Filtering and Inference for State-Space Modelling

Jimmy Lin

1 Introduction

1.1 State-Space Modelling

Within the field of time series analysis and applied probability, we may find that the variable we are observing is not directly dependent on its past value, but depends on some unobserved variable which evolves following a Markov process. This is commonly known as **state-space modelling** (SSM), or sometimes also termed a **hidden Markov model** (HMM). Since its introduction, this class of models has had a variety of applications in many fields, for example, in finance, social sciences, reinforcement learning (POMDP).

Let us first mathematically define the general homogeneous state-space model. Suppose we have a latent Markov processes defined by $\{X_t\}_{t\geq 0}$ such that,

$$X_0 \sim \mu(x_0) \text{ and } X_t|(X_{t-1} = x_{t-1}) \sim f(x_t|x_{t-1}), \tag{1.1}$$

where X is the latent variable, which is unobserved. Following this, we have a stochastic process for $\{Y_t\}_{t\geq 1}$ such that,

$$Y_t|(X_t = x_t) \sim g(y_t|x_t), \tag{1.2}$$

and Y is a the random variable which we do observe.

There exists a variety of problems in state-space modelling. We will first look at the problem known as *filtering*, where we find $p(x_t|y_{1:t})$, as new observations y_t come in, something that can be useful in an online setting. After this section, we will look more in-depth at parameter inference, where we now say the distribution within f and g has some θ parameter it is dependent on, assumed known in the filtering problem, and we want to identify the parameter value when they are unknown.

1.2 Examples and Applications

1.2.1 Dynamic Linear Model

The Gaussian distribution is a simple distribution that can always be applied due to its properties. We can define the distribution as follows,

$$\begin{aligned} X_0 &\sim \mathcal{N}(a_0, b_0), \\ X_t|(X_{t-1} = x_{t-1}) &\sim \mathcal{N}(\gamma x_{t-1}, \sigma^2), \\ Y_t|(X_t = x_t) &\sim \mathcal{N}(x_t, \tau^2). \end{aligned} \tag{1.3}$$

This problem is tractable and can be solved optimally using the Kalman filter. This makes the method an interesting case study for the sequential Monte Carlo methods. In practice, when working

with unknown model dynamics, statisticians typically model such problems initially as a dynamic linear model (DLM).

1.2.2 Stochastic Volatility

One way we can showcase the use of state-space models is through the financial use of stochastic volatility. In finance, the term volatility typically refers to how unpredictable a movement in stock is, or in statistics term the 'variance'. A standard approach in the Black-Scholes model is to assume constant variance, but we can easily see through the rapidly changing environment that this cannot be a reasonable assumption, for example a large volatility would be more likely during financial crashes or the pandemic. However, if we attempt to split up volatility into periods perhaps in a rolling window approach, we run into the issue of selecting optimal window length to reflect current volatility, but also having sufficient data such that the volatility is good measurement. One alternative is to use simulation based approaches and to model volatility as a state-space model.

First, we treat our Y as an observed variable and most commonly used in financial markets is log returns. Since we treat volatility as the variance, we can say that,

$$Y_t|(X_t = x_t) \sim \mathcal{N}(0, \exp(x_t)), \tag{1.4}$$

with the assumption that the mean of a log-return is 0. Now we can just treat X as a simple AR process such that,

$$X_t|(X_{t-1} = x_{t-1}) \sim \mathcal{N}(\gamma x_{t-1}, \sigma^2). \tag{1.5}$$

There are many possible further extensions to this which are useful within finance. Perhaps we would prefer to model returns with some non-zero mean to capture relationships with past returns or trends and seasonalities. We may also choose an alternative distributions to capture heavier tails or skewed distributions.

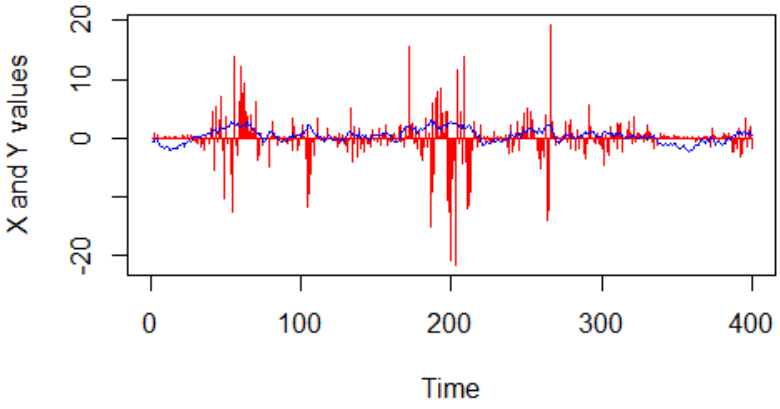

Figure 1: Stochastic Volatility Example: X (blue), Y (red).

2 Likelihood Functions

We will begin this chapter by diving investigating the theory behind filtering. Firstly from equation (1.1), we can derive the joint distribution of the latent variable $x_{0:t}$ such that,

$$p(x_{0:t}) = p(x_1) \prod_{k=2}^{t} f(x_k|x_{k-1}), \tag{2.1}$$

and the joint distribution of observations conditioned on the latent $y_{1:t}|x_{0:t}$ such that,

$$p(y_{1:t}|x_{0:t}) = \prod_{k=1}^{t} g(y_k|x_k). \tag{2.2}$$

Utilising the equations (2.1) and (2.2), We can employ Bayes theorem to derive the likelihood of the joint density functions between all observed and latent variables using Bayes theorem to obtain,

$$p(x_{0:t}, y_{1:t}) = \mu(x_0) \prod_{i=1}^{t} f(x_i|x_{i-1}) \prod_{j=1}^{t} g(y_j|x_j). \tag{2.3}$$

We can use this result to also derive the marginal joint density function of all the observation values which would be given by,

$$p(y_{1:t}) = \int \mu(x_0) \prod_{i=1}^{t} f(x_i|x_{i-1}) \prod_{j=1}^{t} g(y_j|x_j) \mathrm{d}x_{1:t}. \tag{2.4}$$

For a recursive definition, which is a key formulation in a sequential Monte Carlo, we should define a likelihood for a **prediction** step $x_t|y_{1:t-1}$, or what may be easier to begin with is the joint prediction likelihood $x_{0:t}|y_{1:t-1}$, before defining the filtering likelihood. This is key in the propagating in sequential Monte Carlo which we will see later.

$$p(x_{0:t}|y_{1:t-1}) = p(x_{0:t-1}|y_{1:t-1}) f(x_t|x_{t-1}), \tag{2.5}$$

and as a result, using the Bayes theorem we can derive the joint filtering likelihood distribution such that,

$$\begin{aligned} p(x_{0:t}|y_{1:t}) &= \frac{p(x_{0:t}, y_{1:t})}{p(y_{1:t})} \\ &= \frac{p(x_{0:t}, y_{1:t-1}) g(y_t|x_t)}{p(y_{1:t-1}) p(y_t|y_{1:t-1})} \\ &= p(x_{0:t}|y_{1:t-1}) \frac{g(y_t|x_t)}{p(y_t|y_{1:t-1})}, \end{aligned} \tag{2.6}$$

where we have that,

$$p(y_t|y_{1:t-1}) = \int g(y_t|x_t) p(x_t|y_{1:t-1}) \mathrm{d}x_t. \tag{2.7}$$

By integrating out the unnecessary variables from the joint filtering and predictive distribution, we can arrive at the marginal filtering and predictive likelihood function which are given by,

$$p(x_t|y_{1:t-1}) = \int p(x_{0:t-1}|y_{1:t-1}) f(x_t|x_{t-1}) \mathrm{d}x_{0:t-1}, \tag{2.8}$$

and,

$$p(x_t|y_{1:t}) = p(x_t|y_{1:t-1})\frac{g(y_t|x_t)}{p(y_t|y_{1:t-1})} = \frac{g(y_t|x_t)p(x_t|y_{1:t-1})}{\int g(y_t|x_t)p(x_t|y_{1:t-1})\mathrm{d}x_t}. \tag{2.9}$$

However, because of the denominator, it makes most situations of this problem intractable. Therefore, oftentimes, we will just denote this solely to a constant of proportionality, and use Monte Carlo methods to sample such a distribution. We will next discuss how to sample from a distribution known only up to a constant of proportionality.

