1 State Space Models

State space models have wide applications in applied statistics, signal processing and time series analysis. Formally, a state space model is constructed by a transition equation and a measurement equation. The transition equation describes the prior distribution of a hidden Markov state process $\{x_t\}_{t\geq 0}$ while the likelihood of observations are derived by the measurement equation. It is often assumed that the latent process is Markovian so that x_t is independent of the history of states and measurements if the previous state x_{t-1} is known.

$$p(x_t|x_{0:t-1}, y_{1:t-1}) = p(x_t|x_{t-1}, x_{0:t-2}, y_{1:t-1}) = f_\theta(x_t|x_{t-1}) \tag{1.1}$$

where x_0 follows an initial prior density $\mu_0(x)$. The observation is also assumed to be independent of the history of states and previous measurements though this is not always the case.

$$p(y_t|x_{0:t}, y_{1:t-1}) = p(y_t|x_t, x_{0:t-1}, y_{1:t-1}) = g_\theta(y_t|x_t) \tag{1.2}$$

In equations above, $z_{i:j}$ denotes sequence $\{z_i, z_{i+1}, \ldots, z_j\}$ and θ is the parameter vector of the model. In next notations, we use z^k to represent the consecutive components of a sequence $\{z_t\}$ that start from the initial one and end with z_k. For example, y^t equally denotes the sequence of $y_{1:t}$.

In many situations, we are interested in the posterior distribution of state variables when observations are obtained. Except in special cases such as linear Gaussian and hidden finite state space Markov chains, it is impossible to evaluation the posterior distribution analytically. MC simulations are a class of methods that can help estimate the posterior distributions. In the next sections, we will introduce how to solve filtering, smoothing and parameter learning problems in state space models by sequential Monte Carlo simulation which is also called particle filtering. [1].

2 Particle Filtering

Particle filtering methods are implemented to estimate posterior distribution $p_\theta(x_{1:t}|y_{1:t})$ or $p(x_t|y_{1:t})$ and likelihood $p_\theta(y_{1:t})$.

2.1 Exact Particle Filter (EPF)

The easiest way to understand particle filtering is to consider situations where importance sampling is not required. Exact sampling, which leads to the exact particle filter[10], is accessible because direct sampling from $\widehat{p}(x_{t+1}|y_{t+1})$ is feasible. To explain how it works, we firstly have

$$p(y_{t+1}, x_{t+1}|x_t) \propto p(y_{t+1}|x_t)p(x_{t+1}|x_t, y_{t+1}) \tag{2.1}$$

[1] Except for parameter learning problems, we assume all parameters in the model are known and fixed

which indicates that the filtering recursion can be constructed as

$$\begin{aligned} p(x_{t+1}|y_{1:t+1}) &= \int p(x_{t+1}, x_t|y_{1:t}, y_{t+1})dx_t \\ &\propto \int p(x_{t+1}, x_t, y_{1:t}, y_{t+1})dx_t \\ &\propto \int p(y_{t+1}, x_{t+1}|x_t, y_{1:t})p(x_t|y_{1:t})dx_t \\ &= \int p(y_{t+1}, x_{t+1}|x_t)p(x_t|y_{1:t})dx_t \\ &\propto \int p(y_{t+1}|x_t)p(x_{t+1}|x_t, y_{t+1})p(x_t|y_{1:t})dx_t \end{aligned} \tag{2.2}$$

where $p(y_{t+1}|x_t) = \int p(y_{t+1}|x_{t+1})p(x_{t+1}|x_t)d\,x_{t+1}$ is the predictive likelihood and $p(x_{t+1}|x_t, y_{t+1})$ is the posterior distribution of the new state given the previous state and the new observation. Then, given a particle approximation of to $\widehat{p}(x_t|y_{1:t})$, estimation is updated as

$$\begin{aligned} \widehat{p}(x_{t+1}|y_{1:t}) &\propto \sum_{i=1}^{N} p(y_{t+1}|x_t^{(i)})p(x_{t+1}|x_t^{(i)}, y_{t+1}) \\ &= \sum_{i=1}^{N} q_t^{(i)} p(x_{t+1}|x_t^{(i)}, y_{t+1}) \end{aligned} \tag{2.3}$$

where the normalized first stage weights are $q_t^{(i)} = \dfrac{p(y_{t+1}|x_t^{(i)})}{\sum_{i=1}^{N} p(y_{t+1}|x_t^{(i)})}$.

It is noted that $\widehat{p}(x_{t+1}|y_{1:t+1})$ is a standard discrete mixture distribution as the first stage weights are known when the new observation y_{t+1} reaches. Sampling from a discrete mixture distribution is straightforward by first selecting the mixture index and then secondly propagate new particles from that chosen mixture distribution. Given that N particles $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ are available at time t and the new observation y_{t+1} arrives, the goal is to update the particle system to get $\{x_{t+1}^{(i)}, w_{t+1}^{(i)}\}_{i=1}^N$ and $\hat{p}(x_{t+1}|y_{1:t+1})$.

The **preconditions** of **EPF** are: in **step 1.1**, the predictive likelihood $p(y_{t+1}|x_t)$ can be computed and $p(x_{t+1}|x_t, y_{t+1})$ in **step 1.2** can be sampled. The EPF algorithm is listed as below:

step 0 Draw N particles $x_0^{(i)} \sim u(x_0)$ and set $\omega_0^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.

step 1 For each time step, given the new observation, repeat the following steps:

1.1 (Re-sampling) With new observation y_{t+1}, draw $\{z(i)\}_{i=1}^N \sim \text{Mult}_N(\{q_t^{(s)}\}_{s=1}^N)$ where $q_t^{(s)} = \frac{p(y_{t+1}|x_t^{(s)})}{\sum_{j=1}^N p(y_{t+1}|x_t^{(j)})}$. Set $\tilde{x}_t^{(i)} = x_t^{z(i)}$.

1.2 (Propagating) Draw $x_{t+1}^{(i)} \sim p(x_{t+1}|\tilde{x}_t^{(i)}, y_{t+1})$ for $i = 1, \ldots, N$. Set $w_{t+1}^{(i)} = \frac{1}{N}$.

A simple example to show how **EPF** perform is the *linear Gaussian* model in which $p(y_t|x_t) \sim N(x_t, \sigma_y^2)$ and $p(x_{t+1}|x_t) \sim N(ax_t, \sigma_x^2)$. The exact filtering algorithm requires that $p(y_{t+1}|x_t)$ and $p(x_{t+1}|x_t, y_{t+1})$ are available. In this simple *linear Gaussian* model, it is easy to obtain

that

$$p(y_{t+1}|x_t) \sim N(ax_t, \sigma_x^2 + \sigma_y^2)$$
$$p(x_{t+1}|x_t, y_{t+1}) \propto p(y_{t+1}|x_{t+1})p(x_{t+1}|x_t) \sim N(\mu_{t+1}, \sigma_{t+1}^2)$$

where $\frac{1}{\sigma_{t+1}^2} = \frac{1}{\sigma_x^2} + \frac{1}{\sigma_y^2}$ and $\frac{\mu_{t+1}}{\sigma_{t+1}^2} = \frac{y_{t+1}}{\sigma_y^2} + \frac{ax_t}{\sigma_x^2}$.

Assuming $(a, \sigma_x, \sigma_y) = (0.8, 0.4, 0.9)$, we simulate 500 state variables, $\{x_t\}_{t=1}^{500}$, and the corresponding observations, $\{y_t\}_{t=1}^{500}$ from $x_0 = 2$. In the first step of **EPF**, particles $\{x_0^{(i)}\}_{i=1}^{N}$ are drawn from $N(y_1, 3)$ and the number of particles is set as $N = 2000$[2].

The results of **EPF** are displayed in Figure 1, in which the red dash line denotes the true states and the blue solid line represents the filtered estimates $E[x_t|y_{1:t}]$. The lower bound and upper bound of the shade are estimates of the 5% and 95% quantiles respectively.

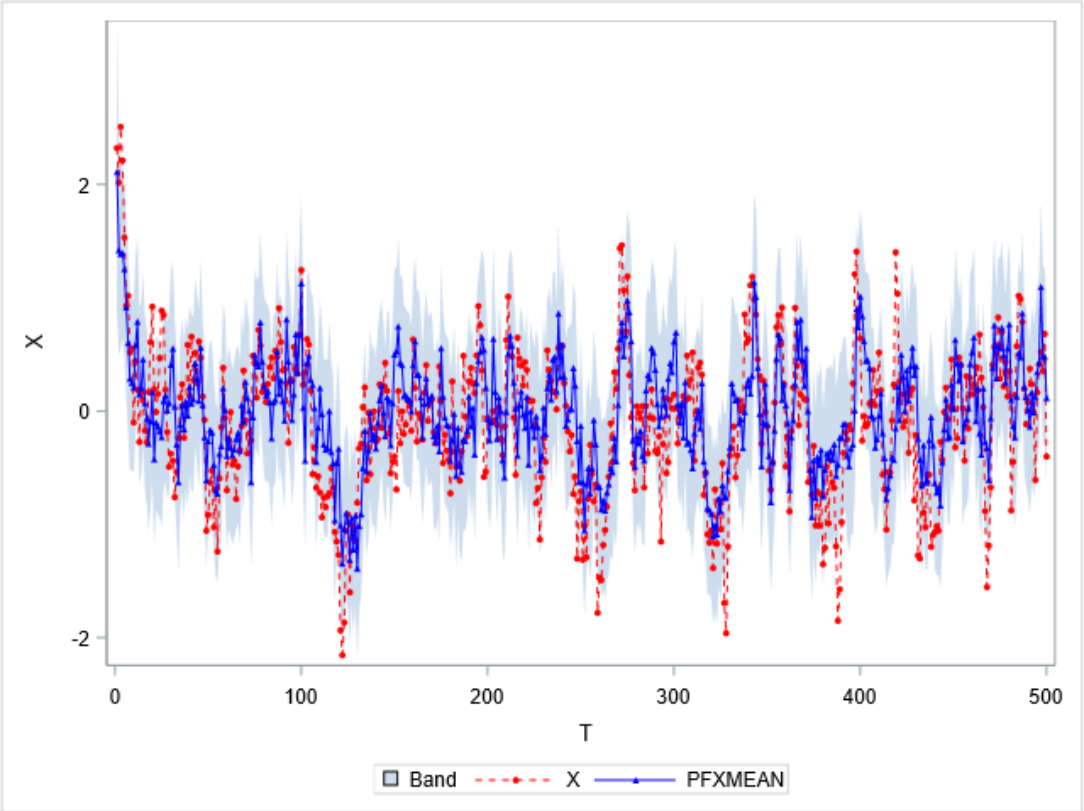

Figure 1: **EPF** distribution for 500 simulated observations, *linear Gaussian* model

2.2 Sequential Importance Sampling(SIS)

As a sequential version of importance sampling, SIS can generate importance sampling approximations to filtering distributions of generic state space models of the following form

$$x_t \sim f(x_t|x_{t-1})$$
$$y_t \sim g(y_t|x_t)$$

[2]For the sake of simplicity, number of particles in filtering examples is always 2000.

where x_t is the state at time step t and y_t is the observation or measurement. The state and measurements may contain both discrete and continuous components. With samples from an importance distribution and their weights $\{\omega_t^{(i)}, x_t^i\}, i = 1, \ldots, N$, the filtering distribution $p(x_t|y^t)$ can be approximated by

$$\hat{p}(x_t|y^t) \approx \sum_{i=1}^{N} \omega_t^{(i)} \mathbb{1}_{\{x_t = x_t^{(i)}\}} \tag{2.4}$$

Then at time step t, the approximation to the expected value of an arbitrary function $g(\cdot)$ can be calculated as the weighted sample average

$$E[g(x_t)|y^t] \approx \sum_{i=1}^{N} \omega_t^{(i)} g(x_t^{(i)}) \tag{2.5}$$

We have the recursion for the posterior distribution

$$p(x_t|y^t) \propto p(x_{t-1}|y^{t-1}) g(y_t|x_t) f(x_t|x_{t-1}) \tag{2.6}$$

If the importance distribution for state x_t is formed recursively as

$$\pi(x_t|y^t) = \pi(x_t|x_{t-1}, y_t)\pi(x_{t-1}|y^{t-1}) \tag{2.7}$$

then the corresponding weights can be acquired by

$$\omega_t^{(i)} \propto \frac{g(y_t|x_t^{(i)}) f(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t^{(i)}|x_{t-1}^{(i)}, y_t)} \frac{p(x_{t-1}^{(i)}|y^{t-1})}{\pi(x_{t-1}^{(i)}|y^{t-1})} \tag{2.8}$$

Assuming we have already drawn $x_{t-1}^{(i)}$ from the importance distribution $\pi(x_{t-1}|y^{t-1})$ and have computed their weights $\omega_{t-1}^{(i)}$, then the new state samples, $x_t^{(i)}$, can be drawn from $\pi(x_t|x_{t-1}^{(i)}, y_t)$. The optimal importance density, in terms of minimizing the variance of importance weights, is $p(x_t|x_{t-1}, y_t)$. Moreover, the importance weights from the previous step, $\omega_{t-1}^{(i)}$, are proportional to $\frac{p(x_{t-1}^{(i)}|y^{t-1})}{\pi(x_{t-1}^{(i)}|y^{t-1})}$ and thus the weights satisfy the following recursion

$$\omega_t^{(i)} \propto \frac{g(y_t|x_t^{(i)}) f(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t^{(i)}|x_{t-1}^{(i)}, y_t)} \omega_{t-1}^{(i)} \tag{2.9}$$

Now, the generic **SIS** can be shown as below:

step 0 Draw N samples $x_0^{(i)} \sim p(x_0)$ and set $\omega_0^{(i)} = \frac{1}{N}$, for $i = 1, \ldots, N$.

step 1 For each time step, given the new observation, we need to

- (Propagating) Draw $x_{t+1}^{(i)}$ from the importance distribution $\pi(x_{t+1}|x_t^{(i)}, y_{t+1})$, $i = 1, \ldots, N$.
- Update weights as $\omega_{t+1}^{(i)} \propto \frac{g(y_{t+1}|x_{t+1}^{(i)}) f(x_{t+1}^{(i)}|x_t^{(i)})}{\pi(x_{t+1}^{(i)}|x_t^{(i)}, y_{t+1})} \omega_t^{(i)}$ and then normalize them to unity.

One **problem** in **SIS** algorithm is that it is easy to encounter a situation called *degeneracy problem* where almost all particles have nearly zero weights. Such problem can be solved by a re-sampling procedure. It refers to a procedure where we draw N new samples from the discrete distribution based on the weights and replace the old set of samples with the new one. The re-sampling procedure can be described by

step 1 Draw $\{z(i)\}_{i=1}^{N} \sim \text{Mult}_N(\{\omega_t^{(s)}\}_{s=1}^{N})$ and set $x_t^{(i)} = x_t^{z(i)}$.

step 2 Set all weights $\omega_t^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.

The objective of re-sampling procedure in **SIS** is to remove particles with small weights and duplicate particles with large weights. Despite unchanged distribution for the particles, re-sampling will introduce addition variance to estimates. A proper choice of re-sampling methods like stratified re-sampling (Kitagawa, 1996)[12] can reduce such additional variance. Stratified re-sampling method is described as

step 1 Generate N ordered random numbers by $u_k = \frac{(k-1)+\tilde{u}_k}{N}$ where $\tilde{u}_k \sim \text{unif}[0, 1]$.

step 2 Allocate n_i copies of the particle $x_t^{(i)}$ to the new distribution. We have $n_i = \#u_k \in (\sum_{s=1}^{i-1} \omega_t^{(s)}, \sum_{s=1}^{i} \omega_t^{(s)}]$, where $\omega_t^{(s)}$ is the weight of sth particle before re-sampling.

step 3 Set all weights $\omega_t^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.

