Stat689: Statistical Methods for Finance

Joint estimation in Dynamic State-Space Models

Instructor: Jianhua Huang *Soma S Dhavala*

Abstract

In this report, we consider the joint estimation of state sequences and the (static) parameters in a dynamic state space model. In particular, we review the algorithms presented in Polson et al (2008) that use rolling-window Sequential Monte Carlo techniques. The joint estimation is performed by repeatedly alternating between parameter estimation *given the states* and the state estimation *given the parameters*. In the state estimation phase, smoothing & filtering densities of states in the current window are updated by using their estimates available outside the window backwards in time. In the parameter estimation phase, sufficient statistics based on the entire history of smoothed states up to the current time point are used to make the algorithm run in linear time. We review Forward-Filtering Backward-Sampling (FFBS) algorithm that forms the back bone of many Monte-Carlo simulation strategies and provide the theory (with proofs) needed to adapt the FFBS to the rolling-window case. We discuss some possible variations and present an ad-hoc (vague in theory) algorithm that was conceived in the due course. We implement the algorithms on a simulated data-set from an AR(1) plus noise model[†].

Introduction

A dynamic state-space model has two components: state evolution equation and the observation equation, described by some parameter. In general, the states are not observed and only some function the these states is observed. It is often of interest to infer the states given the measurements and/or the parameters that describe the state evolution and measurement process. The model can be expressed as:

Model:

$$\begin{aligned}
\text{Observation :} \quad & \mathbf{y}_t \sim g(\mathbf{y}_t|\mathbf{x}_t, \theta) \\
\text{State evolution:} \quad & \mathbf{x}_t \sim f(\mathbf{x}_t|\mathbf{x}_{t-1}, \theta) \\
\text{Initial state:} \quad & \mathbf{x}_0 \sim \pi(\mathbf{x}_0|\theta_0)
\end{aligned}$$

where θ is the parameter vector, $\mathbf{x}_t$ the state vector at time t, $\mathbf{y}_t$ the observation vector at time t, g the observation density and f the evolution density and some assumptions in the model are:

Assumptions:

$$\begin{aligned}
f(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}..., \theta) &= f(\mathbf{x}_t|\mathbf{x}_{t-1}, \theta) \\
g(\mathbf{y}_t|\mathbf{x}_t, \mathbf{x}_{t-1}..., \theta) &= g(\mathbf{y}_t|\mathbf{x}_t, \theta).
\end{aligned}$$

[†]For a general introduction, motivation and background of the work, please refer to the companion report by Krista Rister. You can also find rolling-window Monte-Carlo algorithm with known parameter case in her report. I will spend time in providing proofs and discussing the algorithm instead

As a consequence, the following results will be useful:

Identities:

$$\begin{aligned} f(\mathbf{x}_t|\mathbf{x}_{t+1,\ldots},\mathbf{y}_{1:T}) &= f(\mathbf{x}_t|\mathbf{x}_{t+1,\ldots},\mathbf{y}_{1:t})T \geq t \\ f(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k},\mathbf{y}_{1:T}) &= f(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k},\mathbf{y}_{t-k+1:t}) \end{aligned}$$

The above model is quite general and encompasses several well-studied models such as noisy AR process, for example. The above model is very well studied when θ is known: Kalman filtering and a wide variety of its extensions, for example. Similarly, time-series analysis specifically deals with the case where θ is unknown but the states are known. However, there is not much literature available to tackle the problem when both parameter and states are unknown, relatively speaking. The focus of the report is the joint estimation of states and parameters, specifically the algorithms proposed in Polson et al (2008). In the next Section, we will briefly review sequential Monte Carlo, establish the notation and terminology that will be used in the subsequent Sections.

Sequential Monte Carlo methods

Inference in the state-space model is based on the joint density of the states (and parameters if they are known) given the observed data, i.e., we want to know

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

For example, we can obtain point estimates for the state from the joint density by taking the expecttion over the posterior marginal density. The marginal density can be obtained by integrating-out the nuisance states. However, it is often the case that, analytical form for the joint density is not available or it is very complex to compute. Monte-Carlo simulation techniques offer practical alternatives to analytical computations. Further, they can be suitably adapted to update the joint densities as and when new observations arrive, i.e.,

$$p(\mathbf{x}_{0:t+1},\theta|\mathbf{y}_{1:t+1}) \xleftarrow{y_{t+1}} p(\mathbf{x}_{0:t},\theta|\mathbf{y}_{1:t})$$

Monte Carlo techniques that accomplish task similar to what was mentioned above are usually referred to as sequential Monte Carlo methods (SMC). Three major tasks have to be performed in updating the joint density with the newest observation. The three tasks are (θ is dropped for brevity):

$$\begin{aligned} \text{Filtering :} &\quad p(\mathbf{x}_t,|\mathbf{y}_{1:t}) \\ \text{Prediction :} &\quad p(\mathbf{x}_{t+k},|\mathbf{y}_{1:t}), k > 0 \\ \text{Smoothing :} &\quad p(\mathbf{x}_{t-k},|\mathbf{y}_{1:t}), 1 \leq k < t. \end{aligned}$$

Using the Bayes rule

$$p(A|B) \quad \propto \quad p(B|A)p(A)$$

and the chain rule of probability

$$p(A,B,C) \quad = \quad p(A)p(B|A)p(C|A,B)$$

we can establish any recursive relationships. In particular,

<u>Recursion for the joint density:</u>

Goal:

$$p(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}) \xleftarrow{\mathbf{y}_{1:t}} p(\mathbf{x}_0)$$

Factorize:

$$\begin{aligned} p(\mathbf{x}_{1:T}|\mathbf{y}_{1:T}) &= p(\mathbf{x}_T|\mathbf{y}_{1:T}) \prod_{t=0}^{T-1} p(\mathbf{x}_t|\mathbf{x}_{t+1:T}, \mathbf{y}_{1:T}) \\ p(\mathbf{x}_t|\mathbf{x}_{t+1:T}\mathbf{y}_{1:T}) &= p(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}) \\ &= \frac{p(\mathbf{x}_t|\mathbf{y}_{1:t})f(\mathbf{x}_{t+1}|\mathbf{x}_t)}{p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t})} \\ &\propto p(\mathbf{x}_t|\mathbf{y}_{1:t})f(\mathbf{x}_{t+1}|\mathbf{x}_t) \end{aligned}$$

