Parameter Inference via Pseudo-Marginal Metropolis-Hastings and Sequential Monte Carlo

Andrea Becsek, Mauro Camara, Doug Corbin

Contents

1 Inference in State Space Models

1.1 State Space Models (SSM)

Statistics leverages information contained in the observed data to perform inference on unknown quantities. It is often the case, however, that the data itself is the result of a process that is hidden to the observer. Interest then lies in understanding such a process with the use of the observations.
Throughout this report, we imagine that the unobserved process starts at time $t = 0$ with the realization $\mathbf{x}_0$ of a random variable $\mathbf{X}_0$, which we don't observe. At time $t = 1$, two events occur. First, the hidden process evolves forward in time with a realization $\mathbf{x}_1$ of a new random variable $\mathbf{X}_1$. Next, we receive the first realization $\mathbf{Y}_1 = \mathbf{y}_1$ of the observed process. At a general time $t > 0$, the system advances by first moving the hidden process forward in time $\mathbf{X}_{t+1} = \mathbf{x}_{t+1}$ and then producing the corresponding observation $\mathbf{Y}_{t+1} = \mathbf{y}_{t+1}$, as illustrated in Figure 1.

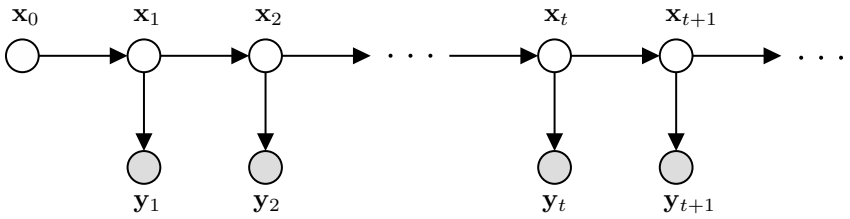


Figure 1: Graphical model of a state-space model. Filled circles represent observed variables, while empty circles represent hidden states.

The system just described can lead to multiple models but the one on which we will be focusing on is called **State Space Model** (SSM) or general Hidden Markov Model (HMM). A SSM consists of an $\mathcal{X}$-valued Markov process $\{\mathbf{X}_t\}_{t\geq 0}$, which is not observable, has initial probability $p(\mathbf{x}_0 \mid \boldsymbol{\theta})$, and transition probability $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta})$, and of a set of random variables $\{\mathbf{Y}_t\}_{t\geq 0}$ which are conditionally independent given the realizations of the Markov process. Practically, this means that we can describe the system with the following equations

$$p(\mathbf{x}_0 \mid \boldsymbol{\theta}) \tag{1}$$
$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta}) \qquad t \geq 1 \tag{2}$$
$$p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) \qquad t \geq 1 \tag{3}$$

where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ is a vector of parameters. Note that for the sake of simplicity we have used the same parameter vector for all three equations, however this must not necessarily be the case. In fact, in a more general sense, the equations could all depend on different parameters, so $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$. Our goal is to estimate various quantities *online* as a new observation $\mathbf{y}_t$ is observed. In general, the subjects of inference are threefold, depending on the application. For notational ease, we use $\mathbf{x}_{0:t} = \{\mathbf{x}_k\}_{k=0}^t$ to denote the sequence of the hidden states up to time t, and similarly for the observations, $\mathbf{y}_{1:t} = \{\mathbf{y}_k\}_{k=1}^t$.

1. *Posterior distribution*: Commonly one wants to be able to infer $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$, i.e. the posterior distribution of all the hidden states up to time t, given all the observations up to time t. This is useful when we want to learn about the "path" of the hidden variable.
2. *Filtering distribution*: One often wants to be able to predict the current hidden state, given all available data $p(\mathbf{x}_t \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$.
3. *Expectations*: Related to the first point, one may want to compute, at every time step t, expectations of functions of the hidden states with respect to the posterior distribution.

$$\mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t},\boldsymbol{\theta})}\left[f_t(\mathbf{x}_{0:t})\right] = \int f_t(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) d\mathbf{x}_{0:t}$$

Some common examples of online applications include navigation and target tracking. Given some object that is in motion we might be interested in either the position, velocity or acceleration of that object at different times, which represent the hidden states of the model. The source of the observations about the object can be either via some internal sensors, in which case we are faced with a navigation problem. On the other hand, when the observations come from an external source, the problem is instead a tracking problem. For example, if we observe the position of a rocket at multiple times, we might want to infer the velocity of it and predict the remaining of its trajectory.

1.2 Parameter Inference via Particle Markov Chain Monte Carlo

Inference of the hidden states is often performed using Sequential Monte Carlo algorithms such as the Bootstap Filter or the Auxiliary Particle Filter, which assume that the parameter vector $\boldsymbol{\theta}$ is known. Such algorithms target the following posterior distribution

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) \propto p(\mathbf{x}_{0:t} \mid \boldsymbol{\theta}) p(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) = p(\mathbf{x}_0 \mid \boldsymbol{\theta}) \prod_{k=1}^{t} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}) \prod_{k=1}^{t} p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}). \tag{4}$$

In practice, one doesn't have access to $\boldsymbol{\theta}$ and needs to determine its value. Noticeably, this is often the main inference goal in SSMs and throughout this manuscript we will address the problem of learning such a parameter using Particle Markov Chain Monte Carlo (PMCMC) which was originally introduced in Andrieu et al. (2010). When $\boldsymbol{\theta}$ is unknown we can place a prior on the parameter $p(\boldsymbol{\theta})$ and consider the following joint distribution as the target of our inference

$$p(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) \propto \left[p(\mathbf{x}_0 \mid \boldsymbol{\theta}) \prod_{k=1}^{t} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{\theta}) \prod_{k=1}^{t} p(\mathbf{y}_k \mid \mathbf{x}_k, \boldsymbol{\theta}) \right] p(\boldsymbol{\theta}). \tag{5}$$

PMCMC algorithms, as we will see later, run an MCMC chain where the proposal distribution is constructed using a run of an SMC algorithm. Before moving towards PMCMC, we will build up the relevant theoretical work needed in order to understand the Bootstrap Filter and the Auxiliary Particle Filter when $\boldsymbol{\theta}$ is known.

2 Hidden State Inference via Sequential Monte Carlo

2.1 The Ideal Scenario: Perfect Monte Carlo Sampling

In few simple cases, such as Kalman Filters (Welch et al., 1995), the posterior distribution $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$ in equation (4) will be easy to sample from. If that's the case, then one can obtain, at time t, N samples from it

$$\mathbf{x}_{0:t}^{(1)}, \ldots, \mathbf{x}_{0:t}^{(N)} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$$

and use them to approximate the posterior using an empirical distribution, as outlined in Appendix A.

$$\widehat{p}_N(\mathbf{x}_{0:t}) = \sum_{i=1}^{N} \frac{1}{N} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})$$

Of course, in practice, we can't actually sample from the posterior distributions and we need to resort to other methods. In non-sequential cases, the preferred choice is either stochastic methods, such as MCMC, or deterministic methods such as variational inference (Blei et al., 2017). However they are not suited, in their vanilla characterizations, to sequential inference. A precursor to MCMC is Importance Sampling, which we will explore in the following section. While not being itself useful for sequential estimation, it can be easily adapted to such scenarios, obtaining straightforward algorithms such as the Bootstrap Filter, described in Section 2.5.

2.2 Importance Sampling

Instead of sampling from $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$ we can choose a density $q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ that is easy to sample from. This density is referred to as an importance or proposal density, and it needs to have the same support as the target density

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) > 0 \quad \Longrightarrow \quad q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) > 0. \tag{6}$$

We will see towards the end of this section that it is also advisable that q has heavier tails than the target distribution. Importance sampling is particularly useful when our goal is to approximate expectations. Then, we can re-write the integral as a fraction of two expectations with respect to $q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ and then approximating such integrals by replacing the proposal with it's empirical representation of N samples.

$$\widehat{q}_N(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}) \qquad \text{where} \qquad \mathbf{x}_{0:t}^{(i)} \overset{\text{i.i.d.}}{\sim} q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$$

Using the definition of the Dirac-delta function given in Appendix A we derive the following expression

$$
\begin{aligned}
\mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t},\boldsymbol{\theta})}\left[f_t(\mathbf{x}_{0:t})\right] &= \int f_t(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \mid \boldsymbol{\theta})\ d\mathbf{x}_{0:t} \\
&= \frac{\int f_t(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \mid \boldsymbol{\theta})\ d\mathbf{x}_{0:t}}{\int p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \mid \boldsymbol{\theta})\ d\mathbf{x}_{0:t}} && \text{normalizing} \\
&= \frac{\int f_t(\mathbf{x}_{0:t}) \frac{p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \mid \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})\ d\mathbf{x}_{0:t}}{\int \frac{p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \mid \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})\ d\mathbf{x}_{0:t}} && \text{using (6)} \\
&\approx \frac{\int f_t(\mathbf{x}_{0:t}) \frac{p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \mid \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} \frac{1}{N} \sum_{i=1}^{N} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})\ d\mathbf{x}_{0:t}}{\int \frac{p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \mid \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} \frac{1}{N} \sum_{i=1}^{N} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})\ d\mathbf{x}_{0:t}} && \text{empirical density} \\
&= \frac{\sum_{i=1}^{N} \int f_t(\mathbf{x}_{0:t}) w_{\text{func}}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}) d\mathbf{x}_{0:t}}{\sum_{i=1}^{N} \int w_{\text{func}}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}) d\mathbf{x}_{0:t}} \\
&= \frac{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) \sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)}) \tilde{w}(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) \sum_{i=1}^{N} \tilde{w}(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})} && \text{definition of } \delta \\
&= \sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)}) w(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t}), && (7)
\end{aligned}
$$

where we have defined the weight function as

$$
w_{\text{func}}(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{x}_{0:t}, \mathbf{y}_{1:t} \mid \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} = \frac{p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) \propto \frac{p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})}.
$$

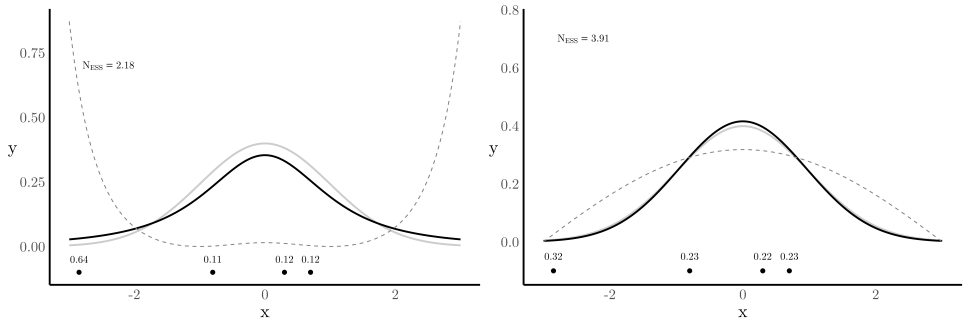
However, in practice the marginal distribution $p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})$ is not known, so instead we first compute the **un-normalized importance weights** as the ratio between the posterior distribution and the proposal distribution

$$
\tilde{w}_t(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t}) \propto \frac{p(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t}, \theta)}{q(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t})}, \tag{8}
$$

which takes into account the discrepancy between the two distributions, and then normalize them, obtaining the **normalized weights**

$$
w_t(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t}) = \frac{\tilde{w}_t(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{\sum_{j=1}^{N} \tilde{w}_t(\mathbf{x}_{0:t}^{(j)}|\mathbf{y}_{1:t})}.
$$