2.2.1 Sequential Importance Re-sampling (SIR)

Adding a re-sampling step to **SIS** leads to **SIR**(Gordon et al.,1993; Kitagawa, 1996; Doucet et al., 2001; Ristic et al., 2004). In **SIR**, re-sampling is not performed in each step but performed only when it is needed. One way to implement this strategy is to do re-sampling every n steps, where n is a predefined constant. In addition, we can apply *adaptive re-sampling* in which "effective" number of particles (Liu and Chen, 1995) helps judge whether a re-sampling is needed in that step. When sample size N is large enough, the effective number of particles at time step t, which is closely related to the variance of particle weights, can be computed from particle weights as

$$N_{\text{eff}} = \frac{N}{1 + \frac{1}{N}\sum_{j=1}^{N}(N\omega_t^{(i)} - 1)^2} \approx \frac{1}{\sum_{j=1}^{N}(\omega_t^{(i)})^2} \tag{2.10}$$

Re-sampling is performed when N_{eff} is significantly less than the total number of particles N, for example, $N_{\text{eff}} < \frac{N}{10}$.

The algorithm of **SIR** is almost same as the one of **SIS** except that a re-sampling should be performed if needed at the end of **step 1** for each non-zero time step. There still exist some **problems** embedded in **SIR**. Firstly, weight degeneracy problem may be just hidden rather than fixed. Moreover, the tails of $p(x_t|y^t)$ are usually poorly approximated by $\sum_{i=1}^{N} \omega_t^{(i)} \mathbb{1}_{\{x_t = x_t^{(i)}\}}$ because of the use of mixture approximation. This problem holds for particle filters in general.

Bootstrap Filter (BF) The bootstrap filter (Gordon et al., 1993)[9] is a special case of SIR method where $p(x_t|x_{t-1})$ is used as the importance distribution. Such setting makes the implementation of the algorithm very easy, however, it may require a very large number of Monte Carlo samples for accurate estimation results due to the inefficiency of that special importance distribution. In **BF** the re-sampling is normally done at each time step. **BF** algorithm can be shown as below:

step 0 Draw N samples $x_0^{(i)} \sim p(x_0)$ and set $\omega_0^{(i)} = \frac{1}{N}$, for $i = 1, \ldots, N$.

step 1 For each time step, given the new observation, we should

- (Propagating) Draw $x_{t+1}^{(i)}$ from $f(x_{t+1}|x_t^{(i)})$, $i = 1, \ldots, N$.
- Update weights as $\omega_{t+1}^{(i)} \propto p(y_{t+1}|x_{t+1}^{(i)})$ for $i = 1, \ldots, N$ and normalize them to unity.
- Do re-sampling and set the weights of all new particles to be $\frac{1}{N}$.

In addition to the *linear Gaussian* model, we also explore the bootstrap filtering results of the following *stochastic volatility* model:

$$X_{t+1} = \phi X_t + \sigma V_t$$
$$Y_t = \beta \exp(X_t/2) U_t$$

The parameter vector is set as $\theta = (\phi, \sigma, \beta) = (0.8, 0.9, 1)$. Observations are generated by simulation from $x_0 = 0.5$. Again, we draw $\{x_0\}$ from $N(y_1, 3)$.

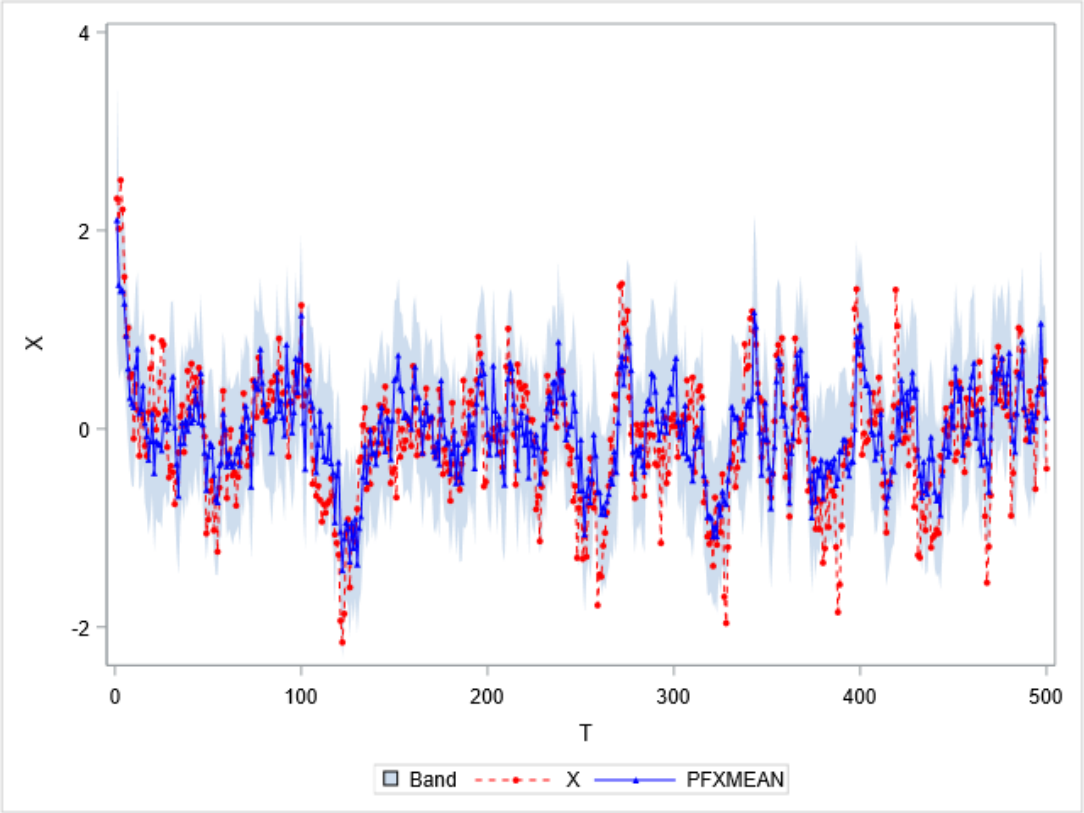

Figure 2: **BF** distribution for 500 simulated observations, *linear Gaussian* model

It is also worth noting that in **BF** the states are drawn from the prior distribution, $p(x_{t+1}|x_t)$, without considering the new observation y_{t+1}. Then the simulated states may not be in high likelihood regions. When there is an outlier new observation, the weights for the particles will be very unevenly distributed especially when the measurement density $f(y_t|x_t)$ is quite sensitive to x_t. Auxiliary particle filtering (Pitt, Shephard, 1999)[16] provides a more efficient sampling method for the approximated posterior distribution, $\hat{p}(x_t|y^t)$. We will discuss that method in next section.

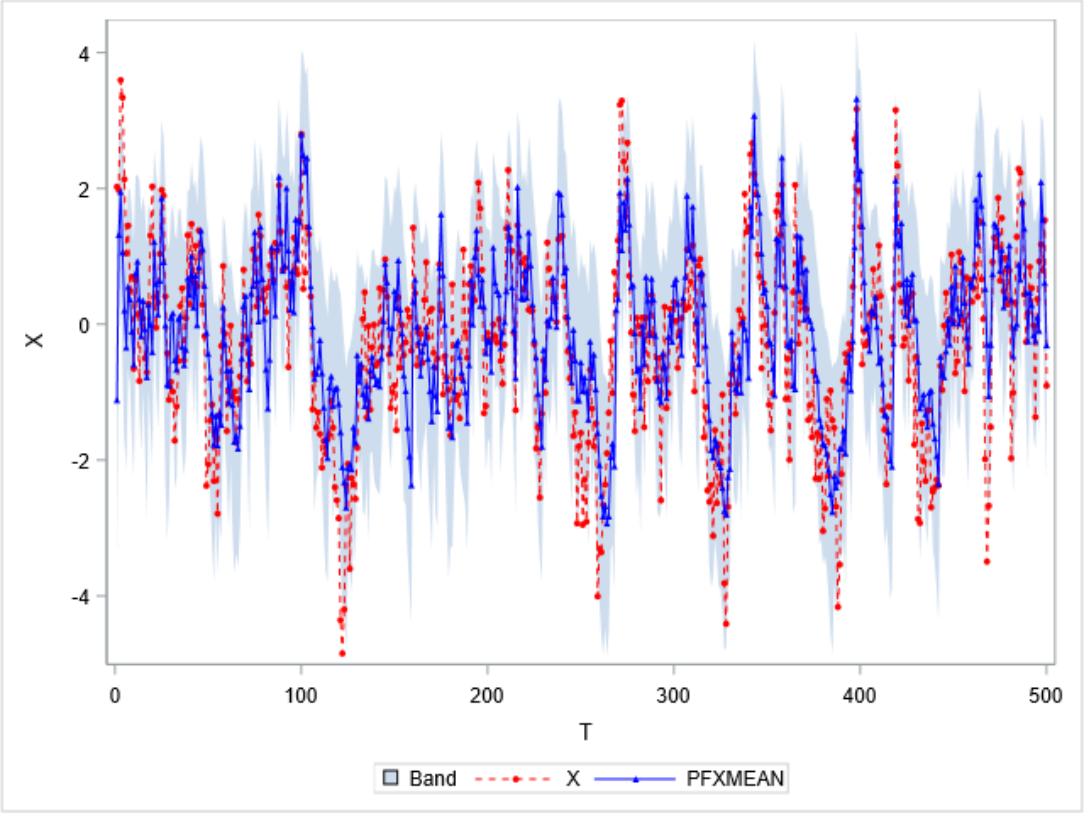

Figure 3: **BF** distribution for 500 simulated observations, *Stochastic Volatility* model

2.2.2 Auxiliary Particle Filtering (APF)

Introducing the particle index into the posterior distribution, **APF** gives a new sampling method. Firstly, let us define the following joint distribution by

$$p(x_{t+1}, k|y^{t+1}) \propto p(y_{t+1}|x_{t+1})p(x_{t+1}|x_t^{(k)})\omega_t^{(k)}, \quad k = 1, \ldots, N \tag{2.11}$$

where k is the index of particles. We call k an auxiliary variable as it assists us in sampling. With the idea of **SIR**, we can generate N new particles $\{x_{t+1}^{(i)}, k^{(i)}\}$ from a importance distribution $\pi(x_{t+1}, k|y^{t+1})$. To avoid blind sampling and make weights even, such $\pi(\cdot)$ should depend on the new observation, y_{t+1}, and kth particle at time t, $x_t^{(k)}$.

A convenient choice of importance distribution is designed as

$$\pi(x_{t+1}, k|y^{t+1}) \propto g(y_{t+1}|\mu_{t+1}^k)f(x_{t+1}|x_t^{(k)}) \tag{2.12}$$

where μ_{t+1}^k is the conditional expected value, the mode, a draw or other likely values associated with the density $p(x_{t+1}|x_t^{(k)})$. When applying this importance distribution, we need to determine the index k at first by

$$\begin{aligned}
\pi(k|y^{t+1}) &\propto \omega_t^{(k)} \int g(y_{t+1}|\mu_{t+1}^{(k)})f(x_{t+1}|x_t^{(k)})d\,x_{t+1} \\
&= \omega_t^{(k)} g(y_{t+1}|\mu_{t+1}^{(k)}) \int f(x_{t+1}|x_t^{(k)})d\,x_{t+1} \\
&= \omega_t^{(k)} g(y_{t+1}|\mu_{t+1}^{(k)})
\end{aligned}$$

where $\omega_t^{(k)}$ denotes the weight assigned to $x_t^{(k)}$ again. With determined particle index k, we can sample a new state variable, x_{t+1}^k from the transition distribution $p(x_{t+1}|x_t^{(k)})$ with corresponding weight $\omega_{t+1}^{(k)} = \frac{g(y_{t+1}|x_{t+1}^{(k)})}{g(y_{t+1}|\mu_{t+1}^{(k)})}$. The resulting **APF** algorithm runs as below:

step 0 Draw N samples $x_0^{(i)} \sim p(x_0)$ and set $\omega_0^{(i)} = \frac{1}{N}$, for $i = 1, \ldots, N$.

step 1 For each time step, given the new observation y_{t+1} and recent particles $\{x_t^{(i)}, \omega_t^{(i)}\}$, $i = 1, \ldots, N$, we need to

- Calculate the conditional expected value $\mu_{t+1}^{(i)} = E[x_{t+1}|x_t^{(i)}]$ for $i = 1, \ldots, N$.
- Calculate probabilities for each auxiliary index $p(k^{(i)}) \propto \omega_t^{(i)} g(y_{t+1}|\mu_{t+1}^{(i)})$ for $i = 1, \ldots, N$ and then normalize them to unity.
- (Re-sampling) Sample auxiliary indices $k^{(i)}$ according to $\{p(k^{(j)})\}_{j=1}^N$ and set $x_t^{(i)} = x_t^{k^{(i)}}, \mu_{t+1}^{(i)} = \mu_{t+1}^{k^{(i)}}$ for $i = 1, \ldots, N$.
- (Propagating) Draw $\{x_{t+1}^{(i)}\}$ by $p(x_{t+1}|x_t^{(i)})$ for $i = 1, \ldots, N$.
- Update weights as $\omega_{t+1}^{(i)} \propto \dfrac{g(y_{t+1}|x_{t+1}^{(i)})}{g(y_{t+1}|\mu_{t+1}^{(i)})}$ and then normalize them to unity.

The filtering results of implementing **APF** in *linear Gaussian* and *stochastic volatility* models are shown in Figure 4 and 5.

