)$ :"*as in the case of filtering based parameter estimation, the approximation errors they procedure increase with the data length. The methods we proposed here to approximate the filter derivative are based on the sequence of marginal distributions, $p^N(x_t|y^t, \theta)$, and hence do not suffer from the aforementioned problem.*"

is fixed and does not suffer from exponential degeneracy or "curse of dimensionality" induced by tracking $x^t$. Our contribution is not the introduction of sufficient statistics, as this was done earlier by Storvik (2002) in a pure particle filtering algorithm and Andrieu, de Freitas, and Doucet (1999) and Fearnhead (2002) in the context of the MCMC based resample-move algorithm.

Second, this decomposition implies that there is no need for MCMC methods, which are commonly used in sequential parameter and state learning algorithms (Andrieu, de Freitas, and Doucet (1999), Gilks and Berzuini (2001) or Fearnhead (2002)). In generating particle approximations to the joint posterior, our approach first generates draws from the particle approximation to $p\left(s_{t+1}, x_{t+1}|y^{t+1}\right)$ and then updates parameters. No additional iterative steps are required. Third, unlike sequential importance sampling, we draw the parameters from their appropriate posterior distribution, $p\left(\theta|s_{t+1}\right)$. This is important because it provides fresh, updated parameter values. This additional randomness is crucial when learning parameters through time. As noted by Gilks and Berzuini (2001), traditional sequential importance sampling algorithms rapidly degenerate for sequential parameter estimation because they do not provide an opportunity to draw values for the parameters conditional on the new data.

The key to the performance of our algorithm is the exact or direct sampling. To do this, we express

$$p\left(s_{t+1}, x_{t+1}|y^{t+1}\right) = \int p\left(y_{t+1}|x_t, \theta\right) p\left(s_{t+1}, x_{t+1}|\theta, s_t, x_t, y_{t+1}\right) dp\left(\theta, s_t, x_t|y^t\right), \tag{3}$$

where

$$p\left(s_{t+1}, x_{t+1}|\theta, s_t, x_t, y_{t+1}\right) = p\left(s_{t+1}|x_{t+1}, s_t, y_{t+1}\right) p\left(x_{t+1}|\theta, x_t, y_{t+1}\right).$$

This is a slight abuse of notation since $s_{t+1} = \mathcal{S}\left(s_t, x_{t+1}, y_{t+1}\right)$ implies that $p\left(s_{t+1}|x_{t+1}, s_t, y_{t+1}\right)$ is a degenerate distribution. Since $s_t$ and $x_{t+1}$ are random variables, the sufficient statistics are also random and are be replenished in the filtering step.

Given a particle approximation to $p\left(\theta, s_t, x_t|y^t\right)$, the joint filtering problem is given by

$$p^N\left(s_{t+1}, x_{t+1}|y^{t+1}\right) = \int p\left(y_{t+1}|x_t, \theta\right) p\left(s_{t+1}, x_{t+1}|\theta, s_t, x_t, y_{t+1}\right) dp^N\left(\theta, s_t, x_t|y^t\right) \tag{4}$$

$$= \sum_{i=1}^{N} w_{t+1}\left(\left(\theta, x_t\right)^{(i)}\right) p\left(s_{t+1}, x_{t+1}|\left(\theta, s_t, x_t\right)^{(i)}, y_{t+1}\right), \tag{5}$$

with mixture weights

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

Note that $p^N\left(s_{t+1}, x_{t+1}|y^{t+1}\right)$ is just a standard discrete mixture distribution, with mixing probabilities, $w_{t+1}\left(\theta, x_t\right)$, and $p\left(s_{t+1}, x_{t+1}|\theta, s_t, x_t, y_{t+1}\right)$ is the conditional state distribution. Because of this, we can directly or exactly generate samples from the discrete distribution $p^N\left(s_{t+1}, x_{t+1}|y^{t+1}\right)$ and then update the parameters from $p\left(\theta|s_{t+1}\right)$ using the following algorithm:

---

**Algorithm: Particle Filtering and Parameter Learning**

*Step 1:* For $i = 1, ..., N$, draw $z_t^{(i)} \sim Multi_N\left(\left\{w_{t+1}\left(\left(\theta, x_t\right)^{(i)}\right)\right\}_{i=1}^{N}\right)$, where $Multi_N$ is the multinomial distribution;

*Step 2:* For $i = 1, ..., N$, draw $x_{t+1}^{(i)} \sim p\left(x_{t+1}|\left(x_t, \theta\right)^{z_t^{(i)}}, y_{t+1}\right)$ and compute $s_{t+1}^{(i)} = \mathcal{S}\left(s_t^{z_t^{(i)}}, x_{t+1}^{(i)}, y_{t+1}\right)$;

*Step 3:* For $i = 1, ..., N$, $\theta^{(i)} \sim p\left(\theta|s_{t+1}^{(i)}\right)$.

---

The first step draws a mixture index, and then the next three sample from the appropriate conditional distributions. The mixture index sampling can be done via multinomial sampling (as in the above algorithm) or any of a number of other methods (see Carpenter, Clifford, and Fearnhead (1999), Liu and Chen (1998)).

The output of the particle filtering algorithm can be used to compute a number of important quantities, most notably marginal parameter posteriors, the marginal state filtering distribution, marginal predictive distributions, marginal likelihoods and marginal state smoothing distributions. The advantage of particle filters in this context is that it is straightforward to integrate out both parameter and state uncertainty to compute these distributions. We discuss these issues in greater detail below.

Before providing examples of the algorithm and adapting it to specific conditionally Gaussian models, we provide a general discussion of notable features of the approach with comparisons to existing approaches.

**Discussion**

- In the terminology of Pitt and Shephard (1999), this algorithm is fully-adapted, providing a direct or exact draw from $p^N(\theta, s_t, x_t|y^t)$. Because of this, it will generally outperform other importance sampling algorithms such as Storvik (2002). Because of the direct draw, notice that there is no re-weighting or importance sampling weights at the end of the algorithm.

- This algorithm does not apply in all settings, as it may not be possible to calculate $p(y_{t+1}|x_t, \theta)$ or draw from $p(x_{t+1}|x_t, \theta, y_{t+1})$. In the following sections, we show that this general algorithm can be adapted, via additional auxiliary state variables, to conditionally Gaussian state space models, as analyzed by Carlin, Polson, and Stoffer (1992), Frühwirth-Schnatter (1994), Carter and Kohn (1994, 1996) and Shephard (1994). This is a very wide class of models, that is analytically tractable in our setting because of the discrete or scale mixture of normal distribution errors and the fact that the parameter posterior distributions admit conditional sufficient statistics. In this regard, the algorithm is similar to the Gibbs sampler.

- Sufficient statistics are clearly important, playing two roles. First, sufficient statistics imply that the sequential learning problem is of fixed dimension. This avoids the problems of sampling from increasingly high dimensional distributions. Second, models that admit fixed dimensional sufficient statistics have a certain amount of analytical tractability, implying that the sampling problems are reasonable, in some sense. By analogy, the Gibbs sampler can compute smoothing distributions in many models, provided the parameter posteriors admit conditional sufficient statistics. Also note that while the sufficient statistics have a deterministic recursive update, our algorithm generates draws from $p^N(x_t, s_t|y^t)$, which implies that the sufficient statistics are random. The sufficient statistic distribution would only degenerate if $x_t$ does, but this is unlikely due to the exact nature of our filtering algorithm using APF.

- In terms of particle filtering algorithms, the main competitor algorithm is Storvik (2002). Storvik's algorithm uses the blind sampling/importance resampling algorithm by propagating states via $p(x_{t+1}|x_t, \theta)$, updating sufficient statistics, drawing parameters, and resampling with $p(y_{t+1}|x_{t+1}, \theta)$, generating three main differences. First, our algorithm resamples first and then propagates. As discussed by Pitt and Shephard

(1999), this is generally more efficient as it propagates only high likelihood particles. Second, we propagate states from $p(x_{t+1}|x_t, \theta, y_{t+1})$ instead of $p(x_{t+1}|x_t, \theta)$. By 'adapting' the draws by accounting for $y_{t+1}$, our algorithm tilts the state draws toward those that were likely to have generated $y_{t+1}$. Storvik's algorithms simulates states blindly, without any account for the new observation. Third, since $p(y_{t+1}|x_t, \theta)$ is a flatter distribution than $p(y_{t+1}|x_{t+1}, \theta)$ (the former integrates out $x_{t+1}$), the weights used in resampling will be more evenly distributed for our algorithm than Storvik's. Because of this, our algorithm will have a greater effective sample size, a measure of Monte Carlo efficiency. Carvalho, Johannes, Lopes and Polson (2008) show that even more marginalization is possible, leading to even more efficient algorithms.

- Care must always be taken when using Monte Carlo methods. This algorithm, like MCMC, generates Monte Carlo errors, and it is always important to monitor these errors. Although our approach generates independent exact samples from $p^N(\theta, x_t, s_t|y^t)$, there is still the sampling error since the true target is $p(\theta, x_t, s_t|y^t)$. At this point, two issues are important to discuss.

    1. In comparing different particle filtering methods, metrics such as the effective sample size (ESS) are quite useful. The ESS monitors the diversity of the particles at the resampling stage. If there are $N$ particles, the ESS is the number of unique particles after resampling. If the weights become highly skewed, then the ESS will be small and the algorithm will not provide a good approximation to $p(\theta, x_t, s_t|y^t)$. We are always careful to monitor the ESS and compare ours to those of alternative particle filtering algorithms. As we show later, our approach generates drastic increases in ESS compared to Storvik's algorithm.

    2. For a given algorithm, choosing the number of particles is analogous to diagnosing convergence in MCMC algorithms. As in the case of MCMC methods, the accuracy of the particle filtering algorithms is a function of the dimension of the problem, the model, and the signal-to-noise ratio. One approach that we have used is to run the algorithm multiple times from the same initial conditions using different random seeds and then monitor the output. If the sequential posteriors change substantially across simulations, then the Monte Carlo error is clearly too large. The asymptotics of these algorithms are developed in Carvalho, Johannes, Lopes, and Polson (2008), and are similar to the asymptotics

of normal particle filtering algorithms. In general, for a fixed time-dimension, approximation errors shrink as $N$ increases. Carvalho, Johannes, Lopes, and Polson (2008) also provide comparisons to MCMC algorithms, and show that our algorithm is often preferred to MCMC.

- It is also important to compare our algorithm with the sample-move algorithm, one of the first approaches used for sequential parameter and state learning. Gilks and Berzuini (2001) developed the algorithm, with additional results in Andrieu, de Freitas and Doucet (1999) and Fearnhead (2002). These approaches track the entire history of states, $x^t$, and use a combination of particle filtering and MCMC moves to jointly update $\theta$ and $x^t$. The general resample-move algorithm of Gilks and Berzuini (2001) uses standard importance sampling methods to approximate $p\left(x^t|y^t\right)$. Previous applications of the resample-move algorithm considered only a small number of unknown model parameters. For example, Fearnhead (2002) and Gilks and Berzuini (2001) learn only one unknown parameter. Another way to see the differences is via a Rao-Blackwellization argument for $p\left(\theta|y^{t+1}\right)$. in the resample-move algorithm, the marginal parameter posteriors are given by

$$p\left(\theta|y^{t+1}\right) = \int\limits_{\mathcal{R}^{t+1}} p\left(\theta|x^{t+1}, y^{t+1}\right) p\left(x^{t+1}|y^{t+1}\right) dx^{t+1},$$

which is an integral over a $t+1$ dimensional space, thus the curse of dimensionality. With sufficient statistics,

$$p\left(\theta|y^{t+1}\right) = \int p\left(\theta|s_{t+1}\right) p\left(s_{t+1}|y^{t+1}\right) ds_{t+1},$$

which is a lower-dimensional integral. Since all Monte Carlo algorithms suffer from the curse of dimensionality, parameter estimates using $p\left(s_{t+1}|y^{t+1}\right)$ are more efficient than those based on $p\left(x^{t+1}|y^{t+1}\right)$.

## 2.2 Priors

- From the above representation of the posterior, we have an equivalent prior mixture of the form

$$p(\theta) = \int p(\theta|s_0)p(s_0)ds_0$$

In the case of imformative conditionally conjugate priors we choose $p(s_0)$ to be a point mass at the pre-specified hyperparameters and set the initial particles $s_0^{(i)} = s_0$.