If one could choose the posterior distribution $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$ as the importance density then all the weights would be constant and, most importantly, uniform, and importance sampling would reduce to perfect Monte Carlo sampling. Therefore, it makes sense to choose the proposal distribution in such a way that the importance weights are as uniform as possible. Unfortunately, as dimensionality and complexity of the posterior increase this is a very difficult task to do because its typical set will be harder to capture due to the curse of dimensionality (MacKay, 2002). When weights are not uniform, this can lead to a biased representation of our target distribution and, often, to a handful of weights being much



(a) Proposal: t_2 **(b)** Proposal: $\mathcal{N}(0, 0.96)$

Figure 2: Target distribution $\mathcal{N}(0, 1)$ is shown in black. a) t_2 and b) $\mathcal{N}(0, 0.96)$ are used as IS distributions, shown in grey. The dashed line shows the (re-scaled) value of the weights. We can see how sampling from a similar normal distribution leads to more uniform weights and thus a Effective Sample Size closer to 4.

larger than the rest and dominating the sum in (7). In Figure 2 we see how choosing an importance density with lighter or heavier tails affects the distribution of the weights.
One way around this issue is to utilize a metric to judge how many weights are actually contributing to the sum. A commonly used metric is the **Effective Sample Size** (ESS) as derived by Kish (1965)

$$N_{\text{ESS}} = \frac{1}{\sum_{i=1}^{N} w(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t})^2}.$$

We can see that N_{ESS} captures information about whether the weights are uniform or not from the following observation

$$N_{\text{ESS}} = \begin{cases} N & \text{if } w(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t}) = N^{-1} \quad \forall\, i \\ 1 & \text{if } w(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t}) = 1 \quad \text{and} \quad w(\mathbf{x}_{0:t}^{(j)} \mid \mathbf{y}_{1:t}) = 0 \quad \forall\, i \neq j. \end{cases}$$

In addition, it can easily be shown that $1 \leq N_{\text{ESS}} \leq N$. One of the problems with Importance Sampling is that it is inadequate for recursive estimation. This is because at each time step we need to recompute all the importance weights, leading to a computational complexity that increases at least linearly with time.

2.3 Sequential Importance Sampling

Sequential Importance Sampling provides a better solution to recursive estimation and it does so by computing the weights recursively. In order to achieve this, we assume that the proposal density can be factorized as follows:

$$\begin{aligned} q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) &= q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t}) && (9) \\ &= q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}) && (10) \\ &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}) && (11) \end{aligned}$$

Note that in Equation (10) we eliminate the dependency on the measurement at time t. By doing so, it enables us to compute the density recursively. Furthermore, to get to Equation (11) the proposal density has been chosen such that it only depends on $\mathbf{x}_{t-1}$ and $\mathbf{y}_t$. Making these choices enable us to go from one state to the next without having to consider the full past trajectories, which would become more computationally expensive as the number of states increases.

To obtain the recursive formula for the weights, we first rewrite the target posterior density in terms of $p(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\theta})$, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta})$, and $p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}, \boldsymbol{\theta})$

$$
\begin{aligned}
p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) &= p(\mathbf{x}_{0:t} \mid \mathbf{y}_t, \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) \\
&= \frac{p(\mathbf{x}_{0:t}, \mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})} && \text{def. of conditional prob.} \\
&= \frac{p(\mathbf{y}_t \mid \mathbf{x}_{0:t}, \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})} \\
&= \frac{p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})} && \{\mathbf{Y}_t\} \text{ conditionally ndep.} \\
&= \frac{p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) p(\mathbf{x}_t, \mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})} \\
&= \frac{p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1} \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})} \\
&= \frac{p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})} p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) && \{\mathbf{X}_t\} \text{ Markov Process} \quad (12) \\
&\propto p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) p(\mathbf{x}_t | \mathbf{x}_{t-1}, \boldsymbol{\theta}) p(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t-1}, \boldsymbol{\theta}). && (13)
\end{aligned}
$$

Using equations (11) and (13) we can rewrite the weights in (8) as

$$
\begin{aligned}
w_t &\propto \frac{p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} && (14) \\
&\propto \frac{p(\mathbf{y}_t | \mathbf{x}_t, \boldsymbol{\theta}) p(\mathbf{x}_t | \mathbf{x}_{t-1}, \boldsymbol{\theta})}{q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{y}_t)} \frac{p(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t-1}, \boldsymbol{\theta})}{q(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t-1})} && (15) \\
&\propto \frac{p(\mathbf{y}_t | \mathbf{x}_t, \boldsymbol{\theta}) p(\mathbf{x}_t | \mathbf{x}_{t-1}, \boldsymbol{\theta})}{q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{y}_t)} w_{t-1}. && (16)
\end{aligned}
$$

Now that the weights can be computer sequentially, the joint posterior density can be approximated using the empirical distribution

$$
\widehat{p}_N(\mathbf{x}_{0:t} | \mathbf{y}_{1:t}) = \sum_i w_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}). \quad (17)
$$

And the marginal posterior can be easily obtained by integrating out the previous states

$$
\begin{aligned}
\widehat{p}_N(\mathbf{x}_t | \mathbf{y}_{1:t}) &= \int \widehat{p}_N(\mathbf{x}_{0:t} | \mathbf{y}_{1:t}) d\mathbf{x}_{0:t-1} && (18) \\
&= \sum_i^N w_t^{(i)} \int \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}) d\mathbf{x}_{0:t-1} && (19) \\
&= \sum_i^N w_t^{(i)} \delta_{\mathbf{x}_t^{(i)}}(\mathbf{x}_t). && (20)
\end{aligned}
$$

Sequential Importance Sampling is essentially a constrained version of Importance Sampling which assumes that the trial distribution has a recursive structure, as described in Equation (11). Unfortunately SIS is often inefficient in high-dimensional scenarios. As the number of states increases, the weights become more skewed as most of them will become very small. That means that only a very small number of particles with high weights will contribute to the approximation (Doucet et al., 2001, Chapter 1).

2.4 Resampling

The problem of particle degeneracy described above can be solved by introducing a resampling step. By doing so, the particles with very low weights are eliminated and those with large weights are replicated.

In the case of Importance Sampling we have approximated the target distribution using the weighted samples drawn from the trial distribution $q_t(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$. However, in order to obtain samples approximately

distributed according to the target distribution $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}, \boldsymbol{\theta})$ we sample from the weighted empirical distribution $\widehat{p}_N(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$.

The current samples and their associated weights $(\mathbf{x}_t^{(i)}, w_t^{(i)})$ get mapped to a new set of uniformly weighted samples $(\mathbf{x}_t^{(i^*)}, 1/N)$. The new samples can be obtained by resampling with replacement from $\widehat{p}_N(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ such that $P(\mathbf{x}_t^{(i^*)} = \mathbf{x}_t^{(i)}) = w_t^{(i)}$. By doing so we obtain iid samples from $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}, \boldsymbol{\theta})$.

There are multiple resampling techniques that can be used, however they should all be such that

$$\int f_t(\mathbf{x}_{0:t}) P_N(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \approx \int f_t(\mathbf{x}_{0:t}) \hat{P}_N(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}).$$

One of the most used resampling methods is Systematic Resampling, which is easy to implement and minimizes the Monte Carlo variation. We start by drawing a sample $u_1 \sim U[0, \frac{1}{N}]$ and create an equally spaced grid by defining $u_i = u_1 + \frac{i-1}{N}$, for $i = 2, \cdots, N$. The number of off-springs associated with particle $i = 1$ at time t is given by

$$N_t^i = \left| \left\{ u_j : \sum_{k=1}^{i-1} w_t^k \leq u_j \leq \sum_{k=1}^{i} w_t^k \right\} \right|. \tag{21}$$

As shown in Figure 3, this means that the number of offsprings associated with each particle is given by the number of equally spaced points that fall between the previous and next cumulative sum of the weights. That means that the larger the weights of a particle, the more off-springs it has.

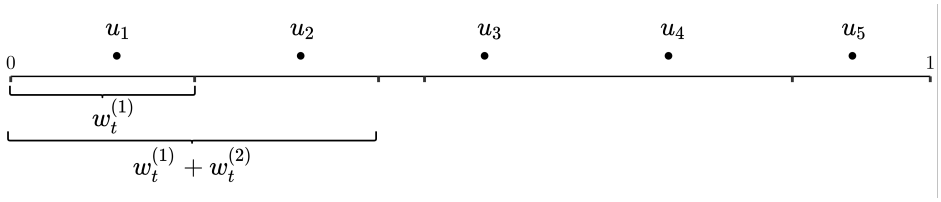


Figure 3: Plot of the cumulative sums of weights along the x axis together with the equally spaced points starting from $u_1 \sim U[0, \frac{1}{N}]$.

The main advantage of resampling is that by eliminating particles with very low weights it allows us to focus on regions with high probability mass. In the next section we explore the Bootstrap and Auxiliary Particle Filter, which both rely on resampling.

2.5 Bootstrap Filter

The Bootstrap filter (Gordon et al., 1993) is peharphs one of the most important Sequential Importance Resampling algorithms. The main assumption of this method is regarding the proposal distribution, which is set to be the transitional prior of the target distribution

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \boldsymbol{\theta}). \tag{22}$$

Substituting this into Equation (16) it can be shown that $w_t^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|\mathbf{x}_t^{(i)}, \boldsymbol{\theta})$. However, after the previous resampling step the particles have uniform weights N^{-1}, and therefore the weights at time t are simply given by the likelihood and then normalized.

The algorithm is described below, with each step being performed for each particle $i = 1, \cdots, N$

Algorithm 1 Bootstrap Filter

1. Initialize $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0, \boldsymbol{\theta})$.
2. For $t = 1, \cdots, T$:

   Importance Sampling

   - Sample $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i^*)}, \boldsymbol{\theta})$.
   - Set $\mathbf{x}_{0:t}^{(i)} = \left[\mathbf{x}_{0:t-1}^{(i^*)}, \mathbf{x}_t{}^{(i)}\right]$.
   - Compute the un-normalized importance weights $\tilde{w}_t^{(i)} = p(\mathbf{y}_t | \mathbf{x}_t^{(i)}, \boldsymbol{\theta})$.
   - Normalize the importance weights.

   Resampling

   - Resample with replacement N particles $\mathbf{x}_{0:t}^{(i^*)}$ from $\mathbf{x}_{0:t}^{(i)}$ with weights $w_t^{(i)}$.

2.6 Auxiliary Particle Filtering

Recall the expression for the full posterior distribution $p(\mathbf{x}_{0:t} | \mathbf{y}_{1:t}, \boldsymbol{\theta})$

$$p(\mathbf{x}_{0:t} | \mathbf{y}_{1:t}, \boldsymbol{\theta}) = p(\mathbf{x}_t | \mathbf{y}_{1:t}, \mathbf{x}_{0:t-1}, \boldsymbol{\theta}) p(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t}, \boldsymbol{\theta}). \tag{23}$$

While formulating the proposal distribution for sequential importance sampling, a subtle assumption was made in (10). That is, using a proposal distribution satisfying

$$q(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t}) = q(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t-1}) \tag{24}$$

implicitly makes no use of the current observation $\mathbf{y}_t$ in sampling particles $\tilde{\mathbf{x}}_{0:t-1}^{(0)}, \ldots, \tilde{\mathbf{x}}_{0:t-1}^{(N)}$ to be propagated. This is clearly the case for the Bootstrap Filter in which every particle is propagated according to $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \boldsymbol{\theta})$, irrespective of $\mathbf{y}_t$. It is only **after** the propagation of (all) the particles that $\mathbf{y}_t$ is used to weight and resample them.