The above factorization helps us in obtaining draws from the joint density. We can see from the above equations that we should be able to efficiently (in a recursive manner) generate the filtering distribution. It can be obtained as:

<u>Recursion for filtering:</u>

Goal:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \xleftarrow{y_t} p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$$

Recursion:

$$\begin{aligned} \text{predict :} \quad & p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})f(\mathbf{x}_t|\mathbf{x}_{t-1})d\mathbf{x}_t \\ \text{update :} \quad & p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto g(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \end{aligned}$$

Now we can describe the algorithm (in terms of the densities) to obtain the joint density that is referred to as forward-filtering, backward-sampling as:

step-1:

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \xrightarrow[\text{for t = 1:T}]{\text{forward-filtering}} p(\mathbf{x}_t|\mathbf{y}_{1:t})$$

step-2:

$$p(\mathbf{x}_t|\mathbf{x}_{t+1:T}, \mathbf{y}_{1:T}) \xleftarrow[\text{for t = T-1:1}]{\text{backward-sampling}} p(\mathbf{x}_t|\mathbf{y}_{1:t})$$

Obtaining draws from the joint densities relies heavily on the filtering densities. There can be several approaches to sampling from the filtering densities and use them to update the future filtering densities (step-1) or updating the smoothing densities (step-2). One technique that efficiently adjusts the samples (realizations) and is easy to update is sequential importance. Suppose that we have samples, $\mathbf{x}_{t-1}^{(i)}$, from the filtering density each with a a weight $w_{t-1}^{(i)}$, i.e, we can approximate the density as a p.m.f given by:

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

where δ is indicator function and $\sum_{i=1}^{N} w_{t-1}^{(i)} = 1$.

Now, in order to obtain the filtering density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, do

- for $i = 1 \dots N$
  - draw $\mathbf{x}_t^{(i)}$ from $f(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$
  - update the weights $w_t^{(i)} = w_{t-1}^{(i)} g(\mathbf{y}_t|\mathbf{x}_t^{(i)})$
  - normalize the weights s.t $\sum_{i=1}^{N} w_t^{(i)} = 1$

Drawing $\mathbf{x}_t^{(i)}$ from the state evolution is just one of the many choices available. In fact, one can choose any proposal distribution based on an educated guess about how close it is to the target distribution and being easy to sample from, at the same time. Weights have to modified accordingly, of course! In fact, the state evolution density may be a poor choice sometimes. It could lead to particle degeneration which means that some weights tend to be too large and often the particles get stuck in their vicinity. Many strategies have been suggested in the literature. However, in our simulations we have chosen the state density itself as it was easy to draw from. Likewise, to obtain the smoothing density $p(\mathbf{x}_{t-1}|\mathbf{x}_{t:T}^{i})\mathbf{y}_{1:T})$, do

- for $i = 1 \dots N$
  - update the weights $w_{t-1}^{(i)} = w_{t-1}^{(i)} f(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})$
  - normalize the weights s.t $\sum_{i=1}^{N} w_t^{(i)} = 1$

Having established the method for regular SMC, we will now discuss the rolling-window SMC in the next Section.

Rolling-window SMC methods

In the regular SMC, we obtained the filtering densities and the smoothening densities by conditioning on the look-ahead observations and states. The forward-filtering goes forward in time all the way up to T, the last observation. And the smoothening distribution samples the states (by updating the weights that were updated in the forward filtering phase) backwards in time to the time origin. As we can see, each state is updated exactly twice. Like any iterative algorithm, we might need to scan the data several times, adjust the weights several times to reach the point of convergence. This might be particularly true when the parameter are unknown. In the rolling-window SMC (rSMC), each state is sampled about twice the length of the window, thereby allowing the particles to mix and settle and reach the stationary distribution. We will first discuss rSMC when the parameters are fixed (known) and reserve the next subsection for the unknown parameter case.

Fixed parameters

The central idea in rSMC is based on the following representation:

$$\begin{aligned} p(\mathbf{x}_{t-k+1:t}|\mathbf{y}_{1:t}) &= \int p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t}) p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t-k} \\ &= \int p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1:t}) p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t}) d\mathbf{x}_{t-k} \\ &\approx \int p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}, \mathbf{y}_{t-k+1:t}) p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-k} \end{aligned}$$

Due the to approximation in the last step above, we can reuse the particles $\mathbf{x}_{t-k}^{(i)}$ from $p(\mathbf{x}_{t-k}|\mathbf{y}_{1:t-1})$. Now, suppose that we can evaluate $p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}^{(i)}, \mathbf{y}_{1:t})$. Then we can evaluate the integral without any difficulty just by performing the monte-carlo average. In fact, our goal is to generate the samples $\mathbf{x}_{t-k+1:t}^{(i)}$, so we do not even need to evaluate the integral. However, a strong requirement here is the ability t draw sampled from $p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}^{(i)}, \mathbf{y}_{1:t})$. It turns out that we can adapt the FFBS in this case too as detailed below:

Recursion for the joint density:

Goal:

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

Factorize:

$$\begin{aligned} p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}, \mathbf{y}_{1:t}) &= p(\mathbf{x}_t|\mathbf{x}_{t-k}, \mathbf{y}_{1:t}) \prod_{j=t-k+1}^{t-1} p(\mathbf{x}_j|\mathbf{x}_{j+1,t-k}, \mathbf{y}_{1:t}) \\ p(\mathbf{x}_j|\mathbf{x}_{j+1,t-k}, \mathbf{y}_{1:t}) &= p(\mathbf{x}_j|\mathbf{x}_{j+1,t-k}, \mathbf{y}_{1:j}) \\ &= \frac{p(\mathbf{x}_j|\mathbf{x}_{t-k}, \mathbf{y}_{1:j}) f(\mathbf{x}_{j+1}|\mathbf{x}_j)}{p(\mathbf{x}_{j+1}|\mathbf{x}_{t-k}, \mathbf{y}_{1:j})} \\ &\propto p(\mathbf{x}_j|\mathbf{x}_{t-k}, \mathbf{y}_{1:j}) f(\mathbf{x}_{j+1}|\mathbf{x}_j) \end{aligned}$$

As before, forward filtering recursion is obtained as:

Recursion for filtering:

Goal:

$$p(\mathbf{x}_t|\mathbf{x}_{t-k}, \mathbf{y}_{1:t}) \xleftarrow{y_t} p(\mathbf{x}_{t-1}|\mathbf{x}_{t-k}, \mathbf{y}_{1:t-1})$$

Recursion:

$$\begin{aligned} \text{predict :} \quad & p(\mathbf{x}_t|\mathbf{x}_{t-k}, \mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_{t-1}|\mathbf{x}_{t-k}, \mathbf{y}_{1:t-1}) f(\mathbf{x}_t|\mathbf{x}_{t-1}) d\mathbf{x}_t \\ \text{update :} \quad & p(\mathbf{x}_t|\mathbf{x}_{t-k}, \mathbf{y}_{1:t}) \propto g(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-k}, \mathbf{y}_{1:t-1}) \end{aligned}$$

Now we can describe the FFBS algorithm (in terms of the particles) to obtain the joint density.

Suppose that we have samples, $(\mathbf{x}_{j-1}^{(i)}, t-k < j \leq \mathrm{t})$, from the filtering density $p(\mathbf{x}_{j-1}|\mathbf{x}_{t-k}, \mathbf{y}_{1:t-1})$ each with a weight $w_{j-1}^{(i)}$. In order to obtain the filtering density $p(\mathbf{x}_j|\mathbf{x}_{t-k}, \mathbf{y}_{1:t})$, do

- for $i = 1 \dots N$
  - draw $\mathbf{x}_j^{(i)}$ from $f(\mathbf{x}_j|\mathbf{x}_{j-1}^{(i)})$
  - update the weights $w_j^{(i)} = w_{j-1}^{(i)} g(\mathbf{y}_t|\mathbf{x}_j^{(i)})$
  - normalize the weights s.t $\sum_{i=1}^N w_t^{(i)} = 1$

To obtain the smoothing density $p(\mathbf{x}_{j-1}|\mathbf{x}_{t-k,j:t}^{(i)}, \mathbf{y}_{1:t})$, do

- for $i = 1 \dots N$
  - update the weights $w_{j-1}^{(i)} = w_{j-1}^{(i)} f(\mathbf{x}_j^{(i)}|\mathbf{x}_{j-1}^{(i)})$
  - normalize the weights s.t $\sum_{i=1}^N w_j^{(i)} = 1$

The algorithm for learning states when the parameters are **known** is described below:

Algorithm 1: Filtering with Fixed parameters

```
• For each time period $t = k+1, \dots, T$
    – For each sample path $i = 1, \dots, N$
        1. Run an (FFBS) MCMC with stationary distribution $p(\mathbf{x}_{t-k+1:t}|\mathbf{x}_{t-k}^{(i)}, \mathbf{y}_{1:t})$
        2. Define $\mathbf{x}_{t-k+1:t}^{(i)}$ as the last value of $(\mathbf{x}_{t-k+1:t})$ in the chain
        3. Store $\mathbf{x}_{t-k+1:t}^{(i)}$ as draw from $p(\mathbf{x}_{t-k+1}|\mathbf{y}_{1:t})$
        4. Report $\mathbf{x}_t^{(i)}$ as draw from $p(\mathbf{x}_t|\mathbf{y}_{1:t})$
```

The algorithm requires three inputs: N, the number sample paths in the MCMC and G, the number of MCMC iterations to obtain one draw from the fixed-lag smoothing distribution (the number times we update weights for each state in the window, in our simulations we used one cycle) and the length of the rolling-window size k. The choice of the window length depends on how well the following approximation holds for that chosen window length:

$$\begin{aligned} p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t}) &= \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t-1}) \\ &\approx p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t-1}) \text{ iff} \\ p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) &\approx p(\mathbf{y}_t|\mathbf{x}_{0:t-k}, \mathbf{y}_{1:t-1}) \end{aligned}$$

The last approximation hold good when $\mathbf{x}_{0:t-k}$ and $\mathbf{t}_t$ are independent *given* $\mathbf{y}_{1:t-1}$. One approach to finding the optimal k is by choosing the k that minimizes the distance between the two densities. The authors suggest that using a window of size 15 to 25 is good enough for practical purposes but it may depend on the specific problem. Also note that the algorithm starts at $t = k+1$. We need to run a regular or full FFBS to get the histories of the states given the data up to $t = k$. Then the *practical filtering* (that is what the authors call it) algorithm can begin. So far, we assumed that we know the parameters. Now we turn our attention the case where the parameters are unknown.

Unknown parameters

Our goal is to obtain the joint density of states as well as the parameters. More specifically, obtain $p(\mathbf{x}_{0:t}, \theta|\mathbf{y}_{1:t})$. We can accomplish this by noting that:

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

However, the requirement is the ability to draw from $p(\mathbf{x}_{t-k+1:t}, \theta|\mathbf{x}_{0:t-k}, \mathbf{y}_{t-k+1:t})$ given the draws from $p(\mathbf{x}_{0:t-k}|\mathbf{y}_{1:t-1})$. We can accomplish this by repeatedly alternating between conditional distribution of states *given* the parameters and *vice versa*, i.e.,

$$p(\mathbf{x}_{t-k+1:t}|\theta, \mathbf{x}_{t-k}, \mathbf{y}_{1:t}) \leftrightarrows p(\theta|\mathbf{x}_{t-k+1:t}, \mathbf{x}_{t-k}, \mathbf{y}_{1:t})$$

We must note that conditioned on θ, we do not need full histories of the states but only states up to $t-k$ are sufficient due to the Markovian property (model assumption). Whereas, the conditional distribution requires all the histories up to t. The algorithm for learning states when the parameters are **unknown** is described below:

Algorithm 2: Filtering with unknown parameters

```
• For each time period $t = k+1, \dots, T$
    – For each sample path $i = 1, \dots, N$
        1. Run an (FFBS) MCMC with stationary distribution $p(\mathbf{x}_{t-k+1:t}, \theta|\mathbf{x}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
            (a) draw from $p(\mathbf{x}_{t-k+1:t}|\theta^{(i)}\mathbf{x}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
            (b) draw from $p(\theta|\mathbf{x}^{(i)}_{t-k+1:t}, \mathbf{x}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
        2. Define $(\mathbf{x}^{(i)}_{t-k+1:t}, \theta^{(i)}_t)$ as the last value of $(\mathbf{x}_{t-k+1:t}, \theta)$ in the chain
        3. Store $\mathbf{x}^{(i)}_{0:t-k+1:t}$ as draw from $p(\mathbf{x}_{0:t-k+1}|\mathbf{y}_{1:t})$
        4. Report $(\mathbf{x}^{(i)}_t, \theta^{(i)})$ as draw from $p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t})$
```