For vague priors we follow Jeffreys and use heavy-tailed cauchy mixtures. For example for a mean or intercept if we assume that $p(\theta|s_0) \sim \mathcal{N}(0, s_0)$ and $s_0 \sim \mathcal{IG}\left(\frac{1}{2}, \frac{1}{2}\right)$ then we have a vauge prior $\theta \sim C(0,1)$ as in Jeffreys (1961). Hence we initialize our particle draws $s_0^{(i)} \sim \mathcal{IG}\left(\frac{1}{2}, \frac{1}{2}\right)$

For the use of half cauchy priors on variances see the discussion in Gelman et al (2007). Remember that priors such as $p(\theta, \sigma) \equiv 1/\sigma$ lead to improper posterior distributions in state space models.

- We can also address the sensitivity of the initial conditions as follows. Suppose that we wish to find the posterior $\pi(\theta|y)$ under a different prior $\pi(\theta)$ than used in the filtering algorithm. By Bayes rule we have

$$\pi(\theta|y^t) \propto \frac{\pi(\theta)}{p(\theta)} p(\theta|y^t) x$$

for any $t$. Hence if we have draws $\theta^{(i)} \sim p(\theta|y^t)$ we can use weights $\pi^{(i)} \propto \pi(\theta^{(i)})/p(\theta^{(i)})$ and use the particle approximation $\pi^N(\theta|y^t) = \sum_{i=1}^N \pi^{(i)} \delta_{\theta^{(i)}}$. This allows you to do sensitivity calculations and to deal with vague priors where accumulation error might be high.

Priors can't be improper and too diffuse (eg $\sigma_v$ in an SV model).

- Accumulation Errors

$C_T/\sqrt{N}$. Cases where uniformly bounded and others not (when observation variance is high). However, re-sampling first will always be better.

- The behaviour of the sufficient statistic distributions are well known in many common situations (West and Harrison, 1987) typically converging to a point and the parameter posteriors being conditionally normal.

## 2.3 algorithm

***Particle Filtering***

1. **(Re-sample)**: Generate an index $k(i) \sim Multi\left(w_n^{(i)}\right)$ where
2. **(Sample)**: Draw or propagate states

### 2.3.1 Illustrative example: AR(1) with noise

For a concrete example, consider the latent autoregressive, AR(1), with noise model:

$$y_{t+1} = x_{t+1} + \sigma\varepsilon_{t+1}^y$$
$$x_{t+1} = \alpha_x + \beta_x x_t + \sigma_x\varepsilon_{t+1}^x,$$

where the shocks are independent standard normal random variables and $\theta = (\alpha_x, \beta_x, \sigma_x^2, \sigma^2)$. There are many parameterizations of this model that could be analyzed. We assume an initial state distribution, $x_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$ and standard conjugate priors for the parameters: $\sigma^2 \sim \mathcal{IG}(a, A)$ and $p(\alpha_x, \beta_x|\sigma_x^2)\, p(\sigma_x^2) \sim \mathcal{N}(c, C)\, \mathcal{IG}(b/2, B/2)$, where $\mathcal{N}$ is the normal distribution and $\mathcal{IG}$ is the inverse gamma distribution.

The algorithm requires the predictive likelihood, the updated state distribution, the sufficient statistics, and the parameter posterior. The predictive likelihood used in the initial resampling step is $p(y_{t+1}|x_t, \theta) \sim \mathcal{N}(\alpha_x + \beta_x x_t, \sigma^2 + \sigma_x^2)$, which implies that

$$w_{t+1}(x_t, \theta) \propto \frac{1}{\sqrt{\sigma^2 + \sigma_x^2}} \exp\left(-\frac{1}{2}\frac{(y_{t+1} - \alpha_x - \beta_x x_t)^2}{\sigma^2 + \sigma_x^2}\right).$$

The updated state distribution is

$$p(x_{t+1}|x_t, \theta, y_{t+1}) \propto p(y_{t+1}|x_{t+1}, \theta)\, p(x_{t+1}|x_t, \theta) \sim \mathcal{N}\left(\mu_{t+1}, \sigma_{t+1}^2\right),$$

where

$$\frac{\mu_{t+1}}{\sigma_{t+1}^2} = \frac{y_{t+1}}{\sigma^2} + \frac{\alpha_x + \beta_x x_t}{\sigma_x^2} \text{ and } \frac{1}{\sigma_{t+1}^2} = \frac{1}{\sigma^2} + \frac{1}{\sigma_x^2}.$$

$p(x_{t+1}|x_t, \theta, y_{t+1})$ shows the sensitivity of state updating to parameters, one reason why the joint learning problem is more difficult than pure state filtering.

For the parameters and sufficient statistics, write $x_{t+1} = Z_t'\beta + \sigma_x\varepsilon_{t+1}^x$, where $Z_t = (1, x_t)'$ and $\beta = (\alpha_x, \beta_x)'$. Then,

$$p(\theta|s_t) = p\left(\beta|\sigma_x^2, s_t\right) p\left(\sigma^2|s_t\right) p\left(\sigma_x^2|s_t\right),$$

where $p\left(\sigma^2|s_{t+1}\right) \sim \mathcal{IG}\left(a_{t+1}/2, A_{t+1}/2\right)$, $p\left(\sigma_x^2|s_{t+1}\right) \sim \mathcal{IG}\left(b_{t+1}/2, B_{t+1}/2\right)$, and $p\left(\beta|\sigma_x^2, s_{t+1}\right) \sim \mathcal{N}\left(c_{t+1}, \sigma_x^2 C_{t+1}^{-1}\right)$. The sufficient statistics are given by $s_{t+1} = (A_{t+1}, B_{t+1}, c_{t+1}, C_{t+1})$, and the recursive mapping is given by

$$A_{t+1} = (y_{t+1} - x_{t+1})^2 + A_t,\ B_{t+1} = B_t + c_t' C_t c_t + x_{t+1}' x_{t+1} - c_{t+1}' C_{t+1} c_{t+1},$$
$$c_{t+1} = C_{t+1}^{-1}\left(C_t c_t + Z_t x_{t+1}\right),\ \text{and}\ C_{t+1} = C_t + Z_t Z_t'.$$

The hyperparameters are deterministic and given by $a_{t+1} = 1 + a_t$ and $b_{t+1} = 1 + b_t$.

The full algorithm consists of the following steps:

---

**Algorithm: AR(1) model state filtering and parameter learning**

*Step 1:* For $i = 1, ..., N$, draw $z_t^{(i)} \sim Multi_N\left[\left\{w_{t+1}\left((x_t, \theta)^{(i)}\right)\right\}_{i=1}^{N}\right]$;

*Step 2:* For $i = 1, ..., N$, draw $x_{t+1}^{(i)} \sim p\left(x_{t+1}|(x_t, \theta)^{z_t^{(i)}}, y_{t+1}\right)$ and $s_{t+1}^{(i)} = \mathcal{S}\left(s_t^{z_t^{(i)}}, x_{t+1}^{(i)}, y_{t+1}\right)$;

*Step 3:* For $i = 1, ..., N$, draw $\left(\sigma^2\right)^{(i)} \sim \mathcal{IG}\left(a_{t+1}/2, A_{t+1}^{(i)}/2\right)$, $\left(\sigma_x^2\right)^{(i)} \sim \mathcal{IG}\left(b_{t+1}/2, B_{t+1}^{(i)}/2\right)$, and $(\beta)^{(i)} \sim \mathcal{N}\left(c_{t+1}^{(i)}, \left(\sigma_x^2\right)^{(i)}\left(C_{t+1}^{-1}\right)^{(i)}\right)$.

---

### 2.3.2 A motivation using the fully adapted APF

To motivate our algorithm and why it outperforms Storvik's (2002) algorithm, consider the simpler case of pure state filtering, assuming parameters are known. Here, we compare and contrast traditional state filtering algorithms: the classic sampling/importance resampling algorithm of Gordon, Salmond, and Smith (1993) and the auxiliary particle filtering algorithm of Pitt and Shephard (1999). The key insight is (a) the ordering of the algorithm matters, as they show that in many cases, resampling first and then propagating is generally preferred and (b) that $y_{t+1}$ should be taken into account when propagating states. In the case of a fully-adapted auxiliary particle, the algorithm provides a direct i.i.d. draw, as opposed to an approximate draw using importance sampling, from $p^N\left(x_{t+1}|y^{t+1}\right)$. The merits of this algorithm have been repeatedly documented in many settings, but especially those with outliers or slight model misspecification.

Traditional particle filtering algorithms are based on the following recursive representation of the filtering distribution:

$$\begin{aligned} p\left(x_{t+1}|y^{t+1}\right) &\propto p\left(y_{t+1}|x_{t+1}\right) p\left(x_{t+1}|y^{t}\right) \\ &\propto p\left(y_{t+1}|x_{t+1}\right) \int p\left(x_{t+1}|x_{t}\right) p\left(x_{t}|y^{t}\right) dx_{t}. \end{aligned}$$

A particle approximation to $p\left(x_t|y^t\right)$ consists of $N-$weights $\pi_t^{(i)}$ and states $x_t^{(i)}$ for $i = 1, ..., N$. In many cases, the weights are equal, $\pi_t^{(i)} = N^{-1}$, which we assume without any loss of generality. With this approximation, the particle filtering approximation to $p\left(x_{t+1}|y^{t+1}\right)$ is

$$p^N\left(x_{t+1}|y^{t+1}\right) \propto \sum\nolimits_{i=1}^{N} p\left(y_{t+1}|x_{t+1}\right) p\left(x_{t+1}|x_t^{(i)}\right).$$

The main challenge is how to sample from this mixture distribution.

The classic algorithm draws from $p^N\left(x_{t+1}|y^{t+1}\right)$ using the sampling/importance resampling algorithm. This provides an approximate importance sample from $p^N\left(x_{t+1}|y^{t+1}\right)$. The algorithm consists of the following steps

*Propagate*: For $i = 1, ..., N$, simulate $x_{t+1}^{(i)} \sim p\left(x_{t+1}|x_t^{(i)}\right)$

*Resample*: For $i = 1, ..., N$, draw $z\left(i\right) \sim Mult_N\left(\left\{w_{t+1}\left(x_{t+1}^{(i)}\right)\right\}_{i=1}^{N}\right)$ and set $x_{t+1}^{(i)} = x_{t+1}^{(z(i))}$,

where

$$w_{t+1}\left(x_{t+1}^{(i)}\right) = \frac{p\left(y_{t+1}|x_{t+1}^{(i)}\right)}{\sum_{j=1}^{N} p\left(y_{t+1}|x_{t+1}^{(j)}\right)}.$$

Notice the order: the algorithm blindly propagates states and then resamples. The problem is that eventually observations arrive generating unbalanced weights: one of the $p\left(y_{t+1}|x_{t+1}^{(i)}\right)$'s is much larger than the others, and resampling replicates a small number of $x_{t+1}^{(i)}$'s. This is known as sample impoverishment. In the limit, all the particles could take a single value. The problem is that the algorithm propagates particles without reference to the new observation, $y_{t+1}$. Similarly, any outlying observations or model misspecification with create serious problems for this algorithm.

An alternative known as the fully-adapted auxiliary particle filter reverses the order: first resampling and then propagating. This algorithm can be motivated by first expressing the filtering density recursively via

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

Notice that this involves $p\left(x_{t+1}|x_t, y_{t+1}\right)$ and $p\left(y_{t+1}|x_t\right)$ instead of $p\left(x_{t+1}|x_t\right)$ and $p\left(y_{t+1}|x_{t+1}\right)$. This generates a different particle approximation:

$$p^N\left(x_{t+1}|y^{t+1}\right) = \sum_{i=1}^N w_{t+1}\left(x_t^{(i)}\right) p\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right),$$

where

$$w_{t+1}\left(x_t^{(i)}\right) = p\left(y_{t+1}|x_t^{(i)}\right) \Big/ \sum_{i=1}^N p\left(y_{t+1}|x_t^{(i)}\right).$$

Again, filtering reduces to a sampling problem: how to sample from $p^N\left(x_{t+1}|y^{t+1}\right)$? In this case, $p^N\left(x_{t+1}|y^{t+1}\right)$ is a standard discrete-mixture distribution, and the mixture weights are functions of $y_{t+1}$ and $x_t$ and not $x_{t+1}$. Assuming that one can evaluate $p\left(y_{t+1}|x_t\right)$ and draw from $p\left(x_{t+1}|x_t, y_{t+1}\right)$, then it is possible to directly or exactly sampled from $p^N\left(x_{t+1}|y^{t+1}\right)$ by first selecting mixture indices, and then simulating the states. The algorithm consists of the following steps

*Resample*: For $i = 1, ..., N$, draw $z\left(i\right) \sim Mult_N\left(\left\{w_{t+1}\left(x_t^{(i)}\right)\right\}_{i=1}^N\right)$ and set $x_t^{(i)} = x_{t+1}^{z(i)}$

*Propagate*: For $i = 1, ..., N$, simulate $x_{t+1}^{(i)} \sim p\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right)$.