An alternative approach first proposed by Pitt and Shephard (1999), known as Auxiliary Particle Filtering (APF), uses the current observation $\mathbf{y}_t$ to determine which particles to propagate. At each time step t a set of "parent" particles are sampled

$$\tilde{\mathbf{x}}_{0:t-1}^{(0)}, \ldots, \tilde{\mathbf{x}}_{0:t-1}^{(N)} \sim q(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t}). \tag{25}$$

Naturally, the i'th parent particle is therefore associated with its "offspring", $\mathbf{x}_{0:t}^{(i)}$, defined by

$$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t | \mathbf{y}_{1:t}, \tilde{\mathbf{x}}_{0:t-1}^{(i)}), \quad \mathbf{x}_{0:t}^{(i)} = \left(\tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{x}_t^{(i)}\right). \tag{26}$$

The difference between APF and Sequential Importance Sampling Resampling (SISR) is best described by the analogy:

- APF chooses which parent particles produce offspring, based on how likely each offspring is to explain the observations $\mathbf{y}_{1:t}$ well.
- SISR chooses which offspring survive based on how well they explain the observations $\mathbf{y}_{1:t}$.

We are therefore required to sample from the distribution $\tilde{\mathbf{x}}_{0:t-1}^{(1)}, \ldots, \tilde{\mathbf{x}}_{0:t-1}^{(N)} \sim q(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t})$ which is dependent on the current observation $\mathbf{y}_t$. Moving forward we derive how resampling can be used to draw approximate samples from $q(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t})$.

By marginalizing with respect to the $\mathbf{x}_t$ we can write $p(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t}, \boldsymbol{\theta})$ as an integral of the full posterior distribution

$$p(\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t}, \boldsymbol{\theta}) = \int p(\mathbf{x}_{0:t} | \mathbf{y}_{1:t}, \boldsymbol{\theta}) \mathrm{d}\mathbf{x}_t. \tag{27}$$

The integrand can be expressed as a recursion using Bayes Theorem.

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t},\boldsymbol{\theta}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t,\boldsymbol{\theta})p(\mathbf{x}_t|\mathbf{x}_{t-1},\boldsymbol{\theta})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1},\boldsymbol{\theta})}p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1},\boldsymbol{\theta}) \tag{28}$$

Combining (27) and (28) we get

$$p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t},\boldsymbol{\theta}) = \frac{p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1},\boldsymbol{\theta})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1},\boldsymbol{\theta})}\int p(\mathbf{y}_t|\mathbf{x}_t,\boldsymbol{\theta})p(\mathbf{x}_t|\mathbf{x}_{t-1},\boldsymbol{\theta})\mathrm{d}\mathbf{x}_t. \tag{29}$$

The time $t-1$ posterior distribution can then be approximated as

$$p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1},\boldsymbol{\theta}) \approx \widehat{p}_N(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}) = \sum_{i=1}^{N} w_{t-1}^{(i)}\delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1}) \tag{30}$$

leading to the approximation

$$\begin{aligned} p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t},\boldsymbol{\theta}) &\approx \frac{1}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1},\boldsymbol{\theta})}\left[\sum_{i=1}^{N} w_{t-1}^{(i)}\delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1})\right]\int p(\mathbf{y}_t|\mathbf{x}_t,\boldsymbol{\theta})p(\mathbf{x}_t|\mathbf{x}_{t-1},\boldsymbol{\theta})\mathrm{d}\mathbf{x}_t \\ &= \sum_{i=1}^{N} v_{t-1}^{(i)}\delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1}) \end{aligned} \tag{31}$$

Where we refer to $v_{t-1}^{(0)},\ldots,v_{t-1}^{(N)}$ as **parent weights** defined by

$$v_{t-1}^{(i)} \propto w_{t-1}^{(i)}\int p(\mathbf{y}_t|\mathbf{x}_t,\boldsymbol{\theta})p(\mathbf{x}_t|\mathbf{x}_{t-1},\boldsymbol{\theta})\mathrm{d}\mathbf{x}_t. \tag{32}$$

Thus we see that resampling from the current particles using the parent weights serves as an approximation to sampling from $p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t})$, as desired. For each iteration we therefore sample a set of parent particles from the empirical distribution

$$\tilde{\mathbf{x}}_{0:t-1}^{(0)},\ldots,\tilde{\mathbf{x}}_{0:t-1}^{(N)} \sim \sum_{i=1}^{N} v_{t-1}^{(i)}\delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1}). \tag{33}$$

Each parent particles is then propagated one step into the future (producing "offspring")

$$\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{y}_{1:t},\tilde{\mathbf{x}}_{0:t-1}). \tag{34}$$

Returning to equation (32), we see that we are faced with an integral which is often difficult to compute. To avoid this, a common approach is to use a deterministic approximation of the transition density

$$p(\mathbf{x}_t|\mathbf{x}_{t-1},\boldsymbol{\theta}) \approx \delta_{f(\mathbf{x}_{t-1})}(\mathbf{x}_t) \quad \Rightarrow \quad v_{t-1}^{(i)} \approx c_1 \cdot w_{t-1}^{(i)}p\left(\mathbf{y}_t|f(\mathbf{x}_{t-1}),\boldsymbol{\theta}\right) \tag{35}$$

for some constant $c_1 \in \mathbb{R}$. In practice a good choice of $f(\mathbf{x}_{t-1})$ is the mean/median/mode. Our final consideration is that of computing the weights $w_t^{(0)},\ldots,w_t^{(N)}$ at each timestep. Each weight is computed as

$$w_t^{(i)} \propto \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t},\boldsymbol{\theta})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}. \tag{36}$$

Using the identity

$$q(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t}) = q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t},\mathbf{x}_{0:t-1}^{(i)})q(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{y}_{1:t}) \tag{37}$$

and that

$$\begin{aligned} q(\mathbf{x}_{0:t-1}^{(j)}|\mathbf{y}_{1:t}) &= \sum_{i=1}^{N} v_{t-1}^{(i)}\delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1}) \\ &= \sum_{i=1}^{N} \frac{v_{t-1}^{(i)}}{w_{t-1}^{(i)}} w_{t-1}^{(i)}\delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1}^{(j)}) \\ (\delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1}^{(j)}) \neq 0 \Leftrightarrow i=j) \quad \Rightarrow \quad &= \frac{v_{t-1}^{(j)}}{w_{t-1}^{(j)}}\sum_{i=1}^{N} w_{t-1}^{(i)}\delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1}^{(j)}) \\ &= \frac{v_{t-1}^{(j)}}{w_{t-1}^{(j)}}\widehat{p}_N(\mathbf{x}_{0:t-1}^{(j)} \mid \mathbf{y}_{1:t-1}) \end{aligned} \tag{38}$$

we can rewrite (36) as

$$\begin{aligned} w_t^{(i)} &= \frac{w_{t-1}^{(i)}}{v_{t-1}^{(i)}} \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t}, \boldsymbol{\theta})}{q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t}, \mathbf{x}_{0:t-1}^{(i)})\widehat{p}_N(\mathbf{x}_{0:t-1}^{(i)})} \\ &\propto \frac{w_{t-1}^{(i)}}{v_{t-1}^{(i)}} \frac{p(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\theta})p(\mathbf{x}_t|\mathbf{x}_{t-1}, \boldsymbol{\theta})}{q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t}, \mathbf{x}_{0:t-1}^{(i)})} \frac{p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}, \boldsymbol{\theta})}{\widehat{p}_N(\mathbf{x}_{0:t-1}^{(i)})} \\ &\approx \frac{w_{t-1}^{(i)}}{v_{t-1}^{(i)}} \frac{p(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\theta})p(\mathbf{x}_t|\mathbf{x}_{t-1}, \boldsymbol{\theta})}{q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t}, \mathbf{x}_{0:t-1}^{(i)})} \cdot 1 \end{aligned} \tag{39}$$

By choosing $q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t}, \mathbf{x}_{0:t-1}^{(i)}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \boldsymbol{\theta})$ this reduces to

$$w_t^{(i)} \approx c_2 \cdot \frac{w_{t-1}^{(i)}}{v_{t-1}^{(i)}} p(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\theta}) \tag{40}$$

for some constant $c_2 \in \mathbb{R}$. Algorithm 2 formally defines the Auxiliary Particle Filter.

Algorithm 2 Auxiliary Particle Filter

1. Initialize $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$.
2. Compute weights $w_0^{(i)} = p(\mathbf{y}_1|f(\mathbf{x}_0^{(i)}), \boldsymbol{\theta})$
3. For $t = 1, \cdots, T$:

   Sample Parents

   - Compute and normalize the parent weights $v_{t-1}^{(i)} \propto w_{t-1}^{(i)} p(\mathbf{y}_t|f(\mathbf{x}_{t-1}^{(i)}), \boldsymbol{\theta})$.
   - Sample N parent particles $\tilde{\mathbf{x}}_{0:t-1}^{(1)}, \ldots, \tilde{\mathbf{x}}_{0:t-1}^{(N)} \sim \sum_{i=1}^N v_{t-1}^{(i)} \delta_{\mathbf{x}_{0:t-1}^{(i)}}(\mathbf{x}_{0:t-1})$.

   Propagate

   - Sample $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\tilde{\mathbf{x}}_{t-1}^{(i)}, \boldsymbol{\theta})$.
   - Set $\mathbf{x}_{0:t}^{(i)} = \left[\tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{x}_t{}^{(i)}\right]$.
   - Compute and normalize the importance weights $w_t^{(i)} \propto \frac{w_{t-1}^{(i)}}{v_{t-1}^{(i)}} p(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\theta})$.

3 Parameter Inference via Particle MCMC

3.1 Overview

Our goal in this section is to develop a methodology to perform Bayesian inference on the vector of parameters $\boldsymbol{\theta}$, which specify the distribution of the initial and transition distributions of the hidden Markov process $\{\mathbf{X}_t\}_{t\geq 0}$. In Section 1.2 we hinted at the possibility of introducing a parameter prior $p(\boldsymbol{\theta})$ and making the parameter posterior $p(\boldsymbol{\theta} \mid \mathbf{x}_{0:t}, \mathbf{y}_{1:t}) \propto p(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ the target of our inference. In what follows we will see how we can use the approximate empirical distribution $\widehat{p}_N(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$ generated by a particle filter algorithm for this purpose.
Since the introduction of Monte Carlo within Metropolis (MCWM) and Grouped-Independence Metropolis-Hastings (GIMH) methods in epidemiology and population genetics (O'Neill et al., 2000; Beaumont, 2003) as a motivation to use Importance Sampling estimates in Metropolis-Hastings ratios, there has been an ever-growing literature on the topic, with work exploring their generalizations and convergence properties (Andrieu and Roberts, 2009; Andrieu and Vihola, 2015), among which Andrieu et al. (2010) gives an extensive overview of the methods and theoretical results. All such algorithms are classified as pseudo-marginal MCMC because they either replace the likelihood terms in the Metropolis-Hastings ratio with an unbiased estimate, or they estimate the whole ratio at once, without bias.