3 Monte Carlo Methods

3.1 Monte Carlo Sampling

Before exploring the realm of sequential Monte Carlo, one must understand the basis of Monte Carlo simulation. Say we have $X_1, X_2, ..., X_n$ which are independent and identically distributed (i.i.d.), and we would like to estimate the properties of such a distribution, we can empirically estimate the distribution properties as follows,

$$\mathbb{E}[h(X)] \approx \frac{1}{n}\sum_{i=1}^{n} h(X_i), \tag{3.1}$$

which converges almost surely as $N \to \infty$, by the strong law of large numbers. Additionally, we can then estimate the density of X using Monte Carlo approximation by,

$$p(x) \approx \hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \delta_{X_i}(x), \tag{3.2}$$

where $\delta_X(x)$ is denoted by the Dirac delta function, where $\delta_{x_i}(x) = 1$ if $x = x_i$ and $\delta_{x_i}(x) = 0$ otherwise. This can be used to estimate properties of any density. However, sometimes sampling directly from p may not be feasible, or may be computationally expensive. We may in that case use what is known as importance sampling.

3.2 Importance Sampling

Importance sampling can be used for sampling a difficult $p(x)$ distribution by using an alternative distribution $q(x)$, which we call the proposal distribution, which typically easy to sample from. We can do this by,

$$\mathbb{E}_p[h(X)] = \int h(x)p(x)\mathrm{d}x = \int h(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x = \mathbb{E}_q\left[\frac{p(X)}{q(X)}h(X)\right]. \tag{3.3}$$

By applying the empirical approximation given in equation (3.1), we can approximate the unknown density, using the known density so that,

$$\mathbb{E}_p[h(X)] = \mathbb{E}_q\left[\frac{p(X)}{q(X)}h(X)\right] \approx \frac{1}{n}\sum_{i=1}^{n}\frac{p(X_i)}{q(X_i)}h(X_i). \tag{3.4}$$

Typically, we denote $w_i^{(*)} = p(x)/q(x)$ as the weighting given to each x value. Hence, the approximation can be re-written as,

$$\hat{\mathbb{E}}_p[h(X)] = \frac{1}{n}\sum_{i=1}^{n} w_i^* h(X_i), \tag{3.5}$$

and the associated empirical estimation of the density $p(x)$ as,

$$p(x) \approx \hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} w_i^* \delta_{X_i}(x). \tag{3.6}$$

A key component to consider within importance sampling is the choice of the proposal distribution $q(x)$, one may imagine that approximating from a proposal distribution that is fairly similar the distribution of interest $p(x)$ will be far more efficient than sampling from a distribution which is nothing alike compared to the distribution of interest. The most common metric to compare the efficiency of a proposal distribution is the **effective sample size** (Liu, 1996), which suggest that for most $h(X)$, it would mostly be dependent on the variance of the weights.

$$\text{ESS} = \frac{n}{1 + \text{Var}(w^*)} = \frac{(\sum_{i=1}^{n} w_i^*)^2}{\sum_{i=1}^{n} (w_i^*)^2}. \tag{3.7}$$

One may imagine a perfect proposal distribution is one with no variance of weights, hence having proposal distribution $q(x) = p(x)$ would be perfect. Of course this would just be the simple Monte Carlo sampling and we are interested in the cases where cannot sample from $p(x)$.

Suppose we are trying to sample from $X_1, ..., X_n \sim \mathcal{N}(0, 1)$, where we have $n = 1000$. Now consider we have two alternative proposal distributions $q_1(x)$ and $q_2(x)$. For the first, we will use the standard Cauchy distribution where $x_0 = 0$ and $\gamma = 1$, where x_0 defines the centre of the distribution as true mean is undefined, and γ defines the shape parameter. The second distribution is a Cauchy distribution with location $x_0 = 6$ and scale $\gamma = 1$. The results are given in figures 2 and 3 where we have the red line representing the target distribution, green line representing the proposal distribution, and the histogram representing the results of the weighted samples. As we can see, the results are fairly poor for the Cauchy distribution that is not centred, as it is fairly far off from the distribution of interest, hence the variance within the importance weights will be large leading to small ESS.

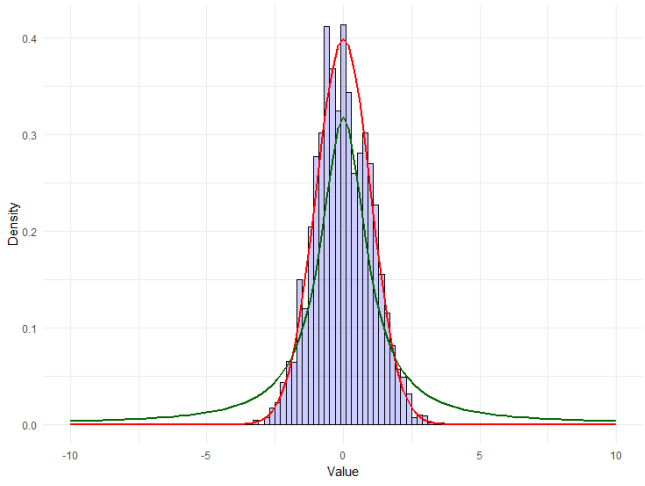

Figure 2: $q(x) \sim \text{Cauchy}(0, 1)$: ESS = 741.492

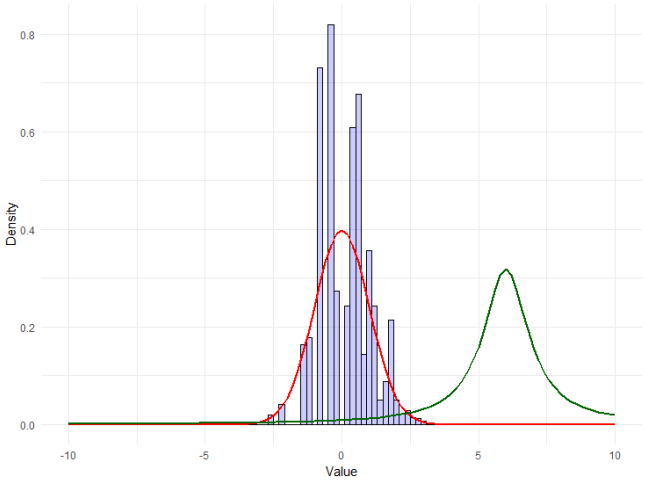

Figure 3: $q(x) \sim \text{Cauchy}(6, 1)$: ESS = 26.830

3.3 Self-Normalised Importance Sampling

Perhaps a more common issue which arises is the target density being unnormalised, something which occurs in Bayesian statistics often where we only know the posterior distribution $p(x|\theta)$ to a

constant of proportionality. Though often applied to posteriors, we will look at simply the general case of $p(x)$ known to a constant of proportionality. We can extend the base importance sampling formulation such that,

$$\mathbb{E}_p[h(X)] = \int h(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x = \frac{\int h(x)\frac{p(x)}{q(x)}q(x)\mathrm{d}x}{\int \frac{p(x)}{q(x)}q(x)\mathrm{d}x} \approx \frac{\sum_{i=1}^n h(x_i)\frac{p(x_i)}{q(x_i)}}{\sum_{i=1}^n \frac{p(x_i)}{q(x_i)}}. \tag{3.8}$$