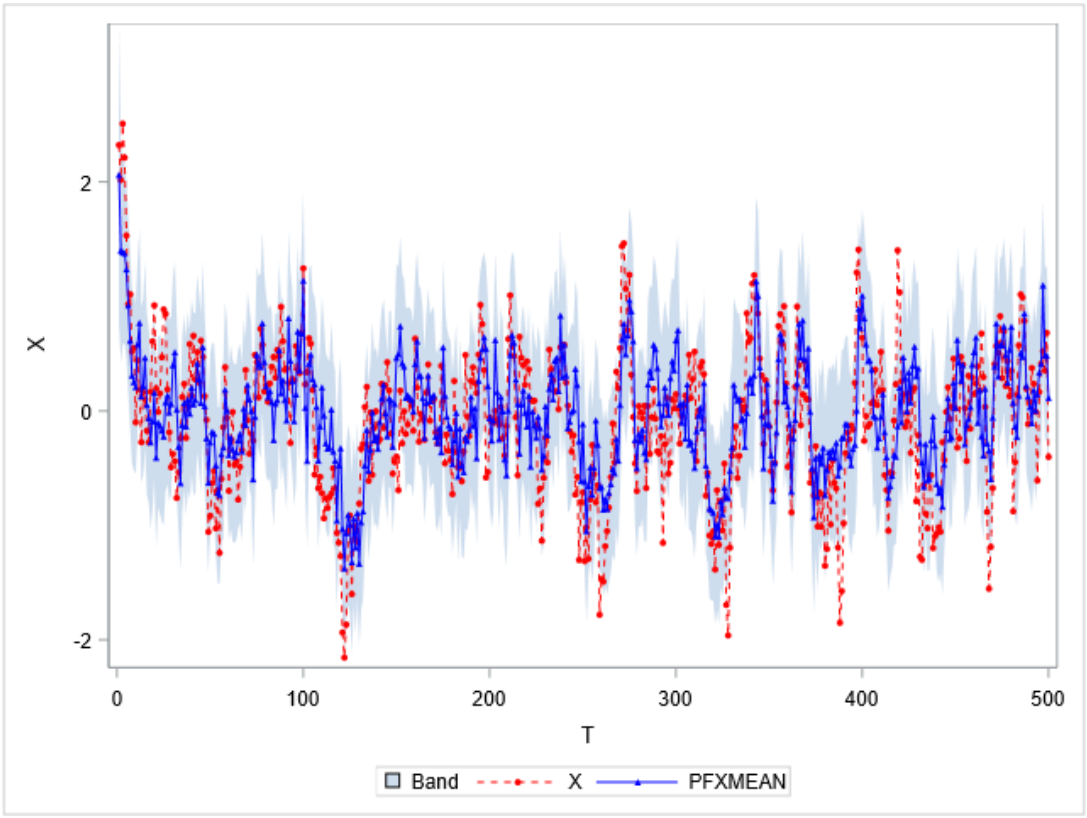

Figure 4: **APF** distribution for 500 simulated observations, *linear Gaussian* model

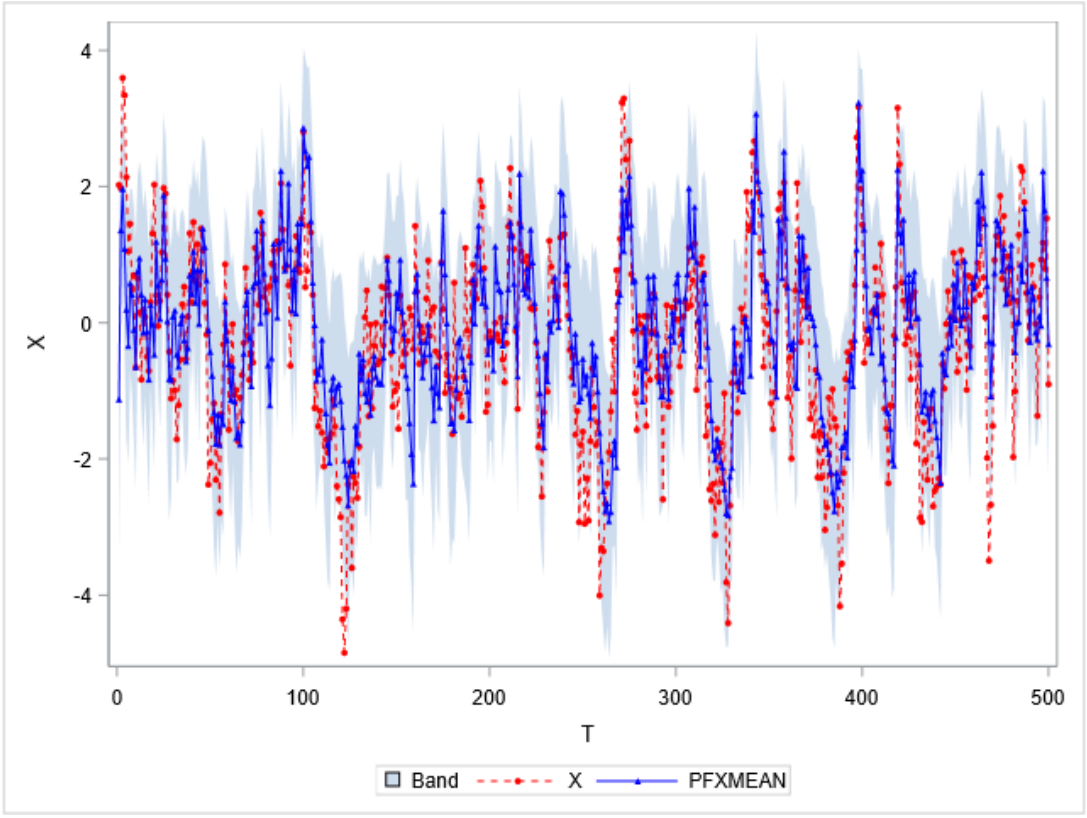

Figure 5: **APF** distribution for 500 simulated observations, *Stochastic Volatility* model

3 Particle Filters Smoothing

We have reviewed how to apply particle filtering methods with known into approximating posterior distribution $p(x_t|y^t)$. Actually, the sequence of densities $p(x^t|y^t)$ can also be approximated in a similar way as long as we re-sampling all historical particles when re-sampling is implemented. The particle filtering algorithm for sequence $x^t = \{x_{1:t}\}$ is listed as below:

step 0 Draw N samples $x_0^{(i)} \sim p(x_0)$ and set $\omega_0^{(i)} = \frac{1}{N}$, for $i = 1, \ldots, N$.

step 1 Repeat the following steps for $t = 1, \ldots, T$

1.1 (Propagating) Draw new particles $x_t^{(i)}$ by a certain filtering method with corresponding weights $\omega_t^{(i)}$ for $i = 1, \ldots, N$ that relate the new particles to the new observation. Normalize those weights so that they sum up to 1.

1.2 Extend $x_{1:t-1}^{(i)}$ by the new particles to have $\tilde{x}_{1:t}^{(i)} = \{x_{1:t-1}^{(i)}, x_t^{(i)}\}$.

1.3 Evaluate the effective number of particles at time step t by (7). If N_{eff} is less than the threshold, go to 1.4. Otherwise, set $x_{1:t}^{(i)} = \tilde{x}_{1:t}^{(i)}$ for $i = 1, \ldots, N$ and enter the iteration of time step $t + 1$.

1.4 (Re-sampling) Draw $\{z(i)\}_{i=1}^N \sim \text{Mult}_N(\{\omega_t^{(s)}\}_{s=1}^N)$ and set $x_t^{(i)} = x_t^{z^{(i)}}$. Set $x_{1:t}^{(i)} = \tilde{x}_{1:t}^{z(i)}$ for $i = 1, \ldots, N$ and $\omega_t^{(i)} = \frac{1}{N}$.

At time T, with particles for each path, we can approximate $p(x_{1:T}|y_{1:T})$ by

$$\hat{p}(d\,x_{1:T}|y_{1:T}) = \sum_{i=1}^{N} \omega_T^{(i)} \delta_{x_{1:T}^{(i)}}(d\,x_{1:T}) \tag{3.1}$$

Then it is straightforward to provide an approximation for $p(x_s|y^T), s \leq T$, by marginalizing $\hat{p}(x_{1:T}|y_{1:T})$, which solves particle filters smoothing problems. Any filtering algorithm can be used and the method inherits $O(N)$ computational complexity. Such approximation, however, is inefficient when $T - s$ is very large as degeneracy problem asks for give only a few different particle paths. Some alternative smoothing methods that stave off these problems will be discussed in this section.

3.1 Fixed-lag Smoother

Firstly, let us introduce a smoothing method by storing state vector (Kitagawa, 1996)[12]. This method which stores all the realized particle in each path is just a modification of the particle filtering steps. In particle filtering process, regardless of which filtering method is implemented, we can have new particles and their corresponding weights after particle propagating. After this step, a re-sampling step may be added to alleviate the concerns about degeneracy problem. Rather than re-sample the new particles, this smoothing method re-sample all historical particles.

Assuming we are given observations till time T, smoothers at each previous time step t ($t \leq T$) can be defined as $\{s_{t|T}^{(i)}\}, i = 1, \ldots, N$ such that $s_{t|T}^{(i)} \sim p(x_t|y^T)$. An algorithm for smoothing by storing state vector is given as below:

step 0 Draw N samples $x_0^{(i)} \sim p(x_0)$ and set $\omega_0^{(i)} = \frac{1}{N}$, for $i = 1, \ldots, N$.

step 1 Repeat the following steps for $t = 1, \ldots, T$

1.1 (Propagating) Draw new particles $x_t^{(i)}$ by a certain filtering method with corresponding weights $\omega_t^{(i)}$ for $i = 1, \ldots, N$ that relate the new particles to the new observation. Normalize those weights.

1.2 (Re-sampling) Generate $\{(s_{1|t}^{(i)}, \ldots, s_{t-1|t}^{(i)}, s_{t|t}^{(i)})^T, i = 1, \ldots N\}$ by re-sampling $\{(s_{1|t-1}^{(i)}, \ldots, s_{t-1|t-1}^{(i)}, x_t^{(i)})^T, i = 1, \ldots N\}$ based on normalized weights $\{\omega_t^{(i)}\}_{i=1}^N$. Set $\omega_t^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.

Even though we can propagate a sufficiently large number of particles in each time step, the repetition of re-sampling will gradually reduce diversity in our samples if T is quite large. In order to fix such problem, the fixed lag smoother is applied to re-sampling method. Choose a lag number L, the steps in fixed lag smoothing when $t \leq L$ are exactly same as smoothing by storing state vector. When $t \geq L + 1$, the re-sampling step should be replaced by step 1.2′

1.2’ (Re-sampling) Generate $\{(s_{t-L|t}^{(i)}, \ldots, s_{t-1|t}^{(i)}, s_{t|t}^{(i)})^T, i = 1, \ldots N\}$ by re-sampling $\{(s_{t-L|t-1}^{(i)}, \ldots, s_{t-1|t-1}^{(i)}, x_t^{(i)})^T, i = 1, \ldots N\}$ based on normalized weights $\{\omega_t^{(i)}\}_{i=1}^N$. Set $\omega_t^{(i)} = \frac{1}{N}$ for $i = 1, \ldots, N$.

Consequently, this fixed-lag smoother relies on the fact that observations collected at times $s > t + L$ bring no additional information to the particles till time t. Then in the algorithm we will not re-sample particles $x_{1:t}$ at times $s > t + L$. Under this method, we consider $p(x_t|y^{t+L})$ to be the approximation of $p(x_t|y^T)$. As $p(x_t|y^{t+L})$ usually converges quickly to $p(x_t|y^T)$, L is not necessarily quite large (say L between 20 and 50).

Apart from fixing degeneracy problem, the fixed-lag smoother also saves storage of historical particles. $N \times L$ particles should be stored in fixed-lag smoother while the original smoother requires storage for all $N \times T$ particles. The computational complexity of fixed-lag smoothing algorithm is $O(N)$ at each step.

Despite the obvious advantages, fixed-lag smoother also brings about a few problems. Firstly, it is worth noting that L needs to be determined carefully. If L is too small, $p(x_t|y^{t+L})$ will give a poor approximation of $p(x_t|y^T)$. On the other hand, degeneracy problem appears when we choose a sufficiently large L. Secondly, the fixed-lag smoother would not converge asymptotically towards the true smoothing distributions as there always exists a bias for the approximation of $p(x_t|y^T)$ with $p(x_t|y^{t+L})$.

3.2 Smoothing Using Backward Recursion

3.2.1 Backward Smoothing

Another alternative smoothing algorithm, proposed by Doucet, Godsill and Andrieu in 2000[4], is called backward smoothing because this method solves marginal smoothing problem by backward recursions starting from the last time step. This smoothing algorithm is on a basis of the following formula (Kitagawa 1987):

$$p(x_s|y^T) = p(x_s|y^s) \int \frac{f(x_{s+1}|x_s)}{p(x_{s+1}|y^s)} p(x_{s+1}|y^T) d\, x_{s+1} \tag{3.2}$$

Let us define an approximation for the marginal smoothing approximation by the following form

$$\hat{p}(d\, x_s|y_{1:T}) \triangleq \sum_{i=1}^{N} \omega_{s|T}^{(i)} \delta_{x_s^{(i)}}(d\, x_s) \tag{3.3}$$

Given this definition, we could simplify the integral in 3.2 by

$$\int \frac{p(x_{s+1}|x_s)}{f(x_{s+1}|y^s)} p(x_{s+1}|y^T) d\, x_{s+1} = \sum_{i=1}^{N} \omega_{s+1|T}^{(i)} \frac{f(x_{s+1}^{(i)}|x_s)}{p(x_{s+1}^{(i)}|y^s)} \tag{3.4}$$

where $p(x_{s+1}^{(i)}|y^s)$ can be approximated by

$$\begin{aligned} p(x_{s+1}^{(i)}|y^s) &= \int f(x_{s+1}^{(i)}|x_s) p(x_s|y^s) d\, x_s \\ &\approx \sum_{j=1}^{N} \omega_s^{(j)} f(x_{s+1}^{(i)}|x_s^{(j)}) \end{aligned}$$

As a result, the approximation $\hat{p}(d\,x_s|y_{1:T})$ is

$$\begin{aligned}\hat{p}(d\,x_s|y_{1:T}) &= \Big[\sum_{i=1}^{N}\omega_s^{(i)}\delta_{x_s^{(i)}}(d\,x_s)\Big]\sum_{j=1}^{N}\omega_{s+1|T}^{(j)}\frac{f(x_{s+1}^{(j)}|x_s)}{\sum_{l=1}^{N}\omega_s^{(l)}f(x_{s+1}^{(j)}|x_s^{(l)})}\\ &= \sum_{i=1}^{N}\omega_s^{(i)}\Big[\sum_{j=1}^{N}\omega_{s+1|T}^{(j)}\frac{f(x_{s+1}^{(j)}|x_s^{(i)})}{\sum_{l=1}^{N}\omega_s^{(l)}p(x_{s+1}^{(j)}|x_s^{(l)})}\Big]\delta_{x_s^{(i)}}(d\,x_s)\\ &\triangleq \sum_{i=1}^{N}\omega_{s|T}^{(i)}\delta_{x_s^{(i)}}(d\,x_s)\end{aligned}$$

The backward recursion is thus $\omega_{s|T}^{(i)} = \omega_s^{(i)}\Big[\sum_{j=1}^{N}\omega_{s+1|T}^{(j)}\frac{f(x_{s+1}^{(j)}|x_s^{(i)})}{\sum_{l=1}^{N}\omega_s^{(l)}f(x_{s+1}^{(j)}|x_s^{(l)})}\Big]$. At time T, given the filtering particles with corresponding weights, $\{x_t^{(i)},\omega_t^{(i)}, t=1,\dots,T\}_{i=1}^{N}$, the weights in backward recursions can be acquired by the following algorithm:

step 1 Set $\omega_{T|T}^{(i)} = \omega_T^{(i)}$ for $i = 1,\dots,N$.

step 2 For $s = T-1,\dots,1$, update the importance weight as
$\omega_{s|T}^{(i)} = \sum_{j=1}^{N}\omega_{s+1|T}^{(j)}\frac{\omega_s^{(i)}f(x_{s+1}^{(j)}|x_s^{(i)})}{\Big[\sum_{l=1}^{N}\omega_s^{(l)}f(x_{s+1}^{(j)}|x_s^{(l)})\Big]}$.

Finally, 3.3 will be implemented to have corresponding $\hat{p}(d\,x_s|y_{1:T}), s \le T$. The problem of this smoothing method is that the approximation still relies on the same sequence of particles as the ones in filtering process despite different weights. If $p(x_s|y^s)$ and $p(x_s|y^T), s \le T$ have high probability densities in different intervals, then the MC approximation will have a very high variance.

The importance weight update step indicates the computational complexity of this marginal backward smoothing is $O(N^2)$ at each step. It is also possible to approximate any fixed interval or joint smoothing distributions with this backward smoother while the computational complexity grows exponentially. Thus, this method is restricted in solving marginal distribution in most practical problems.