3.2 Marginal Metropolis-Hastings

Suppose, for the moment, that we have some way of sampling from the states posterior $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$ and of evaluating the marginal likelihood $p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})$ for every $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. Then, using the Metropolis-Hastings algorithm (Hastings, 1970; Rosenbluth, 2003) we can construct a Markov chain targeting $p(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ by working on the extended, joint space $\{\mathbf{X}_{0:t}, \boldsymbol{\theta}\}$.This is done by choosing a proposal density on such extended space and factorizing it in terms of a parameter proposal distribution $q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$ and a proposal for the hidden states $q(\mathbf{x}^*_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*)$, as shown below

$$q\left((\mathbf{x}^*_{0:t}, \boldsymbol{\theta}^*) \mid (\mathbf{x}_{0:t}, \boldsymbol{\theta})\right) = q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) q(\mathbf{x}^*_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*). \tag{41}$$

In this scenario, one can choose the second term on the right hand side of (41) and set it equal to the states posterior $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$ since we are assuming that we can sample from it. We can then sample from the proposal density

$$q\left((\mathbf{x}^*_{0:t}, \boldsymbol{\theta}^*) \mid (\mathbf{x}_{0:t}, \boldsymbol{\theta})\right) = q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) p(\mathbf{x}^*_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*)$$

by first sampling a candidate parameter $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})$ and then using it to sample $\mathbf{x}^*_{0:t} \sim p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*)$. Finally, we can use the parameter posterior given the data $\mathbf{y}_{1:t}$ as our prior to construct the target distribution $p(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$

$$\begin{aligned} p(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) &= p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t}) \\ &= p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) \frac{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y}_{1:t})}. \end{aligned}$$

Using these choices for the proposal distribution and the parameter posterior we get the following acceptance probability

$$\begin{aligned} \alpha\left((\boldsymbol{\theta}^*, \mathbf{x}^*_{0:t}), (\boldsymbol{\theta}, \mathbf{x}_{0:t})\right) &= \min\left\{1, \frac{p(\boldsymbol{\theta}^*, \mathbf{x}^*_{0:t} \mid \mathbf{y}_{1:t})}{p(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} \cdot \frac{q\left((\mathbf{x}_{0:t}, \boldsymbol{\theta}) \mid (\mathbf{x}^*_{0:t}, \boldsymbol{\theta}^*)\right)}{q\left((\mathbf{x}^*_{0:t}, \boldsymbol{\theta}^*) \mid (\mathbf{x}_{0:t}, \boldsymbol{\theta})\right)}\right\} \\ &= \min\left\{1, \frac{p(\mathbf{x}^*_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*) p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^*) p(\boldsymbol{\theta}^*)}{p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})} \cdot \frac{p(\mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t})} \cdot \frac{q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*) p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})}{q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}) p(\mathbf{x}^*_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*))}\right\} \\ &= \min\left\{1, \frac{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^*) p(\boldsymbol{\theta}^*) q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})}\right\}, \end{aligned} \tag{42}$$

which means that, using the log scale, a candidate $(\boldsymbol{\theta}^*, \mathbf{x}^*_{0:t})$ will be accepted if, for $u \sim \mathcal{U}(0,1)$, the following holds

$$\log(u) \leq \log p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^*) - \log p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}^*) - \log p(\boldsymbol{\theta}) + \log q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*) - \log q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}).$$

The resulting algorithm, is summarized in Algorithm 3.

Algorithm 3 Marginal Metropolis-Hastings

1. Choose $\boldsymbol{\theta}^{[0]}$, $\mathbf{x}^{[0]}_{0:t}$, and M. Sample $u^{[1:M]} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0,1)$ and compute $\log p(\boldsymbol{\theta}^{[0]})$ and $\log p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^{[0]})$.
2. For $i = 1, \cdots, M$:
   (a) Sample a candidate parameter $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{[i-1]})$, a candidate path $\mathbf{x}^*_{0:t} \sim p(\mathbf{x}^*_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*)$ and compute the marginal log-likelihood $\log p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^*)$ and log-prior $\log p(\boldsymbol{\theta}^*)$.
   (b) If $\log(u^{[i]}) \leq \log \alpha\left((\boldsymbol{\theta}^*, \mathbf{x}^*_{0:t}), (\boldsymbol{\theta}^{[i-1]}, \mathbf{x}^{[i-1]}_{0:t})\right)$ accept the move and set:
   - $\boldsymbol{\theta}^{[i]} \longleftarrow \boldsymbol{\theta}^*$
   - $\mathbf{x}^{[i]}_{0:t} \longleftarrow \mathbf{x}^*_{0:t}$
   - $\log p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^{[i]}) \longleftarrow \log p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^*)$
   - $\log p(\boldsymbol{\theta}^{[i]}) \longleftarrow \log p(\boldsymbol{\theta}^*)$

   (c) Otherwise, reject and set:
   - $\boldsymbol{\theta}^{[i]} \longleftarrow \boldsymbol{\theta}^{[i-1]}$
   - $\mathbf{x}^{[i]}_{0:t} \longleftarrow \mathbf{x}^{[i-1]}_{0:t}$
   - $\log p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^{[i]}) \longleftarrow \log p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^{[i-1]})$
   - $\log p(\boldsymbol{\theta}^{[i]}) \longleftarrow \log p(\boldsymbol{\theta}^*)$

We can see that thanks to the choice of our proposal distribution, all dependence on $\mathbf{x}_{0:t}$ has been cancelled out, resulting in a "marginal" algorithm, thus explaining the name. In Algorithm 3 we keep track of $\mathbf{x}_{0:t}^{[i]}$ in order to already have a sample from the posterior with the final parameter $\boldsymbol{\theta}^{[M]}$, once the algorithm ends. Of course, in general, it is not easy to sample from the posterior $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$ and the marginal likelihood requires an intractable integration. We can find an expression for such integral by using equation (13).

$$
\begin{aligned}
1 &= \int p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) d\mathbf{x}_{0:t} \\
&= \int p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) \frac{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta}) p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})} d\mathbf{x}_{0:t} && \text{Using (13)} \\
&= \frac{1}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})} \int p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta}) p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) d\mathbf{x}_{0:t} && \text{Independent of } \mathbf{x}_{0:t}
\end{aligned}
$$

This leads to the following expression for $p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})$

$$
p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) = \int p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta}) p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) d\mathbf{x}_{0:t}. \tag{43}
$$

which can be plugged into the expression below to find an expression for the marginal likelihood

$$
p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) = p(\mathbf{y}_1 \mid \boldsymbol{\theta}) \prod_{k=2}^{t} p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta})
$$

3.3 Particle-Marginal Metropolis-Hastings

Of course, the whole point of SMC, is that the posterior $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta})$ and the marginal likelihood $p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})$ are intractable and we need to approximate them. The Particle-Marginal Metropolis-Hastings is an approximation of the ideal Marginal Metropolis-Hastings algorithm which uses a particle approximation to construct a proposal distribution $p(\mathbf{x}_{0:t}^* \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*)$ and to evaluate $p(\mathbf{y}_{1:t} \mid \theta)$.
As we have seen in the previous sections, the Bootstrap Filter and the Auxiliary Particle Filter, given a parameter vector $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, return a set of N particles $\mathbf{x}_{0:t}^{(1)}, \ldots, \mathbf{x}_{0:t}^{(N)}$ and their respective normalized weights $w_t^{(1)}, \ldots w_t^{(N)}$ which can be used to construct an empirical approximation to the posterior distribution

$$
\widehat{p}_N(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}) = \sum_{i=1}^{N} w_t(\mathbf{x}_{0:t}^{(i)}) \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}).
$$

and, by plugging the empirical distribution in to (43) we obtain a particle approximation of $p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta})$. To do this, we use a simple importance sampling argument to get an approximation to the marginal likelihood. For instance, for Bootstrap we have

$$
\begin{aligned}
p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) &= \int p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta}) p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) d\mathbf{x}_{0:t} \\
&= \int \frac{p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta}) p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta})}{q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})} q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\
&= \int \frac{p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}, \boldsymbol{\theta}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta}) p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta})}{q(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}) p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta})} q(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\
&= \int p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) \tilde{w}_{t-1}(\mathbf{x}_{0:t-1}) \widehat{q}_N(\mathbf{x}_{0:t}) d\mathbf{x}_{0:t} \\
&= \frac{1}{N} \sum_{i=1}^{N} \int p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}) \tilde{w}_{t-1}(\mathbf{x}_{0:t-1}) \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}) d\mathbf{x}_{0:t} \\
&= \frac{1}{N} \sum_{i=1}^{N} p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) \tilde{w}_{t-1}(\mathbf{x}_{0:t-1}^{(i)}) \\
&= \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_t(\mathbf{x}_t^{(i)}). && \text{resampling}
\end{aligned}
$$

The PMMH algorithm is then summarized in Algorithm 4. The acceptance ratio $\widehat{\alpha}_N$ is an approximation of (42) where we have replaced the marginal likelihood by its **unbiased** approximation coming from the Particle Filter used (Moral et al., 2006). Thanks to this estimator being unbiased, this is an *exact approximation*, meaning that the Markov kernel still leaves $p(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ invariant (Andrieu and Roberts, 2009; Andrieu and Vihola, 2015).

Algorithm 4 Particle Marginal Metropolis-Hastings

1. Choose $\boldsymbol{\theta}^{[0]}$, and M. Run an SMC algorithm with $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^{[0]})$ as target. Sample $\mathbf{x}_{0:t}^{[0]}$ from $\widehat{p}_N(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^{[0]})$ and store $\log \widehat{p}_N(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^{[0]})$. Sample $u^{[1:M]} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0,1)$ and compute $\log p(\boldsymbol{\theta}^{[0]})$.
2. For $i = 1, \cdots, M$:
   (a) Sample a candidate parameter $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{[i-1]})$, run an SMC algorithm to get $\mathbf{x}_{0:t}^* \sim \widehat{p}_N(\mathbf{x}_{0:t}^* \mid \mathbf{y}_{1:t}, \boldsymbol{\theta}^*)$ and $\log \widehat{p}_N(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^*)$. Compute the log-prior $\log p(\boldsymbol{\theta}^*)$.
   (b) If $\log(u^{[i]}) \leq \log \widehat{\alpha}_N \left((\boldsymbol{\theta}^*, \mathbf{x}_{0:t}^*), (\boldsymbol{\theta}^{[i-1]}, \mathbf{x}_{0:t}^{[i-1]}) \right)$ accept the move and set:
      - $\boldsymbol{\theta}^{[i]} \longleftarrow \boldsymbol{\theta}^*$
      - $\mathbf{x}_{0:t}^{[i]} \longleftarrow \mathbf{x}_{0:t}^*$
      - $\log \widehat{p}_N(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^{[i]}) \longleftarrow \log \widehat{p}_N(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^*)$
      - $\log p(\boldsymbol{\theta}^{[i]}) \longleftarrow \log p(\boldsymbol{\theta}^*)$
   (c) Otherwise, reject and set:
      - $\boldsymbol{\theta}^{[i]} \longleftarrow \boldsymbol{\theta}^{[i-1]}$
      - $\mathbf{x}_{0:t}^{[i]} \longleftarrow \mathbf{x}_{0:t}^{[i-1]}$
      - $\log \widehat{p}_N(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^{[i]}) \longleftarrow \log \widehat{p}_N(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^{[i-1]})$
      - $\log p(\boldsymbol{\theta}^{[i]}) \longleftarrow \log p(\boldsymbol{\theta}^*)$