In the above implementation, we need to draw samples from the conditional distribution of the parameters as well (that is the major difference from Algorithm 1). Another variation that is possible is the use of plug-in estimators while alternating between conditional distributions, i.e.,

$$p(\mathbf{x}_{t-k+1:t}|E[\theta], \mathbf{x}_{t-k}, \mathbf{y}_{1:t}) \leftrightarrows p(\theta|E[\mathbf{x}_{t-k+1:t}], \mathbf{x}_{t-k}, \mathbf{y}_{1:t})$$

It is particularly suitable when the dynamic state-space model reduces to the familiar linear models given the states and closed-form solutions exist for $E[\theta]$ and Monte-Carlo estimates can be replaced in the place of $E(\mathbf{x}_t)$. This is ad-hoc because, I did study the properties of the algorithm or established any theoretical results in this. It would be interesting to study those properties. With the plug-in estimator based algorithm is summarized here:

Algorithm 2a[§]: Filtering with unknown parameters (plug-in estimators)

```
• For each time period $t = k+1, \ldots, T$
    – For each sample path $i = 1, \ldots, N$
        1. Run an (FFBS) MCMC with stationary distribution $p(\mathbf{x}_{t-k+1:t}, \theta | \mathbf{x}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
            (a) draw from $p(\mathbf{x}_{t-k+1:t} | \hat{\theta}^{\mathbf{t-1}} \mathbf{x}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
            (b) update $\hat{\theta}^t = E\left[p(\theta | \mathbf{x}^{(i)}_{t-k+1:t}, \mathbf{x}^{(i)}_{t-k}, \mathbf{y}_{1:t})\right]$
        2. Define $(\mathbf{x}^{(i)}_{t-k+1:t}, \theta^{(i)}_t)$ as the last value of $(\mathbf{x}_{t-k+1:t}, \theta)$ in the chain
        3. Store $\mathbf{x}^{(i)}_{0:t-k+1:t}$ as draw from $p(\mathbf{x}_{0:t-k+1} | \mathbf{y}_{1:t})$
        4. Report $(\mathbf{x}^{(i)}_t, \theta^{(i)})$ as draw from $p(\mathbf{x}_t, \theta | \mathbf{y}_{1:t})$
```

However, the technical difficulty is, in the FFBS step,we are using sequential importance sampling and not Markov-Chain Monte Carlo methods. This makes the computation in 1(a) above difficult. Instead, we can modify the above algorithm so that sequential Monte-Carlo weights can be effectively used in the parameter estimation phase. The importance weights in the FFBS step are reweighed in the *backward-sampling* step to obtain $p(\mathbf{x}_{j-1} | \theta, \mathbf{x}^{(i)}_{t-k,j:t}, \mathbf{y}_{1:t})$ as:

- for $i = 1 \ldots N$
  - update the weights $w^{(i)}_{j-1} = w^{(i)}_{j-1} f(\mathbf{x}^{(i)}_j | \mathbf{x}^{(i)}_{j-1})$
- re-sample $\mathbf{x}^{(i)}_{j-1}$ so that $w^{(i)}_j \approx \frac{1}{N} \forall i$
- normalize the weights s.t $\sum_{i=1}^{N} w^{(i)}_j = 1$

. Then Algorithm 2a can be modified so that we can use FFBS for the states as:

Algorithm 2b[§] : Filtering with unknown parameters (plug-in estimators)

```
• For each time period $t = k+1, \ldots, T$
    – For each sample path $i = 1, \ldots, N$
        1. Run an (FFBS) MCMC with stationary distribution $p(\mathbf{x}_{t-k+1:t} | \theta^t \mathbf{x}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
        2. Define $(\mathbf{x}^{(i)}_{t-k+1:t}, \theta_t)$ as the last value of $(\mathbf{x}_{t-k+1:t}, \theta)$ in the chain
        3. Store $\mathbf{x}^{(i)}_{0:t-k+1:t}$ as draw from $p(\mathbf{x}_{0:t-k+1} | \mathbf{y}_{1:t})$
        4. Report $(\mathbf{x}^{(i)}_t, \theta^{(i)})$ as draw from $p(\mathbf{x}_t, \theta | \mathbf{y}_{1:t})$
• obtain $\hat{\mathbf{x}}^t_{\mathbf{t-k+1:t}} = \frac{1}{N} \sum_1^N \mathbf{x}^{(i)}_{t-k+1:t}$
• update $\hat{\theta}^t = E\left[p(\theta | \hat{\mathbf{x}}_{t-k+1:t}, \hat{\mathbf{x}}_{t-k}, \mathbf{y}_{1:t})\right]$
```

[§]These algorithms are ad-hoc and are the outcome of my research

Another variation possible is due to to the following observations: in the FFBS steps, parameter estimate θ_i does not change. It essentially means that, the filtering and smoothing are carried on based on the histories available at $t-k$ and the parameter estimates are not update while running the FFBS inside the window. We modify Algorithm 2 so that parameters can be updated as states are updated in the window. Care must be taken while updating the parameters because, the estimates might be based on over-smoothened states, so we reset the state such that the at the end of the window, state estimates would be identical to FFBS. The only change we make is to update the parameters on the fly instead of waiting till the FFBS is completed in the window. The algorithm is summarized below:

Algorithm 2c§: Filtering with unknown parameters

```
For each time period $t = k+1, \ldots, T$:

• Initialize the forward filter: For $i = 1, \ldots, N$, $\mathbf{x}_{t-k+1}^{(i,f)}$ from $p(\mathbf{x}_{t-k+1}|\theta_{t-k}^{(i)}, \mathbf{x}_{t-k}, \mathbf{y}_{1:t-k+1})$
• Initialize $\theta$: run MCMC for $p(\theta|\mathbf{x}_{t-k}^{(i,f)}, \mathbf{x}_{t-k+1}^{(i)}, \mathbf{y}_{1:t-k+1})$ and assign it to $\theta_{t-k}^{(i,1)}$
• For each time state in the window $p = 2 \ldots, k-1$, $\forall i$
    1. run forward-filtering $p(\mathbf{x}_{t-k+p}|\theta_{t-k}^{(i,p-1)}\mathbf{x}_{t-k}, \mathbf{y}_{1:t-k+p})$ for $\mathbf{x}_{t-k+p}^{(i,f)}$
    2. run backward-sampler for $q = p-1, \ldots, 1$
       $p(\mathbf{x}_{t-k+q}|\theta_{j-1}^{(i,p-1)}, \mathbf{x}_{t-k}^{(i)}, \mathbf{x}_{t-k+q+1:t-k+p}^{(i,f)}, \mathbf{y}_{1:t-k+p})$ for $\mathbf{x}_{t-k+1:t-k+p}^{(i,b)}$
    3. run an MCMC for $p(\theta|\mathbf{x}_{t-k}^{(i)}, \mathbf{x}_{t-k+1:t-k+p}^{(i,b)}, \mathbf{y}_{1:t-k+p})$ and assign it to $\theta_{t-k}^{(i,p-1)}$
• For $p = k$
    1. run forward-filtering and store $\mathbf{x}_t^{(i,f)}$
    2. run backward-sampler for $q = k-1, \ldots, 1$ and store $\mathbf{x}_{t-k+1:t-1}^{(i,b)}$
    3. run an MCMC for $p(\theta|\mathbf{x}_{t-k}^{(i)}, \mathbf{x}_{t-k+1}^{(i,b)}, \mathbf{y}_{1:t})$ and assign it to $\theta_{t-k+1}^{(i)}$
• Store $\mathbf{x}_{0:t-k+1}^{(i)}$ as draw from $p(\mathbf{x}_{0:t-k+1}|\mathbf{y}_{1:t})$
• Report $(\mathbf{x}_t^{(i)}, \theta_{t-k+1}^{(i)})$ as draw from $p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t})$
```

In all the algorithms (Algorithm 2,2a and 2b), estimation of θ either using a Monte-Carlo technique or plug-in method requires the full histories. Often times, the distribution of the parameter may just depend on the sufficient statistics and if those sufficient statistics can be obtained recursively, the algorithm will be linear in time and efficient too. The central idea in the Algorithm 3 can be described as:

$$p(\mathbf{x}_{t-k+1:t}, \theta|\mathbf{x}_{0:t-k}, \mathbf{y}_{t-k+1:t}) \quad = \quad p(\mathbf{x}_{t-k+1:t}, \theta|\mathbf{T}_{t-k})$$

where $\mathbf{T}_t = h(\mathbf{T}_{t-1}, \mathbf{x}_t, \mathbf{y}_t)$ are the sufficient statistics. That is the third algorithm mentioned in the paper which is summarized below:

Algorithm 3: Filtering with unknown parameters (sufficient statistics)

```
• For each time period $t = k+1, \ldots, T$
```

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

1. Run an (FFBS) MCMC with stationary distribution $p(\mathbf{x}_{t-k+1:t}, \theta | \mathbf{T}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
   (a) draw from $p(\mathbf{x}_{t-k+1:t} | \theta^{(i)} \mathbf{T}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
   (b) draw from $p(\theta | \mathbf{x}^{(i)}_{t-k+1:t}, \mathbf{T}^{(i)}_{t-k}, \mathbf{y}_{1:t})$
2. Define $(\mathbf{x}^{(i)}_{t-k+1:t}, \theta^{(i)}_t)$ as the last value of $(\mathbf{x}_{t-k+1:t}, \theta)$ in the chain
3. Set $\mathbf{T}^{(i)}_{t-k} = h(\mathbf{T}^{(i)}_{t-1}, \mathbf{x}^{(i)}_{t-k+1}, \mathbf{y}_t)$
4. Store $\mathbf{T}^{(i)}_{t-k+1}$ as draw from $p(\mathbf{T}_{t-k+1} | \mathbf{y}_{1:t})$
5. Report $(\mathbf{x}^{(i)}_t, \theta^{(i)})$ as a draw from $p(\mathbf{x}_t, \theta | \mathbf{y}_{1:t})$

Sufficient statistics based algorithms run in linear time and if a recursive relation exist, they may be efficient too. For example, consider the model of the form

$$\begin{aligned} \mathbf{y}_t &= \mathbf{H}'_t \alpha + \epsilon_t \\ \mathbf{x}_t &= \mathbf{F}'_t \beta + \mathbf{w}_t \\ \mathbf{e}_t &\sim \mathcal{N}(\mathbf{0}, \tau^2 \mathbf{I}) \\ \mathbf{w}_t &\sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}) \end{aligned}$$

where $\mathbf{H}'_t = H(\mathbf{x}_t)$ and $\mathbf{F}'_t = F(\mathbf{x}_{t-1})$ are matrices whose elements can be some nonlinear functions of the states and $\theta = (\alpha, \beta, \tau^2, \sigma^2)$ is the vector of the unknown parameters. Suppose, we elicit conjugate priors of the parameters (Normal-Inverse Gamma family in this case), the posterior distribution of the parameters can be described in terms of the sufficient statistics, given by:

$$\begin{aligned} \beta | \sigma^2, \mathbf{x}_t, \mathbf{y}_{1:t} &\sim \mathcal{N}(\mathbf{B}_t^{-1} \mathbf{t}, \sigma^2 \mathbf{B}_t^{-1}) \\ \boldsymbol{\sigma^2} | \mathbf{x}_t, \mathbf{y}_{1:t} &\sim \mathcal{IG}(\frac{u_t}{2}, \frac{v_t}{2}) \end{aligned}$$

The sufficient statistics can be obtained recursively as:

$$\begin{aligned} \mathbf{B}_t &= \mathbf{B}_{t-k} + \mathbf{F}'\mathbf{F} \\ \mathbf{b}_t &= \mathbf{b}_{t-k} + \mathbf{F}'\mathbf{x} \\ u_t &= u_{t-k} + kp \\ v_t &= v_{t-k} + \mathbf{b}'_{t-k} \mathbf{B}_{t-k} \mathbf{b}_{t-k} + \mathbf{x}'\mathbf{x} - \mathbf{b}'_t \mathbf{B}_t \mathbf{b}_{t-k} \end{aligned}$$

where $\mathbf{F} = [\mathbf{F}'_{t-k+1}, \ldots, \mathbf{F}'_t]$, $\mathbf{x} = [\mathbf{x}'_{t-k+1}, \ldots, \mathbf{x}'_t]$, and p is the dimension of $\mathbf{x}_t$, k the window length. Similar expressions can be derived for α, τ^2 as well. We will use AR(1) model that is a special case of the above state-space model to study all the three algorithms described in this report so far.