3.2.2 Smoothing with Backward Simulation

Unlike smoothing by backward smoothing that is used to approximate marginal smoothing distribution $p(d\,x_s|y_{1:T}), s \le T$, backward simulation[5] can be implemented to get a approximation of a joint smoothing density. Assuming we have already performed particle filtering process and have approximations for filtering distributions, $p(d\,x_t|y_{1:t})$, for each time step $t \in 1,\dots,T$ from weighted particles $\{x_t^{(i)},\omega_t^{(i)}\}_{i=1}^{N}$. This smoothing method is on a basis on the following factorization

$$p(x_{1:T}|y_{1:T}) = p(x_T|y_{1:T})\prod_{t=1}^{T-1}p(x_t|x_{t+1:T},y_{1:T}) \tag{3.5}$$

where, given Markovian assumption in the spate-space model (SSM), the last component can be represented as

$$\begin{aligned} p(x_t|x_{t+1:T}, y_{1:T}) &= p(x_t|x_{t+1}, y_{1:T}) \quad \text{(SSM Markovian assumption)} \\ &= p(x_t|x_{t+1}, y_{1:t}) \quad \text{(conditional independence property in SSM)} \\ &= \frac{p(x_t, x_{t+1}|y_{1:t})}{p(x_{t+1}|y_{1:t})} \\ &= \frac{p(x_t|y_{1:t})f(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} \\ &\propto p(x_t|y_{1:t})f(x_{t+1}|x_t) \end{aligned} \tag{3.6}$$

With approximated filtering distribution, $\hat{p}(d\,x_t|y_{1:t}) = \sum_{i=1}^{N} \omega_t^{(i)} \delta_{x_t^{(i)}}(d\,x_t)$ in (3.6), we have backward particle approximation

$$p(x_t|x_{t+1:T}, y_{1:T}) \approx \sum_{i=1}^{N} \Big[\frac{\omega_t^{(i)} f(x_{t+1}|x_t^{(i)})}{\sum_{j=1}^{N} \omega_t^{(j)} f(x_{t+1}|x_t^{(j)})} \Big] \delta_{x_t^{(i)}}(x_t) \tag{3.7}$$

Then the corresponding weight for backward particle is thus defined as

$$\widetilde{\omega}_{t|t+1}^{(i)} \triangleq \frac{\omega_t^{(i)} f(x_{t+1}|x_t^{(i)})}{\sum_{j=1}^{N} \omega_t^{(j)} f(x_{t+1}|x_t^{(j)})} \tag{3.8}$$

Given a starting point $p(x_T|y_{1:T})$, a backward recursion for joint smoothing density can be built by

$$p(x_{t:T}|y_{1:T}) = p(x_t|x_{t+1}, y_{1:t}) p(x_{t+1:T}|y_{1:T}) \tag{3.9}$$

Then backward particles can be used to generate states recursively conditioning on the future states. Given $\widetilde{x}_{t+1:T}$ drawn from $p(x_{t+1:T}|y_{1:T})$, $\widetilde{x}_t$ can be sampled in the reverse-time direction from $p(x_t|\widetilde{x}_{t+1}, y_{1:T})$. The new augmented backward particles $(\widetilde{x}_t, \widetilde{x}_{t+1:T})$ is then an approximation a sample from $p(x_{t:T}|y_{1:T})$. Repeating this process till recursively over time till $t = 1$, all backward particles will be required. Such smoothing algorithm using backward simulation is listed as below

step 0 Sample independently $\{b_T(j)\}_{j=1}^{M} \sim \text{Mult}_M(\{\omega_T^{(i)}\}_{i=1}^{N})$. Set $\widetilde{x}_T^{(j)} = x_T^{b_T(j)}$ for $j = 1, \ldots, M$.

step 1 Repeat the following steps for $t = T-1, \ldots, 1$,

1.0 For each $j \in 1, \ldots, M$, do the following steps:

1.01 Calculate $\widetilde{\omega}_{t|t+1}^{i,j} \propto \omega_t^{(i)} f(\widetilde{x}_{t+1}^{(j)}|x_t^{(i)})$ for $i = 1, \ldots, N$.

1.02 Normalize the backward smoothing weights $\{\widetilde{\omega}_{t|t+1}^{i,j}\}_{i=1}^{N}$ to sum up to 1.

1.03 Draw $b_t(j) \sim \text{Mult}_1(\{\widetilde{\omega}_{t|t+1}^{i,j}\}_{i=1}^{N})$. Set $\widetilde{x}_t^{(j)} = x_t^{b_t(j)}$ and $\widetilde{x}_{t:T}^{(j)} = (\widetilde{x}_t^{(j)}, \widetilde{x}_{t+1:T}^{(j)})$.

Such backward simulation smoothing algorithm gives an equally weighted point-mass approximation of joint smoothing distribution by

$$\hat{p}(d\,x_{1:T}|y_{1:T}) = \frac{1}{M} \sum_{j=1}^{M} \delta_{\widetilde{x}_{1:T}^{(j)}}(d\,x_{1:T}) \tag{3.10}$$

This approximation can also be applied to estimate any marginal $(p(x_t|y_{1:T}), t \leq T)$ or fixed-interval smoothing $(p(x_{s:t}|y_{1:T}), s < t \leq T)$ distribution. The computational complexity is $O(MN)$ for each time step as index i is from 1 to N and j ranges from 1 to M. The number of backward realizations, M, needs to be large enough so that we can have an accurate approximation of the smoothing distribution. Since the backward realizations can be produced independently, such algorithm is well suitable for parallel computation.

Some approaches are available to derive a smoothing method with backward recursions with $O(N\,logN)$ computational complexity. Fast multipole expansions (Greengard & Rokhlin, 1987), box-sum approximations (Felzenswalb et al., 2003)[7] and spatial-index methods (Moore, 2000) are generally applied to give additional approximations and then decrease $O(N^2)$ computational bottleneck to $O(N\,logN)$. Moreover, Randal Douc et al. (2011) proposed a faster version of backward simulation smoother by using an accept-reject procedure without introducing any further approximations. Such modification of backward simulation smoother has a complexity that grows only linearly in N rather than in NM.

3.3 Generalized Two-filter Smoother

The two-filter smoother[1] gives approximation of the marginal smoothing density $p(x_t|y_{1:T})$ by combining particle filtering with a backward information filtering. Firstly, such marginal smoothing density can be factorized as

$$\begin{aligned} p(x_t|y_{1:T}) &= p(x_t|y_{1:t-1}, y_{t:T}) \\ &= \frac{p(x_t, y_{t:T}|y_{1:t-1})}{p(y_{t:T}|y_{1:t-1})} \\ &= \frac{p(x_t|y_{1:t-1})p(y_{t:T}|x_t, y_{1:t-1})}{p(y_{t:T}|y_{1:t-1})} \\ &\propto p(x_t|y_{1:t-1})p(y_{t:T}|x_t) \quad \text{(SSW Markovian assumption)} \end{aligned} \tag{3.11}$$

The first term in (3.11) is the prediction result of particle filtering at step $t-1$ and the second term needs to be evaluate by following backward recursions:

$$\begin{aligned} p(y_{t+1:T}|x_t) &= \int p(y_{t+1:T}, x_{t+1}|x_t) d\,x_{t+1} \\ &= \int p(y_{t+1:T}|x_{t+1}, x_t) f(x_{t+1}|x_t) d\,x_{t+1} \\ &= \int p(y_{t+1:T}|x_{t+1}) f(x_{t+1}|x_t) d\,x_{t+1} \\ p(y_{t:T}|x_t) &= p(y_t, y_{t+1:T}|x_t) \\ &= p(y_t|y_{t+1:T}, x_t) p(y_{t+1:T}|x_t) \\ &= g(y_t|x_t) p(y_{t+1:T}|x_t) \\ &= g(y_t|x_t) \int p(y_{t+1:T}|x_{t+1}) f(x_{t+1}|x_t) d\,x_{t+1} \end{aligned} \tag{3.12}$$

There is a probability that $\int p(y_{t:T}|x_t) d\,x_t$ is not finite as $p(y_{t:T}|x_t)$ is not probability density function in x_t. One solution, proposed by Briers et al. (2004), to this problem is constructing a generalized version of two-filter smoothing algorithm by introducing an artificial prior distribution $\gamma_t(x_t), t = 0, \ldots, T$. Such artificial prior distribution can be derived recursively from $\gamma_0(x_0)$

by

$$\gamma_t(x_t) = \int f(x_t|x_{t-1})\gamma_{t-1}(x_{t-1})d\,x_{t-1} \tag{3.13}$$

where $\gamma_0(x_0)$ is the starting point and we have to guarantee that $\gamma_t(x_t) > 0$ if $p(y_{t:T}|x_t) > 0$ for $t = 0, \ldots, T$. $\gamma_0(x_0)$ is not necessarily equal to $p(x_0)$, which is used in the filtering recursion. If the actual prior distribution $p(x_0)$ is used as $\gamma_0(x_0)$, then $\frac{f(x_{t+1}|x_t)\gamma_t(x_t)}{\gamma_{t+1}(x_{t+1})} = p(x_t|x_{t+1})$. It is worth noting that, based on Bayes rule, we have

$$p(x_t|x_{t+1}) = \frac{f(x_{t+1}|x_t)p(x_t)}{p(x_{t+1})} \neq cf(x_{t+1}|x_t), \quad \text{c is any constant}$$

The reverse transition is thus not proportional to $f(x_{t+1}|x_t)$, which is a common mistake.

If a Markovian kernel $p(x_{t+1}|x_t)$ has an invariant distribution $\pi(x)$, we can define $\gamma_0(x_0) = \pi(x)$. Then $\gamma_t(x_t) = \pi(x), t \geq 1$ even though $p(x_0) \neq \pi(x)$. At that situation, $\frac{f(x_{t+1}|x_t)\gamma_t(x_t)}{\gamma_{t+1}(x_{t+1})} = f(x_t|x_{t+1})$ if f is $\pi-$reversible. Given linear-Gaussian states, $\gamma_t(x_t)$ is available in closed form. If the prior distribution $p(x_0)$ is also Gaussian, it can be directly used as $\gamma_0(x_0)$.

To realize the backward filtering in this smoothing algorithm, an auxiliary probability density, $\widetilde{p}(x_t|y_{t:T})$ needs to be defined. Given the initial condition, $\widetilde{p}(x_T|y_T) = \frac{g(y_T|x_T)\gamma_T(x_T)}{p(y_T)}$, let us define the sequence of auxiliary probability density as

$$\widetilde{p}(x_{t:T}|y_{t:T}) \propto \gamma_t(x_t) \prod_{i=t+1}^{T} f(x_i|x_{i-1}) \prod_{i=t}^{T} g(y_i|x_i), \quad t < T \tag{3.14}$$

Then we have

$$\begin{aligned}
p(y_{t:T}|x_t) &= \int \cdots \int p(y_{t:T}, x_{t+1:T}|x_t)d\,x_{t+1:T} \\
&= \int \cdots \int p(x_{t+1:T}|x_t)p(y_{t:T}|x_{t:T})d\,x_{t+1:T} \\
&= \int \cdots \int \frac{\gamma_t(x_t)}{\gamma_t(x_t)} \prod_{i=t+1}^{T} p(x_i|x_{i-1}) \prod_{i=t}^{T} p(y_i|x_i)d\,x_{t+1:T} \\
&\propto \int \cdots \int \frac{\widetilde{p}(x_{t:T}|y_{t:T})}{\gamma_t(x_t)} d\,x_{t+1:T} \\
&= \frac{\widetilde{p}(x_t|y_{t:T})}{\gamma_t(x_t)}
\end{aligned} \tag{3.15}$$

Thus the auxiliary probability density is chosen such that $\widetilde{p}(x_t|y_{t:T}) \propto \gamma_t(x_t)p(y_{t:T}|x_t)$. To derive a recursion for this auxiliary probability density, (3.12) is expanded in the following way:

$$\begin{aligned}
p(y_{t:T}|x_t) &= g(y_t|x_t) \int p(y_{t+1:T}|x_{t+1})f(x_{t+1}|x_t)d\,x_{t+1} \\
&\propto \int \frac{\widetilde{p}(x_{t+1}|y_{t+1:T})}{\gamma_{t+1}(x_{t+1})} f(x_{t+1}|x_t)g(y_t|x_t)d\,x_{t+1} \\
&= \frac{g(y_t|x_t)}{\gamma_t(x_t)} \int \widetilde{p}(x_{t+1}|y_{t+1:T}) \frac{f(x_{t+1}|x_t)\gamma_t(x_t)}{\gamma_{t+1}(x_{t+1})} d\,x_{t+1}
\end{aligned} \tag{3.16}$$

It indicates that one backward recursive filtering can be obtained by comparison between (3.15) and (3.16), which is

$$\widetilde{p}(x_t|y_{t:T}) \propto p(y_t|x_t) \int \widetilde{p}(x_{t+1}|y_{t+1:T}) \frac{f(x_{t+1}|x_t)\gamma_t(x_t)}{\gamma_{t+1}(x_{t+1})} d\, x_{t+1}$$

Such backward recursion stems from the initial point $\widetilde{p}(x_T|y_T) \propto g(y_T|x_T)\gamma_T(x_T)$. With respect to particle realizations, let $\{x_t^{(i)}, \omega_t^{(i)}\}_{i=1}^N$ denotes the forward filtering particles at time t while $\{\widetilde{x}_t^{(i)}, \widetilde{\omega}_t^{(i)}\}_{i=1}^N$ are used to approximate the backward filtering density, $\widetilde{p}(x_t|y_{t:T})$. Then the generalized two-filter smoothing algorithm is show as below:

1 ***Forward Filtering*** For $t = 1, \ldots, T$, perform particle filtering to obtain weighted realizations $\{x_t^{(i)}, \omega_t^{(i)}\}_{i=1}^N$.

2 ***Backward Filtering***

2.1 *Initialization:*

I From an importance distribution, draw $\widetilde{x}_T^{(i)} \sim q(\cdot|y_T)$ for $i = 1, \ldots, N$.

II Compute the importance weights $\widetilde{\omega}_T^{(i)} \propto \frac{g(y_T|\widetilde{x}_T^{(i)})\gamma_T(\widetilde{x}_T^{(i)})}{q(\widetilde{x}_T^{(i)}|y_T)}$ for $i = 1, \ldots, N$.

2.2 For $t = T, \ldots, 2$, repeat the following steps:

I Draw $\widetilde{x}_{t-1}^{(i)} \sim q(\cdot|\widetilde{x}_t^{(i)}, y_{t-1})$ for $i = 1, \ldots, N$.

II Compute backward importance weights

$$\widetilde{\omega}_{t-1}^{(i)} \propto g(y_{t-1}|\widetilde{x}_{t-1}^{(i)})\widetilde{\omega}_t^{(i)} \frac{f(\widetilde{x}_t^{(i)}|\widetilde{x}_{t-1}^{(i)})\gamma_{t-1}(\widetilde{x}_{t-1}^{(i)})}{\gamma_t(\widetilde{x}_t^{(i)})q(\widetilde{x}_{t-1}^{(i)}|\widetilde{x}_t^{(i)}, y_{t-1})}$$

Plugging (3.15) into (3.11), we obtain $p(x_t|y_{1:T}) \propto p(x_t|y_{1:t-1})\frac{\widetilde{p}(x_t|y_{t:T})}{\gamma_t(x_t)}$. Consequently, the smoothing density can be approximated by

$$\begin{aligned}
\widehat{p}(d\, x_t|y_{1:T}) &\propto \sum_{j=1}^N p(\widetilde{x}_t^{(j)}|y_{1:t-1}) \frac{\widetilde{p}(\widetilde{x}_t^{(j)}|y_{t:T})}{\gamma_t(\widetilde{x}_t^{(j)})} \delta_{\widetilde{x}_t^{(j)}}(d\, x_t) \\
&= \sum_{j=1}^N \sum_{i=1}^N f(\widetilde{x}_t^{(j)}|x_{t-1}^{(i)})\omega_{t-1}^{(i)} \frac{\widetilde{p}(\widetilde{x}_t^{(j)}|y_{t:T})}{\gamma_t(\widetilde{x}_t^{(j)})} \delta_{\widetilde{x}_t^{(j)}}(d\, x_t) \\
&= \sum_{j=1}^N \sum_{i=1}^N f(\widetilde{x}_t^{(j)}|x_{t-1}^{(i)})\omega_{t-1}^{(i)} \frac{\widetilde{\omega}_t^{(j)}}{\gamma_t(\widetilde{x}_t^{(j)})} \delta_{\widetilde{x}_t^{(j)}}(d\, x_t)
\end{aligned} \tag{3.17}$$

As the approximation above shows, this algorithm has $O(N^2)$ computational complexity. In addition, this smoothed distribution is being approximated from two sets of realizations of backward and forward filters. That is why it is expected to outperform those backward recursion smoothing methods.