The fully-adapted APF has two key insights. First, the APF resamples first, then propagates. This is exactly the reverse order of the SIR algorithm and many other sequential importance sampling algorithms. At this initial stage, the algorithm resamples the old $x_t^{(i)}$ particles by their predictive weights, $p\left(y_{t+1}|x_t\right)$, propagating those particles that were most likely to have generated the next period's observation. Second, these high-likelihood particles are propagated forward using $p\left(x_{t+1}|x_t, y_{t+1}\right)$, the posterior distribution of the state given past states and new data. This algorithm also coincides with an algorithm that uses the optimal importance function for propagating states, $p\left(x_{t+1}|x_t, y_{t+1}\right)$ (see Doucet et al. (2001) or Doucet, Godsill, and Andrieu (2000)) with initial resampling. As mentioned earlier, since $p\left(y_{t+1}|x_t\right)$ is a flatter distribution than $p\left(y_{t+1}|x_{t+1}\right)$, the algorithm has a greater effective sample size than the SIR algorithm.

This fully-adapted APF provides direct sample from the distribution, $p^N\left(x_{t+1}|y^{t+1}\right)$, without resorting to an approximate sample via importance sampling. Thus, the fully-adapted APF only suffers from the error of approximating $p\left(x_{t+1}|y^{t+1}\right)$ via $p^N\left(x_{t+1}|y^{t+1}\right)$ and not additional in generating an approximate sample from $p^N\left(x_{t+1}|y^{t+1}\right)$. This is crucial in any sequential setting, as errors can accumulate. Direct or exact sampling is generally

preferred to importance sampling whenever possible, except in certain degenerate cases. Our contribution is to extend this idea to the setting with parameter learning. In this regard, our approach can be viewed as an extension of the fully-adapted APF to the case of sampling from particle approximations to $p(\theta, s_t, x_t|y^t)$, which is why the algorithm performs better than existing algorithms.

## 2.4 Computing Bayes factors for model and hypothesis testing

In addition to sequential parameter estimation, particle filters are extremely useful for Bayesian model specification and hypothesis testing. Bayesian model comparison and hypothesis testing utilizes the Bayes factor, essentially a likelihood ratio between competing specifications. Formally, given a number of competing model specifications, generically labeled as model $\mathcal{M}_i$ and $\mathcal{M}_j$, the Bayesian approach computes the Bayes factor:

$$\mathcal{BF}_{i,j}^t = \frac{p(y^t|\mathcal{M}_i)}{p(y^t|\mathcal{M}_j)},$$

providing a measure of the relative merits of the competing models. These model specifications could be nested or non-nested and could involve hypothesis tests regarding parameter values. In state-space models, it is useful to recursively define the Bayes factor as

$$\mathcal{BF}_{i,j}^{t+1} = \frac{p(y_{t+1}|y^t, \mathcal{M}_i)}{p(y_{t+1}|y^t, \mathcal{M}_j)}\mathcal{BF}_{i,j}^t,$$

where the change in the Bayes factor is driven by the new information embedded in the marginal likelihoods. Most applications utilize full-sample likelihood ratios, $\mathcal{BF}_{i,j}^T$, providing an overall measure of model fit for entire sample.

The main computational problem is one of computing $p(y_{t+1}|y^t, \mathcal{M}_i)$. This is the time $t+1$ marginal likelihood component,

$$p(y_{t+1}|y^t, \mathcal{M}_i) = \int p\left(y_{t+1}|x_t, \theta\right) p\left(x_t, \theta|y^t\right) d\left(x_t, \theta\right),$$

which is not known analytically and is difficult to compute, even using MCMC methods. Since our algorithm provides approximate samples from $p(x_t, \theta|y^t)$, it is straightforward to compute Monte Carlo estimates of marginal likelihoods:

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

where $\left\{(x_t, \theta)^{(i)}\right\}_{i=1}^{N}$ is a particle sample from our algorithm. Notice, there are no importance sampling weights due to the exact or i.i.d. particle sampling.

Computing marginal likelihoods is a clear advantage of particle filters over competing methods such as MCMC. MCMC methods generally provide samples from the full-sample posterior, $p\left(\theta, x^T | y^T\right)$, and in many cases it is difficult to use the output to compute marginal likelihoods, which are an integral against the prior distribution (see Han and Carlin (2001) for a review). A number of papers have analyzed model choice using particle filters, assuming the parameters are known (e.g., Kim, Shephard, and Chib (1998) and Johannes, Polson, and Stroud (2008)).

## 2.5 Conditionally Gaussian state space models

The algorithm given earlier can be modified in a straightforward manner to handle errors that are discrete or continuous mixtures of normal distributions. This class of shocks has a long history in state space modeling. In the case of the smoothing problem, models with scale and/or discrete mixture errors have been extensively analyzed using MCMC methods, as mentioned above. The scale mixture class is quite broad, including the cases of implies that we allow for $t-$distributed errors, stable errors, double exponential errors, and discrete mixture errors. This latter case includes the important class of log-stochastic volatility models using the representation of Kim, Shephard, and Chib (1998), which we analyze below.

One of the examples that we considered is a fat-tailed autoregressive model:

$$\begin{aligned} y_{t+1} &= x_{t+1} + \sigma\sqrt{\lambda_{t+1}^y}\varepsilon_{t+1} \\ x_{t+1} &= \alpha_x + \beta_x x_t + \sigma_x\sqrt{\lambda_{t+1}^x}\varepsilon_{t+1}^x, \end{aligned}$$

where the specification of $\lambda_{t+1} = \left(\lambda_{t+1}^y, \lambda_{t+1}^x\right)$ determines the error distribution. For example, if $\lambda_{t+1}^x$ and $\lambda_{t+1}^y$ are independent inverse gamma variates $\left(\mathcal{IG}\left(\nu/2, \nu/2\right)\right)$, then the model has t-distributed errors. This is a robust version of the linear Gaussian state space model commonly used, with the important extension that we are interested in additionally learning the parameters. Data augmentation is the key in this setting, not only for state filtering but also for generating sufficient statistics. With data augmentation, the sufficient statistic recursions are defined as

$$s_{t+1} = \mathcal{S}\left(s_t, x_{t+1}, \lambda_{t+1}, y_{t+1}\right).$$

It is important to note that the parameter posteriors generally do not admit sufficient statistics unless we introduce the latent auxiliary variables.

The algorithm outlined in Section 2.1 requires an analytical form for $p\left(y_{t+1}|x_t,\theta\right)$ and an ability to simulate from $p\left(\theta|s_{t+1}\right)$ and $p\left(\lambda_{t+1},x_{t+1}|x_t,\theta,y_{t+1}\right)$. For the discrete mixture models, these densities are analytically known and a straightforward modification can be used for the continuous mixture models.

### 2.5.1 Discrete mixtures

For the discrete mixture case, $\lambda_{t+1}$ is an indicator variable taking a finite number of values. For simplicity, we focus the discussion on the case with mixture errors in the observation equation. The case with mixture errors in the state equation or both equations is handled similarly, although they are notationally more complicated. We also assume the discrete mixture transitions are i.i.d. with probabilities $(p_1,...,p_K)$.

A popular example of this class of models is an accurate approximation of the log-stochastic volatility first used in Kim, Shephard, and Chib (1998). The traditional log SV model (Jacquier, Polson and Rossi (1994)) is given by

$$\begin{aligned} r_{t+1} &= \exp\left(\frac{x_{t+1}}{2}\right)\varepsilon_{t+1} \\ x_{t+1} &= \alpha_x + \beta_x x_t + \sigma_x \varepsilon^x_{t+1}, \end{aligned}$$

where the errors are independent standard normal random variables. This model is Gaussian, but the observation equation is nonlinear. Defining $y_t = \ln r_t^2$,

$$\begin{aligned} y_{t+1} &= x_{t+1} + \epsilon_{t+1} \\ x_{t+1} &= \alpha_x + \beta_x x_t + \sigma_x \varepsilon^x_{t+1}, \end{aligned}$$

where $\epsilon_t$ is a $log\left(\chi_1^2\right)$ random variable. Kim, Shephard, and Chib (1998) and Omori, Chib, and Shephard (2006) show that the distribution of $\epsilon_t$ can be accurately approximated via a $K$-component discrete normal mixture approximation with fixed weights, $\sum_{j=1}^K p_j Z_j$ where $Z_j \sim N(\mu_j,\sigma_j^2)$. The indicator variable $\lambda_{t+1}$ tracks the mixture components, with, for example, $\lambda_{t+1}=j$ indicating a current state in mixture component $j$.

For the general class of discrete mixture models, if we assume that the distribution of $y_{t+1}$ conditional $x_t$ and on being in mixture $j$ is $\mathcal{N}\left(\alpha_x+\beta_x x_t+\mu_j,\sigma_j^2+\sigma_x^2\right)$, then the

predictive likelihood is

$$p\left(y_{t+1}|x_t,\theta\right)=\sum_{j=1}^{K}p_j\mathcal{N}\left(\alpha_x+\beta_x x_t+\mu_j,\sigma_j^2+\sigma_x^2\right).$$

To propagate the states, note that

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

The distribution $p\left(x_{t+1}|x_t,\theta,\lambda_{t+1},y_{t+1}\right)$ is Gaussian and $p\left(\lambda_{t+1}|x_t,\theta,y_{t+1}\right)$ is a discrete distribution, both of which can be easily sampled. Finally, the sufficient statistics are updated with $s_{t+1}=\mathcal{S}\left(s_t,x_{t+1},\lambda_{t+1},y_{t+1}\right)$.

The full algorithm for discrete mixtures is given by

---

**Algorithm: Particle filtering and parameter learning for discrete mixtures**

*Step 1:* For $i=1,...,N$, draw $z_t^{(i)}\sim Multi_N\left(\left\{w_{t+1}\left((x_t,\theta)^{(i)}\right)\right\}_{i=1}^{N}\right)$;

*Step 2:* For $i=1,...,N$, draw $\lambda_{t+1}^{(i)}\sim p\left(\lambda_{t+1}|\left(x_t,\theta\right)^{z_t^{(i)}},y_{t+1}\right)$, $x_{t+1}^{(i)}\sim p\left(x_{t+1}|\left(x_t,\theta\right)^{z_t^{(i)}},\lambda_{t+1}^{(i)},y_{t+1}\right)$, and set $s_{t+1}^{(i)}=\mathcal{S}\left(s_t^{z_t^{(i)}},x_{t+1}^{(i)},\lambda_{t+1}^{(i)},y_{t+1}\right)$;

*Step 3:* For $i=1,...,N$, draw $\theta^{(i)}\sim p\left(\theta|s_{t+1}^{(i)}\right)$.

---

**Stochastic volatility errors** Consider the stochastic volatility model from above

$$\begin{aligned}y_t&=x_t+\epsilon_t\\x_t&=\alpha_x+\beta_x x_{t-1}+\sigma_x v_t\end{aligned}$$

where $p\left(\epsilon_t\right)=\sum_{j=1}^{K}p_j\phi_j$ and each distributional component, $\phi_j$, is normal. We assume standard conjugate priors for the parameters. Thus, conjugate: $p\left(\alpha_x,\beta_x|\sigma_x^2\right)p\left(\sigma_x^2\right)\sim\mathcal{N}\left(c,C\sigma_x^2\right)\mathcal{IG}\left(\frac{b}{2},\frac{B}{2}\right)$.

To implement the algorithm, the predictive likelihood is given by

$$p(y_{t+1}|x_t,\theta)=\sum_{j=1}^{K}p_j N(\alpha+\beta x_t,\sigma_j^2+\sigma_x^2),$$

which implies that the weights, $w_{t+1}\left((x_t,\theta)^{(i)}\right)$, are easy to compute. The other required distributions for propagating the states are also easy to compute. First, $p\left(\lambda_{t+1}|x_t,\theta,y_{t+1}\right)$ is a known discrete distribution since

$$p\left(\lambda_{t+1}=j|x_t,\theta,y_{t+1}\right)\propto p\left(y_{t+1}|x_t,\theta,\lambda_{t+1}=j\right)p_j,$$

where the conditional likelihood is

$$p\left(y_{t+1}|x_t,\theta,\lambda_{t+1}=j\right)=\mathcal{N}\left(\alpha_x+\beta_x x_t+\mu_j,\sigma_x^2+\sigma_j^2\right).$$

Drawing from this distribution is straightforward.