Simulation details

We generated 500 data points from a AR(1) model with the following parameters:

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{x}_t + \epsilon_t \\
\mathbf{x}_t &= 0.9\mathbf{x}_{t-1} + \mathbf{w}_t \\
\mathbf{e}_t &\sim \mathcal{N}(\mathbf{0}, 0.1) \\
\mathbf{w}_t &\sim \mathcal{N}(\mathbf{0}, 0.04) \\
\mathbf{x}_0 &\sim \mathcal{N}(\mathbf{0}, 1) \text{ initial state,}
\end{aligned}$$

The parameter vector then becomes $(\beta, \tau^2, \sigma^2) = (1, [0, 0.9], 0.1, 0.04')$. The simulated states are shown in Fig. 1(a) and the observations in Fig. 1(b). In all the algorithms, 1000 sample paths were generated. In running-window algorithms, a window of size 15 is chosen. Regular FFBS was used in the burn-in period in all the cases. While implementing Algorithms2-3, we assumed the noise variances to be known to allow easy implementation. The algorithms are implemented in MATLAB®and the code is provided in the Appendix.

Similar to Algorithm 2c, we can modify Algorithm 3 to exploit the sufficient statistics as:

Algorithm 3a[§]: Filtering with unknown parameters

```
For each time period $t = k+1, \ldots, T$:

• Initialize the forward filter: For $i = 1, \ldots, N$, $\mathbf{x}_{t-k+1}^{(i,f)}$ from $p(\mathbf{x}_{t-k+1}|\theta_{t-k}^{(i)}, \mathbf{T}_{t-k}^{(i,f)}, \mathbf{y}_{1:t-k+1})$

• Initialize $\theta$: run MCMC for $p(\theta|\mathbf{T}_{t-k}^{(i)}, \mathbf{x}_{t-k+1}^{(i)}, \mathbf{y}_{1:t-k+1})$ and assign it to $\theta_{t-k}^{(i,1)}$

• For each time state in the window $p = 2 \ldots, k-1$, $\forall i$

    1. run forward-filtering $p(\mathbf{x}_{t-k+p}|\theta_{t-k}^{(i,p-1)}, \mathbf{T}_{t-k}^{(i)}, \mathbf{y}_{1:t-k+p})$ for $\mathbf{x}_{t-k+p}^{(i,f)}$
    2. run backward-sampler for $q = p-1, \ldots, 1$
       $p(\mathbf{x}_{t-k+q}|\theta_{j-1}^{(i,p-1)}, \mathbf{x}_{t-k}^{(i)}, \mathbf{x}_{t-k+q+1:t-k+p}^{(i,f)}, \mathbf{y}_{1:t-k+p})$ for $\mathbf{x}_{t-k+1:t-k+p}^{(i,b)}$
    3. $\mathbf{T}_{t-k}^{(i,b)} = f(\mathbf{T}_{t-k}^{(i)}, \mathbf{x}_{t-k+1:t-k+p}^{(i,b)}, \mathbf{y}_{1:t-k+p})$
    4. run an MCMC for $p(\theta|\mathbf{T}_{t-k}^{(i,b)})$ and assign it to $\theta_{t-k}^{(i,p-1)}$

• For $p = k$

    1. run forward-filtering and store $\mathbf{x}_t^{(i,f)}$
    2. run backward-sampler for $q = k-1, \ldots, 1$ and store $\mathbf{x}_{t-k+1:t-1}^{(i,b)}$
    3. $\mathbf{T}_{t-k+1}^{(i)} = f(\mathbf{T}_{t-k}^{(i)}, \mathbf{x}_{t-k+1}^{(i,b)}, \mathbf{y}_{1:t})$
    4. run an MCMC for $p(\theta|\mathbf{T}_{t-k+1}^{(i)}$ and assign it to $\theta_{t-k+1}^{(i)}$

• Store $\mathbf{x}_{0:t-k+1}^{(i)}$ as draw from $p(\mathbf{x}_{0:t-k+1}|\mathbf{y}_{1:t})$

• Report $(\mathbf{x}_t^{(i)}, \theta_{t-k+1}^{(i)})$ as draw from $p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t})$
```

We can run a full-FFBS at each iteration to learn about the parameters, this leads to the Algorithm-3b