In both backward recursion and two-filter smoothers, degeneracy problems still exist when $f(x_t|x_{t-1})$ is quite close to zero for most combinations of possible x_t and x_{t-1}. This problem is worse for two-filter smoother as the forward and backward filter particles are sampled independently. It is likely that no pairs of forwards and backwards filter particles can match. Fearnhead et al. (2010)[6] also came up a new two-filter smoother with $O(N^2)$ complexity that

can overcome such shortcoming in two-filter smoother. They attempted to propagate new particles from marginal smoothing densities directly rather than re-weight particles generated from backward recursion. This new algorithm will have a $O(N^2)$ complexity, however, one simple approach can be used make it $O(N)$ in computational complexity.

3.4 Maximum a Posteriori (MAP) Smoother

The estimation of MAP sequence and the marginal fixed-lag MAP sequence is a complex global optimization problem, where MAP sequence is defined as

$$x_{1:t}^{MAP}(t) \triangleq \arg\max_{x_{1:t}} p(x_{1:t}|y_{1:t}) \tag{3.18}$$

and the marginal fixed-lag MAP sequence is defined as

$$x_{t-L+1:t}^{MMAP}(t) \triangleq \arg\max_{x_{t-L+1:t}} p(x_{t-L+1:t}|y_{1:t}) \tag{3.19}$$

A simple method to find the MAP smoother relies on sample paths of $x_{1:t}^{(i)}, i = 1, \ldots, N$. Such paths are generated from the posterior distribution $p(x_{1:t}|y_{1:t})$ or any other distributions that has the same global maximums as the posterior distribution, such as $p(x_{1:t}|y_{1:t})^{\gamma}$ $(\gamma > 0)$. Based on the weighted filtering sequences $\{x_{1:t}^{(i)}, \omega_t^{(i)}\}$, one simple approximation of the MAP sequence can be obtained as

$$\widehat{x}_{1:t}^{MAP} = \arg\max_{x_{1:t} \in \{x_{1:t}^{(i)}, i=1,\ldots,N\}} \sum_{i=1}^{N} \omega_t^{(i)} \delta_{x_{1:t}^{(i)}}(x_{1:t}) \tag{3.20}$$

Even though this approximation is easy to implement and requires linear computational convexity $O(N)$, it suffers from severe degeneracy problems discussed in the previous sections. The approximation of MAP sequence is poor when t is quite large, so is the marginal fixed-lag MAP sequence when we have a considerable L.

Godsill et al. (2001)[8] came up with a dynamic programming approach to estimate MAP sequence based on weighted state simulators at each time step which can be acquired by a normal particle filtering. Then we can estimate (3.18) by locating a sequence of maximum probability. The MAP approximation is thus defined as

$$\widetilde{x}_{1:t}^{MAP}(t) \triangleq \arg\max_{x_{1:t} \in \otimes_{k=1}^{t} \{x_k^{(i)}\}_{i=1}^{N}} p(x_{1:t}|y_{1:t}) \tag{3.21}$$

The MAP sequence search needs to take into account all particles so the number of possible paths grows exponentially with time. One property of smoothing distribution, however, can be used to simplify this search process. We have already learned that $p(x_{1:t}|y_{1:t})$ can be factorized as

$$p(x_{1:t}|y_{1:t}) \propto \prod_{i=1}^{t} f(x_i|x_{i-1}) g(y_i|x_i) \tag{3.22}$$

Therefore, the search of MAP sequence is equivalent to the following optimization problem.

$$x_{1:t}^{MAP}(t) \triangleq \arg\max_{x_{1:t}} \sum_{k=1}^{t} [\log g(y_k|x_k) + \log f(x_k|x_{k-1})] \tag{3.23}$$

A dynamic programming technique, Viterbi algorithm, can be used to solve such maximization problem by calculating

$$\widetilde{x}_{1:t}^{MAP}(t) \triangleq \underset{x_{1:t}\in\otimes_{k=1}^{t}\{x_k^{(i)}\}_{i=1}^{N}}{\arg\max} \sum_{k=1}^{t}[\log g(y_k|x_k) + \log f(x_k|x_{k-1})] \tag{3.24}$$

The corresponding algorithm to approximate MAP sequence is listed as blow:

1 ***Forward Filtering*** For $t = 1, \ldots, T$, perform particle filtering to obtain weighted realizations $\{x_t^{(i)}, \omega_t^{(i)}\}_{i=1}^{N}$.

2 ***Initialization*** For $i = 1, \ldots, N$,

$$\delta_1(i) = \log p(x_1^{(i)}) + \log p(y_1|x_1^{(i)})$$

3 ***Recursion*** For $t = 2, \ldots, T$

$$\delta_t(j) = \log p(y_t|x_t^{(j)}) + \max_i[\delta_{t-1}(i) + \log p(x_t^{(j)}|x_{t-1}^{(i)})]$$

$$\psi_t(j) = \arg\max_i[\delta_{t-1}(i) + \log p(x_t^{(j)}|x_{t-1}^{(i)})]$$

4 ***Termination*** $\imath_T = \arg\max_i \delta_T(i), \quad , \widetilde{x}_T^{MAP} = x_T^{i_T}$

5 ***Backtracking*** For $t = T-1, \ldots, 1$, $i_t = \psi_{t+1}(i_{t+1}), \quad \widetilde{x}_t^{MAP} = x_t^{i_t}$

6 ***Aggregation*** $\widetilde{x}_{1:t}^{MAP} \triangleq \{\widetilde{x}_1^{MAP}, \ldots, \widetilde{x}_T^{MAP}\}$

Due to **step 3**, this MAP smoothing algorithm has computational complexity $O(N^2)$ and memory requirement of oder $O(N)$ as weighted filtering realizations $\{x_t^{(i)}, \omega_t^{(i)}, i = 1, \ldots, N\}_{t=1}^{T}$ need to be recorded.

This MAP smoothing algorithm can be easily modified to give the estimation of fixed-lag MAP sequence, which solves the maximization of $\log p(x_{T-L+1:T}|y_{1:T}) +$
$\log g(y_{t-L+1}|x_{t-L+1}) + \sum_{k=T-L+2}^{T}[\log g(y_k|x_k) + \log f(x_k|x_{k-1})]$. Therefore, the search of fixed-lag MAP sequence is almost same as the MAP smoothing algorithm except that the initial point is at $t = T - L + 1$.

3.5 Forward Smoothing for Additive Functionals

Let $s_k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, k \in \mathbb{N}$, be a sequence of functions and $S_n : \mathcal{X}^n \to \mathbb{R}, n \in \mathbb{N}$, donate the corresponding sequence of additive functionals constructed from s_k as

$$S_n(x_{0:n}) = \sum_{j=1}^{n} s_j(x_{j-1}, x_j)$$

The so-called *smoothed additive functional* is defined by

$$\mathcal{S}_n^{\theta} = E_{\theta}[S_n(X_{0:n})|y_{1:n}] = \int_{\mathcal{X}^{n+1}}[\sum_{j=1}^{n} s_j(x_{j-1}, x_j)]p_{\theta}(x_{0:n}|y_{1:n})dx_{0:n}.$$

It is much likely that we need to calculate such smoothed additive functional recursively especially when implementing maximum-likelihood parameter estimation procedures.

As we have introduced before, the standard 'forward filter backward smoothing'(FFBS) gives one way to solve marginal smoothing problem and it can be implemented to compute $\mathcal{S}_n^\theta$ by two steps. The first step is the forward filtering process in which filtering densities $p_\theta(x_k|y_{1:k}), k = 1,\dots,T$ are approximated by particles. The second step, a backward recursion, calculates the following marginal smoothed densities that are needed when evaluating $s_k(x_{k-1}, x_k)$:

$$p_\theta(x_{k-1}, x_k|y_{1:n}) = p_\theta(x_k|y_{1:n})p_\theta(x_{k-1}|y_{1:k-1}, x_k), \quad 1 \le k \le n \tag{3.25}$$

where $p_\theta(x_{k-1}|y_{1:k-1}, x_k) = \frac{f_\theta(x_k|x_{k-1})p_\theta(x_{k-1}|y_{1:k-1})}{p_\theta(x_k|y_{1:k-1})}$. Decrementing k, we can calculate this probability recursively from $k = n$ to $k = 1$. The sequential monte carlo version of FFBS is straightforward to implement with approximations of components in (3.25). In forward filtering process, the particles at each time step are stored and the corresponding weights $\omega_k^{(i)}$ is an approximation of $p_\theta(x_k|y_{1:k})$ which will be plugged into in (3.25) in the backward recursion. Initialized at $k = n$, let us define

$$\widehat{p}_\theta(dx_k|y_{1:n}) = \sum_{i=1}^N W_{k|n}^{(i)} \delta_{x_k^{(i)}}(x_k) \tag{3.26}$$

to be the SMC approximation of $p_\theta(dx_k|y_{1:n})$. Using the forward filtering particle weights, we obtain

$$\widehat{p}_\theta(dx_{k-1}|y_{1:k-1}, x_k) = \frac{\sum_{j=1}^N \omega_{k-1}^{(j)} f_\theta(x_k|x_{k-1}^{(j)})\delta_{x_{k-1}^{(j)}}(dx_{k-1})}{\sum_{l=1}^N \omega_{k-1}^{(l)} f_\theta(x_k|x_{k-1}^{(l)})} \tag{3.27}$$

Combining with defined $\widehat{p}_\theta(dx_k|y_{1:n})$, we have

$$\widehat{p}_\theta(dx_{k-1:k}|y_{1:n}) = \sum_{i=1}^N \sum_{j=1}^N W_{k|N}^{(i)} \frac{\omega_{k-1}^{(j)} f_\theta(x_k^{(i)}|x_{k-1}^{(j)})}{\sum_{l=1}^N \omega_{k-1}^{(l)} f_\theta(x_k^{(i)}|x_{k-1}^{(l)})} \delta_{x_k^{(i)}, x_{k-1}^{(j)}}(dx_{k-1:k}) \tag{3.28}$$

Integrating (3.25) with regard to x_k yields $p_\theta(x_{k-1}|y_{1:n})$. Therefore, marginalising this approximation gives approximation of $p_\theta(dx_{k-1}|y_{1:n})$ in the form of definition (3.26), which is

$$W_{k-1|n}^{(j)} = \sum_{i=1}^N W_{k|N}^{(i)} \frac{\omega_{k-1}^{(j)} f_\theta(x_k^{(i)}|x_{k-1}^{(j)})}{\sum_{l=1}^N \omega_{k-1}^{(l)} f_\theta(x_k^{(i)}|x_{k-1}^{(l)})} \tag{3.29}$$

Then the SMC approximation of $\mathcal{S}_n^\theta$ is given by

$$\widehat{\mathcal{S}}_n^\theta = \sum_{k=1}^N \int s_k(x_{k-1}, x_k)\widehat{p}_\theta(dx_{k-1:k}|y_{1:n}) \tag{3.30}$$

To avoid the need for backward recursion when computing $\mathcal{S}_n^\theta$, an auxiliary function is introduced as below

$$T_n^\theta(x_n) := \int S_n(x_{0:n})p_\theta(x_{0:n-1}|y_{1:n-1}, x_n)dx_{0:n-1} \tag{3.31}$$

Then we have $\mathcal{S}_n^\theta = \int T_n^\theta(x_n) p_\theta(x_n|y_{1:n})dx_n$. It can be shown that $T_n^\theta(x_n) = \int [T_{n-1}^\theta + s_n(x_{n-1}, x_n)] p_\theta(x_{n-1}|y_{1:n-1,x_n})dx_{n-1}$ where $T_0^\theta(x_0) := 0$.

$$\begin{aligned}
T_n^\theta(x_n) &= \int [S_{n-1}(x_{0:n-1}) + s_n(x_{n-1}, x_n)] p_\theta(x_{0:n-1}|y_{1:n-1}, x_n)dx_{0:n-1} \\
&= \int [\int S_{n-1}(x_{0:n-1}) p_\theta(x_{0:n-2}|y_{1:n-2}, x_{n-1})dx_{0:n-2}] p_\theta(x_{n-1}|y_{1:n-1}, x_n)dx_{n-1} \\
&+ \int s_n(x_{n-1}, x_n) p_\theta(x_{0:n-1}|y_{1:n-1}, x_n)dx_{0:n-1} \\
&= \int T_{n-1}^\theta(x_{n-1}) p_\theta(x_{n-1}|y_{1:n-1}, x_n)dx_{n-1} + \int s_n(x_{n-1}, x_n) p_\theta(x_{n-1}|y_{1:n-1}, x_n)dx_{n-1} \\
&= \int [T_{n-1}^\theta(x_{n-1}) + s_n(x_{n-1}, x_n)] p_\theta(x_{n-1}|y_{1:n-1}, x_n)dx_{n-1} \qquad (3.32)
\end{aligned}$$

Deriving from (3.31) and (3.32), an algorithm is given to approximate $\mathcal{S}_n^\theta$ as below:

1 Initialize $\widehat{T}_0^\theta(x_0^{(i)}) = 0$ and obtain weighted filtering particles $\{x_0^{(i)}, \omega_0^{(i)}\}_{i=1}^N$.

2 Repeat the following steps for time steps $k = 1, 2, \ldots, n$.

2.1 Obtain weighted filtering particles $\{x_k^{(i)}, \omega_k^{(i)}\}$ for $i = 1, \ldots, N$.

2.2 $\widehat{T}_k^\theta(x_k^{(i)}) = \dfrac{\sum_{j=1}^N \omega_{k-1}^{(j)} f_\theta(x_k^{(i)}|x_{k-1}^{(j)})[\widehat{T}_{k-1}^\theta(x_{k-1}^{(j)}) + s_k(x_{k-1}^{(j)}, x_k^{(i)})]}{\sum_{j=1}^N \omega_{k-1}^{(j)} f_\theta(x_k^{(i)}|x_{k-1}^{(j)})}, i = 1, \ldots, N.$

2.3 $\widehat{\mathcal{S}}_k^\theta = \sum_{i=1}^N \omega_k^{(i)} \widehat{T}_k^\theta(x_k^{(i)})$.

Westerborn and Olsson (2014)[17] proposed a variation of this algorithm by approximating $\widehat{T}_k^\theta(x_k^{(i)})$ with a MC estimate. Assuming we have particles with weights $\{\omega_{k-1}^{(i)}, x_{k-1}^{(i)}\}_{i=1}^N$ and the estimates $\{T_{k-1}^\theta(x_{k-1}^{(i)})\}_{i=1}^N$ at time step $k-1$, then new filtering particles, $\{\omega_k^{(i)}, x_k^{(i)}\}_{i=1}^N$, with weights are simulated at first. For each new particle $x_k^{(i)}$, we draw $\tilde{N} \geq 2$ indices $J_k^{(i,j)}$ through

$$P(J_k^{i,j} = l) = \frac{\omega_{k-1}^{(l)} f_\theta(x_k^{(i)}|x_{k-1}^{(l)})}{\sum_{l=1}^N \omega_{k-1}^{(l)} f_\theta(x_k^{(i)}|x_{k-1}^{(l)})}$$

Then the estimator $\widehat{T}_k^\theta(x_k^{(i)}) = \frac{1}{\tilde{N}} \sum_{j=1}^{\tilde{N}} \left(\widehat{T}_{k-1}(x_{k-1}^{(J_k^{i,j})}) + s_{k-1}(x_{k-1}^{(J_k^{i,j})}, x_k^{(i)})\right)$. This variation decrease the original complexity $O(N^2)$ to $O(N\tilde{N})$ where $\tilde{N}$ is often chosen to be 2.