With the substitution of $w^* = p(x)/q(x)$, we have that,

$$\hat{\mathbb{E}}_p[h(X)] = \frac{\sum_{i=1}^n h(x_i)w_i^*}{\sum_{i=1}^n w_i^*}. \tag{3.9}$$

This can be simplified further by normalising the weights, which we define as $w_i = w_i^*/\sum_{j=1}^n w_j^*$. The formula gets simplified to,

$$\hat{\mathbb{E}}_p[h(X)] = \sum_{i=1}^n h(x_i)w_i, \tag{3.10}$$

and we would have the density estimation given by,

$$p(x) \approx \hat{p}(x) = \sum_{i=1}^n w_i\delta_{X_i}(x). \tag{3.11}$$

This is actually very similar in the behaviour to a standard importance sampling, with the additional benefit of normalising the resulting posterior, making it a fundamental sampling method for Bayesian statistics. We can rearrange the ESS formulation into normalised weights such that,

$$\mathrm{ESS} = \frac{(\sum_{i=1}^n w_i^*)^2}{\sum_{i=1}^n (w_i^*)^2} = \frac{1}{\sum_{i=1}^n (w_i)^2}. \tag{3.12}$$

As this method does not particularly deal with problems regarding poor proposal distribution choices, therefore it would act similarly to standard importance sampling where a good selection for the proposal distribution is important.

4 Filtering with Sequential Monte Carlo

Since we have now defined standard Monte Carlo methods, we can return to our filtering problem. Consider the posterior equation $p(x_t|y_{1:t})$ in equation (2.9), we can simplify this now to a constant of proportionality such that,

$$p(x_t|y_{1:t}) \propto g(y_t|x_t)p(x_t|y_{1:t-1}). \tag{4.1}$$

We can then approximate this value using the self-normalised importance sample. Say we have n number of weighted samples $\{(x_t^{(i)}, w_t^{(i)})\}_{i=1}^n$, the sampling approximation is then given by,

$$p(x_t|y_{1:t}) \approx \hat{p}(x_t|y_{1:t}) = \sum_{i=1}^n w_t^{(i)}\delta_{x_t^{(i)}}(x_t). \tag{4.2}$$

Note that $w_t^{(i)}$ will denote the normalised weights such that $\sum_{i=1}^n w_t^{(i)} = 1$. For an updated notation as the old notation may cause clutter, we will denote $\tilde{w}_t^{(i)}$ if we need to denote an unnormalised weight. In the filtering problem, each sample is typically called a 'particle', due to the fact they move along in time,

and can disappear or duplicate, sometimes in geophysics they may call these ensembles (Fearnhead and Künsch, 2018).

4.1 Sequential Importance Sampling

First take a look at the prediction distribution, we can extend equation (4.2) by propagating our previous particles forward in time following the distribution given in equation (1.1). This means that assuming i.i.d. assumption in the propagation distribution, we have,

$$p(x_t|y_{1:t-1}) \approx \hat{p}(x_t|y_{1:t-1}) = \sum_{i=1}^{n} w_{t-1}^{(i)} f(x_t|x_{t-1}^{(i)}) = \sum_{i=1}^{n} w_{t-1}^{(i)} \delta_{x_t^{(i)}}(x_t). \tag{4.3}$$

After defining the predictive distribution, we can find the weights associated with individual particles, which is required for sampling the filtering distribution. Suppose we use the proposal distribution $q(x_t^{(i)}) = f(x_t^{(i)}|x_{t-1}^{(i)})$, then we have,

$$\tilde{w}_t^{(i)} = \frac{\tilde{p}(x_t^{(i)}|y_{1:t})}{q(x_t^{(i)})} \approx \frac{g(y_t|x_t^{(i)})\hat{p}(x_t^{(i)}|y_{1:t-1})}{f(x_t^{(i)}|x_{t-1}^{(i)})} = w_{t-1}^{(i)} g(y_t|x_t^{(i)}). \tag{4.4}$$

The next step is to simply normalise the updated weights given in equation (4.4) as required by the self-normalised importance sampling,

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{n} \tilde{w}_t^{(j)}}. \tag{4.5}$$

Of course, given this we still have not defined how we start the sampling algorithm. This can be done simply by taking n particle samples from the prior distribution on X_0, $x_0^{(i)} \sim \mu(X_0)$. When we sample directly from the target distribution, all importance weights are set equal, hence $w_0^{(i)} = 1/N, \forall i \in \{1, ..., n\}$. The pseudocode fully describing this algorithm is described in algorithm 1.

Algorithm 1 Sequential Importance Sampling

Require: : Number of particles n. Time length T. Observations $y_{1:T}$. Transition probability of latent variable $f(x_t|x_{t-1})$. Conditional probability of observation to latent variable $g(y_t|x_t)$. Initial distribution $\mu(X_0)$.

1: Draw prior samples $x_0^{(i)} \sim \mu(X_0)$. Assign weights such that $w_0^{(i)} = 1/n$ for $i = 1, ..., n$.
2: **for** $t = 1, ..., T$ **do**
3: (**Propagate**) Draw $x_t^{(i)} \sim f(x_t|x_{t-1}^{(i)})$.
4: (**Assign Weight**) Set $\tilde{w}_t^{(i)} = w_t^{(i)} g(y_t|x_t^{(i)})$ and normalise such that $w_t^{(i)} = \tilde{w}_t^{(i)} / \sum_{j=1}^{n} \tilde{w}_t^{(j)}$.
5: **end for**
6: **return** $\hat{p}(x_T|y_{1:T-1}) = \sum_{i=1}^{n} w_T^{(i)} \delta_{x_T^{(i)}}(x_T)$

4.2 Particle Degeneracy in Sequential Importance Sampling

In an earlier chapter on importance sampling, we talked about importance sampling and how it performed poorly with poor choices of proposal distributions, one of the metrics used to measure this was the effective sample size (ESS) which suggests that a poor performance in importance sampling is a result of large variances in the weights. Within sequential importance sampling, simulate multiple trajectories following the distribution describing evolution of the latent variable, but we never update

this based off the observation, and oftentimes, this can lead to large variances in trajectories leading to far larger variance in weights, hence, smaller ESS values.