Algorithm 3b§: Filtering with unknown parameters

```
For each time period $t = k+1, \dots, T$

for $i = 1, \dots, N$

  • For $j = 1, \dots, k$
      – run FF to get $\mathbf{x}^{(i)}_{t-k+1}$ from $p(\mathbf{x}_{t-k+1}|\mathbf{T}^{(i)}_{t-k}, \theta^{(i)}_{t-k+j-1}\mathbf{y}_{1:t-k+p})$
      – $\mathbf{T}^{(i,1)}_{t-k+j} = h(\mathbf{T}^{(i)}_{t-k}, \mathbf{x}^{i}_{t-k+1}, \mathbf{y}_{1:t-k+1})$
      – run MCMC for $\theta^{(i)}_{t-k+j} \sim p(\theta|\mathbf{T}^{(i,1)}_{t-k+j})$
      – For $p = 2, \dots, j-1$
          1. run FF to get $\mathbf{x}^{(i)}_{t-k+p}$ from $p(\mathbf{x}_{t-k+p}|\theta^{(i)}_{t-k+j}\mathbf{y}_{1:t-k+p})$
          2. run BS to get $\mathbf{x}^{(i)}_{t-k+1:t-k+p-1}$ from $p(\mathbf{x}_{t-k+1:t-k+p-1}|\theta^{(i)}_{t-k+j}\mathbf{y}_{1:t-k+p})$
          3. $\mathbf{T}^{(i,p)}_{t-k+j} = f(\mathbf{T}^{(i)}_{t-k}, \mathbf{x}^{(i)}_{t-k+1:t-k+p}, \mathbf{y}_{1:t-k+p})$
          4. run MCMC to get $\theta^{(i)}_{t-k+j}$ from $p(\theta|\mathbf{T}^{(i,p)}_{t-k+j})$
      – run FF to get $\mathbf{x}^{(i)}_{t-k+j}$ from $p(\mathbf{x}_{t-k+j}|\theta^{(i)}_{t-k+j}\mathbf{y}_{1:t-k+j})$
      – run BS to get $\mathbf{x}^{(i)}_{t-k+1:t-k+j-1}$ from $p(\mathbf{x}_{t-k+1:t-k+j-1}|\theta^{(i)}_{t-k+j}\mathbf{y}_{1:t-k+j})$
  • $\mathbf{T}^{(i)}_{t-k+1} = f(\mathbf{T}^{(i)}_{t-k}, \mathbf{x}^{(i)}_{t-k+1}, \mathbf{y}_{1:t-k+1})$
  • run MCMC to get $\theta^{(i)}_{t-k+1}$ from $p(\theta|\mathbf{T}^{(i)}_{t-k+1})$
  • Store $\mathbf{x}^{(i)}_{0:t-k+1}$ as draw from $p(\mathbf{x}_{0:t-k+1}|\mathbf{y}_{1:t})$
  • Report $(\mathbf{x}^{(i)}_{t}, \theta^{(i)}_{t-k+1})$ as draw from $p(\mathbf{x}_t, \theta|\mathbf{y}_{1:t})$
```

Results and discussion

We compare the FFBS and Algo-1 in Fig. 2, . FFBS is closely following the true states compared to Algo-1. Some reasons could be the size of the window and the choice of the proposal density in the importance sampling stage. Because in algo-1 each state is visited several times, the states could be oversmoothed. The parameter estimates from the algo-2 are shown in Figs. 3(a-b) and the state estimates are shown in Fig. 4. The parameter estimates are converging even though they are not the true values. We might have to run Monte-Caro analysis to assess the performance all the algorithms in terms of the MSE. We can not conclusively argue about the merits based a single realization of the data. The parameter estimates and 95%credible intervals are shown in Fig. 5(a-b). Compared to algo-2b, the parameters are closer to the true values. in Fig. 6 the states estimated by the algorithm are shown. Both in algo-2 and algo-3, the states are smoothed compared to the true states. This is because, we have been traversing the states multiple times during the estimation process. Finally, in Fig. 7, we plot the states estimated by all the algorithms. In general, algo-3 is able to estimate the parameters and states well, cosidering the fact that it is a joint estimation problem. FFBS methods do not oversmooth the states as much as the windowing-based methods are doing.

Conclusions

We have provided a brief review of the sequential Monte Carlo techniques and discussed the necessary tools required to derive new algorithms. In particular, Forward-Filtering Backward sampling strategy is discussed. We reviewed the algorithms presented to Polson et al (2008) that consider running-window Sequential Monte carlo techniques to estimate the states as well the parameters jointly. In this light, we have provided the details of how to adapt the FFBS so that sequential importance sampling can be used as opposed to MCMC based algorithms. We have suggested few variations that are possible with-in the frame-work set-up by Polson et al. Estimation algorithms in all the cases: parameters known and unknown, using full histories or sufficient statistic are implemented for comparison purposes. To compare different algorithms, we have simulated the data from a AR(1) process observed in noise. Joint estimation is challenging problem and there seems to lot of scope for developing new algorithms. Running a full Monte-Carlo simulation study to asses the performance of all the algorithms is needed for objectively assessing their the merits and demerits

References

Clapp, T. C. and Godsill, S. (1999). Fixed-lag smoothing using sequential importance sampling. *Bayesian Statistics*, 6, 743-752.

Doucet, Arnaud., Freitas Nando, N. and Gordon Neil, eds, (2001), *Sequential Monte Carlo methods in Practice*, Springer, New York

Djurić, P. M., et. al. (2003). Particle Filtering: A review of the theory and how it can be used for solving problems in wireless communications. *IEEE Signal Processing Magazine*, 20, 5, 19-38.

Geir Storvik. (2002), Particle Filters for state-space models with the presence of unknown static parameters, *IEEE Trans. on Sig. Proc.*, 50(2), 281-289

Godsill, S. J., Doucet, A., and West, M. (2004). Monte Carlo smoothing for non-linear time series. *Journal of the American Statistical Association*, 99, 465, 156-168.

Kitagawa, G. and Sato. S. (2001). Monte Carlo smoothing and self-organizing state-space model. *Sequential Monte Carlo Methods in Practice.* Springer, New York.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing.* Springer Series in Statistics, New York.

Pitt, M. K. and Shephard, N. (2001). Auxiliary variable based particle filters. *Sequential Monte Carlo Methods in Practice.* Springer, New York.

Polson, N. G., Stroud, J. R., and Müller, P. (2008). Practical Filtering with Sequential Parameter Learning, *Journal of the Royal Statistical Society*, Series B, 70, 413-428

Sylvia Früwith-Schnatter. (1992), Data augmentation and dynamic linear models, *J. Time Series Anal*, 15, 183-202

Figures

(a) states (b) observations

Figure 1: Data simulated from an AR(1) plus noise model

Figure 2: estimated states from different algorithms (with fixed parameters)

(a) β_0 (b) β_1

Figure 3: estimated parameters from Algo-2b

Figure 4: estimated states from algo-2

(a) CIs for β_0 (b) CIs for β_1

Figure 5: CIs for parameters in Algo-3

(a) Algo-2 (b) All the Algorithms

Figure 6: estimates from different algorithms

Appendix B: MATLAB$^{®}$ pseudo code

Data generation

```
% generate data
bet = [0,0.9]';
sig2 = (0.04);
sig = sqrt(sig2);
tau2 = (0.1);
tau = sqrt(tau2);
T = 500;
N = 2000; % number of sample paths
x = zeros(T,length(sig2));
y = zeros(T,length(tau2));
% initialize x0 state vector
x0 = normrnd(0,0.1);
x(1) = normrnd([1,x0]*bet,sig);
y(1) = normrnd(x(1),tau);
for p = 2:T+1
     x(p) = normrnd([1,x(p-1)]*bet,sig);
     y(p) = normrnd(x(p-1),tau);
end
% save the data for all simulations
```