To update the persistent states,

$$p\left(x_{t+1}|\lambda_{t+1}=j,x_t,\theta,y_{t+1}\right)\propto p\left(y_{t+1}|x_{t+1},\lambda_{t+1}=j,\theta\right)p\left(x_{t+1}|x_t,\theta\right)\sim\mathcal{N}\left(\mu_{t+1},\sigma_{t+1}^2\right)$$

where

$$\frac{\mu_{t+1}}{\sigma_{t+1}^2}=\frac{y_{t+1}-\mu_j}{\sigma_j^2}+\frac{\alpha_x+\beta_x x_t}{\sigma_x^2}\text{ and }\frac{1}{\sigma_{t+1}^2}=\frac{1}{\sigma_j^2}+\frac{1}{\sigma_x^2}.$$

Conditional on $\lambda_{t+1}$, the sufficient statistics and parameter posteriors are similar to those for the AR(1) example and are omitted.

### 2.5.2 Continuous mixtures

Continuous-mixtures are a bit more complicated and require a slight modification of the algorithm. This the case, for example, with t-distributed errors, stable errors, or double exponential errors. In this case, we write the particle distribution as

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

by first updating $\lambda_{t+1}$, then $x_{t+1}$, then $s_{t+1}$, and finally $\theta$. As in Section 2.1, we rely on the factorization and careful marginalization arguments to generate samples from the joint distribution. Again, for notational simplicity, we consider the case with scale mixture errors in the observation equation, although we discuss the more general case with t-errors in both equations below in an example. The same argument directly applies to scale mixture errors additional in the state evolution.

For scale mixture errors, $p\left(s_{t+1},x_{t+1},\lambda_{t+1}|y^{t+1}\right)$ can be expressed as

$$p\left(s_{t+1},x_{t+1},\lambda_{t+1}|y^{t+1}\right)=\int p\left(s_{t+1},x_{t+1}|s_t,x_t,\theta,\lambda_{t+1},y_{t+1}\right)p\left(y_{t+1}|x_t,\theta,\lambda_{t+1}\right)dp\left(s_t,x_t,\theta,\lambda_{t+1}|y^t\right).$$

Note that the filtering distribution is now defined over $p\left(s_t, x_t, \theta, \lambda_{t+1}|y^t\right)$, which has a particularly simple form since $\lambda_{t+1}$ is i.i.d.

$$p\left(s_t, x_t, \theta, \lambda_{t+1}|y^t\right) = p\left(\lambda_{t+1}\right) p\left(s_t, x_t, \theta|y^t\right).$$

The particle approximation is then given by

$$\begin{aligned} p^N\left(s_{t+1}, x_{t+1}, \lambda_{t+1}|y^{t+1}\right) &= \int p\left(s_{t+1}, x_{t+1}|\theta, x_t, \lambda_{t+1}, y_{t+1}\right) p\left(y_{t+1}|\theta, x_t, \lambda_{t+1}\right) dp^N\left(\theta, s_t, x_t, \lambda_{t+1}|y^t\right) \\ &= \sum_{i=1}^{N} w_{t+1}\left(\left(\theta, x_t, \lambda_{t+1}\right)^{(i)}\right) p\left(s_{t+1}, x_{t+1}|\left(\theta, s_t, x_t, \lambda_{t+1}\right)^{(i)}, y_{t+1}\right), \end{aligned}$$

where

$$w_{t+1}\left(\left(\theta, s_t, x_t, \lambda_{t+1}\right)^{(i)}\right) = \frac{p\left(y_{t+1}|\left(\theta, s_t, x_t, \lambda_{t+1}\right)^{(i)}\right)}{\sum_{i=1}^{N} p\left(y_{t+1}|\left(\theta, s_t, x_t, \lambda_{t+1}\right)^{(i)}\right)}.$$

For common models with scale mixtures, all of these quantities can be sampled directly..

The algorithm for scale mixtures is

---

**Algorithm: Particle filtering and parameter learning for continuous scale mixture**

*Step 1:* For $i = 1, ..., N$, draw $z_t^{(i)} \sim Multi_N\left(\left\{w_{t+1}\left(\left(\theta, x_t, \lambda_{t+1}\right)^{(i)}\right)\right\}_{i=1}^{N}\right)$, where $\lambda_{t+1} \sim p\left(\lambda_{t+1}\right)$

*Step 2:* For $i = 1, ..., N$, draw $x_{t+1}^{(i)} \sim p\left(x_{t+1}|\left(x_t, \theta, \lambda_{t+1}\right)^{z_t^{(i)}}, y_{t+1}\right)$ and set

$s_{t+1}^{(i)} = \mathcal{S}\left(s_t^{z_t^{(i)}}, x_{t+1}^{(i)}, \lambda_{t+1}^{z_t^{(i)}}, y_{t+1}\right)$,

*Step 3:* For $i = 1, ..., N$, draw $\theta^{(i)} \sim p\left(\theta|s_{t+1}^{(i)}\right)$.

---

This algorithm provides a direct sample from $p^N\left(\theta, s_{t+1}, \lambda_{t+1}, x_{t+1}|y^{t+1}\right)$. Of course, it is also possible to add an additional 'MCMC' step before drawing the parameters to draw $\lambda_{t+1}$ conditional on $x_{t+1}$ and then $x_{t+1}$ given $\lambda_{t+1}$, as in Gilks and Berzuini (2001).

### 2.5.3 Illustrative example: T-distributed errors

Assume that the state and observation error distributions are $t-$distributed with $\nu$ and $\nu^x$ degrees of freedom, respectively. Using the scale mixture representation,

$$y_{t+1} = x_{t+1} + \sigma\sqrt{\lambda_{t+1}}\varepsilon_{t+1}$$
$$x_{t+1} = \alpha_x + \beta_x x_t + \sigma_x\sqrt{\omega_{t+1}}\varepsilon^x_{t+1}$$

where the auxiliary variables are independent, $\lambda_{t+1} \sim \mathcal{IG}(\nu/2, \nu/2)$, and $\omega_{t+1} \sim \mathcal{IG}(\nu^x/2, \nu^x/2)$. Conditional on $\lambda_{t+1}$ and $\omega_{t+1}$, the model is conditionally Gaussian. The parameter distributions are the same as in the previous examples.

Masreliez and Martin (1977) develop approximate robust state filters for models with $t-$distributed errors in either the state or observation equation, but not both. West (1981) and Gordon and Smith (1993) analyze the pure filtering problem. All of these papers assume the parameters are known. Storvik (2002) uses importance sampling to analyze sequential state and parameter learning assuming the observation errors, but not state errors, are $t-$distributed. Johannes, Polson, and Yae (2007) consider extensions that generate quantile errors, for robust estimation criterion.

Applying the general algorithm from earlier, $p(y_{t+1}|\theta, \lambda_{t+1}, \omega_{t+1}, x_t)$ and $p(x_{t+1}|\theta, \lambda_{t+1}, \omega_{t+1}, x_t)$ are required to implement our algorithm. The first defines the weights:

$$w_{t+1}\left((x_t, \theta)^{(i)}\right) \propto \frac{1}{\sqrt{(\sigma^2)^{(i)}\lambda^{(i)}_{t+1} + (\sigma^2_x)^{(i)}\omega^{(i)}_{t+1}}} \exp\left(-\frac{1}{2}\frac{\left(y_{t+1} - \alpha^{(i)}_x - \beta^{(i)}_x x^{(i)}_t\right)^2}{(\sigma^2)^{(i)}\lambda^{(i)}_{t+1} + (\sigma^2_x)^{(i)}\omega^{(i)}_{t+1}}\right).$$

For the states,

$$p(x_{t+1}|\theta, \lambda_{t+1}, \omega_{t+1}, x_t) \propto p(y_{t+1}|\lambda_{t+1}, x_{t+1}, \theta)\, p(x_{t+1}|\omega_{t+1}, x_t, \theta) \sim \mathcal{N}\left(\mu_{t+1}, \sigma^2_{t+1}\right),$$

where

$$\frac{\mu_{t+1}}{\sigma^2_{t+1}} = \frac{y_{t+1}}{\sigma^2\lambda_{t+1}} + \frac{\alpha_x + \beta_x x_t}{\sigma^2_x\omega_{t+1}} \text{ and } \frac{1}{\sigma^2_{t+1}} = \frac{1}{\sigma^2\lambda_{t+1}} + \frac{1}{\sigma^2_x\omega_{t+1}},$$

are straightforward modifications of the Kalman recursions.

For the parameter posteriors and sufficient statistics, re-write the state equation as

$$x_{t+1} = Z_t'\beta + \sigma_x\sqrt{\omega_{t+1}}\epsilon_{t+1}$$

where $Z_t = (1, x_t)'$ and $\beta = (\alpha_x, \beta_x)'$. This takes the form of a heteroskedastic regression, condiitonal on the states. Defining $s_{t+1} = (A_{t+1}, B_{t+1}, c_{t+1}, C_{t+1})$, $p(\sigma^2|s_{t+1}) \sim$

$\mathcal{IG}(a_{t+1}, A_{t+1})$, $p\left(\sigma_x^2|s_{_{t+1}}\right) \sim \mathcal{IG}(b_{t+1}, B_{t+1})$, and $p(\beta|\sigma_x^2, s_{t+1}) \sim \mathcal{N}\left(c_{t+1}, \sigma_x^2 C_{t+1}^{-1}\right)$, where $a_t = \frac{1}{2} + a_{t-1}$, $b_t = \frac{1}{2} + b_{t-1}$,

$$
\begin{aligned}
A_{t+1} &= \frac{(y_{t+1} - x_{t+1})^2}{\lambda_{t+1}} + A_t \\
B_{t+1} &= B_t + c_t' C_t c_t + \frac{x_{t+1}^2}{\omega_{t+1}} - c_{t+1}' C_{t+1} c_{t+1} \\
c_{t+1} &= C_{t+1}^{-1}\left(C_t c_t + \frac{Z_t' x_{t+1}}{\omega_{t+1}}\right), \text{ and} \\
C_{t+1} &= C_t + \frac{Z_t Z_t'}{\omega_{t+1}}.
\end{aligned}
$$

The t-distributed error model requires the specification of the degrees of freedom parameter in the $t$-distributions. We assume that $(\nu, \nu^x)$ are known parameters. The posterior distributions for the degrees of freedom do not admit fixed-dimensional sufficient statistics. However, it is possible to approximate the posterior by discretizing the support using our methodology. In simulations, we have found this approach to work well.

# 3 Illustrative examples

In this section, we provide a number of examples. The simulated data examples are used primarily to document the performance of the algorithms, and the real data applications provide new insights about stochastic volatility and inflation dynamics.

## 3.1 Stochastic volatility

### 3.1.1 Simulated data

We simulate 50 samples of length $T = 100$ from the stochastic volatility model for a range of parameters values to gauge the efficiency of our particle algorithm in comparison to Storvik's algorithm, the main competitor. For each series, we run particle filters with $N = 5000$ particles. The relatively short time spans are due to the computational demands of recursively running the particle filters on multiple datasets and parameterizations for two algorithms. Table 1 reports effective sample sizes for the two algorithms, reported as a percentage of the total particle numbers, with a cross-sectional estimate of the Monte Carlo error given in parentheses. For baseline parameters, we assume $\alpha_x = 0$ and consider a range

| | $\sigma_x$ | | | |
|---|---|---|---|---|
| | .2 | .5 | 1 | 2 |
| Exact sampling algorithm | 93 (1.1) | 89 (2.0) | 86 (2.4) | 84 (1.4) |
| Storvik's algorithm | 38 (1.6) | 35 (1.7) | 29 (1.5) | 22 (1.1) |

Table 1: Effective particle size of the filtering with sequential parameter learning. All numbers are expressed in the percentage of the physical particle size. Numbers in parenthesis are standard errors for the mean of effective particle size from 50 simulated time-series of length 100 which has SV errors. Physical particle size is 5000, and $\alpha = 0$.

of parameters determining the signal-to-noise ratios. We use the following hyperparameters generating proper but relatively diffusive initial priors: $\alpha_x|\sigma_x^2 \sim \mathcal{N}(0, \sigma_x^2/30)$, $\beta_x|\sigma_x^2 \sim \mathcal{N}(0.95, 0.1\sigma_x^2)$, and $\sigma_x^2 \sim \mathcal{IG}(8, 0.35)$.

The results in Table 1 document a drastic improvement in the effective sample size of the exact or i.i.d. sampling algorithm when compared Storvik's algorithm. In some cases, the improved algorithm generates an almost four-fold increase in effective sample size, and in all cases the differences are large. The reason for this should be clear. Both algorithms use the same sufficient statistics, but the exact or i.i.d. sampling algorithm resamples first using $p(y_{t+1}|x_t)$ and then propagates using the exact state posterior, conditional on $x_t$ and $y_{t+1}$. Storvik's algorithm resamples using $p(y_{t+1}|x_{t+1})$, which is much more peaked than $p(y_{t+1}|x_t)$, generating highly unbalanced weights.