First, we can look at an example following a DLM described earlier, in equation (1.3), but with defined parameters,

$$\begin{aligned} X_0 &\sim \mathcal{N}(0,1), \\ X_t|(X_{t-1}=x_{t-1}) &\sim \mathcal{N}(x_{t-1},1), \\ Y_t|(X_t=x_t) &\sim \mathcal{N}(x_t,1). \end{aligned} \tag{4.6}$$

By simulating this over a time period of 5, then simulating 10 particles, we get the result seen in figure 4 where we have denoted the dot size by the normalised weight, each line denotes a particle trajectory except the red dotted line which denotes the observed values. We can see from this figure that the ESS over time gradually decreases, even after the first time step, the ESS value halves. This is a phenomena known as **particle degeneracy**.

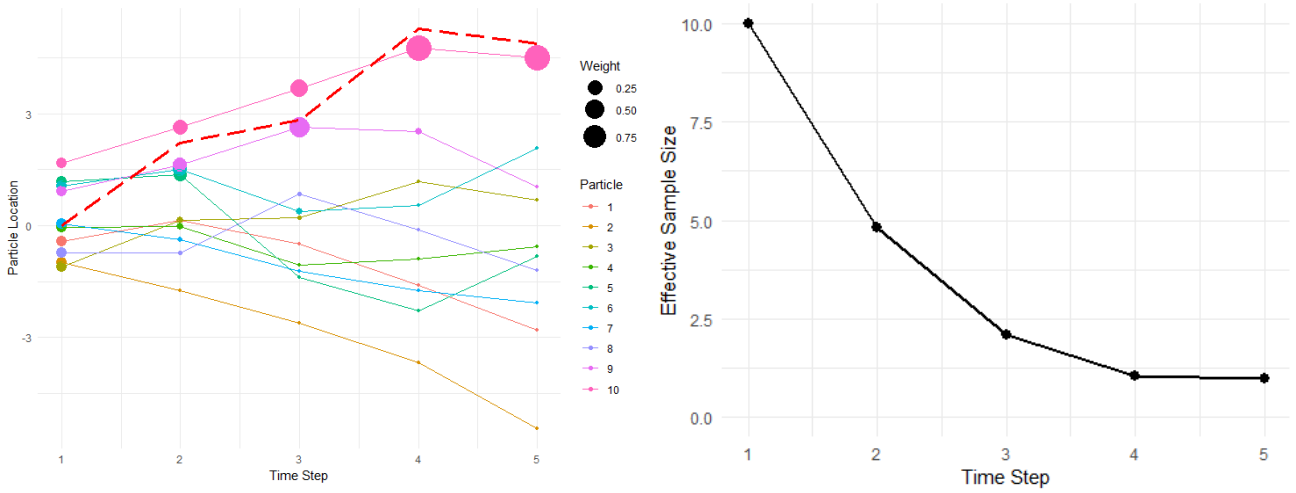

Figure 4: Sequential Importance Sampling for formulation given in equation (4.6) with (left) importance weights associated with each particle and (right) ESS values over time.

One of the major causes of this is that at each time step, we are not taking into consideration the values of y we have observed, which narrows down the probability for where the latent state may be. Hence the variance of the simulation will be large after just a few steps. One way to mitigate this problem is to consider resampling at each time step based on our observations y.

4.3 Bootstrap Filter

The bootstrap filter, sometimes known as the sequential importance resampling (SIR) method, is a term given to the method of resampling from a set of observations we receive with replication, 'bootstrapping' the particles at a particular observation time t. Here, what we would like to do is to mitigate the effects of particle degeneracy by bootstrapping at each time step with probability given by the importance weights of each particle. This at each point attempts recreate the probability $p(x_t|y_{1:t})$ and allows us to re-distribute the particle weights evenly to $1/n$. The intuition behind this is to remove particles that are less likely to follow the trajectory of the true latent variable movement given our knowledge of the observation, and create more particles at trajectories that are more likely to follow true latent variable trajectory.

4.3.1 Multinomial Resampling

There are many resampling methods, however what is often used in practice and the simplest is what is known as **multinomial resampling**. Say at the time step of interest t, we want to conduct the resampling, and we have the variables $\{(x_t^{(i)}, w_t^{(i)})\}_{i=1}^n$. When we resample, we simply draw

$$\{O^{(i)}\}_{i=1}^n \sim \text{Multinomial}\left(n, \{w_t^{(i)}\}_{i=1}^n\right), \tag{4.7}$$

where $\{O^{(i)}\}_{i=1}^n = (O^{(1)}, ..., O^{(n)})$ with $O^{(i)}$ denoting the number of 'offspring', or copies of particle $x_t^{(i)}$ are resampled.

If we look at the properties of the resample, we see that the expected number of copies regarding particle $x_t^{(i)}$ is $\mathbb{E}[O^{(i)}] = nw_t^{(i)}$, therefore unbiased. We can then substitute this into the approximation for the filtering equation,

$$p(x_t|y_{1:t}) \approx \hat{p}(x_t|y_{1:t}) = \frac{1}{n}\sum_{i=1}^{n} O^{(i)}\delta_{x_t^{(i)}}(x_t). \tag{4.8}$$

This is approximately equivalent to the pre-resampled distribution, and almost surely convergent as the number of particles tend to infinity according to the strong law of large numbers.

Constructing this algorithm is not trivial, and requires what is known as the **inverse CDF algorithm**. Most programming languages will have some form of this pre-built or as part of a package, but regardless I will still mention the details of this algorithm.

Within this algorithm, the first step is to create a empirical approximation to the CDF for the filter, which can be done by using the weights such that,

$$C^j = \sum_{i=1}^{j} w_t^{(i)}. \tag{4.9}$$

We can see that this is monotonically increasing CDF, since we have $C^0 = 0$ and due to normalisation $C^n = 1$.

The next step is to generate n uniform samples such that $U \sim \text{Uniform}(0, 1)$, and order it such that $U^{(1)} \leq U^{(2)} \leq \cdots \leq U^{(n)}$. We can then set the resampled set of particles such that,

$$\tilde{x}_t^{(i)} = x_t^{(j)}, \quad \text{s.t. } j = \min\{k : C_k \geq U^{(i)}\}. \tag{4.10}$$

The pseudocode for the inverse CDF algorithm for multinomial sampling is formalised in 2.

Algorithm 2 Multinomial Resampling

Require: : Set of weighted samples $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^n$.
1: Set $C^j = \sum_{i=1}^{j} w_t^{(i)}$.
2: Draw $U \sim \text{Uniform}(0, 1)$ and reorder such that $U^{(1)} \leq \cdots \leq U^{(n)}$.
3: **for** i = 1,...,n **do**
4: Set $j = \min\{k : C_k \geq U^{(i)}\}$.
5: Set $\tilde{x}_t^{(i)} = x_t^{(j)}$.
6: **end for**
7: **return** $\{\tilde{x}_t^{(i)}, \frac{1}{n}\}_{i=1}^n$.