FFBS

```
load ar1.mat; % use the same data in all simulations
xt = zeros(T+1,N);
wf = zeros(T+1,N);% forward weights
wb = zeros(T+1,N); % backward weights

% run forward filtering
xt(1,:) = normrnd(0,1,[1,N]);
wf(1,:) = 1/N;
for p = 2:T+1
    xt(p,:) = normrnd(bet(1)+ (bet(2)*xt(p-1,:)),sig);
    wf(p,:) = wf(p-1,:).*normpdf(y(p-1),xt(p,:),tau);
    wf(p,:) = wf(p,:)/sum(wf(p,:));
end

% run backward smoothing (t=T-1 to 1)
wb(T+1,:) = wf(T+1,:);
for p = T:-1:1
   % wb(p,:) = wf(p+1,:).*normpdf(x(p+1),xt(p,:),sig);
    wb(p,:) = wf(p+1,:).*normpdf(x(p+1,:),(bet(1)+bet(2)*xt(p,:)),sig);
    wb(p,:) = wb(p,:)/sum(wb(p,:));
end
% full FFBS is done. % Now implement algorithm-1
```

Algorithm-1

```
k = 15;
for t = k+2 : T+1
    % you are actually at t=k+1
    fprintf(1,'in %d\n',t);
    for p = t-k+1:t
        xt(p,:) = normrnd(bet(1)+ (bet(2)*xt(p-1,:)),sig);
        wf(p,:) = wf(p-1,:).*normpdf(y(p-1),xt(p,:),tau);
        wf(p,:) = wf(p,:)/sum(wf(p,:));
    end
    % now smooth them
    for p = t-1:-1:t-1-(k-1)+1
        %wb(p,:) = wf(p+1,:).*normpdf(x(p+1),xt(p,:),sig);
        wb(p,:) = wf(p,:).*normpdf(x(p+1,:),(bet(1)+bet(2)*xt(p,:)),sig);
        wb(p,:) = wb(p,:)/sum(wb(p,:));
        wf(p,:) = wb(p,:);
    end
end
```

Algorithm-2b

```
%run rolling window FFBS on this
k = 30;
for t = k+2 : T+1
    % you are actually at t=k+1
    fprintf(1,'in %d\n',t);
    for p = t-k+1:t
        xt(p,:) = normrnd(b0Hat(t-1)+ (b1Hat(t-1)*xt(p-1,:)),sigHat(t-1));
        wf(p,:) = wf(p-1,:).*normpdf(y(p-1),xt(p,:),tauHat(t-1));
        wf(p,:) = wf(p,:)/sum(wf(p,:));
    end
    % now smooth them
    wb(t,:) = wf(t,:);
    for p = t-1:-1:t-1-(k-1)+1
        %wb(p,:) = wf(p+1,:).*normpdf(x(p+1),xt(p,:),sig);
        wb(p,:) = wf(p,:).*normpdf(xt(p+1,:),(b0Hat(t-1)+b1Hat(t-1)*xt(p,:)),sigHat(t-1));
        wb(p,:) = wb(p,:)/sum(wb(p,:));
        % resmaple the states
        [wTemp,xTemp] = resample(wb(p,:),xt(p,:));
         wf(p,:) =  wTemp;
         xt(p,:) = xTemp;
    end
    % estimate theta (plug-in)
    ind = [t-k+1:t];
    xHat(ind) = sum(xt(ind,:).*wb(ind,:),2);
    tauHat(t) = tau; %std(y(1:t-1)'-xHat(2:t));
    s = regstats(xHat(ind),xHat(ind-1),'linear');
    sigHat(t) = sig;%0.2;sqrt(s.mse);
    b0Hat(t) = bet(1);s.beta(1);
    b1Hat(t) = bet(2);s.beta(2);
```

```
end
```

Algorithm-3

```
% define the sufficient statistic vevtors
b0rec = zeros(T+1,N);
b1rec = zeros(T+1,N);
Bt = cell(N,1);
bt = cell(N,1);

for p = 1:N
    Bt{p} = eye(2);
    bt{p} = [0;0.9];
end

b0rec(1:k+1,:) = repmat(b0Hat(1:k+1),1,N);
b1rec(1:k+1,:) = repmat(b1Hat(1:k+1),1,N);

for t = k+2 : T+1
    % you are actually at t=k+1
    fprintf(1,'in %d\n',t);
    for p = t-k+1:t
        xt(p,:) = normrnd(b0rec(t-1,:)+ (b1rec(t-1,:).*xt(p-1,:)),sigHat(t-1));
        wf(p,:) = wf(p-1,:).*normpdf(y(p-1),xt(p,:),tauHat(t-1));
        wf(p,:) = wf(p,:)/sum(wf(p,:));
    end
    % now smooth them
    wb(t,:) = wf(t,:);
    for p = t-1:-1:t-1-(k-1)+1
        %wb(p,:) = wf(p+1,:).*normpdf(x(p+1),xt(p,:),sig);
        wb(p,:) = wf(p,:).*normpdf(xt(p+1,:),(b0rec(t-1,:)+b1rec(t-1,:).*xt(p,:)),sigHat(t-1));
        wb(p,:) = wb(p,:)/sum(wb(p,:));
        [wTemp,xTemp] = resample(wb(p,:),xt(p,:));
        wf(p,:) =  wTemp;
        xt(p,:) = xTemp;
    end
    % now obtain recursively sample b0 and b1
    ind = [t-k+1:t];
    for p = 1:N
        xTemp = xt(ind,p);
        F = [ones(k,1), xt(ind-1,p)];
        Bt{p} = Bt{p} + F'*F;
        bt{p} = bt{p} + F'*xTemp;
        BtInv = inv(Bt{p}+diag(1e-3));
        bTemp = mvnrnd(BtInv*bt{p},sig2*BtInv); % update posterior distribution
        b0rec(t,p) =bTemp(1); % posterior draws for unknown parameters
        b1rec(t,p) =bTemp(2); %
    end
    sigHat(t) = sig; % we assume noise variances are known
    tauHat(t) = tau;
end
```