4 Parameter Learning

4.1 Maximum Likelihood Approach

4.1.1 Gradient Ascent Algorithm

Off-line Method To maximize the log-likelihood $\log p_\theta(y_{0:n})$ with respect to parameters θ, a gradient ascent algorithm can be implemented. Assuming after iteration i the parameter estimate θ_i is obtained, then at iteration $i+1$ we can update the parameter by

$$\theta_{i+1} = \theta_i + \gamma_{i+1} \nabla \log p_\theta(y_{0:n})|_{\theta=\theta_i} \qquad (4.1)$$

where $\nabla \log p_{\theta_i}(y_{0:n})$ is the score vector computed at $\theta = \theta_i$ and $\{\gamma_i\}_{i\geq 1}$ is a sequence of non-increasing step-size. One important choice for this sequence is $\gamma_n = n^{-\alpha}, 0.5 < \alpha \leq 1$. Using Fisher's identity, the score vector can be linked to a sequence of additive funtionals as

$$\begin{aligned}
\nabla \log p_\theta(y_{1:n}) &= \int \nabla_\theta \log p_\theta(x_{0:n}, y_{1:n}) p_\theta(x_{0:n}|y_{1:n}) dx_{0:n} \\
&= \int \nabla_\theta \log \left(\mu_\theta(x_0) \prod_{k=1}^{n} f_\theta(x_k|x_{k-1}) \prod_{k=1}^{n} g_\theta(y_k|x_k)\right) p_\theta(x_{0:n}|y_{0:n}) dx_{0:n} \\
&= \int \Big[\nabla_\theta \log \mu_\theta(x_0) + \nabla_\theta \log f_\theta(x_1|x_0) + \nabla_\theta \log g_\theta(y_1|x_1) \\
&+ \sum_{k=2}^{n} \left(\nabla_\theta \log f_\theta(x_k|x_{k-1}) + \nabla_\theta \log g_\theta(y_k|x_k)\right)\Big] p_\theta(x_{0:n}|y_{0:n}) dx_{0:n}
\end{aligned}$$

Therefore, the score vector is in line with the form of a sequence of additive funtionals among which $s_1(x_0, x_1) = \nabla_\theta \log \mu_\theta(x_0) + \nabla_\theta \log f_\theta(x_1|x_0) + \nabla_\theta \log g_\theta(y_1|x_1)$ and $s_k(x_{k-1}, x_k) = \nabla_\theta \log f_\theta(x_k|x_{k-1}) + \nabla_\theta \log g_\theta(y_k|x_k)$ for $k \geq 2$. At the end of each iteration, $\widehat{\mathcal{S}}_n^{\theta_i} = \nabla \log p_{\theta_i}(y_{1:n})$ can be estimated by the algorithm we have discussed.

On-line Method For a long sequence of observations, each iteration of off-line gradient ascent algorithm is quite time-consuming. An alternative is the on-line gradient method in which we update the parameter estimate when we have a new observation at each time step. The on-line method is a recursive procedure to update parameter that exploits all observations sequentially once. It is natural to update the parameter at time $k+1$ by

$$\theta_{k+1} = \theta_k + \gamma_{k+1} \nabla \log p_\theta(y_n|y_{1:k-1})|_{\theta=\theta_k} \tag{4.2}$$

where $\{\gamma_l\}_{l\geq 1}$ is a non-increasing sequence of step size such that $\sum_l \gamma_l = \infty$ and $\sum_l \gamma_l^2 < \infty$. $\gamma_l = l^{-\alpha}, 0.5 < \alpha \leq 1$ is again a common choice. This parameter updating scheme is, however, not suitable for the on-line implementation as we need to re-calculate particles from time 0 to k with the current parameter estimate θ_k. Fortunately, Le Gland and Mevel (1997)[13] suggested a *recursive maximum likelihood estimator* to solve this problem. Under this new scheme, the parameter is updated according to

$$\theta_{k+1} = \theta_k + \gamma_{k+1} \nabla \log p_{\theta_{0:k}}(y_k|y_{1:k-1}) \tag{4.3}$$

where $\nabla \log p_{\theta_{0:k}}(y_k|y_{1:k-1}) = \nabla \log p_{\theta_{0:k}}(y_{1:k}) - \nabla \log p_{\theta_{0:k-1}}(y_{1:k-1})$. On receiving y_k, the previous parameter estimate θ_k is updated along the direction of ascent of the predictive conditional density of this new observation. Such parameter updating rule takes into consideration all previous parameter estimates, which is indicated in the notation $p_{\theta_{0:k}}(y_k|y_{1:k-1})$. We can see $\nabla p_{\theta_{0:k}}(y_k|y_{1:k-1})$ as a 'time-varying' score which applies θ_l in time l. In other words, the weighted particles $\{\omega_k^{(i)}, x_k^{(i)}\}_{i=1}^N$ at time k are propagated from weighted particles at time $k-1$ and parameter estimate $\theta = \theta_k$. Moreover, $\{\omega_k^{(i)}, x_k^{(i)}\}_{i=1}^N$ construct a particle approximation of of $p_{\theta_{0:k}}(dx_k|y_{1:k})$.

We can also apply the algorithm of forward smoothing for additive functionals with $s_k(x_{k-1}, x_k) = \nabla \log f_\theta(x_k|x_{k-1})|_{\theta=\theta_k} + \nabla \log f_\theta(y_k|x_k)|_{\theta=\theta_k}$. Estimation of $\nabla \log p_{\theta_{0:k}}(y_k|y_{1:k-1})$ is now the difference of $\widehat{\mathcal{S}}_k^{\theta_k}$ with $\widehat{\mathcal{S}}_{k-1}^{\theta_{k-1}}$. For the initial point, we start with θ_0 and use it to generate particles $x_0^{(i)} \sim \mu_{\theta_0}(x), i = 1, \ldots, N$ with same weights. The parameters start to be updated after a few time steps. The detailed algorithm is listed as below:

1 Initialize $\widehat{T}_0^{\theta_0}(x_0^{(i)}) = 0$ and obtain weighted filtering particles $\{x_0^{(i)}, \omega_0^{(i)} = \frac{1}{N}\}$ for $i = 1, \ldots, N$. Set $\widehat{\mathcal{S}}_0^{\theta_0} = 0$.

2 Repeat the following steps for time steps $k = 1, 2, \ldots, n$:

2.1 Obtain weighted particles $\{x_k^{(i)}, \omega_k^{(i)}\}$ for $i = 1, \ldots, N$ with parameters θ_k.

2.2 For $i = 1, \ldots, N$,

$$\widehat{T}_k^{\theta_k}(x_k^{(i)}) = \frac{\sum_{j=1}^N \omega_{k-1}^{(j)} f_\theta(x_k^{(i)}|x_{k-1}^{(j)})[\widehat{T}_{k-1}^{\theta}(x_{k-1}^{(j)}) + s_k(x_{k-1}^{(j)}, x_k^{(i)})]}{\sum_{j=1}^N \omega_{k-1}^{(j)} f_\theta(x_k^{(i)}|x_{k-1}^{(j)})}\Big|_{\theta=\theta_k} \tag{4.4}$$

where $s_k(x_{k-1}, x_k) = \nabla \log f_\theta(x_k|x_{k-1})|_{\theta=\theta_k} + \nabla \log f_\theta(y_k|x_k)|_{\theta=\theta_k}$.

2.3 Set $\widehat{\mathcal{S}}_k^{\theta_k} = \sum_{i=1}^N \omega_k^{(i)} \widehat{T}_k^{\theta_k}(x_k^{(i)}) = \widehat{\nabla \log p_{\theta_{0:k}}}(y_{1:k})$, $\widehat{\nabla \log p_{\theta_{0:k}}}(y_k|y_{1:k-1}) = \widehat{\mathcal{S}}_k^{\theta_k} - \widehat{\mathcal{S}}_{k-1}^{\theta_{k-1}}$. ($\widehat{\nabla \log p_{\theta_{0:k}}}(y_1) = \widehat{\mathcal{S}}_1^{\theta_1} - \widehat{\mathcal{S}}_0^{\theta_0}$.)

2.4 $\theta_{k+1} = \theta_k + \gamma_{k+1} \widehat{\nabla \log p_{\theta_{0:k}}}(y_k|y_{1:k-1})$.

4.1.2 Expectation-maximization Algorithm

Off-line Method Gradient ascent algorithm has a wider application than expectation maximization (EM) algorithm. However, in practice, it is difficult to determine a proper scale of components in computed gradient vector so gradient ascent algorithm can be numerically unstable. That is why EM method is usually more preferred whenever it is applicable. EM algorithm is an important procedure for maximizing the log-likelihood function.

Given the parameter estimation θ_k at iteration k, we can update the parameter according to the M-step:

$$\theta_{k+1} = \arg\max_\theta Q(\theta_k, \theta) \tag{4.5}$$

where $Q(\theta_k, \theta)$, the expectation of E-step can be expanded as

$$\begin{aligned} Q(\theta_k, \theta) &= \int \log p_\theta(x_{0:n}, y_{1:n}) p_{\theta_k}(x_{0:n}|y_{1:n}) dx_{0:n} \\ &= \int \log u_\theta(x_0) p_{\theta_k}(x_0|y_{1:n}) dx_0 \\ &+ \sum_{k=1}^n \int (\log f_\theta(x_k|x_{k-1}) + \log g_\theta(y_k|x_k)) p_{\theta_k}(x_{k-1:k}|y_{1:n}) dx_{k-1:k} \end{aligned} \tag{4.6}$$

Firstly, when $p_\theta(x_{0:n}|y_{1:n})$ is not in an exponential family, SMC smoothing techniques are needed to approximate $Q(\theta_k, \theta)$ numerically. For example, when applying forward filtering backward smoothing method, $p_{\theta_k}(dx_{k-1:k}|y_{1:n})$ can be approximated by (3.28) and an approximation of $Q(\theta_k, \theta)$ is then obtained. Moreover, it is also trivial to estimate $p_{\theta_k}(dx_{k-1:k}|y_{1:n})$ by marginalizing if the joint smoothing distribution $p_{\theta_k}(x_{0:n}|y_{1:n})$ has been approximated by some SMC smoothing methods such as fixed-lag and forward filtering backward simulation. In order to maximize $Q(\theta_k, \theta)$, we may refer to optimization software or gradient ascent algorithm. We can also estimate the gradient of $Q(\theta_k, \theta)$ at a certain θ by an approximation that is similar to (4.6) except each component is the corresponding gradient.

As $Q(\theta_k, \theta)$ is in the form of additive functionals, its gradients in M-step can also be approximated by off-line forward smoothing for additive functionals. The weighted particles needs to be kept unchanged in M-step of θ_k until the maximum solution, θ_{k+1}, is recognized.

EM algorithm seems quite complicated especially in the M-step when the model does not belong to an exponential family. Fortunately, maximizing process is quite straightforward if $p_\theta(x_{0:n}, y_{1:n})$ is in the exponential family. Let $s^h : \mathcal{X} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, $h = 1, \ldots, m$ be a collection of sufficient statistics, which is also the component in the following additive functionals:

$$S_{h,n}(x_{0:n}, y_{1:n}) = \sum_{k=1}^{n} s^h(x_{k-1}, x_k, y_k), \qquad h = 1, \ldots, m \tag{4.7}$$

and let $S^\theta_{h,n} = \int S_{h,n}(x_{0:n}, y_{0:n}) p_\theta(x_{0:n}|y_{1:n}) dx_{0:n}$. Then the maximizing step can be completed explicitly through a suitable function:

$$\theta_{k+1} = \Lambda(n^{-1} S^{\theta_k}_n) \tag{4.8}$$

where $S^{\theta_k}_n$ is a m-dimensional vector whose hth element is $S^{\theta_k}_{h,n}$. According to Cappe (2011)[2], the lambda function can be solved by equating corresponding gradients to be zero. If the model is in an exponential family, we can transform $p_\theta(x_k, y_k|x_{k-1})$ into the following form

$$p_\theta(x_k, y_k|x_{k-1}) = v(x_k, y_k) \exp(\langle \psi(\theta), s(x_{k-1}, x_k, y_k) \rangle - A(\theta)) \tag{4.9}$$

where $\langle \cdot \rangle$ denotes the scalar product and $s(x_{k-1}, x_k, y_k) = [s^1(x_{k-1}, x_k, y_k), \ldots, s^m(x_{k-1}, x_k, y_k)]$ is the m-dimensional vector of sufficient statistics. Let $\nabla_\theta \psi(\theta) s(x_{k-1}, x_k, y_k) - \nabla_\theta A(\theta) = 0$, the $\Lambda(s)$ is the unique solution of this equation.

On-line Method Assuming $p_\theta(x_{0:k}|y_{1:k})$ is in the exponential family, then running averages of sufficient statistics are computed in on-line EM implementation. Let $\{\theta_k\}_{k=0}^n$ be the sequence of parameter estimates based on $y_{0:k}$. When the new observation, y_{k+1}, is received, we have

$$\begin{aligned} S_{h,k+1} = \gamma_{k+1} &\int s^h(x_k, x_{k+1} y_{k+1}) p_{\theta_{0:k}}(x_k, x_{k+1}|y_{1:k+1}) dx_{k:k+1} \\ &+ (1 - \gamma_{k+1}) \int \sum_{j=1}^{k} \Big(\prod_{i=j+1}^{k} (1 - \gamma_i) \Big) \gamma_j s^h(x_{j-1}, x_j, y_j) p_{\theta_{0:k}}(x_{0:k}|y_{1:k+1}) dx_{0:k} \end{aligned} \tag{4.10}$$

where $S_{h,k+1}$ is the hth element in the vector S_{k+1} and $\{\gamma_k\}_{k \geq 1}$ is the step-size sequence which reaches the same requirement as that one for gradient ascent method. The parameter estimate for next time step is then updated as $\theta_{k+1} = \Lambda(S_{k+1})$. $\prod_{i=j+1}^{k}(1 - \gamma_i)$ in (4.10) reduces the influence of early parameter estimates. Moral et al (2009) [3] proposed a SMC on-line EM algorithm based on forward smoothing as below:

1 Initialize $\widehat{T}_0^{(i)} = 0$ and obtain weighted filtering particles $\{x_0^{(i)}, \omega_0^{(i)} = \frac{1}{N}\}_{i=1}^N$

2 Repeat the following steps for time steps $k = 1, 2, \ldots, n$:

2.1 Obtain weighted filtering particles $\{x_k^{(i)}, \omega_k^{(i)}\}$ for $i = 1, \ldots, N$.

2.2 For $i = 1, \ldots, N$,

$$\widehat{T}_k^{(i)} = \frac{\sum_{j=1}^N \omega_{k-1}^{(j)} f_\theta(x_k^{(i)}|x_{k-1}^{(j)}) [(1 - \gamma_k) \widehat{T}_{k-1}^{(j)} + \gamma_k s(x_{k-1}^{(j)}, x_k^{(i)}, y_k)]}{\sum_{j=1}^N \omega_{k-1}^{(j)} f_\theta(x_k^{(i)}|x_{k-1}^{(j)})} \Big|_{\theta = \theta_{k-1}} \tag{4.11}$$

2.3 Set $\widehat{\mathcal{S}}_k = \sum_{i=1}^N \omega_k^{(i)} \widehat{T}_k^{(i)}$. Update the parameter estimate as $\theta_k = \Lambda(\widehat{\mathcal{S}}_k)$.