4.3.2 Dangers of Resampling

What are the potential **drawbacks** of this method? Although theoretically, the unbiasedness and convergence property may imply that resampling may always be a good idea. However, when we resample, we are using a random distribution which can often add noise, especially when in practice we will be working with finite particle count. Suppose we have equivalent weighting such that $\{w_t^{(i)}\}_{i=1}^n = \{1/n\}_{i=1}^n$, and we were to resample. Since the marginal of a multinomial distribution is a binomial, if we were to look at particle i we have the distribution,

$$O^{(i)} \sim \text{Binomial}\left(n, \frac{1}{n}\right). \tag{4.11}$$

We can see that here, $\mathbb{P}\{O^{(i)} = 0\} = (1 - 1/n)^n \approx e^{-1} = 0.37$, meaning that there is a 0.37 chance that when we have equivalent weights where a particle is not resampled. This can be considered wasteful as we are potentially discarding particles which are worth equally as much as each other. A common heuristical approach to deal with this is a choice of ESS value where only when we reach below this would we choose to resample, hence **only resampling to deal with significant particle degeneracy**.

4.3.3 General Bootstrap Filter Algorithm

Say we set a threshold minimum value for $\text{ESS}_{\min}$ where below this value we resample. We can combine ideas in this chapter into sequential importance sampling, and out comes a bootstrap filter algorithm which can be seen in algorithm 3, which alters the sequential importance sampling by implementing a resampling algorithm.

Algorithm 3 General Bootstrap Filter

Require: : Number of particles n. Time length T. Observations $y_{1:T}$. Transition probability of latent variable $f(x_t|x_{t-1})$. Conditional probability of observation to latent variable $g(y_t|x_t)$. Initial distribution $\mu(X_0)$. Effective sample size threshold $\text{ESS}_{\min}$. Resampling Algorithm.

1: Draw prior samples $x_0^{(i)} \sim \mu(X_0)$. Assign weights such that $w_0^{(i)} = 1/n$ for $i = 1, ..., n$.
2: **for** $t = 1, ..., T$ **do**
3: (**Propagate**) Draw $x_t^{(i)} \sim f(x_t|x_{t-1}^{(i)})$.
4: (**Assign Weight**) Set $\tilde{w}_t^{(i)} = w_t^{(i)} g(y_t|x_t^{(i)})$ and normalise such that $w_t^{(i)} = \tilde{w}_t^{(i)} / \sum_{j=1}^n \tilde{w}_t^{(j)}$.
5: **if** $\left[\sum_{i=1}^n (w_t^{(i)})^2\right]^{-1} < \text{ESS}_{\min}$ **then**
6: (**Resample**) $\{\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}\}_{i=1}^n = \text{Resample}(\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^n)$.
7: **end if**
8: **end for**
9: **return** $\hat{p}(x_T|y_{1:T-1}) = \sum_{i=1}^n w_T^{(i)} \delta_{x_T^{(i)}}(x)$.

4.4 Alternative Resampling Methods

As mentioned previously, when our weighting is approximately equal, most of the time we may lose valuable particles that are unnecessary. Another way to mitigate this problem is with alternative resampling methods. Every sampling method will have to fulfill the condition of $\mathbb{E}[O^{(i)}] = nw_t^{(i)}$ to satisfy asymptotic convergence. Ideally, our alternative resampling can be considered an improvement if the variance of the resampling method is lower than one for multinomial resampling.

4.4.1 Residual Resampling

One alternative resampling method is known as residual resampling. Suppose we have normalised the weights $\sum_{i=1}^{n} w_t^{(i)} = 1$. The intuition behind residual sampling is to split $O^{(i)}$ into a deterministic section with pre-defined offsprings according to the filtering distribution, and a stochastic section which uses multinomial resampling. This can be thought of as a middle ground to no resampling and standard multinomial resampling.

Since the goal is to obtain $\mathbb{E}[O^{(i)}] = nw_t^{(i)}$, we can always obtain this without resampling by simply setting part of the offspring to $\lfloor nw_t^{(i)} \rfloor$. However, this leaves the remaining fractional component left to find as we cannot have a fraction of an offspring to a particle. Lets denote the remaining fractional component as $\varepsilon^{(i)} = nw_t^{(i)} - \lfloor nw_t^{(i)} \rfloor = \text{frac}(nw_t^{(i)})$. We can simply find $\varepsilon^{(i)}$ via multinomial resampling with probabilities $\varepsilon^{(i)} / \sum_{j=1}^{n} \epsilon^{(j)}$. By denoting the stochastic offsprings component with

$$\{O^{(i)}\}_{i=1}^{n} \sim \text{Multinomial}\left(n - \sum_{i=1}^{n} \lfloor nw_t^{(i)} \rfloor, \left\{ \frac{\varepsilon^{(i)}}{\sum_{j=1}^{n} \varepsilon^{(j)}} \right\}_{i=1}^{n} \right), \tag{4.12}$$

but actually it is trivial that $n - \sum_{i=1}^{n} \lfloor nw_t^{(i)} \rfloor = \sum_{j=1}^{n} \varepsilon^{(j)}$, hence we can see that $\mathbb{E}[\tilde{O}^{(i)}] = \varepsilon^{(i)} = \text{frac}(nw_t^{(i)})$ is satisfied. Hence when we combine the deterministic section and the stochastic section, this results in the final number of offsprings being determined by,

$$O^{(i)} = \lfloor nw_t^{(i)} \rfloor + \tilde{O}^{(i)}, \tag{4.13}$$

where we have the result of $\mathbb{E}[O^{(i)}] = nw_t^{(i)}$. This means that during the case of equivalent weight, we are guaranteed to keep all of the particles when weights are equal as $\sum_{i=1}^{n} \lfloor nw_t^{(i)} \rfloor = \sum_{i=1}^{n} 1 = n$, hence there will be no stochastic element in this scenario. This residual resampling is detailed in algorithm 4, but described in a manner to return $\{\tilde{x}_t^{(i)}, \frac{1}{n}\}_{i=1}^{n}$.

Algorithm 4 Residual Resampling

Require: : Set of weighted samples $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{n}$.

Set $k = 1$.
for $i = 1, ..., n$ **do**
  if $\lfloor nw_t^{(i)} \rfloor > 0$ **then**
    for $j = 1, ..., \lfloor nw_t^{(i)} \rfloor$ **do**
      Set $\{\tilde{x}_t^{(k)}, \frac{1}{n}\} = \{x_t^{(i)}, \frac{1}{n}\}$.
      Set $k \leftarrow k + 1$.
    end for
  end if
end for
Set $\tilde{\boldsymbol{\varepsilon}} = \text{frac}(n\boldsymbol{w}_t)$.
Normalise such that $\boldsymbol{\varepsilon} = \frac{\tilde{\boldsymbol{\varepsilon}}}{|\tilde{\boldsymbol{\varepsilon}}|}$.
Set $\{\tilde{x}_t^{(i)}, \frac{1}{n}\}_{i=\sum_{j=1}^{n} \lfloor nw_t^{(j)} \rfloor + 1}^{n} = \text{Multinomial}(x_t^{(i)}, \varepsilon^{(i)})$ and discard the rest.
return $\{\tilde{x}_t^{(i)}, \frac{1}{n}\}_{i=1}^{n}$.

4.4.2 Stratified Resampling

If we return to the standard multinomial sampling, we can reduce the variability by fixing U to an extent, as to evenly cover the $[0, 1]$ line as much as possible. The idea is similar to one within standard stratified sampling, where we split the $[0, 1]$ line evenly into n strata, and we can take one sample from each strata, which can reduce the potential variability in the samples as it ensures uniformity in the resamples. This is formalised as follows,

$$U^{(i)} \sim \text{Uniform}\left(\frac{i-1}{n}, \frac{i}{n}\right). \tag{4.14}$$

There are many benefits to this resampling method, firstly its reduction in complexity as its ordering of the uniform compared to multinomial resampling, but also due to enforcing the variability uniform distribution with equal length strata can guarantee no losses of particles under equivalent weights. Additionally, it also satisfies $\mathbb{E}[O^{(i)}] = nw_t^{(i)}$. One may see how this may occur just by comparing to the simple multinomial framework. Algorithm 5 details how the resampling may work.