3.4 Pseudo-Marginal Exactness Intuition

Previously, we have claimed that if we substitute $p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})$ with an unbiased (and non-negative) estimator then the resulting MCMC algorithm will still target the desired distribution (Del Moral, 2004; Moral et al., 2006). We now aim to present a sketch of the proof of this statement.
Exact Bayesian inference on $\boldsymbol{\theta}$ would usually be based on the posterior of the parameter given the data $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})$, without considering the hidden states $\mathbf{x}_{0:t}$. In such case the acceptance ratio would, as already noted in Section 3.2, be identical to (42)

$$\frac{p(\boldsymbol{\theta}^* \mid \mathbf{y}_{1:t}) q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t}) q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})} = \frac{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}^*) p(\boldsymbol{\theta}^*) q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^*)}{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) q(\boldsymbol{\theta}^* \mid \boldsymbol{\theta})}.$$

We want to show that going from the Marginal Algorithm 3 to the Particle Marginal Algorithm 4 we are still leaving $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})$ invariant, and thus the resulting samples $\boldsymbol{\theta}^{[1]}, \ldots, \boldsymbol{\theta}^{[M]}$ (as M grows) are distributed according to $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})$. The key result that we use is that of unbiasedness. That is, it can be shown that the likelihood estimate coming from the particle filter is an unbiased estimator of the true marginal likelihood

$$\mathbb{E}_{p(\mathbf{x}_{0:t} \mid \boldsymbol{\theta})} \left[\widehat{p}(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) \right] = \mathbb{E}_{p(\mathbf{x}_{0:t} \mid \boldsymbol{\theta})} \left[\widehat{p}(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) \right] = p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}).$$

In the expression above we have made explicit the dependence of our marginal likelihood estimator on the hidden states $\mathbf{x}_{0:t}$, indeed remember that we use the product of the mean of the unnormalized weights at every time step to estimate $p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})$

$$\widehat{p}(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) = \prod_{k=1}^{t} \left[\frac{1}{N} \sum_{i=1}^{N} \tilde{w}_k(\mathbf{x}_{0:t}^{(i)}) \right] = \prod_{k=1}^{t} \left[\frac{1}{N} \sum_{i=1}^{N} p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}) \right].$$

The final observation is that when we are working on the extended space $(\mathbf{X}_{0:t}, \boldsymbol{\theta})$ and we sample from the joint posterior $p(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$, if we consider the parameter samples $\boldsymbol{\theta}^{[1]}, \ldots, \boldsymbol{\theta}^{[M]}$ on their own, this

will be equivalent to "marginalizing" out the $\mathbf{x}_{0:t}$ samples. For this reason our aim is to show that by considering only such samples (and thus marginalizing the hidden states out) we recover the posterior parameter distribution $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})$

$$\begin{aligned}
\int \widehat{p}(\boldsymbol{\theta}, \mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t} &= \int \frac{\widehat{p}(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) p(\mathbf{x}_{0:t} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y}_{1:t})} d\mathbf{x}_{0:t} \\
&= \int \frac{\widehat{p}(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) p(\mathbf{x}_{0:t} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})} d\mathbf{x}_{0:t} \\
&= \int \frac{p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t}) \widehat{p}(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) p(\mathbf{x}_{0:t} \mid \boldsymbol{\theta})}{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})} d\mathbf{x}_{0:t} \\
&= \frac{p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})} \int \widehat{p}(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) p(\mathbf{x}_{0:t}, \boldsymbol{\theta}) d\mathbf{x}_{0:t} \\
&= \frac{p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})} \mathbb{E}_{p(\mathbf{x}_{0:t} \mid \boldsymbol{\theta})} \left[\widehat{p}(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) \right] \\
&= \frac{p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta})} p(\mathbf{y}_{1:t} \mid \boldsymbol{\theta}) \qquad \text{unbiasedness} \\
&= p(\boldsymbol{\theta} \mid \mathbf{y}_{1:t})
\end{aligned}$$

4 Applications

Code used to produce these results can be found at https://github.com/dfcorbin/SeqInference.

4.1 Stochastic Volatility Model

The Stochastic Volatility Model is a well known state-space model, commonly applied to financial time series in order to predict its underlying volatility over time e.g. the volatility of a stock price. The underlying states are assumed to follow an Auto-Regressive process of order 1 and the volatility of the observations is given by a non-negative function of the states:

$$X_t = \alpha X_{t-1} + \sigma \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, 1) \tag{44}$$

$$Y_t = \beta \cdot \exp\left(\frac{X_t}{2}\right) \cdot \zeta_t, \quad \zeta_t \sim \mathcal{N}(0, 1). \tag{45}$$

which is equivalent to taking $p(x_t|x_{t-1}) = \mathcal{N}(\alpha X_{t-1}, \sigma^2)$ and $p(y_t|x_t) = N\left(0, \beta^2 \exp(x_t)\right)$. For this project we focus on inference of the latent states and the model parameters α, σ and β. Figure 4 shows some synthetic data generated according to (44) and (45). From this figure one can see that the observations display a very low/high volatility as the states decrease/increase in value.

4.2 Results

While applying the above methods to the Stochastic Volatility Model, we experimented with two different approaches, both of which are presented and assessed here. The main difference between them is the choice of distributions used for the priors and proposals.

Example 1: Gamma/Beta Priors with Gamma/Beta Proposals.

The data used in this example was simulated from the model described above with 500 states and parameters set to $\alpha = 0.9$, $\beta = 1$ and $\sigma = 1$. For the purposes of this demonstration we fix $\beta = 1$ and we focus on estimating α and σ. This it motivated by the fact that in our experimentation, β with a gamma (or a normal) prior/transition would not converge, differently from Example 2. To estimate the parameters we run a Particle Marginal Metropolis-Hastings with 800 iterations. The proposal distributions are constructed using the Bootstrap filter described in Section 2.5 with $N = 300$ particles. The initial values of the parameters were set to $\alpha^{[0]} = 0.3$ and $\alpha^{[0]} = 0.5$ The priors for the parameters were set to

$$p(\alpha) = \text{Beta}\left(6, \frac{6}{0.8} - 6\right)$$

$$p(\sigma) = \text{Gamma}\left(5, \frac{0.5}{5}\right)$$

And the candidate parameters are drawn from the following proposal distributions are

$$q(\alpha^*|\alpha) = \text{Beta}\left(100, \frac{100}{\alpha} - 100\right)$$

$$q(\sigma^*|\sigma) = \text{Gamma}\left(200, \frac{\sigma}{200}\right).$$

As seen in Figure 5, the prior and the proposal are chosen such that they have the right support. In general, for Stochastic volatility models we expect α to be in the range $(0, 1)$ and hence the Beta distribution seems like a natural choice. On the other hand, the only restriction placed on σ is that it is non-negative so we have used a Gamma distribution. Note how in both cases the prior is mostly flat, whereas the proposal distributions are chosen such that they are quite concentrated around the current parameter samples.

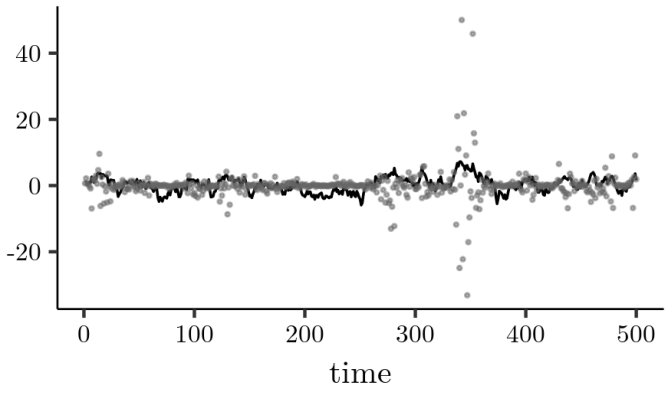


Figure 4: Synthetic data generated from the Stochastic Volatility Model defined in (44), (45), with parameters $\alpha = 0.9$, $\beta = 1$ and $\sigma = 1$.

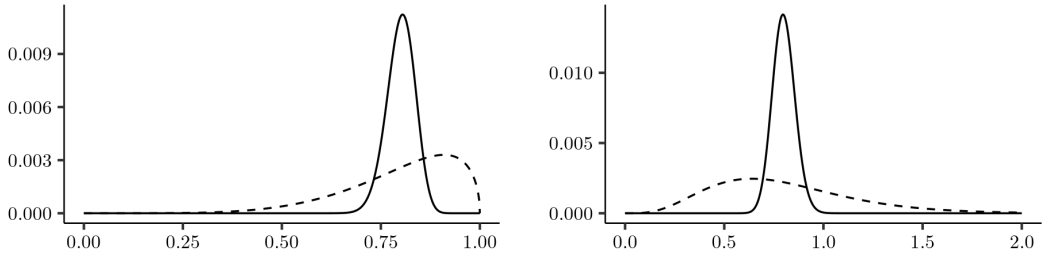


Figure 5: The prior (dashed line) and proposal (solid) distributions for α (left) and σ (right). The proposals are both centered at the current parameter samples, both set to be 0.8.

The number of iterations was chosen to be 800 after trying larger number of iterations and noticing that the Markov Chain does little progress after that point, due to sticking. This is probably due to the choice of transition distribution for α: the Beta distribution becomes highly skewed at the boundaries of its support, and the true value of α is found almost at such a boundary. This results in the PMMH suggesting values that are clustered around a very narrow portion of the support, between 0.91 and 1.0. Despite presenting the results for one set on starting values, using different starting values leads to very similar results, both in terms of the behaviour of the Markov Chains and the parameter posteriors. Furthermore, the auto-correlation plots show that the samples are highly correlated to one another, so

the Markov property is not . This makes sense as every sample is generated based on the previous one and the proposal distributions are quite concentrated around the previous value.
From the trace plots in Figure 7 it can be seen that the posterior means of the parameter samples, $\hat{\alpha} = 0.918$ and $\hat{\sigma} = 0.943$, obtained after discarding the burn-in iterations, get close to the real values. This is also visible in Figure 8 as the posteriors are approximately centered around the true values. Finally, in Figure 9 we note how as we progress along the Markov Chain, the parameter samples lead to increasingly better state estimates.

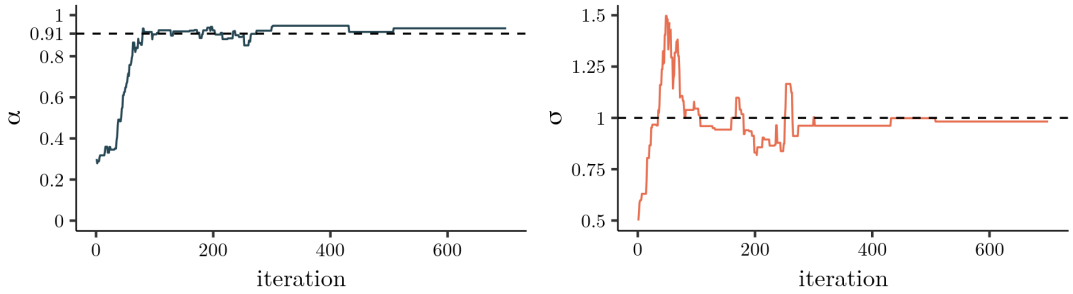


Figure 6: Trace of the Markov Chain for α (left) and σ (right). The dotted lines represent the true values of the parameters.