Again, (4.11) can be approximated by a simple MC estimate (Olsson & Westerborn (2015)) so that the step 2.2 above is replaced by

2.2 For each new particle $x_k^{(i)}, i = 1, \ldots, N$, we draw $\tilde{N} \geq 2$ indices $J_k^{(i,j)}$ through

$$P(J_k^{i,j} = l) = \frac{\omega_{k-1}^{(l)} f_\theta(x_k^{(i)}|x_{k-1}^{(l)})}{\sum_{l=1}^N \omega_{k-1}^{(l)} f_\theta(x_k^{(i)}|x_{k-1}^{(l)})}\Big|_{\theta=\theta_{k-1}}$$

Update $\widehat{T}_k^{(i)} = \tilde{N}^{-1} \sum_{j=1}^{\tilde{N}} \big((1-\gamma_k)\widehat{T}_{k-1}^{(J_k^{i,j})} + \gamma_k s(x_{k-1}^{(J_k^{i,j})}, x_k^{(i)}, y_k\big)$.

To guarantee the particle estimates are stable, we start updating parameter estimates after a few observations has been processed in practical implementations.

4.2 Liu & West Filter

Gordon et al (1993) came up with an approach to reducing degeneracy problem by adding small extra random disturbances to state particles between time steps. Such idea has been used to estimate fixed model parameters by generating particles under posterior distribution at for parameters and adding small disturbances to those particles when evolving to the next time step. If all parameters, however, are pretended to have independent random shocks that each time step, the posteriors would be eventually too diffuse to guarantee the inference precision. To avoid this problem, West (1993) suggested a modification of the original method with kernel smoothing.

Given weighted parameter particles $\{\theta_t^{(j)}, \omega_t^{(j)}\}_{j=1}^N$, a discrete Monte Carlo approximation to $p(\theta|y_{0:t})$ is then provided. It is worth noting that t suffix on θ means the particles are for time t posterior rather than they are time-varying. Let us use $\bar{\theta}_t$ and V_t denote weighted sample mean and variance matrix for the parameters. The smooth kernel density is then given by

$$p(\theta|y_{1:t}) \approx \sum_{j=1}^N \omega_t^{(j)} N(\theta|m_t^{(j)}, h^2 V_t) \tag{4.12}$$

with defining components. Such approximation is a mixture of multivariate normal distributions weighted by particle weights $\omega_t^{(j)}$. The kernel locations $m_t^{(j)}$ are developed with a shrinkage rule. Standard kernel methods would suggest $m_t^{(j)} = \theta_t^{(j)}$ so that current particles are directly used as kernel locations. The variance of the resulting mixture of normals is $(1+h^2)V_t$ that is always larger than V_t. Therefor the kernel density function is is over-dispersed relative to the posterior sample. To see this, introducing a index variable I for where the next particle comes, we have

$$\begin{aligned}
E[\theta] &= E[E[\theta|I]] = E[\theta^{(I)}] = \sum_{j=1}^N \omega_t^{(j)} \theta_t^{(j)} = \bar{\theta}_t \\
Var(\theta) &= E[Var(\theta|I)] + Var[E(\theta|I)] \\
&= E[h^2 V_t] + Var[\theta^{(I)}] \\
&= h^2 V_t + V_t \\
&= (1+h^2) V_t
\end{aligned}$$

In order to correct this problem, West (1993)[14] proposed the ideal of shrinkage of kernel locations. Let us define

$$m_t^{(j)} = a\theta_t^{(j)} + (1-a)\bar{\theta}_t \tag{4.13}$$

where $a = \sqrt{1-h^2} \in (0,1)$. It is common to set a to be 0.98 or higher. With the new kernel locations, the mixture of normals retains the mean $\bar{\theta}_t$ and variance V_t, which can be shown by

$$\begin{aligned}
E[\theta] &= E[E[\theta|I]] = E[a\theta_t^{(I)} + (1-a)\bar{\theta}_t] = a\bar{\theta}_t + (1-a)\bar{\theta}_t = \bar{\theta}_t \\
Var(\theta) &= E[Var(\theta|I)] + Var[E(\theta|I)] \\
&= E[h^2 V_t] + Var[a\theta_t^{(I)} + (1-a)\bar{\theta}_t] \\
&= h^2 V_t + Var(a\theta_t^{(I)}) \\
&= h^2 V_t + a^2 V_t \\
&= V_t
\end{aligned}$$

Based on this kernel smoothing, Liu and West (2001) gives a fixed parameter learning algorithm by incorporating APF. For state vector (x_t, θ_t), a recursion equation can be written as

$$\begin{aligned}
p(x_t, x_{t-1}, \theta_t, \theta_{t-1}|y_t, y_{0:t-1}) = \quad & p(y_t|x_{t-1}, \theta_{t-1}) p(x_{t-1}, \theta_{t-1}|y_{0:t-1}) \\
& \times p(x_t|x_{t-1}, \theta_t, y_{0:t}) p(\theta_t|\theta_{t-1}, y_{0:t})
\end{aligned} \tag{4.14}$$

Similar to APF, Liu and West use $p(y_t|g(x_{t-1}), m_{t-1})$ as the proposal weight to re-sample particles at time $t-1$ and then propagate θ_t from the proposal propagation density $p(\theta_t|\theta_{t-1})$ and propagate x_t conditional on x_{t-1}, θ_t from transition density $p(x_t|x_{t-1}, \theta_t)$. The Liu and West filtering algorithm that incorporates APF can be shown as below.

1 Given $\{x_t^{(j)}, \theta_t^{(j)}\}_{j=1}^N$ with weights $\{\omega_t^{(j)}\}_{j=1}^N$, calculate $\mu_{t+1}^{(j)} = E(x_{t+1}|x_t^{(j)}, \theta_t^{(j)})$, $m_t^{(j)} = a\theta_t^{(j)} + (1-a)\bar{\theta}_t$ and V_t.

2 ***Re-sampling*** Sample auxiliary integers $\{k^{(j)}\}_{j=1}^N$ from the set $\{1, \ldots, N\}$ with normalized probabilities $\pi_{t+1}^{(j)} \propto \omega_t^{(j)} p(y_{t+1}|\mu_{t+1}^{(j)}, m_t^{(j)})$.

3 ***Propagating parameters*** For $j = 1, \ldots, N$, sample $\theta_{t+1}^{(j)} \sim N(\cdot|m_t^{(k^{(j)})}, h^2 V_t)$.

4 ***Propagating states*** For $j = 1, \ldots, N$, sample $x_{t+1}^{(j)} \sim p(\cdot|x_t^{(k^{(j)})}, \theta_{t+1}^{(j)})$.

5 ***Re-weight*** For $j = 1, \ldots, N$, $\omega_{t+1}^{(j)} \propto \dfrac{p(y_{t+1}|x_{t+1}^{(j)}, \theta_{t+1}^{(j)})}{p(y_{t+1}|\mu_{t+1}^{(k^{(j)})}, m_t^{(k^{(j)})})}$. Normalize those weights so that they sum up to 1.

6 New particles $\{x_{t+1}^{(j)}, \theta_{t+1}^{(j)}\}_{j=1}^N$ with corresponding weights $\{\omega_{t+1}^{(j)}\}_{j=1}^N$ are acquired. If the effective number is less than a threshold, re-sample the particles based on their weights and assign $1/N$ to all particles. Set $t = t+1$ and go to step **1**.

5 Simulations

Simulations comparing forward only FFBSm-based on-line EM algorithm with PaRIS-based method are implemented on a *linear Gaussian* model and a *stochastic volatility* model. In

linear Gaussian model, we are able to compute the prediction likelihood $p(y_{t+1}|x_t)$ and sample from $p(x_{t+1}|x_t, y_{t+1})$ so **EPF** is used to generate new weighted particles at each time step. *stochastic volatility* model, such simplicities do not exist, so **APF** is applied when propagating particles with corresponding weights.

Example 1 Consider the *linear Gaussian* SSM:

$$\begin{aligned} X_{t+1} &= aX_t + \sigma_V V_t \\ Y_t &= X_t + \sigma_U U_t \end{aligned}$$

where $\{V_t\}_{t\in N}$ and $\{U_t\}_{t\in N}$ are mutually independent sequences of independent standard Gaussian variables. The parameters of this model are $\theta = (a, \sigma_V^2, \sigma_U^2)$.

$$\begin{aligned} p_\theta(x_{t+1}, y_{t+1}|x_t) &= \frac{1}{\sqrt{2\pi}\sigma_U}\exp(\frac{(y_{t+1}-x_{t+1})^2}{-2\sigma_U^2})\frac{1}{\sqrt{2\pi}\sigma_V}\exp[\frac{(x_{t+1}-ax_t)^2}{-2\sigma_V^2}] \\ &= \frac{1}{2\pi}\exp[-\frac{a^2}{2\sigma_V^2}x_t^2 + \frac{a}{\sigma_V^2}x_t x_{t+1} - \frac{1}{2\sigma_V^2}x_{t+1}^2 - \frac{1}{2\sigma_U^2}(y_{t+1}-x_{t+1})^2 \\ &\quad - \frac{1}{2}\ln(\sigma_V^2) - \frac{1}{2}\ln(\sigma_U^2)] \end{aligned}$$

Let us put this likelihood function into the following form

$$p_\theta(x_{t+1}, y_{t+1}|x_t) = h(x_{t+1}, y_{t+1})\exp(\langle\psi(\theta), s(x_t, x_{t+1}, y_{t+1})\rangle - A(\theta))$$

We have $h(x_{t+1}, y_{t+1}) = \frac{1}{2\pi}$, $\theta = (a, \sigma_V^2, \sigma_U^2)$, $s(x_t, x_{t+1}, y_{t+1}) = (x_t^2, x_t x_{t+1}, x_{t+1}^2, (y_{t+1} - x_{t+1})^2)' = (z_1, z_2, z_3, z_4)'$, $\psi(\theta) = (\frac{a^2}{-2\sigma_V^2}, \frac{a}{\sigma_V^2}, \frac{1}{-2\sigma_V^2}, \frac{1}{-2\sigma_U^2})$ and $A(\theta) = \frac{1}{2}\ln(\sigma_V^2) + \frac{1}{2}\ln(\sigma_U^2)$.

The complete-data maximum likelihood equation $\nabla_\theta\psi(\theta)s - \nabla_\theta A(\theta) = 0$ has a unique solution, which is derived by the following steps.

$$\nabla_\theta\psi(\theta) = \begin{bmatrix} \frac{a}{-\sigma_V^2} & \frac{1}{\sigma_V^2} & 0 & 0 \\ \frac{a^2}{2\sigma_V^4} & \frac{-a}{\sigma_V^4} & \frac{1}{2\sigma_V^4} & 0 \\ 0 & 0 & 0 & \frac{1}{2\sigma_U^4} \end{bmatrix}, \quad s = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}, \quad \nabla_\theta A(\theta) = \begin{bmatrix} 0 \\ \frac{1}{2\sigma_V^2} \\ \frac{1}{2\sigma_U^2} \end{bmatrix}$$

$$\nabla_\theta\psi(\theta)s = \begin{bmatrix} \frac{a}{-\sigma_V^2}z_1 + \frac{1}{\sigma_V^2}z_2 \\ \frac{a^2}{2\sigma_V^4}z_1 + \frac{-a}{\sigma_V^4}z_2 + \frac{1}{2\sigma_V^4}z_3 \\ \frac{1}{2\sigma_U^4}z_4 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{2\sigma_V^2} \\ \frac{1}{2\sigma_U^2} \end{bmatrix}$$

Thus, the unique solution is $\theta(s) = \Lambda(z_1, z_2, z_3, z_4) = (\frac{z_2}{z_1}, z_3 - \frac{z_2^2}{z_1}, z_4)$. Let us compare the parameter estimates produced by the PaRIS-based algorithm and those obtained by FFBSm-based algorithm. The observations $\{Y_t\}_{t\geq 1}$ are generated by simulation with parameter $\theta = (0.8, 0.4^2, 0.9^2)$. We use 400 particles for the FFBSm-based algorithm and 400 particles with $\tilde{N} = 6$ for the PaRIS-based algorithm. Assuming the third parameter $\sigma_U^2 = 0.9^2$ is known and fixed, then only the first unknown parameters require estimations.

The initial parameters in each algorithm are set to be $\theta_0 = (0.2, 1.5^2, 0.9^2)$ and we start updating a and σ_V^2 after 60 steps. In addition, we set the step size $\gamma_t = t^{-0.6}$ in both algorithms. Figure 6 and 7 display outputs of these two algorithms. It clearly shows both algorithms tend towards the true parameters at about 10000 observations and their estimates are convergent for the remaining observations from 10000 to 50000. PaRIS-based algorithm use a Monte Carlo

estimate in one step so it has a $O(N\tilde{N})$ complexity while the other algorithm requires a $O(N^2)$ complexity.

Revisiting our simulations, in IML environment, the Paris-based algorithm (20min) takes two thirds of the time needed for the FFBSm-based algorithm (30min) though they both have 400 particles. Therefore, given a fixed time interval to implement parameter estimates, it is expected that PaRIS-based algorithm would have a lower variance as we can assign it a larger particle sample size.

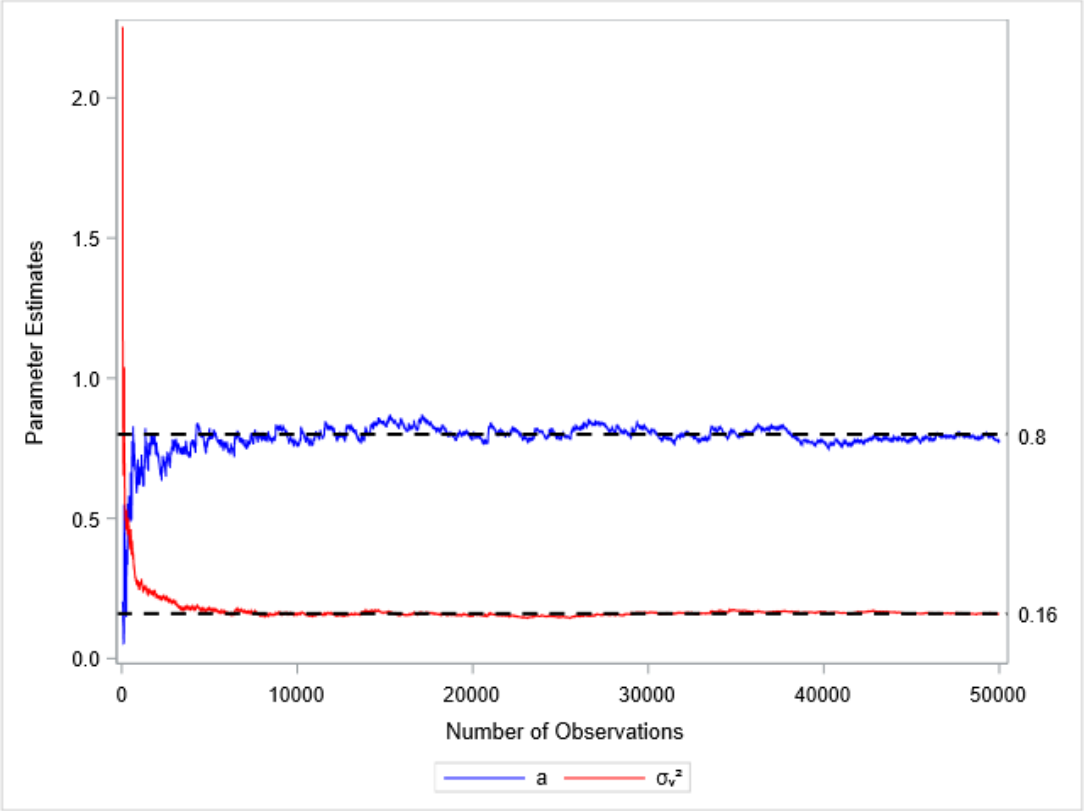

Figure 6: FFBSm-based on-line EM Algorithm, *linear Gaussian* model

Example 2 Consider the *Stochastic Volatility* SSM:

$$X_{t+1} = \phi X_t + \sigma V_t$$
$$Y_t = \beta \exp(X_t/2) U_t$$

where $\{V_t\}_{t\in N}$ and $\{U_t\}_{t\in N}$ are mutually independent sequences of independent standard Gaussian variables. The parameters of this model are $\theta = (\phi, \sigma^2, \beta^2)$.