Figure 1 provides a graphical view of the algorithm, displaying results from a single simulated sample path of length $T = 300$ using $N = 5000$ particles and the following parameters: $\alpha_x = 0$, $\beta = 0.98$, $\sigma_x = \sqrt{0.04} = 0.2$. These values are representative of those observed when modeling daily index returns and generate highly persistent but volatile volatility. The top panel provides the simulated returns, with clear evidence of time-varying volatility generated by the accordion pattern of increasing and decreasing volatility. The second panel show the true simulated volatility state, $v_t = exp(x_t/2)$, on a daily scale in percentages, via a thick line and the $(5, 50, 95)\,\%$ posterior quantiles of $p(v_t|y^t)$. The bottom panels track the same quantiles of the parameter posterior distributions over time.

There are a number of notable features of this simple example. First, for all of the estimates, the reduction in posterior resulting from increased data is clear. This is most

easily seen for the parameters, as the posterior bands shrink over time. The sampling variability is large, but the posterior means toward the end of the sample are still well within the bands throughout. In the first portion of the sample, volatility was relatively high, which is reflected in the parameter posteriors via a higher value of $\alpha_x$, a lower value of $\beta_x$, and a slightly higher value of $\sigma_x$. This variation in posterior means, as well as quantiles, is natural as the state variables move through their state space as more data arrives. Second, parameter learning is quite smooth, with no evidence of particle degeneracy or "stickyness" in the estimates. Particle filters that perform poorly estimating parameters often generate estimates that do not move for many time periods. Third, the algorithm estimates all of the parameters accurately over time, converging to the true parameter estimates as new data arrives.

### 3.1.2 Real data

Next, consider a stochastic volatility model of daily Nasdaq 100 stock index returns from 1999 to August 2006. This time period is interesting because it includes an extremely high variance period in the first portion of the sample and a much lower variance portion in the second half of the sample. The prior hyperparameters are $\alpha_x|\sigma_x^2 \sim \mathcal{N}(0, 0.1\sigma_x^2)$, $\beta_x|\sigma_x^2 \sim \mathcal{N}(0.7, \sigma_x^2/4)$, and $\sigma_x^2 \sim \mathcal{IG}(30, 0.725)$. These priors allow for explosive volatility, as $\beta_x$ could be greater than one and are consistent with priors used previously in the literature. The algorithm was run using $N = 5000$ particles. This example is also of interest as the size of the dataset is much larger than those considered in the simulations, roughly 1500 data points.

The parameter estimation results are given in Figures 2, in the same format as Figure 1. The filtered volatility states vary considerably over the sample, ranging from as high as 6% daily (95% annualized) during the busting of the 'Dot-Com' bubble to less then 1% daily (15% annualized) in 2006 during the quite period of the 'Great Moderation.' Again, the uncertainty over these estimates changes over time as a function of the overall level of volatility and the changing uncertainty over the parameter values.

The results also display substantial variation in the parameter estimates, most clearly in the degree of persistence, $\beta_x$, and the mean parameter, $\alpha_x$. As volatility falls throughout the sample, it also becomes more persistent. This is picked up by a generally increasing posterior mean of $\beta_x$. It is worth to note that the parameter posterior mean at the end of the sample is generally within the bands for the entire sample. Similar variation is clear

for $\alpha_x$. With more data that is less volatile, the estimates of $\alpha_x$ fall. Since the long run mean of $x_t$ is $\alpha_x/(1-\beta_x)$ (if it exists), the mean of $x_t$ does not necessarily change that much because the decrease in $\alpha_x$ coincides with a general increase in $\beta_x$. Again, there do not appear to be any degeneracy in the algorithm over time, as the posterior estimates are smooth.

Next, we consider two model specification/hypothesis testing examples, both executed sequentially through time. The first compares two competing specifications, one with normally distributed errors and the other with stochastic volatility errors. The second compares a model with a unit root in the state specification $\beta_x = 1$, with an unrestricted version. In both cases, we report the logarithm (base 10) of the Bayes factor.

The top panel of Figure 3 provides the model specification tests. The results indicate a strong preference for the SV specification, with the log-likelihood ratio declining linearly approximately linearly throughout the sample. This occurs even with the substantial uncertainty over the parameters, indicating that unknown static parameters that are slowly learned over are insufficient, on their own, to generate the observed data. The bottom panel provides the results for the tests of the unit root. Formally, we evaluate the odds of $H_0 : \beta_x = 1$ vs. $H_1 : \beta_x \neq 1$, reported on a log-scale. Odds are related to probabilities via

$$\text{Prob}(\beta_x = 1|y^t) = \frac{odds\,(H_0 : H_1|y^t)}{1 + odds\,(H_0 : H_1|y^t)}.$$

Thus, a log-odds ratio of -1.5 (-3.5) corresponds to a posterior probability of the null of roughly 16.8% and 2.9%.

At first glance, this would seem to be in conflict with the results in Figure 2, which shows that the $(5, 95)$ % posterior confidence band is slightly below 1, indicating that $\beta_x$ is statistically significantly less than 1. This is not true, however, and is just another example of Lindley's paradox, whereby $p$-values and Bayes factors give different conclusions when faced with the same data. Lindley's paradox largely arises because the t-tests, p-values, and even a Bayesian posterior confidence band for a given parameter all ignore the alternative hypothesis. Odds ratios take into account the alternative, which can lead to different conclusions when testing sharp null hypotheses.

| $\sigma_x$ | .2 | | | | .5 | | | | 1 | | | | 2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma$ | .2 | .5 | 1 | 2 | .2 | .5 | 1 | 2 | .2 | .5 | 1 | 2 | .2 | .5 | 1 | 2 |
| Exact sampling | 78 | 76 | 79 | 81 | 84 | 78 | 76 | 78 | 89 | 83 | 78 | 78 | 92 | 87 | 83 | 78 |
| Storvik | 53 | 68 | 76 | 80 | 35 | 52 | 65 | 74 | 21 | 38 | 52 | 66 | 12 | 25 | 39 | 52 |

Table 2: Effective particle size of the filtering with sequential parameter learning. Numbers are expressed in the percentage of the physical particle size. Standard errors for the mean of effective particle size are less than 3.5% from 50 simulated time-series of length 100 which are AR(1) with t-distributed observation errors with degree of freedom 3. Physical particle size is 5000. $\alpha = 0$, $\beta = 0.9$.

## 3.2 T-errors

### 3.2.1 Simulated data

Next, we consider a model with t-distributed errors. As in the previous examples, we first compare our algorithm to Storvik's and then analyze a specific simulated time series. We simulate $T$ observations assuming t-errors in the observation equation with $\nu = 3$ with $\alpha_x = 0$ and $\beta_x = 0.9$ and a range of volatilities generating different signal-to-noise ratios. We use the following prior hyperparameters: $\alpha_x|\sigma_x^2 \sim \mathcal{N}\left(0, 0.1\sigma_x^2\right)$, $\beta_x|\sigma_x^2 \sim \mathcal{N}\left(0.9, 2\sigma_x^2\right)$, $\sigma_x^2 \sim \mathcal{IG}\left(10, 0.36\right)$, and $\sigma^2 \sim \mathcal{IG}\left(10, 0.9\right)$. The algorithm was run with $N = 5000$ particles.

Table 2 provides the effective sample sizes for our algorithm and Storvik's algorithm, again, demonstrating the drastic improvement in the effective sample size of using our exact or i.i.d. sampling algorithm. The orders of magnitude are similar to those in the stochastic volatility case, and the reasoning is the same as before, although the magnitude of the difference depends on the signal-to-noise ratio. The key is that our approach uniformly dominates Storvik's approach.

Next, we simulate a single time series of length $T = 300$ assuming $\alpha_x = 0$, $\beta_x = 0.9$, $\sigma = \sqrt{0.1} \approx 0.316$, and $\sigma_x = \sqrt{0.04} = 0.2$, and implement our particle filtering algorithm with $N = 5000$ particles. The results are are summarized in Figure 4, the format matching those given earlier. The results are qualitatively similar to those in the previous section. The speed of learning varies across the parameters, with relatively more rapid learning for $\alpha_x$, $\beta_x$, and $\sigma$ than $\sigma_x$. This is due to the relatively low signal-to-noise ratio. Notably, the fat tails in the observation errors do not appear to create any problems for the algorithm.

### 3.2.2 Core PCE Inflation data

Next, we consider modeling the Federal Reserve's preferred inflation measure, the core personal consumption expenditures (PCE) price index using quarterly data from 1959 to 2006 for a total of $T = 188$ observations. The motivation for this application comes from Stock and Watson (2005) and Cecchetti, Hooper, Kasman, Schoenholtz, Stock, and Watson (2007), who posit simple univariate state space models with transitory and persistent processes.

In our setting, we consider specifications with t-errors in either the observation or the state equation and allow $\beta_x < 1$ (the previously mentioned papers constrain $\beta_x = 1$). In particular, we are interested if there is evidence for fat-tailed innovations. This is important for a number of reasons, but maybe most importantly because it is common to specify normal shocks and quadratic objective functions for central bank policy makers. The presence of non-normal shocks questions the use of these quadratic objective functions.

We posit relatively uninformative priors that are given by $\alpha_x|\sigma_x^2 \sim \mathcal{N}(0, 0.5\sigma_x^2)$, $\beta_x|\sigma_x^2 \sim \mathcal{N}(0.95, 10\sigma_x^2)$, $\sigma_x^2 \sim \mathcal{IG}(5, 0.0048)$, and $\sigma^2 \sim \mathcal{IG}(5, 0.0048)$. We assume $\nu = 5$, for both errors. The algorithm was run with $N = 5000$ particles. The results are given in Figures 5 and 6.

The filtered $x_t$ tracks the process relatively closely, as the first and second panels indicate. The filtered $x_t$ series clearly indicates that the increase in core inflation in the mid and late 1970s were due to rapid increases in the state, and not as much outliers in the observation equation. This indicates that the predictable component generates a large portion of the variability of the series. In fact, the the fact that variance decomposition

$$var(y_{t+1}|\theta) = \frac{\sigma_x^2}{1-\beta_x^2} + \sigma^2,$$

the persistent component is responsible for about 90% of the variation in core PCE, at least evaluated at posterior means at the end of the sample. This also corresponds to signal-to-noise entries in Table 2 that indicate our algorithm is vastly more efficient than Storvik's algorithm.

Parameter learning proceeds relatively smoothly, although there are rapid upward revisions in beliefs about $\beta_x$ and $\sigma_x$ in response to the large shocks in the mid 1970s. For $\beta_x$, the posterior mean is temporarily greater than 1, providing strong support for the unit root hypothesis in the 1970s. Over time, the posterior mean and bands on $\beta_x$ decrease, finishing below 1 at the end of the sample. The estimates of $\alpha_x$ and $\sigma_x$ vary over time, but

the posterior means are generally within the bands of the entire period. This indicates that there are not drastic changes in beliefs and parameter estimates, though moving around over time, are not unstable.

Figure 6 provides the Bayes factors for model comparison and the unit root tests. For model comparison, we consider three specifications: (a) normal errors in both the observation and state equation, (b) normal state errors and $t-$observation errors, and (c) normal observation errors and $t-$state errors. In the first 25 years of the sample, there is not strong evidence supporting either specification. Only after the mid 1980s, does reasonable strong evidence emerge for $t-$errors in the state equation. Regarding the unit root tests, there is increasing evidence against a unit root that has largely accumulated since the early 1980s, and this is true for each of the specifications, but the evidence is not overwhelming. The strongest results generate a Bayes factors at the end of the sample of about -1.65, which implies that the probability that $\beta_x < 1$ is about 16%. This is clear evidence against the unit root hypothesis, but is not by any means overwhelming.

Together, these results paint an interesting picture of the inflation process. There is strong evidence for large outliers in the state equation, but these shocks are extremely persistent but not permanent, since the bulk of the evidence supports values of $\beta_x$ less than one.

## 3.3 Learning in large samples

Last, we consider learning in very long time series. As discussed earlier, traditional importance sampling based algorithms for learning parameters will rapidly degenerate, as noted by Andrieu, Doucet and Tadic (2005): "with limited resources, i.e. $N$ fixed and finite, it is not possible to approximate properly the sequence of distributions" as the sample sizes increase (p. 2). This criticism does not apply to the algorithms here, because we approximate $p(\theta, s_t, x_t|y^t)$ instead of $p\left(\theta, x^T|y^T\right)$, but it does raise an interesting issue about the performance of these methods in large samples.

In large samples, the marginal distributions of the parameters and sufficient statistics (although not the states) will eventually collapse, as standard, likelihood-based asymptotics would indicate. However, this does not imply, however, that the particle filtering methods developed in this paper will not work for large samples. The asymptotics referenced earlier indicate that for any fixed $t$, that large $N$ will mitigate Monte Carlo error. The practical question then is how large does $N$ have to be for practical applications?