Algorithm 5 Stratified Resampling

Require: : Set of weighted samples $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^n$.
1: Set $C^j = \sum_{i=1}^n w_t^{(i)}$.
2: **for** $k = 1, ..., n$ **do**
3: Draw $U^{(k)} \sim$ Uniform $\left(\frac{k-1}{n}, \frac{k}{n}\right)$.
4: **end for**
5: **for** $i = 1, ..., n$ **do**
6: Set $j = \min\{k : C_k \geq U^{(i)}\}$.
7: Set $\tilde{x}_t^{(i)} = x_t^{(j)}$.
8: **end for**
9: **return** $\{\tilde{x}_t^{(i)}, \frac{1}{n}\}_{i=1}^n$.

4.4.3 Systematic Resampling

Systematic resampling works very similarly to stratified resampling. However, instead of sampling uniformly in each strata, the systematic resampling only conducts one sample in an interval, and uses the equivalent point in each strata, hence further reducing stochasticity in the algorithm. We can notate this as,

$$U^{(i)} = \frac{(i-1+U)}{n}, \quad U \sim \text{Uniform}(0, 1). \tag{4.15}$$

There is not a lot of difference in terms of performance compared to residual resampling, however this is often preferred due to its simplicity. Algorithm 6 describes this, but one may be able to see it is similar to algorithm 5 from stratified sampling, but just this time we only simulate U once over $[0, 1]$.

4.4.4 Empirical Resampling Experiment

Our experiment involves applying the resampling methodologies to two different dynamic linear models, both of which have the form,

$$X_t|(X_{t-1} = x_{t-1}) \sim \mathcal{N}(x_{t-1}, 2^2) \quad \text{and} \quad Y_t|(X_t = x_t) \sim \mathcal{N}(x_t, \tau^2). \tag{4.16}$$

Algorithm 6 Systematic Resampling

Require: : Set of weighted samples $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^n$.

Set $C^j = \sum_{i=1}^n w_t^{(i)}$.
Draw $U \sim \text{Uniform}(0, 1)$.
for $k = 1, ..., n$ **do**
    Set $U^{(k)} = \frac{(k-1+U)}{n}$.
end for
for $i = 1, ..., n$ **do**
    Set $j = \min\{k : C_k \geq U^{(i)}\}$.
    Set $\tilde{x}_t^{(i)} = x_t^{(j)}$.
end for
return $\{\tilde{x}_t^{(i)}, \frac{1}{n}\}_{i=1}^n$.

The difference will be in the τ parameter where experiment 1 will set $\tau = 10$ and experiment 2 will set $\tau = 0.5$. We want to see how well the resampling methods compare when observations are more closely related to the latent process. We will simulate 20 different state-space models, and use particle filters without resampling and with resampling on the same 20 models to compare. Figure 5 and 6 shows us the results of this simulation experiment.

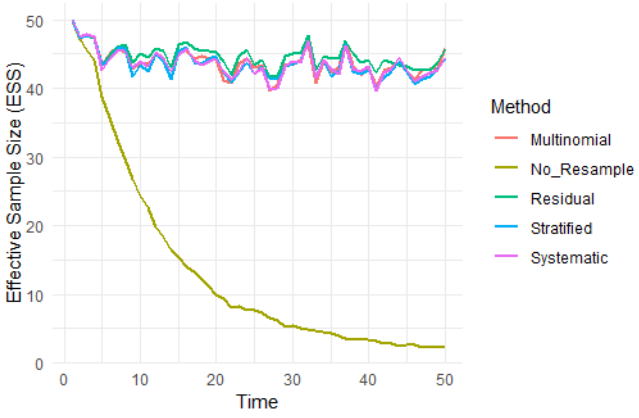

Figure 5: $\tau = 10$

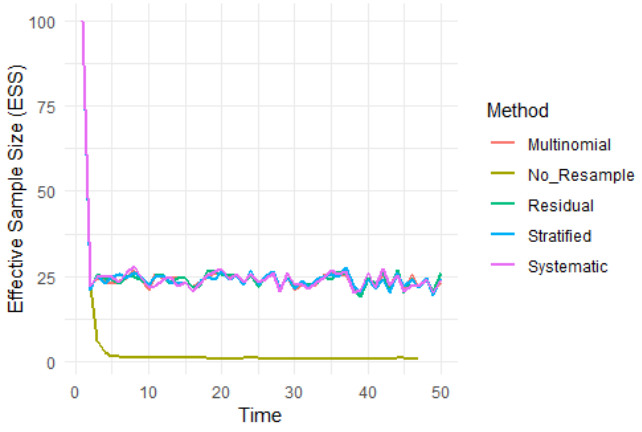

Figure 6: $\tau = 0.5$

In figure 5, we can see that resampling is often a much better strategy that without resampling, we do see also residual resampling typically performing better than most other resampling problems, likely due to its deterministic property. However, in practice, it is often recommended to use systematic sampling due to its simplicity (Chopin and Papaspiliopoulos, 2020) and to avoid multinomial resampling methods for the problems stated previously.

In figure 6 however, we see that all resampling methods are performing approximately equally as poorly. This is due to the stronger dependencies of observed variable but a variable propagation function, hence particle degeneracy is significant even in one step thus hard to save with resampling. One possible solution proposed the use of auxiliary particle filters (Pitt and Shephard, 1999), which uses draws from a distribution similar to the posterior, using knowledge of y_t to help guide the propagation of the latent variable. We will not go into detail in this report on this method in this report.

5 Parameter Inference with Particle MCMC

5.1 The Parameter Inference Problem

One key problem is to consider the parameters within state-space modelling. Prior to this we assume that for some parameter θ is known for our propagation or dependencies of latent to observation, but this may not always be the case. We can redefine equation (1.1) and (1.2) to include the parameter such that,

$$X_t|(X_{t-1} = x_{t-1}) \sim f(x_t|x_{t-1}, \theta) \quad \text{and} \quad Y_t|(X_t = x_t) \sim g(y_t|x_t, \theta). \tag{5.1}$$

We will describe a method known as Particle Markov Chain Monte Carlo (PMCMC), which is often used for parameter estimations within state-space modelling. First, we will begin by defining what is Markov Chain Monte Carlo (MCMC), and follow this by describing how it can extend to state-space modelling.

5.2 Markov Chain Monte Carlo

Standard MCMC is typically used within Bayesian statistics in a similar light to importance sampling for more complex distributions. Suppose we have a posterior distribution $\pi(x)$ which we know to a constant of proportionality, the idea behind MCMC is to create a Markov chain over the real line, which we traverse through our proposal density $q(y|x)$, where y is the new generated point which we propose to travel to. We then accept or reject this using the acceptance ratio seen in equation (5.2), which is the key step for the true stationary distribution. This is known as the **Metropolis-Hastings algorithm**, detailed in algorithm 7.

$$\alpha(x, y) = 1 \wedge \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}, \tag{5.2}$$

where $a \wedge b = \min\{a, b\}$.