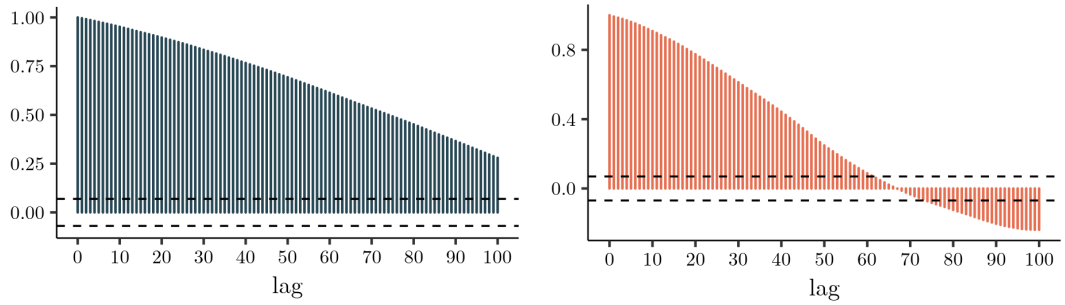


Figure 7: Auto-correlation plots for α (left) and σ (right).

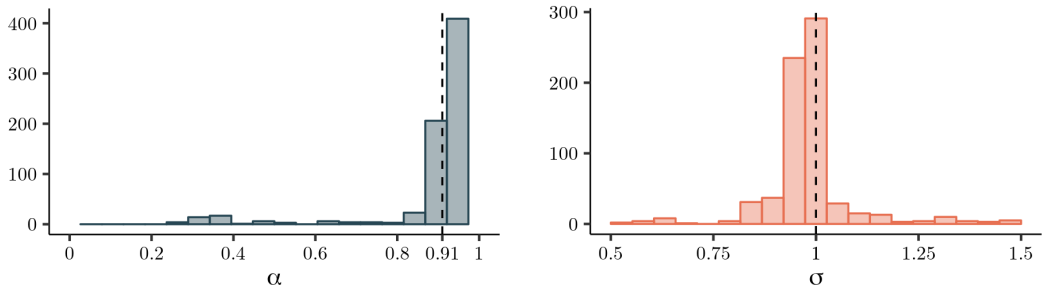


Figure 8: Posterior estimates obtained for α (left) and σ (right) obtained from the PMH algorithm.

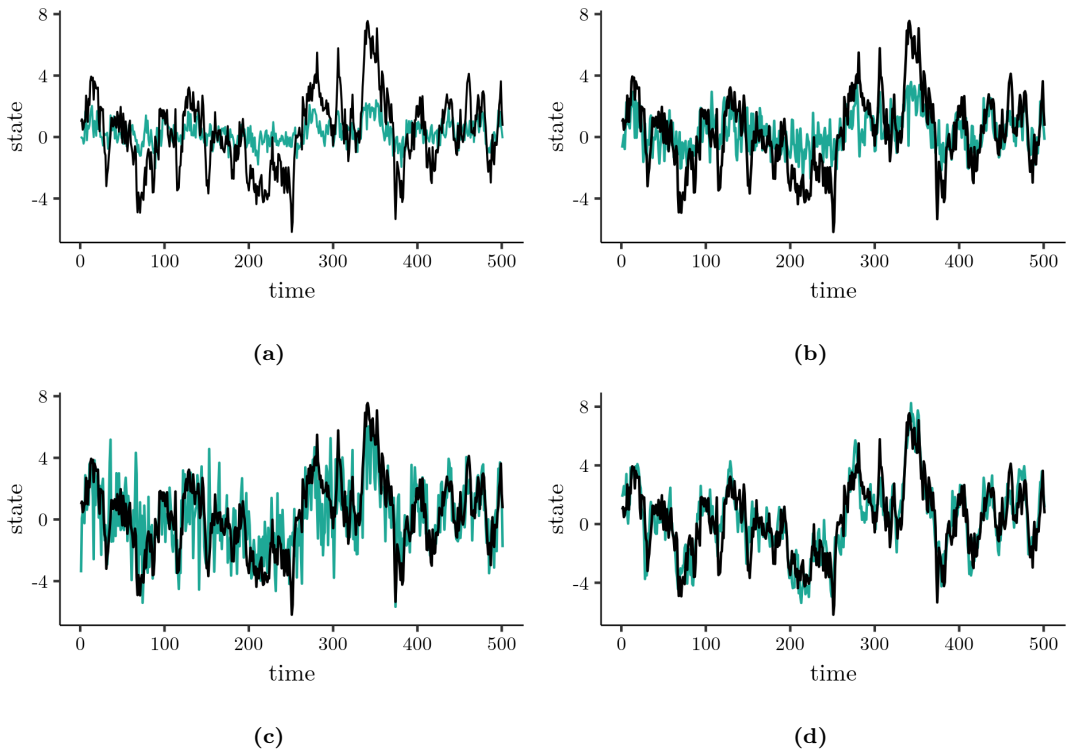


Figure 9: True state values (black) against the estimates (green) based on the PMH at iteration 0 (a.), 50 (b.), 100 (c.), and 800 (d.).

Example 2: Gamma/Truncated Normal Priors with Reflective Random Walk Proposal.

As with the previous example, we choose $\boldsymbol{\theta} = (0.91, 1, 1)$, however we now take a larger data set containing 1000 observations shown in Figure 10. Since we are working with more data, we increase the number of particles in the filter to $N = 800$.

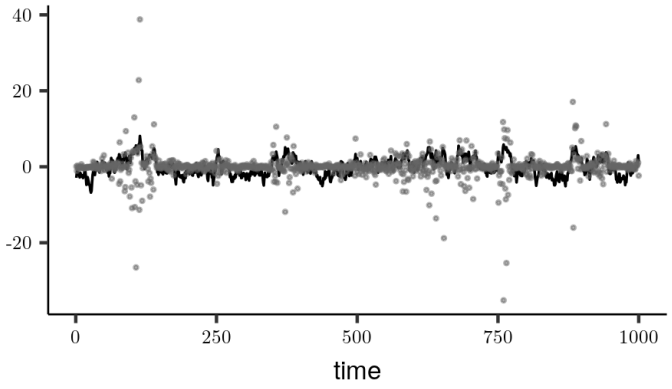


Figure 10: Simulated data based on the Stochastic Volatility Model with 1000 states based on $\theta = (0.91, 1, 1)$.

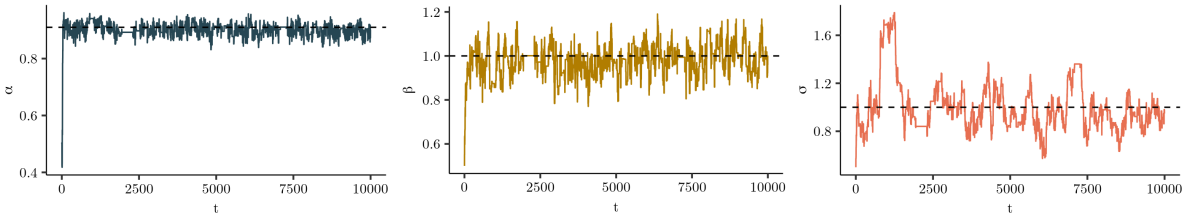
Once again, we will use a Bootstrap Particle Filter within the Metropolis Hastings algorithm in order to estimate the model parameters. In contrast to Example 1, we relax the constraint on α to $|\alpha| < 1$, allowing us to model data where the states are negatively correlated. As a result, we use the prior distribution

$$p(\alpha) = \text{Trunc-Normal}_{(-1,1)}(0.9, 0.5)$$
$$p(\beta) = p(\sigma) = \text{Gamma}(2, 2)$$

with the proposal

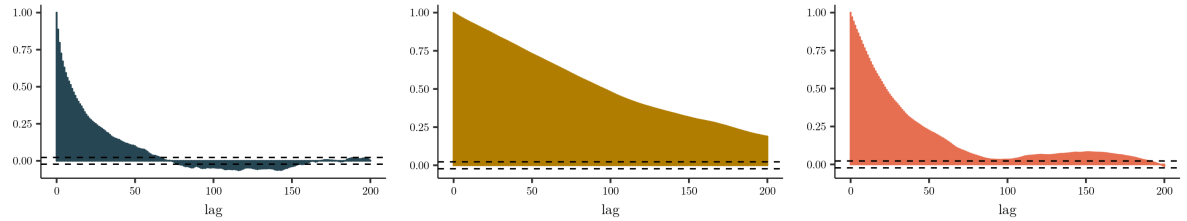
$$q(\theta^*|\theta) = \mathcal{N}(\theta, \sigma_{\text{prop}}\boldsymbol{I}).$$

Since we are working with bounded parameters, we apply reflection at each of their respective boundaries. For example, suppose we propose $\tilde{\alpha}^* = 1.3$; we know that $|\alpha| < 1$ so we reflect the excess back in the opposite direction to get $\alpha^* = 1-(1.3-1) = 0.7$. Initialising $\boldsymbol{\theta}$ at $\boldsymbol{\theta}_0 = (0.5, 0.5, 0.5)$, Figure 11 shows the trace plots generated from the PMH algorithm with the above prior/proposal distributions. From these plots we can be reasonably certain that α and σ have converged to their stationary regime. However, β's trace plot still displays some mildly erratic behaviour. To asses the convergence further, we turn to the ACF plots (Figure 12). It appears that only β's ACF plot has failed to decay, suggesting that it may not have fully converged. Regardless, the approximated posterior distributions shown in Figure 13 are all (roughly) centred on the true underlying model parameters. Overall, it appears that the PMH algorithm has done a good job at approximating samples from the posterior distribution.



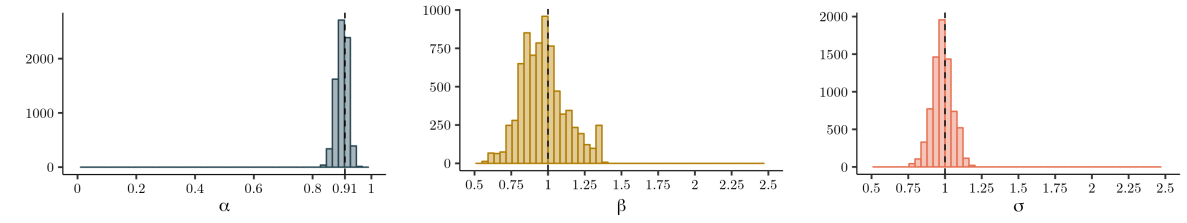
(a) Trace plot for α **(b)** Trace plot for β **(c)** Trace plot for σ

Figure 11: Trace plots for model parameters (chain length = 10000) generated using the Truncated-Normal/Gamma prior distributions with the reflective random walk proposal. Acceptance rate of 21%.



(a) ACF plot for α **(b)** ACF plot for β **(c)** ACF plot for σ

Figure 12: ACF plots for model parameters generated using the Truncated-Normal/Gamma prior distributions with the reflective random walk proposal. Acceptance rate of 21%.



(a) α **(b)** β **(c)** σ

Figure 13: Histogram of samples drawn from the approximated posterior distributions del parameters generated using the Truncated-Normal/Gamma prior distributions with the reflective random walk proposal. Acceptance rate of 21%.