It is first useful to review the asymptotics for MCMC methods, where the target distribution is $p\left(\theta, x^T|y^T\right)$. Here, most theorems hold fixed the dimension of the dataset and let $N$ (the Monte Carlo sample size increase). These asymptotics indicate that convergence is generally a function of the dimensionality of the problem, the signal-to-noise ratio, and even the parameterization. With MCMC, there are convergence issues associated with the Markov Chain, and then traditional Monte Carlo error in estimating expectations. Since the asymptotics typically fix $T$, and then study how many iterations are needed, they say little regarding how $N$ should grow as $T$ increases. Few results are available for this trade-off. Simulated maximum likelihood or simulated method of moments are typically justified using similar asymptotics, assuming N is large enough to ignore Monte Carlo errors and then studying the behavior of estimators as $T$ grows. Few of these methods provide explicit computable bounds, thus there is always uncertainty over the size of the Monte Carlo samples required to provide accurate inference.

With particle filters, the issues are simpler, at least along one dimension. They are simpler because the sampling is i.i.d. with no issues associated with Markov Chain convergence. The potential problem with our particle filtering approach is one of error propagation, a problem inherent in every particle filtering algorithm. If there is any error in the state filtering distribution at time $t$, this error can translate into the sufficient statistic distribution. Errors in sufficient statistic distribution could translate into errors in the parameter posteriors, which can feed back into errors in state filtering distribution. Although the errors can propagate, theory indicates that these errors are negligible for large $N$. Again, there are not explicit computable bounds.

The easiest way to analyze this issue is via simulations. We performed a large number of simulation studies to analyze this issue. We found little evidence of any degeneracies, and the algorithms accurately estimated the parameters. As an example, we provide one such simulation example using a Gaussian latent AR(1) model with parameters that are broadly consistent with monthly or quarterly macroeconomic data: $\beta = 0.9$, $\sigma = 0.3$, and $\sigma_x = 0.3$. We simulated $T = 2500$ months of data and implemented our algorithm using $N = 10000$ particles. The results are in Figure (7). Each of the parameters is accurately estimated in large samples and there is no evidence of any degeneracies. Like MCMC, the key is minimizing the Monte Carlo error. Since the algorithms that we develop are computationally simple, it is easy to use a large number of particles in practice. This example shows the accuracy of the methods, even in extremely large samples.

# 4 Conclusions

In this paper, we provide an exact sampling algorithm for performing sequential parameter learning and state filtering for nonlinear, non-Gaussian state space models. The implication of this is that we do not resort to importance sampling, and thus avoid the well known degeneracies associated with sequential importance sampling methods. Formally, the only assumption we require is that the parameter posterior admits a sufficient statistic structure. We analyze the class of linear non-Gaussian models in detail, and exact state filtering is a special case of our algorithm. Thus, we provide an exact sampling alternative to algorithms such as the auxiliary particle filter of Pitt and Shephard (1999) and mixture Kalman filter of Chen and Liu (2000) We provide both simulation and real data examples to document the efficacy of the approach.

We are currently investigating a number of extensions and issues related to particle filters for parameter learning. First, Johannes and Polson (2007) and Johannes, Polson, and Yae (2007) consider multivariate extensions and extensions to robust state space models, those errors that coincide with popular quantile objective functions. Second, Carvalho, Johannes, Lopes, and Polson (2008) provide detailed comparisons between the algorithm developed here and other algorithms, such as MCMC or Liu and West (2001). The results indicate that the approach developed in this paper is as accurate as MCMC, for a fraction of the computational cost, and uniformly dominates Liu and West (2001), at least for conditionally Gaussian models. Additionally, this paper shows how to compute marginal smoothing distributions, $p\left(x^T|y^T\right)$ using a variant of the FFBS algorithm (Frühwirth-Schnatter (1994), Carter and Kohn (1994, 1996) or Shephard (1994)) and how to extend the idea of sufficient statistics to state variables. Finally, we are studying the behavior of particle approximations in settings with large $T$, to understand how to increase $N$ as $T$ increases.

# 5 References

Andrieu, C., N. de Freitas, A. Doucet, 1999, Sequential MCMC for Bayesian Model Selection. IEEE Signal Processing Workshop on Higher Order Statistics. Ceasarea, Israel, June 14-16.

Andrieu, C., A. Doucet, and V.B. Tadic, 2005, Online simulation-based methods for pa-

rameter estimation in non linear non Gaussian state-space models, *Proceedings of the 44th Conference on Decision and Control*.

Carlin, B., N.G. Polson, and D. Stoffer, 1992, A Monte Carlo Approach to Nonnormal and Nonlinear State-Space Modeling, *Journal of the American Statistical Association*, 87, 493-500.

Carpenter, J., P. Clifford, and P. Fearnhead, 1999, An Improved Particle Filter for Nonlinear Problems. *IEE Proceedings – Radar, Sonar and Navigation*, 1999, 146, 2–7.

Carter, C.K., and R. Kohn, 1994, On Gibbs Sampling for State Space Models, *Biometrika*, 81, 541-553.

Carter, C.K., and R. Kohn, 1996, Markov chain Monte Carlo in conditionally Gaussian state space models, *Biometrika* 83, 589-601.

Carvalho, C., M. Johannes, H. Lopes and N.G. Polson, 2008, Particle Learning and Smoothing. *Working Paper*.

Cecchetti, Stephen G., Peter Hooper, Bruce C. Kasman, Kermit L. Schoenholtz, and Mark W. Watson, 2007, Understanding the Evolving Inflation Process, paper prepared for the U.S. Monetary Policy Forum 2007.

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

Chopin, N., 2004, Central Limit Theorem for sequential Monte Carlo methods and its application to Bayesian inference, *Annals of Statistics* 32:6, 2385-2411.

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

Doucet, A., N. de Freitas, and N. Gordon, 2001, *Sequential Monte Carlo Methods in Practice*, New York: Springer-Verlag, Series Statistics for Engineering and Information Science.

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.

Gilks, W. R. and C. Berzuini, C., 2001, Following a moving target: Monte Carlo inference for dynamic Bayesian models. *J. R. Statist. Soc. B* 63, 127–46.

Gordon N. and A. Smith, 1993, Approximate non-Gaussian Bayesian estimation and modal consistency, *Journal of the Royal Statistical Association, Series B,* 913-918.

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

Han, C. and B. Carlin, 2001, Markov Chain Monte Carlo Methods for Computing Bayes Factors: A Comparative Review, *Journal of the American Statistical Association* 96, p. 1122-1132.

Jacquier, E., N.G. Polson, and P. Rossi, 1994, Bayesian analysis of Stochastic Volatility Models, (with discussion). *Journal of Business and Economic Statistics* 12(4), 371-89.

Johannes, M., and Polson, N.G., 2007, Multivariate sequential parameter learning and state filtering, working paper, University of Chicago.

Johannes, M., Polson, N.G., and J. Stroud, 2008, Optimal Filtering of Jump Diffusions: Extracting Latent States from Asset Prices, forthcoming, *Review of Financial Studies*.

Johannes, M., Polson, N.G., and S. Yae, 2007, Robust sequential parameter learning and state filtering, working paper, University of Chicago.

Kim, S., N. Shephard and S. Chib, 1998, Stochastic volatility: likelihood inference and comparison with ARCH models, *Review of Economic Studies* 65, 361-93.

Klaas, M., N. de Frietas and A. Doucet, 2005. Toward a practical $N^2$ Monte Carlo filter: the marginal particle filter. In *Proc of Artifical Intelligence*.

Liu, J. and Chen, R., 1998, Sequential Monte Carlo Methods for Dynamical Systems. *Journal of the American Statistical Association*, 93, 1032-1044.

Liu, J. and M. West, 2001, Combined parameter and state estimation in simulation -based filtering, in *Sequential Monte Carlo Methods* in Practice, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York: SpringerVerlag, 197-217.

Masreliez, C.J. and R.D. Martin, 1977, Robust Bayesian Estimation for the linear model and robustifying the Kalman filter, *IEEE Transactions on Automatic Control* 22, 361-371.

Omori, Y., S. Chib, N. Shephard, and J. Nakajima, 2006, Stochastic Volatility with Leverage: Fast and Efficient Likelihood Inference, forthcoming *Journal of Econometrics*.

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

Poyadjis, G., A. Doucet, and S.S. Singh, 2005, Maximum Likelihood Parameter Estimation using Particle Methods, working paper.

Shephard, N. 1994, Partial non-Gaussian time series models, *Biometrika* 81, 115-31.

Stock, J. and M. Watson, 2005, Has inflation become harder to forecast?, "Has Inflation Become Harder to Forecast," working paper.

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

West, M. (1981) Robust Sequential Approximate Bayesian Estimation. *Journal of Royal Statistical Society, B.*, 43(2), 157-166.

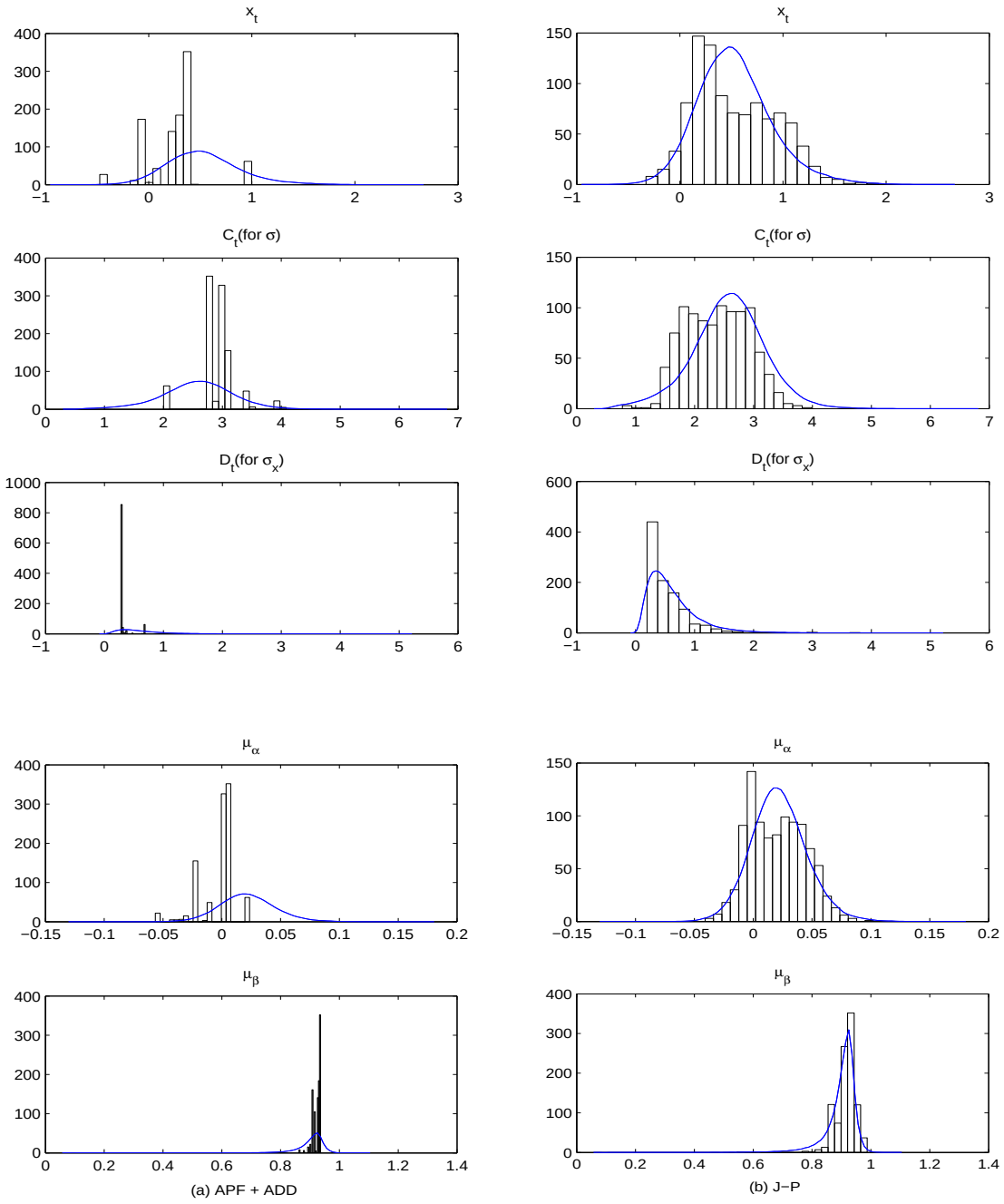


Figure 1: Comparison between (a) ADD's algorithm with auxiliary variable and (b) JP's. This figures provides the particle posterior approximation of sufficient statistics with 1,000 particles at the moment of a huge shock. Histograms of each algorithm's sample are normalized to be compared with MCMC posterior densities (Blue line) with 100,000 samples after 10,000 burn-in.