Algorithm 7 General Metropolis-Hastings Algorithm

Require: : Target distribution π. Proposal distribution q. Starting point x_0. Termination period n. (Optional: Burn in period b).

1: **for** $i = 1, ..., n$ **do**
2: Set $x \leftarrow x_{i-1}$
3: Generate $x' \sim q(x'|x)$.
4: Calculate the acceptance probability

$$\alpha(x, x') = 1 \wedge \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)}.$$

5: Generate $U \sim \text{Uniform}(0, 1)$.
6: **if** $U \leq \alpha(x, x')$ **then**
7: (**Accept**) Set $x_i \leftarrow x'$.
8: **else**
9: (**Reject**) Set $x_i \leftarrow x$.
10: **end if**
11: **end for**
12: **return** $\{x_i : i \geq b\}$.

A simple way we may set up our algorithm is with a random walk proposal distribution where

$q(y|x) \sim \mathcal{N}(x, h)$ where h can be a tuning parameter. Using such a proposal distribution known as the **random walk Metropolis-Hastings**, often used for its simplicity as $q(x|y) = q(y|x)$ with such a proposal. Of course, it is not perfect and it does not perform under certain situations, but we can prove asymptotic properties regarding Metropolis-Hastings algorithm and the fact that it converges in stationary distribution to the true distribution.

To prove stationary distribution follows the distribution π, it is sufficient to prove the **detailed balance** condition is satisfied,

$$\pi(x)K(x, y) = \pi(y)K(y, x), \tag{5.3}$$

where K is the transition density of the Markov chain. When detail balance is satisfied, the Markov chain is considered π-reversible, meaning the dynamics forward in time is the same backwards in time. This also guarantees stationary distribution of π, meaning that the Markov chain is also π-invariant, which can be seen since,

$$\int \pi(x)K(x, y)\mathrm{d}x = \int \pi(y)K(y, x)\mathrm{d}y = \pi(y) \int K(y, x)\mathrm{d}y = \pi(y). \tag{5.4}$$

We can prove that the Metropolis-Hastings algorithm satisfies the detailed balance condition as follows,

$$\begin{aligned} \pi(x)K(x, y) &= \pi(x)q(y|x)\alpha(x, y), \\ &= \pi(x)q(y|x)\left\{1 \wedge \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}\right\}, \\ &= \pi(x)q(y|x) \wedge \pi(y)q(x|y), \end{aligned} \tag{5.5}$$

and because we can similarly that $\pi(y)K(y, x) = \pi(y)q(x|y) \wedge \pi(x)q(y|x)$, and trivially $a \wedge b = b \wedge a$. Now we can see that the detailed balance condition is satisfied.

5.3 Pseudo-Marginal Metropolis-Hastings

When we think to our particle filtering problem, our interest lies in estimating the density of parameter $\pi(\theta) \propto \pi_0(\theta)p(y_{1:T}|\theta)$. In this case, we take T to be the time horizon of our data, and assume this as a method used in an offline setting. We cannot directly estimate since we can not directly evaluate the true value for $p(y_{1:T}|\theta)$, but only an estimate. The **pseudo-marginal Metropolis-Hastings** algorithm (Andrieu and Roberts, 2009) can be used for inference with density estimator, as long as $\hat{\pi}(\theta)$ is an unbiased estimator, this typically means that $\hat{p}(y_{1:T}|\theta)$ should also be unbiased. We can define,

$$\hat{\pi}(\theta; U) = \pi_0(\theta)\hat{p}(y_{1:T}|\theta; U), \tag{5.6}$$

where U denotes an auxiliary random variable from some distribution $q_*(u|\theta)$. When we are finding this estimate, we take u to be the particle filter simulated with parameter θ. To find the unbiased estimator, first note the quantity $p(y_{1:T}|\theta)$ can be factorised such that,

$$p(y_{1:T}|\theta) = \prod_{t=1}^{T} p(y_t|y_{1:t-1}, \theta) = \int g(y_t|x_t, \theta)p(x_t|y_{1:t-1}, \theta)\mathrm{d}x_t. \tag{5.7}$$

It is now clear that we have the empirical Monte Carlo estimation,

$$\hat{p}(y_t|y_{1:t-1},\theta) = \frac{1}{n}\sum_{i=1}^{n} g(y_t|x_t^{(i)}) = \frac{1}{n}\sum_{i=1}^{n} \tilde{w}_t^{(i)}. \tag{5.8}$$

Hence we can estimate $p(y_{1:T}|\theta)$ such that,

$$\hat{p}(y_{1:T}|\theta) = \prod_{t=1}^{T}\left(\frac{1}{n}\sum_{i=1}^{n}\tilde{w}_t^{(i)}\right). \tag{5.9}$$

This is trivially unbiased proven similarly to proofs in the importance sampling section. Algorithm 8 details how the pseudo-marginal Metropolis-Hasting algorithm work.

Algorithm 8 Pseudo-Marginal Metropolis-Hastings

Require: : Target distribution unbiased estimator $\hat{\pi}$. Proposal distribution q. Auxiliary proposal q_*. Starting point θ_0. Termination period m. (Optional: Burn in period b).

1: Generate $u_0 \sim q_*(u_0|\theta)$
2: **for** $i = 1, ..., m$ **do**
3: Set $\theta \leftarrow \theta_{i-1}$ and $u \leftarrow u_{i-1}$.
4: Generate $\theta' \sim q(\theta'|\theta)$.
5: Generate $u' \sim q_*(u'|\theta')$
6: Compute $\hat{\pi}(\theta'; u')$
7: Calculate the acceptance probability

$$\alpha(\theta, u; \theta', u') = 1 \wedge \frac{\hat{\pi}(\theta'; u')q(\theta|\theta')}{\hat{\pi}(\theta; u)q(\theta'|\theta)}.$$

8: Generate $U \sim \text{Uniform}(0, 1)$.
9: **if** $U \leq \alpha(\theta, u; \theta', u')$ **then**
10: (**Accept**) Set $\theta_i \leftarrow \theta'$ and $u_i \leftarrow u'$.
11: **else**
12: (**Reject**) Set $\theta_i \leftarrow \theta$ and $u_i \leftarrow u$.
13: **end if**
14: **end for**
15: **return** $\{\theta_i : i \geq b\}$.

How do we know that this algorithm works? First we can check that the algorithm targets the correct distribution. First, denote the target distribution as the joint distribution of θ and u such that $\tilde{\pi}(\theta, u) \propto q_*(u|\theta)\hat{\pi}(\theta; u)$

$$\int \tilde{\pi}(\theta, u)\mathrm{d}u = \int q_*(u|\theta)\hat{\pi}(\theta; u)\mathrm{d}u = \mathbb{E}_{q_*}[\hat{\pi}(\theta; u)] \propto \pi(\theta), \tag{5.10}$$

and this works due to the unbiased estimator within $\hat{\pi}(\theta; u)$. With respects to our target distribution $\tilde{\pi}(\theta, u)$, we can see that it also satisfies the conditions required for detailed balance in equation (5.11).

$$\begin{aligned}
\tilde{\pi}(\theta, u)q(\theta'|\theta)q_*(u'|\theta')\alpha(\theta, u; \theta', u') &= q_*(u|\theta)\hat{\pi}(\theta; u)q(\theta'|\theta)q_*(u'|\theta')\left\{1 \wedge \frac{\hat{\pi}(\theta'; u')q(\theta|\theta')}{\hat{\pi}(\theta; u)q(\theta'|\theta)}\right\}, \\
&= q_*(u|\theta)q_*(u'|\theta')\left\{\hat{\pi}(\theta; u)q(\theta'|\theta) \wedge \hat{\pi}(\theta'; u')q(\theta|\theta')\right\}, \\
&= \tilde{\pi}(\theta', u')q_*(\theta|\theta')q_*(u|\theta)\alpha(\theta', u'; \theta, u).
\end{aligned} \tag{5.11}$$

Despite this, it is important to note that the acceptance rate of the pseudo-marginal algorithm is always

lower than to use Metropolis-Hastings with the true likelihood (Andrieu and Vihola, 2015).