5 Conclusion

We started by introducing state space models and, although we focused on the Stochastic Volatility model, the methods described in this report can be applied to a wide range of State Space Models, both for state and parameter inference. Both the Bootstrap and Auxiliary Particle Filter were successful at inferring the latent states from synthetic data; we observed no strong difference in performance between the two. Unfortunately both of the aforementioned algorithms suffer from a common limitation - as the number of observations (or equivalently hidden states) grows, we must increase the number of particles accordingly to avoid degeneracy. A failure to do this would result in a small ESS and a poor approximation of the states. Increasing the number of particles then comes at a significant computational cost.

For the case where the number of observations T is relatively small (e.g. $T \leq 1000$), the Particle Metropolis Hastings algorithm proved highly effective at approximately sampling states and parameters from their respective posterior distributions. As the number of observations increased, for the reasons explained above, the particle filters run at each iteration became increasingly costly. A more in-depth analysis is required to understand the influence the choice of proposal distributions have on the results.

A Continuous approximation of the Sampling Distribution

A.1 Discrete Distributions, PMF and CDF

Suppose we have a discrete random variable X taking k different values

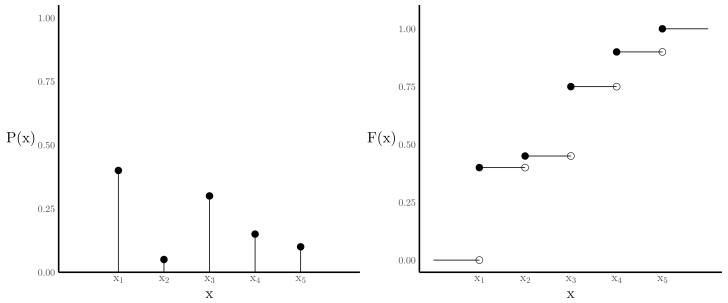
$$\mathbb{P}[X = x_i] = p_i \qquad \forall\, i = 1, \ldots, k,$$

where $0 \leq p_i \leq 1$ and $\sum_{i=1}^{k} p_i = 1$. It's **probability mass function** is then equal to p_i at $X = x_i$ and 0 everywhere else, as shown in Figure 14a

$$P_X(x) = \begin{cases} p_i & x = x_i \\ 0 & x \neq \{x_1, \ldots, x_k\} \end{cases}.$$

The **cumulative distribution function** of X, denoted $F_X(x)$, gives us the probability that the state of X is less than or equal to x (see Figure 14b)

$$F_X(x) = \sum_{x_i \leq x} \mathbb{P}[X = x_i] \qquad\qquad -\infty < x < \infty.$$



(a) Probability Mass Function **(b)** Cumulative Distribution Function

Figure 14: a) PMF, b) CDF for a categorical random variable with $K = 5$ different states.

A.2 Rewriting Discrete Distributions as Continuous Distributions

We can also express the CDF as a sum of step functions. For this purpose we define $u_y(x)$

$$u_y(x) = \begin{cases} 1 & x \geq y \\ 0 & x < y \end{cases},$$

and then rewrite the CDF as follows

$$F_X(x) = \sum_{i=1}^{N} \mathbb{P}\left[X = x_i\right] u_{x^{(i)}}(x) = \sum_{i=1}^{N} p_i u_{x^{(i)}}(x)$$

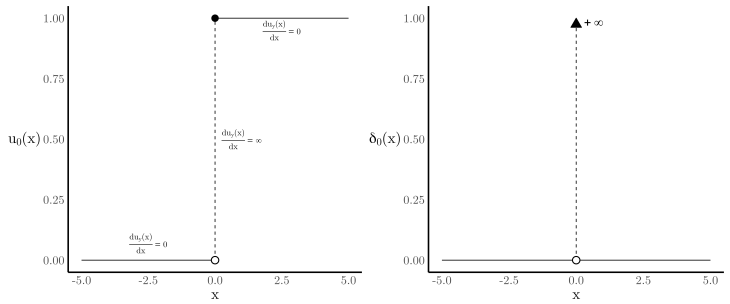
This way of writing the CDF is more useful because, if we can find a suitable derivative for the step function, then we can find an approximation to the PDF. Unfortunately, the step function has a discontinuity at $x = y$. The solution to this problem lies in defining the derivative of the step function ourselves. We call this the **Dirac Delta function** and it satisfies the following properties (for any function $g(x)$)

$$\frac{d\, u_y(x)}{dx} = \delta_y(x) = \begin{cases} \infty & x = y \\ 0 & x \neq y \end{cases}$$

$$\int g(x)\delta_y(x)dx = g(y)$$

Now we can find an approximate continuous PDF for X by taking the derivative of $F_X(x)$

$$p_X(x) = \frac{dF_X(x)}{dx} = \sum_{i=1}^{N} p_i \frac{du_{x^{(i)}}(x)}{dx} = \sum_{i=1}^{N} p_i \delta_{x^{(i)}}(x)$$



(a) Step Function $u_0(x)$ **(b)** Dirac Delta $\delta_0(x)$

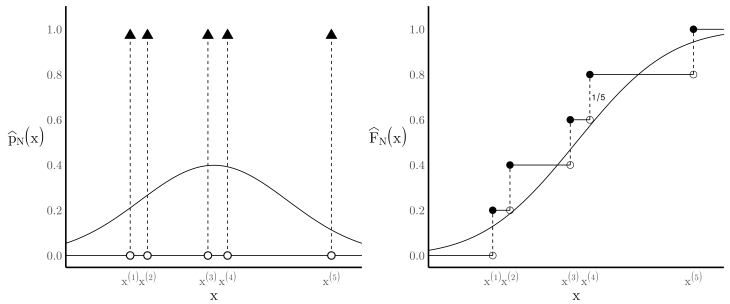
Figure 15: The Dirac-delta function is the derivative of the step function.

A.3 Continuous approximation of an Empirical PDF

Suppose now that we obtain realizations $x^{(1)}, \ldots, x^{(N)}$ of a sample $X^{(1)}, \ldots, X^{(N)}$ from a **probability density function** $p_X(x)$. Since we only have a finite number N of such samples, we need to find a way of obtaining an approximate representation of $p_X(x)$ from them. One step towards this goal is to construct the **empirical distribution function**

$$\widehat{F}_N(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(X^{(i)} \leq x) = \sum_{i=1}^{N} \frac{1}{N}\mathbb{I}(X^{(i)} \leq x),$$

which essentially uses the indicator function to count how many observations of the sample are less than or equal to x and then divides it by N, the number of samples. Another way of interpreting it is that is assigns probability $\frac{1}{N}$ to each sample. Practically, this means that $\widehat{F}_N(x)$ will be conceptually similar to Figure 14b, but with jumps of $\frac{1}{N}$ at every sample.
For instance, suppose that we sample $N = 5$ times from a standard normal $\mathcal{N}(0, 1)$ and we sort the samples in increasing order, as shown in Figure 16a. We can build a CDF with jumps of $\frac{1}{5}$ at every sample $x^{(i)}$ (see Figure 16b), and we can see how it gives a decent approximation of the true CDF. Importantly, by the Glivenko-Cantelli theorem (Billingsley, 1986, Chapter 4) we know that the empirical CDF converges to the true CDF as N increases.



(a) Samples from $\mathcal{N}(0, 1)$ **(b)** Empirical CDF

Figure 16: a) The true PDF $\mathcal{N}(0, 1)$ is being approximated by a weighted sum of Dirac-delta functions. b) The true CDF in b) can be approximated by a weighted sum of step functions.

At this point, we can follow the same steps as above: rewrite the empirical CDF as a weighted sum of step functions, and then take the derivative to find an approximate PDF using the Dirac-delta function.

$$\widehat{F}_N(x) = \sum_{i=1}^{N} \frac{1}{N} u_{x^{(i)}}(x) \quad \Longrightarrow \quad \widehat{p}_N(x) = \sum_{i=1}^{N} \frac{1}{N} \delta_{x^{(i)}}(x)$$

References

C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient monte carlo computations. *Ann. Statist.*, 37(2):697–725, 04 2009. doi: 10.1214/07-AOS574. URL https://doi.org/10.1214/07-AOS574.

C. Andrieu and M. Vihola. Convergence properties of pseudo-marginal markov chain monte carlo algorithms. *Ann. Appl. Probab.*, 25(2):1030–1077, 04 2015. doi: 10.1214/14-AAP1022. URL https://doi.org/10.1214/14-AAP1022.

C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010. doi: 10.1111/j.1467-9868.2009.00736.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2009.00736.x.

M. A. Beaumont. Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160, 2003. ISSN 0016-6731. URL https://www.genetics.org/content/164/3/1139.

P. Billingsley. *Probability and Measure*. John Wiley and Sons, second edition, 1986.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080/01621459.2017.1285773.

P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems With Applications*, volume 100. 05 2004. ISBN 0387202684. doi: 10.1007/978-1-4684-9393-1.

A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

N. J. Gordon, D. J. Salmond, and A. F. 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.

W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444. URL http://www.jstor.org/stable/2334940.

L. Kish. *Survey sampling*. J. Wiley, 1965. URL https://books.google.co.uk/books?id=3xVHAQAAIAAJ.

D. J. C. MacKay. *Information Theory, Inference Learning Algorithms*. Cambridge University Press, USA, 2002. ISBN 0521642981.

P. D. Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 68(3):411–436, 2006. ISSN 13697412, 14679868. URL http://www.jstor.org/stable/3879283.

P. D. O'Neill, D. J. Balding, N. G. Becker, M. Eerola, and D. Mollison. Analyses of infectious disease data from household outbreaks by markov chain monte carlo methods. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 49(4):517–542, 2000. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2680786.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999. ISSN 01621459. URL http://www.jstor.org/stable/2670179.

M. N. Rosenbluth. Genesis of the monte carlo algorithm for statistical mechanics. *AIP Conference Proceedings*, 690(1):22–30, 2003. doi: 10.1063/1.1632112. URL https://aip.scitation.org/doi/abs/10.1063/1.1632112.

G. Welch, G. Bishop, et al. An introduction to the kalman filter. 1995.

As a part of our project we have written an R package, `SMC`, which is mostly written in C++ using the `Rcpp/RcppArmadillo` package. In this tutorial we provide an overview of the different algorithms implemented in the `SMC` package. For a description of the algorithms used within this tutorial, we refer the reader to the primary report.