$$\begin{aligned} p_\theta(x_{t+1}, y_{t+1}|x_t) &= \frac{1}{\sqrt{2\pi}\beta\exp(x_{t+1}/2)} \exp\left(\frac{y_{t+1}^2}{-2\beta^2\exp(x_{t+1})}\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(x_{t+1}-\phi x_t)^2}{-2\sigma^2}\right) \\ &= \frac{1}{2\pi\exp\left(x_{t+1}/2\right)} \exp\left(-\frac{\phi^2 x_t^2}{2\sigma^2} + \frac{\phi x_t x_{t+1}}{\sigma^2} - \frac{x_{t+1}^2}{2\sigma^2} - \frac{y_{t+1}^2\exp(-x_{t+1})}{2\beta^2} - \frac{1}{2}\ln(\beta^2\sigma^2)\right) \end{aligned}$$

Let us put this likelihood function into the following form

$$p_\theta(x_{t+1}, y_{t+1}|x_t) = h(x_{t+1}, y_{t+1}) \exp(\langle \psi(\theta), s(x_t, x_{t+1}, y_{t+1})\rangle - A(\theta))$$

where $h(x_{t+1}, y_{t+1}) = \dfrac{1}{2\pi\exp(x_{t+1}/2)}$, $s(x_t, x_{t+1}, y_{t+1}) = \left(x_t^2, x_t x_{t+1}, x_{t+1}^2, y_{t+1}\exp(-x_{t+1})\right) = (z_1, z_2, z_3, z_4)$, $\psi(\theta) = \left(-\dfrac{\phi^2}{2\sigma^2}, \dfrac{\phi}{\sigma^2}, -\dfrac{1}{2\sigma^2}, -\dfrac{1}{2\beta^2}\right)$ and $A(\theta) = \frac{1}{2}\ln(\sigma^2) + \frac{1}{2}\ln(\beta^2)$.

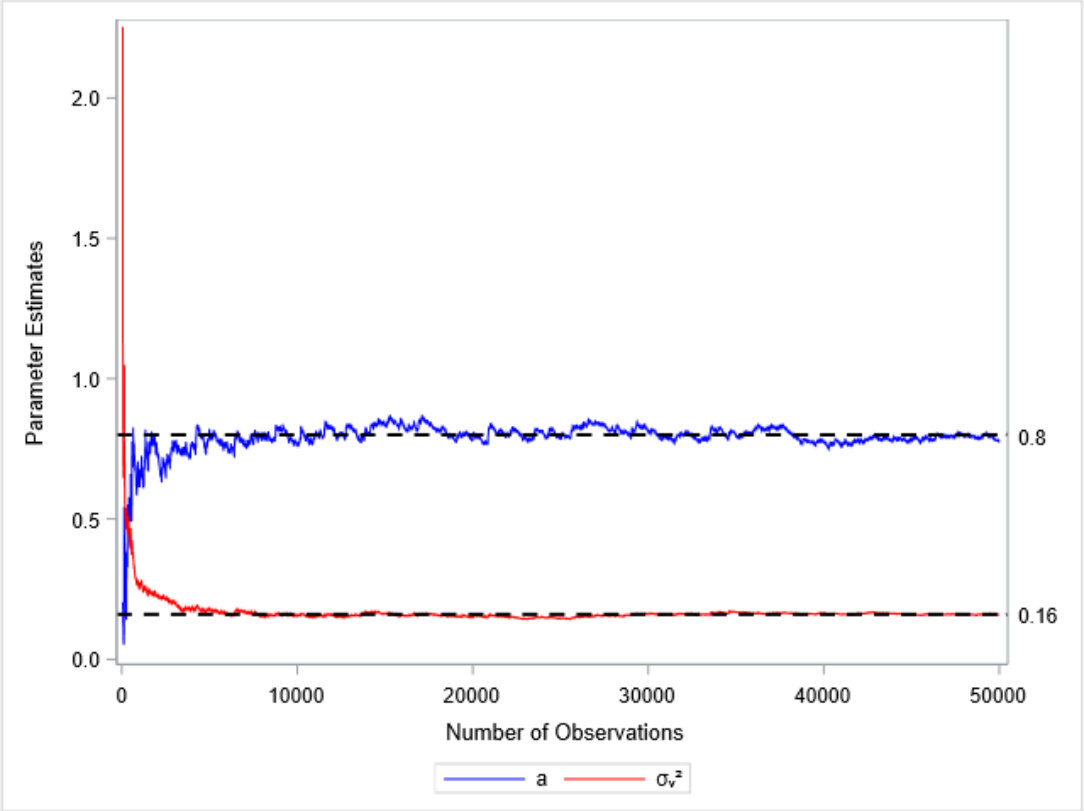

Figure 7: PaRIS-based on-line EM Algorithm, *linear Gaussian* model

The complete-data maximum likelihood equation $\nabla_\theta \psi(\theta) s - \nabla_\theta A(\theta) = 0$ has a unique solution, which is derived by the following steps.

$$\nabla_\theta \psi(\theta) = \begin{bmatrix} -\frac{\phi}{\sigma^2} & \frac{1}{\sigma^2} & 0 & 0 \\ \frac{\phi^2}{2\sigma^4} & -\frac{\phi}{\sigma^4} & \frac{1}{2\sigma^4} & 0 \\ 0 & 0 & 0 & \frac{1}{2\beta^4} \end{bmatrix}, \quad s = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}, \quad \nabla_\theta A(\theta) = \begin{bmatrix} 0 \\ \frac{1}{2\sigma^2} \\ \frac{1}{2\beta^2} \end{bmatrix}$$

$$\nabla_\theta \psi(\theta) s = \begin{bmatrix} -\frac{\phi}{\sigma^2} z_1 + \frac{1}{\sigma^2} z_2 \\ \frac{\phi^2}{2\sigma^4} z_1 - \frac{\phi}{\sigma^4} z_2 + \frac{1}{2\sigma^4} z_3 \\ \frac{1}{2\beta^4} z_4 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{2\sigma^2} \\ \frac{1}{2\beta^2} \end{bmatrix}$$

Thus, the unique solution is $\theta(s) = \Lambda(z_1, z_2, z_3, z_4) = (\frac{z_2}{z_1}, z_3 - \frac{z_2^2}{z_1}, z_4)$. Let us compare the parameter estimates produced by the PaRIS-based algorithm and those obtained by FFBSm-based algorithm. The observations $\{Y_t\}_{t \geq 1}$ are generated by simulation with parameter $\theta = (0.8, 0.1, 1)$. We use 250 particles for the FFBSm-based algorithm and 400 particles with $\tilde{N} = 6$ for the PaRIS-based algorithm.

The initial parameters in each algorithm are set to be $\theta_0 = (0.1, 1, 2)$ and we start updating parameter estimates after 60 steps. In addition, we set the step size $\gamma_t = t^{-0.6}$ for $t \leq 10^5$ and $(t - 20000)^{-0.6}$ for $t > 10^5$ in both algorithms.

Figure 8 and 9 display outputs of these two algorithms. It clearly shows both algorithms tend towards the true parameters at about 4×10^5 observations and their estimates are convergent for the remaining observations from 4×10^5 to 10^6. PaRIS-based algorithm use a Monte Carlo estimate in one step so it has a $O(N\tilde{N})$ complexity while the previous algorithm requires a

$O(N^2)$ complexity. Given a fixed time interval to implement parameter estimates, it is expected that PaRIS-based algorithm would have a lower variance as we can assign it a larger particle sample size.

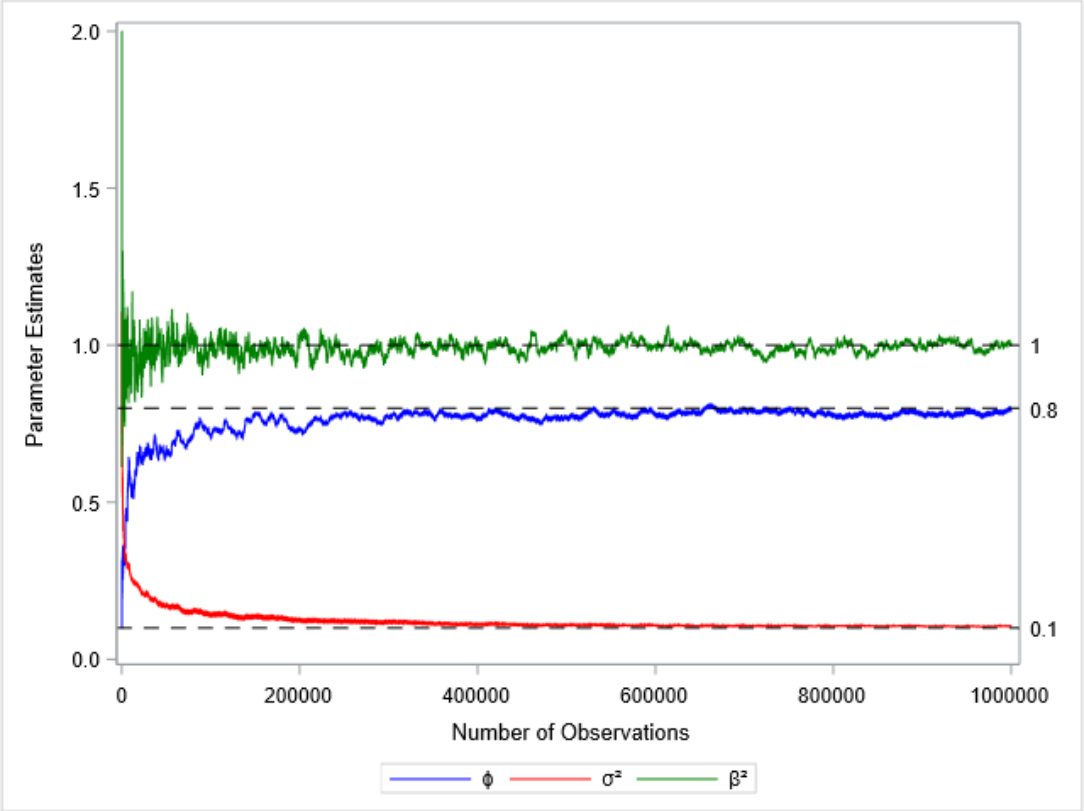

Figure 8: FFBSm-based on-line EM Algorithm, *stochastic volatility* model

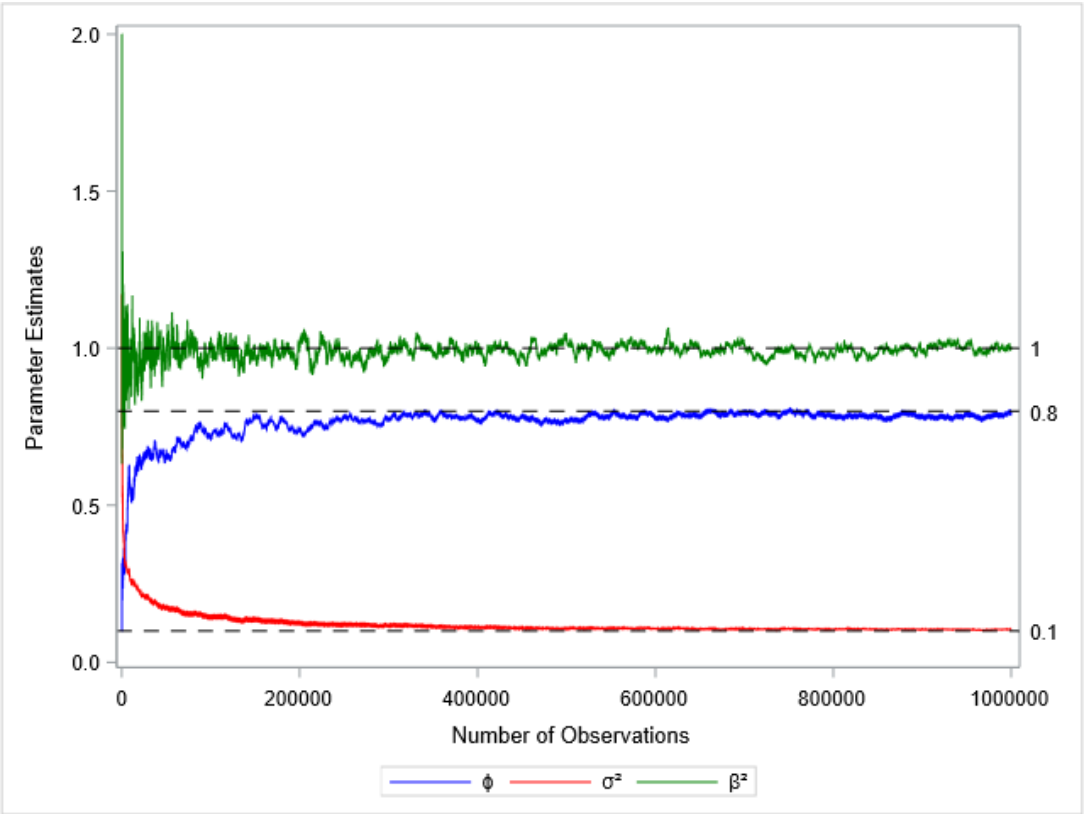

Figure 9: PaRIS-based on-line EM Algorithm, *stochastic volatility* model

References

[1] Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89, 2010.

[2] Olivier Cappé. Online em algorithm for hidden markov models. *Journal of Computational and Graphical Statistics*, 20(3):728–749, 2011.

[3] Pierre Del Moral, Arnaud Doucet, and Sumeetpal Singh. Forward smoothing using sequential monte carlo. *arXiv preprint arXiv:1012.5390*, 2010.

[4] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[5] Arnaud Doucet, Simon J Godsill, and Mike West. Monte carlo filtering and smoothing with application to time-varying spectral estimation. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II701–II704. IEEE, 2000.

[6] Paul Fearnhead, David Wyncoll, and Jonathan Tawn. A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447–464, 2010.

[7] Pedro F Felzenszwalb, Daniel P Huttenlocher, and Jon M Kleinberg. Fast algorithms for large-state-space hmms with applications to web usage analysis. In *Advances in neural information processing systems*, pages 409–416, 2004.

[8] Simon Godsill, Arnaud Doucet, and Mike West. Maximum a posteriori sequence estimation using monte carlo particle filters. *Annals of the Institute of Statistical Mathematics*, 53(1):82–96, 2001.

[9] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

[10] Michael Johannes and Nicholas Polson. Particle filtering. *Handbook of Financial Time Series*, pages 1015–1029, 2009.

[11] Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3):328–351, 2015.

[12] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[13] François LeGland and Laurent Mevel. Recursive estimation in hidden markov models. In *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, volume 4, pages 3468–3473. IEEE, 1997.

[14] Jane Liu and Mike West. Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo methods in practice*, pages 197–223. Springer, 2001.

[15] Jimmy Olsson and Johan Westerborn. An efficient particle-based online em algorithm for general state-space models** this work is supported by the swedish research council, grant 2011-5577. *IFAC-PapersOnLine*, 48(28):963–968, 2015.

[16] Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

[17] Johan Westerborn and Jimmy Olsson. Efficient particle-based online smoothing in general hidden markov models. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 8003–8007. IEEE, 2014.