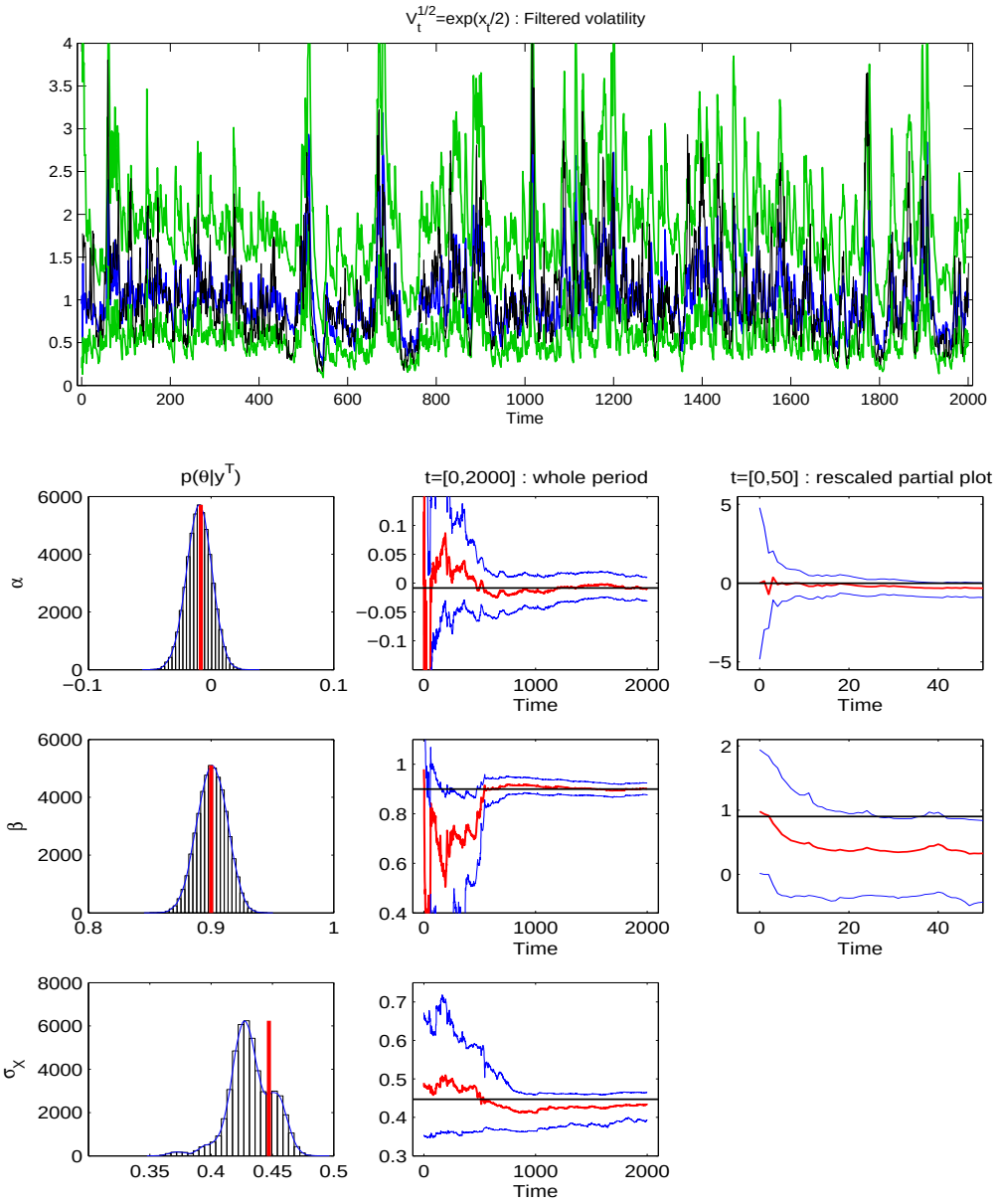


Figure 2: The top panel plots the observed time series, $y_t$, simulated from the stochastic volatility model. The second panel plots the true simulated $x_t$ series (thick line) as well as the $(5, 50, 95)$ posterior quantiles of $p\left(x_t|y^t\right)$. Three bottom panels display sequential summarizes of the parameter posterior, $p\left(\theta|y^t\right)$. The horizontal line denotes the true parameters used in simulation.

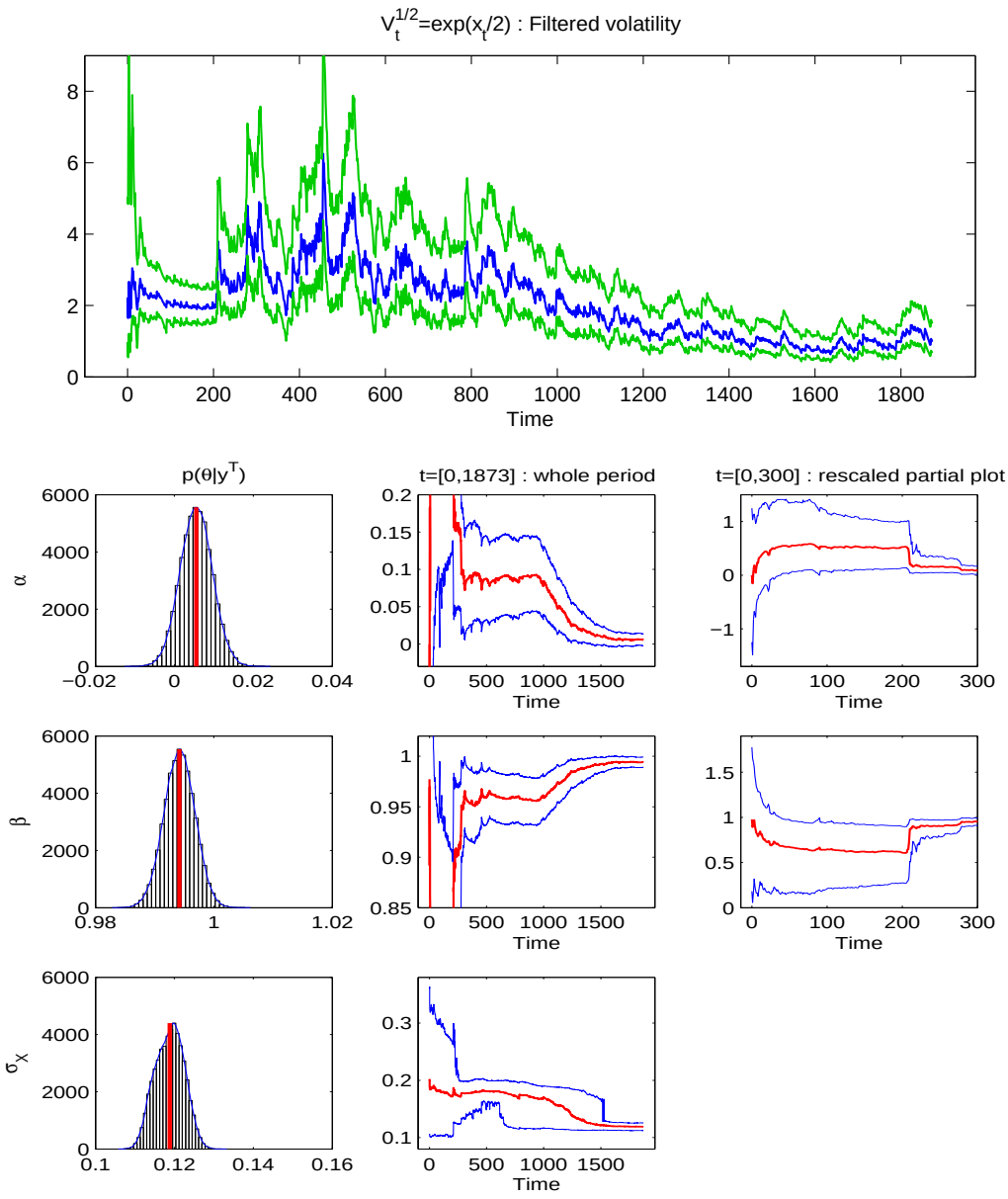


Figure 3: The top panel plots the observed time series, $y_t$, of Nasdaq 100 stock returns. The second panel plots the filtered stochastic volatility state (thick line) as well as the $(5, 50, 95)$ posterior quantiles of $p\left(x_t|y^t\right)$. Three bottom panels display sequential summarizes of the parameter posterior, $p\left(\theta|y^t\right)$. $N = 5000$ particles were used.

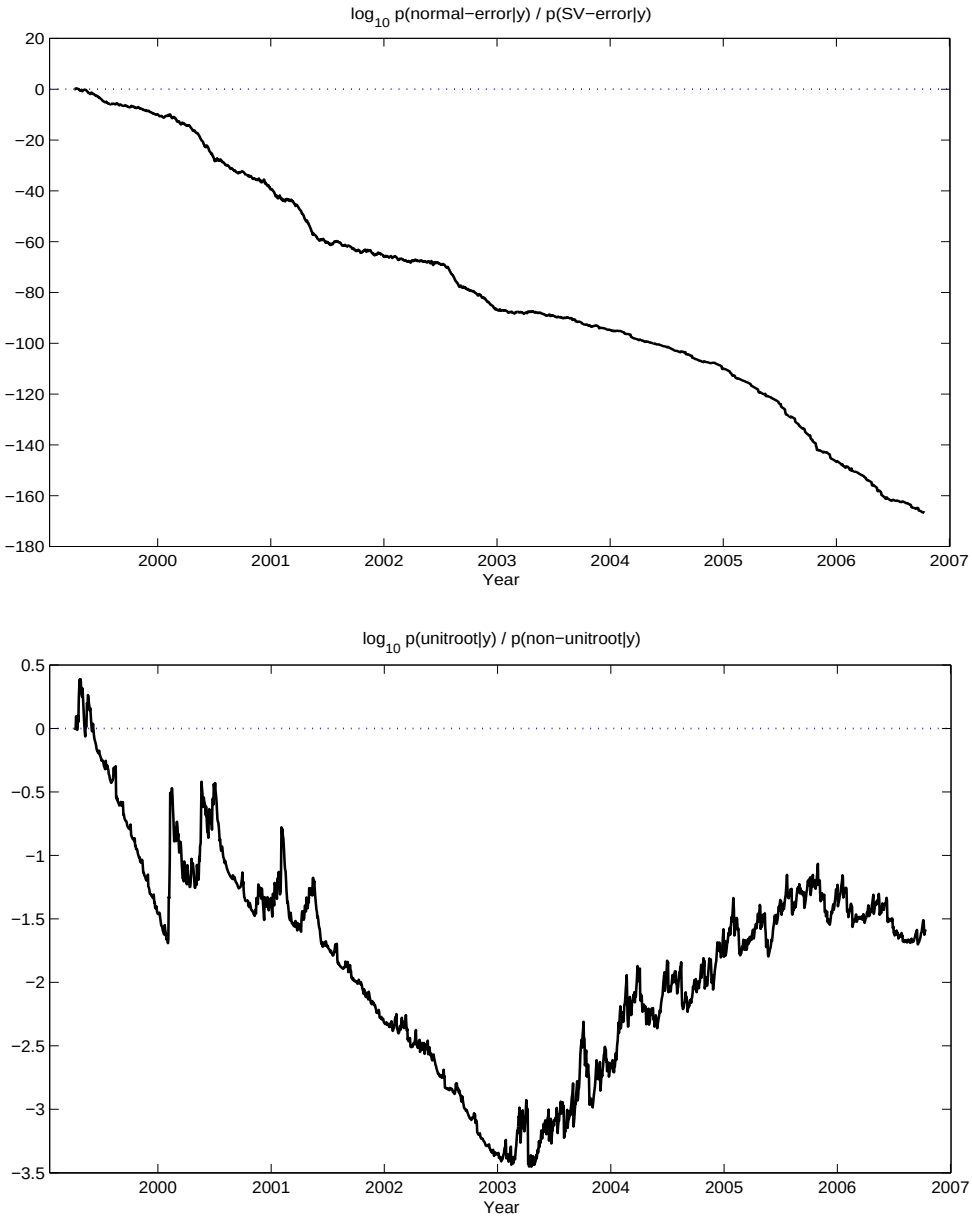


Figure 4: Sequential model choice example with log scale. Nasdaq 100 daily stock return data are fitted by the SV model. Top panel: Posterior Odds of normal error versus SV-error, $\frac{p(normal\ error|y)}{p(SV\text{-}error|y)}$. Bottom panel: Posterior Odds of unitroot versus non-unitroot, $\frac{P(unitroot|y)}{P(non\text{-}unitroot|y)}$.