The algorithms within `SMC` have been written in order to conduct analysis on data which can be modelled by a Stochastic Volatility Model (SVM):

$$X_t = \alpha X_{t-1} + \epsilon_t, \quad \epsilon \sim \mathcal{N}(0,1) \tag{1}$$

$$Y_t = \beta \exp(X_t/2)\zeta_t, \quad \zeta_t \sim \mathcal{N}(0,1) \tag{2}$$

We start by defining our model parameters $\theta = (\alpha, \beta, \sigma)$ and generating synthetic data from the associated SVM.

```
theta <- c(0.91, 1, 1)
tmax <- 1000
set.seed(1234)
data <- SMC::stochastic_volatility(tmax, theta)
head(data)
```

```
##            x          y
## 1 -2.9113404  0.0647087
## 2 -1.5648786 -1.0726622
## 3 -0.9949148  0.3077198
## 4 -1.4801125 -0.2607910
## 5 -1.9113543 -0.3422655
## 6 -2.2165251 -0.3295994
```

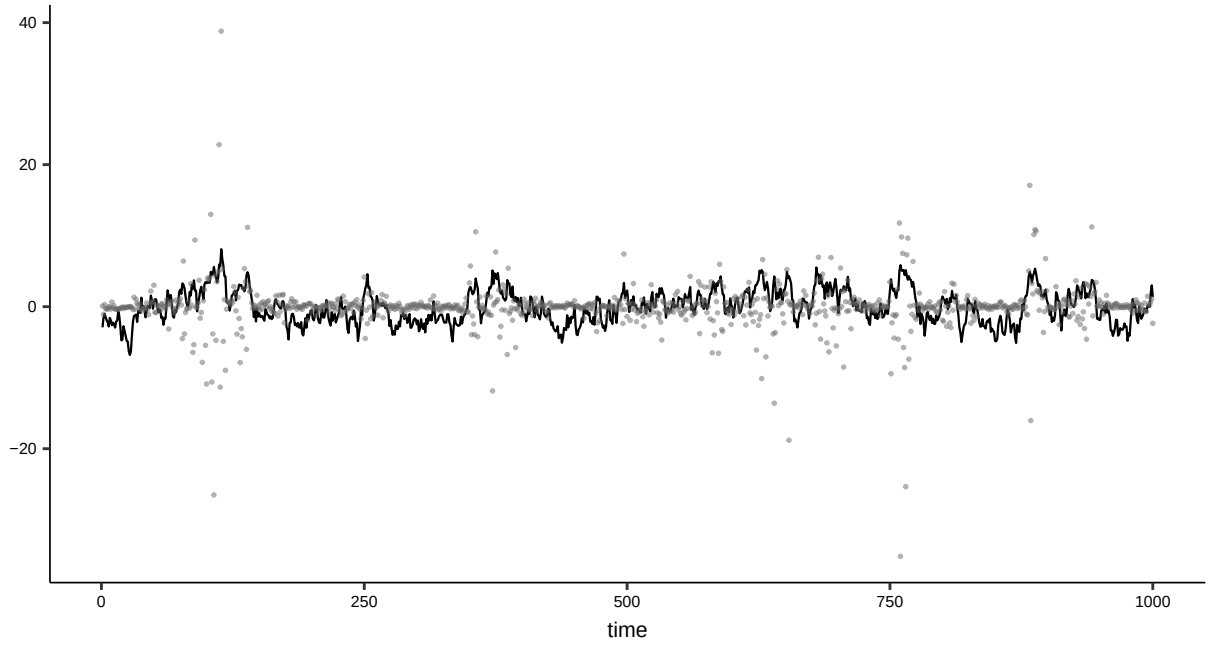


Figure 1: Synthetic data generated according to the Stochastic Volatility model with model parameters theta = (0.91, 1, 1).

Figure 1 shows a plot of the data generated using `SMC::stochastic_volatility()`. One can clearly see that the volatility of the observations, represented by the grey points, decreases as the states decrease. In practice we do not have access to the unknown states $x_{0:t}$, however we can estimate them through the use of the **Bootstrap Partcile Filter** (BSF) and the **Auxiliary Particle Filter**.

```r
obs <- data$y
N <- 400 # We use 400 particles.
BSF_fit <- SMC::BSF(obs, N, theta) # Assumes we know the true model paramters.
APF_fit <- SMC::APF(obs, N, theta)

# Compute the MSE:
cat("Bootstrap MSE: ", sum((BSF_fit$states[-1] - data$x)^2), "\n",
    "Auxiliary MSE: ", sum((APF_fit$states[-1] - data$x)^2), sep = "")
```

```
## Bootstrap MSE: 1553.523
## Auxiliary MSE: 1569.248
```

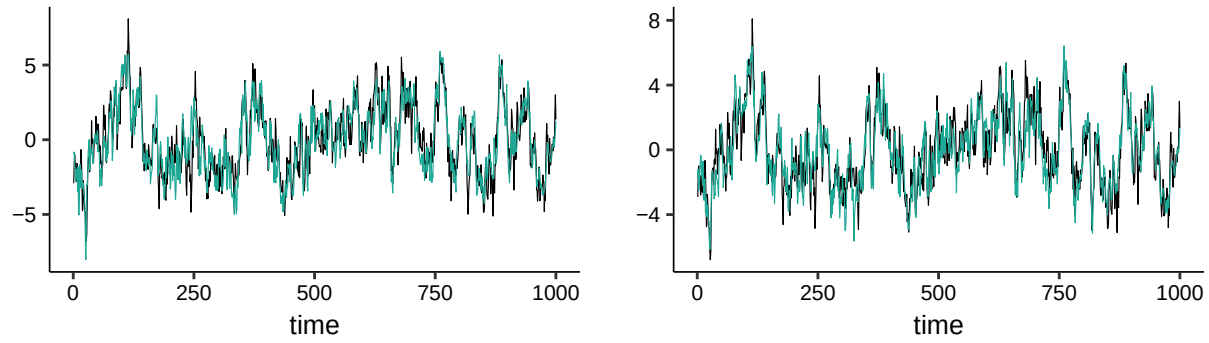


Figure 2: Results of running running the BSF and APF (Left/Right) on the synthetic data. The true states are given by the solid black line, and the filtered states are given by the dark green line.

Figure 2 shows the filtered states plotted against the true synthetic states. In this context there does not seem to be a huge difference in performance. We have of course taken for granted that we know the true model parameters used to generate the data. In practice we often must estimate the model parameters from the data. Working in the offline setting, we can do this using a Pseudo-Marginal Metropolis Hastings (PMMH) algorithm. In short, one can use the above particle filters to approximate the marinal likelihood of the observations

$$\hat{P}_N(y_{0:t}|\boldsymbol{\theta}) \approx p(y_{0:t}|\boldsymbol{\theta}). \tag{3}$$

We can incorporate this approximation into the Metropolis Hastings algorithm, and use it to accept/reject proposed values. For further details of this algorithm we once again refer the reader to the main report.

There are two versions of PMMH algorithm implemented within the `SMC` package. The first, `pmmh1`, estimates parameters α and σ **only**, using the prior/proposal

$$p(\alpha) = \text{Beta}\left((6, \frac{6}{0.8} - 6\right), \quad q(\alpha^*|\alpha) = \text{Beta}\left(100, \frac{100}{\alpha} - 100\right) \tag{4}$$

$$p(\sigma) = \text{Gamma}(5, \frac{0.5}{5}), \quad q(\sigma^*|\sigma) = \text{Beta}(200, \frac{\sigma}{200}). \tag{5}$$

The second, `pmmh2` estimates all three parameters $\theta = (\alpha, \beta, \sigma)$ and uses the prior

$$p(\alpha) = \text{Trunc-Normal}_{(-1,1)}(0.9, 0.5) \tag{6}$$

$$p(\beta) = p(\sigma) = \text{Gamma}(2, 2) \tag{7}$$

with the proposal

$$q(\theta^*|\theta) = \mathcal{N}(\theta, \sigma_{\text{prop}}\boldsymbol{I}). \tag{8}$$

Since all of the parameters are bounded (σ and β must be positive; $\|\alpha\| < 1$), we have applied reflection at each of their respective boundaries. This ensures proposals are only ever in feasible regions. In this tutorial we will show the use of `pmmh2`:

```
pmmh2_fit <- SMC::pmmh2(10000, data$y, 800, c(0.5, 0.5, 0.5), 0.045)
```

This function outputs a list containing the number of accepted steps and a matrix whose columns represent a markov chain for each parameter.

```
acceptance_rate <- pmmh2_fit$Accepted / nrow(pmmh2_fit$chain[-(1:2500),])
cat("Acceptance rate: ", acceptance_rate * 100, "%", sep = "")
```

```
## Acceptance rate: 20.98387%
```

The acceptance rate should ideally be around 23.4%. Let's examine the trace plots:

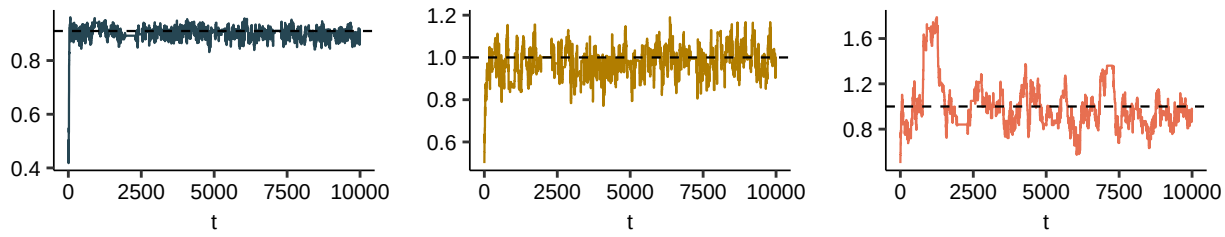


Figure 3: Trace plots for each model parameter (alpha/beta/sigma resp.) generated using pmmh2().

For α and σ, the traceplots in Figure 3 indicate that the chain has converged to its sationary regime. Another method of assessing convergence is through the ACF plots, which should also decay to 0. Figure 4 shows the ACF plots for each parameter after taking a burn in of 2500 iterations.

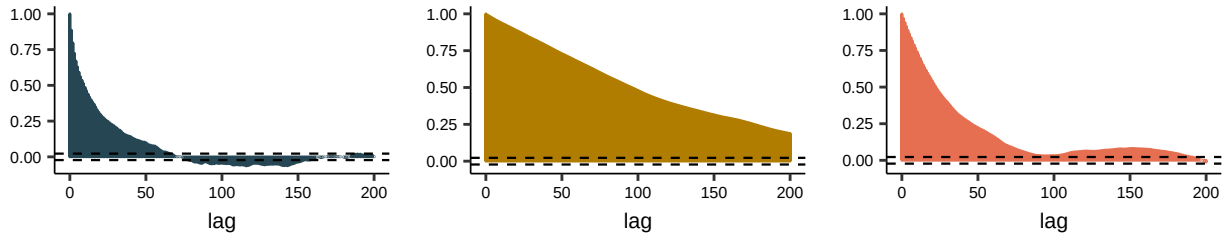


Figure 4: ACF plots for each model parameter (alpha/beta/sigma resp.) generated using pmmh2().

By lag 100, both α and σ's ACF appear to have decayed sufficiently. However, β's ACF plot suggestes that there is still a large level of correlation between values by lag 200. As a result, one might consider running the chain for longer. Finally, Figure 5 gives the histograms of the approximated posterior distributions, which are all roughly centred on the correct values.

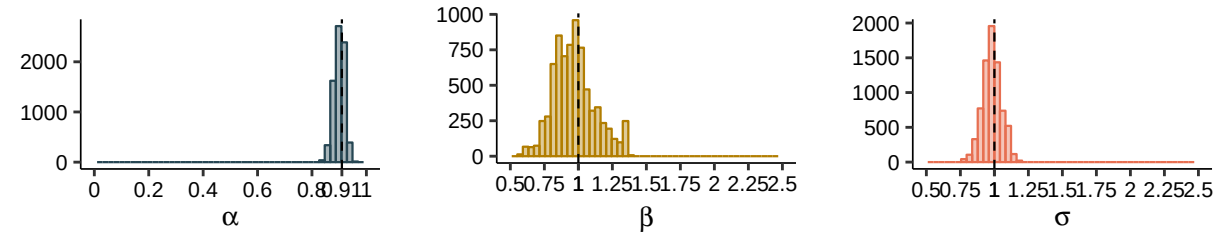


Figure 5: Sample histograms for each model parameter (alpha/beta/sigma resp.); generated using pmmh2().