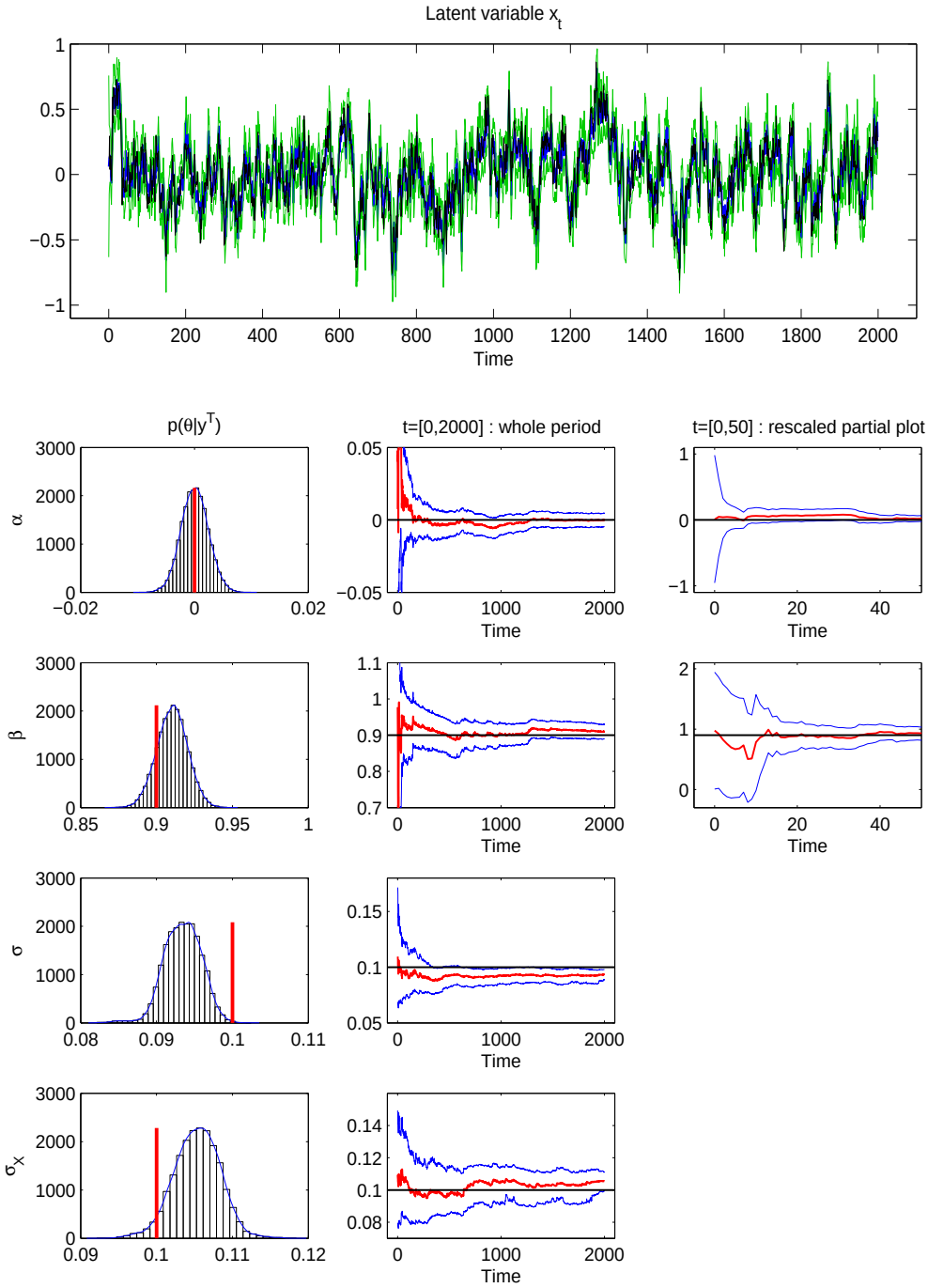


Figure 5: The top panel plots the observed time series, $y_t$, simulated from the t-distributed AR(1) model. The second panel plots the true simulated $x_t$ series (thick line) as well as the $(5, 50, 95)$ posterior quantiles of $p\left(x_t|y^t\right)$. Four bottom panels display sequential summarizes of the parameter posterior, $p\left(\theta|y^t\right)$. The horizontal line denotes the true parameters used in simulation.

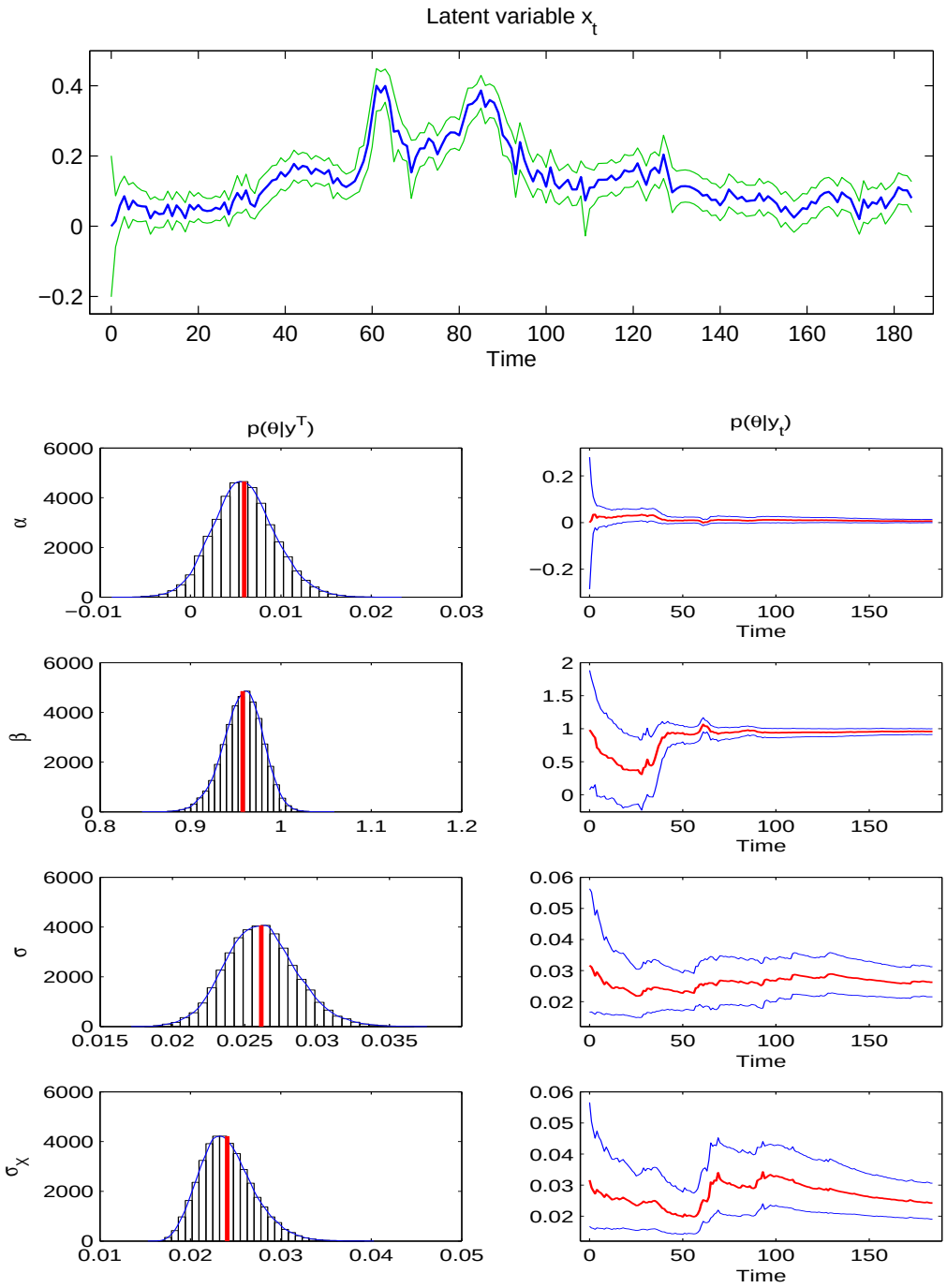


Figure 6: The top panel plots the observed time series, $y_t$, of PCE inflation series. The second panel plots the filtered $x_t$ series (thick line) as well as the $(5, 50, 95)$ posterior quantiles of $p\left(x_t|y^t\right)$. Four bottom panels display sequential summarizes of the parameter posterior, $p\left(\theta|y^t\right)$. AR(1) model with the t-distributed error is used.

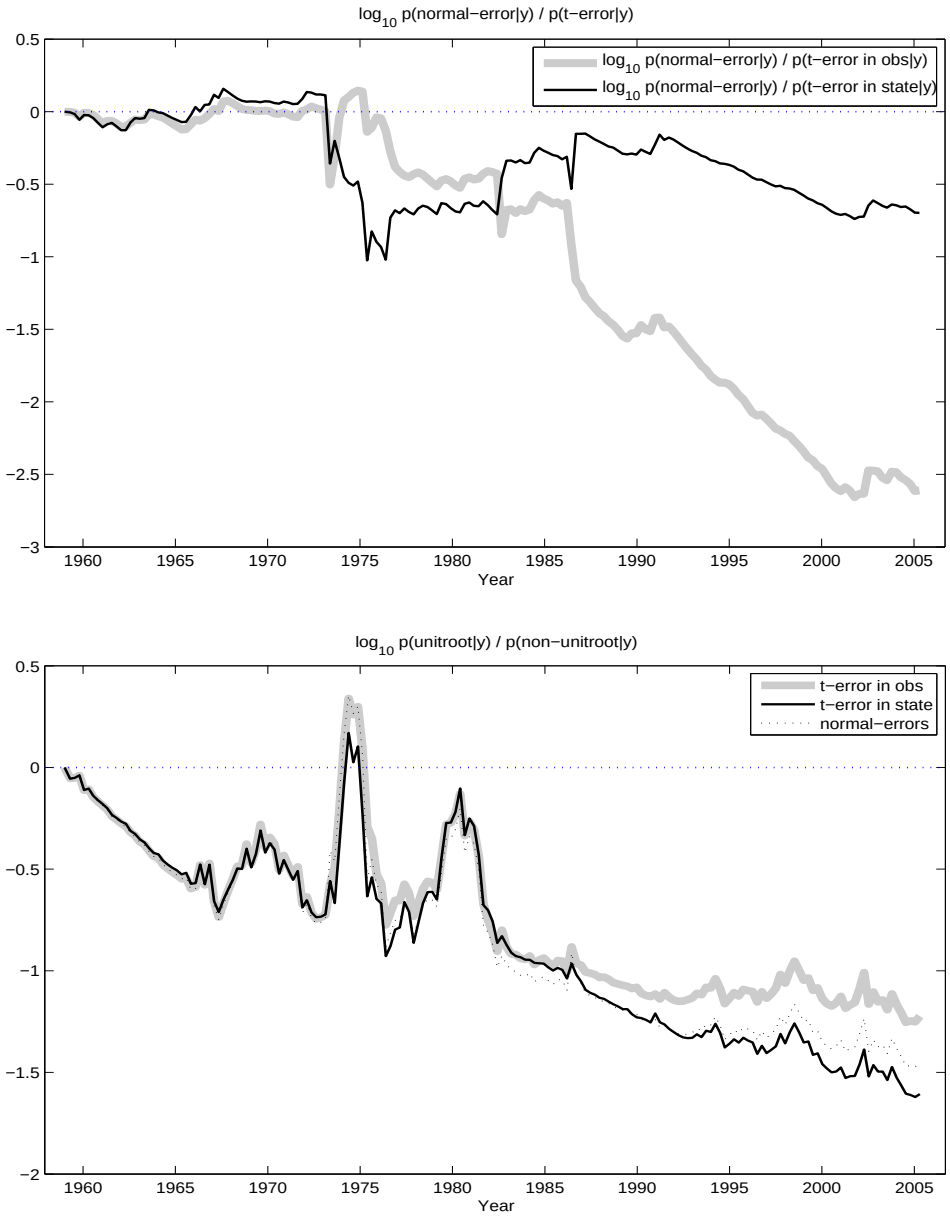


Figure 7: Sequential model choice example with log scale. PCE inflation data are fitted by the AR(1) with observation errors. Top panel: Posterior Odds of normal error versus t-error, $\frac{p(normal\ error|y)}{p(t\text{-}error|y)}$. The thick gray line is the case t-error is in the observation equation and the black line is in the state equation. Bottom panel: Posterior Odds of unitroot versus non-unitroot, $\frac{p(unitroot|y)}{p(non\text{-}unitroot|y)}$. The thick gray line is the case t-error is in the observation equation, the black line is in the state equation, and the dash line is normal error case.