5.4 Particle Marginal Metropolis-Hastings

Now we will specify the application of pseudo-marginal Metropolis-Hastings framework towards particle filters. Something often of interest within state-space modelling is the combination of inference for $p(\theta, x_{1:T}|y_{1:T})$. However, this is of course not possible to calculate in most scenarios hence the need for MCMC algorithms to sample from it. First we require an introduction to **backwards path**, and its proposal probability.

Suppose at time t, we have a weighted, pre-resampled k-th particle $x_t^{(k)}$. This particle arose from propagating the unweighted post-resampled particle from time $t-1$, $\tilde{x}_{t-1}^{(k)}$, and pre-resampling, this was one of $x_{t-1}^{(1)}, ..., x_{t-1}^{(n)}$, perhaps we can call it $x_{t-1}^{(b_{t-1}^k)}$. Similarly, this particle would have arisen from $\tilde{x}_{t-2}^{(b_{t-1}^k)}$. Here, we can start tracing back what is called the '**ancestry**' of each particle, where each particle came from. We can then denote the path back from k as $x_{1:t}^{(b_{1:t}^k)} := (x_1^{(b_1^k)}, ..., x_t^{(b_t^k)})$. We can then set the probability of sampling the path back from k to be,

$$q(k|u,\theta) \equiv q(x_{1:t}^{(b_{1:t}^k)}|u,\theta) = \frac{w_t^{(k)}}{\sum_{j=1}^n w_t^{(j)}} \tag{5.12}$$

Hence we can denote the particle marginal Metropolis-Hastings algorithm as follows in algorithm 9.

Algorithm 9 Particle Marginal Metropolis-Hastings

Require: : Target distribution unbiased estimator $\hat{\pi}$. Proposal distribution q. Auxiliary proposal q_*. Starting point θ_0. Termination period m. (Optional: Burn in period b).

1: Generate $u_0 \sim q_*(u_0|\theta)$
2: Generate $k_0 \sim q(k_0|u_0, \theta_0)$, and then set $x_{0,1:T} \leftarrow x_{1:T}^{(b_{1:T}^{k_0})}$.
3: **for** $i = 1, ..., m$ **do**
4: Set $\theta \leftarrow \theta_{i-1}$, $u \leftarrow u_{i-1}$, and $x_{1:T} \leftarrow x_{i-1,1:T}$.
5: Generate $\theta' \sim q(\theta'|\theta)$.
6: Generate $u' \sim q_*(u'|\theta')$
7: Generate $k' \sim q(k'|u', \theta')$ and set $x'_{1:t} \leftarrow x_{1:t}^{(b_{1:t}^{k'})}$.
8: Compute $\hat{\pi}(\theta'; u')$
9: Calculate the acceptance probability

$$\alpha(\theta, u; \theta', u') = 1 \wedge \frac{\hat{\pi}(\theta'; u')q(\theta|\theta')}{\hat{\pi}(\theta; u)q(\theta'|\theta)}.$$

10: Generate $U \sim \text{Uniform}(0, 1)$.
11: **if** $U \leq \alpha(\theta, u; \theta', u')$ **then**
12: (**Accept**) Set $\theta_i \leftarrow \theta'$, $u_i \leftarrow u'$ and $x_{i,1:T} \leftarrow x'_{1:T}$.
13: **else**
14: (**Reject**) Set $\theta_i \leftarrow \theta$, $u_i \leftarrow u$ and $x_{i,1:T} \leftarrow x_{1:T}$.
15: **end if**
16: **end for**
17: **return** $\{\theta_i, x_{i,1:T} : i \geq b\}$.

The algorithm very similar to the pseudo-marginal algorithm, except with an additional step for sampling and updating the paths. This algorithm however also requires minimal particle degeneracy, as one may see that a backward path can lead back to a small number of particles in the presence of

major particle degeneracy.

The element of the algorithm which brings it all together is the auxiliary variable of $u = (x_{1:T}^{1:n}, a_{1:T-1}^{1:n})$ where $x_{1:T}^{1:n}$ is the set of all weighted particles, and $a_{1:T-1}^{1:n}$ are the set of all ancestor indices. These are things we can find through the simulation of particle filters. Each time we are sampling this conditioned on a θ parameter. This algorithm targets our required joint posterior of

$$\tilde{\pi}(\theta, u, k) \propto \pi_0(\theta)\hat{p}(y_{1:T}|\theta; u)q_*(u|\theta)q(k|u, \theta), \tag{5.13}$$

which we can see as an extension of equation (5.6). This equation can be seen to satisfy detailed balance as follows,

$$\tilde{\pi}(\theta, u, k)q(\theta'|\theta)q_*(u'|\theta')q(k'|u', \theta')\alpha(\theta, u; \theta', u') = \tilde{\pi}(\theta', u', k')q(\theta|\theta')q_*(u|\theta)q(k|u, \theta)\alpha(\theta', u'; \theta, u). \tag{5.14}$$

5.5 Empirical Experiment with Stochastic Volatility

In this part we will look at an empirical experiment using the particle marginal Metropolis-Hastings algorithm, using the random walk proposal $\theta' \sim q(\theta'|\theta) = \mathcal{N}(\theta, h\boldsymbol{I})$. We will apply this to the stochastic volatility problem mentioned in the earlier section where we have log-returns X and stochastic volatility Y through a simulated dataset where,

$$\begin{aligned} X_0 &\sim \mathcal{N}(0, \sigma^2), \\ X_t|(X_{t-1} = x_{t-1}) &\sim \mathcal{N}(\gamma x_{t-1}, \sigma^2), \\ Y_t|(X_t = x_t) &\sim \mathcal{N}(0, \exp(x_t)). \end{aligned} \tag{5.15}$$

Here, we can define the parameters required as $\theta = \{\gamma, \log(\sigma)\}$. For our experiment, we will set true $\theta = \{\gamma, \log(\sigma)\} = \{0.95, -1\}$ with initial conditions $\theta_0 = \{0.5, 0\}$ and $h = 0.005$. For each particle filter, we use $n = 500$ particles for a good but not excessive approximation of likelihood (Sherlock et al., 2015). For a good result for our visualisations we have used termination period $m = 10,000$. However, due to its complexity, we found that it can suffice with far fewer iterations for finding likely parameters.

The first step is to create a trace plot so that we can find a good burn in parameter b. Figure 7 and 8 shows the trace plots only up to the first 500 iterations, and just by looking at these figures, we can approximately set a $b = 100$.

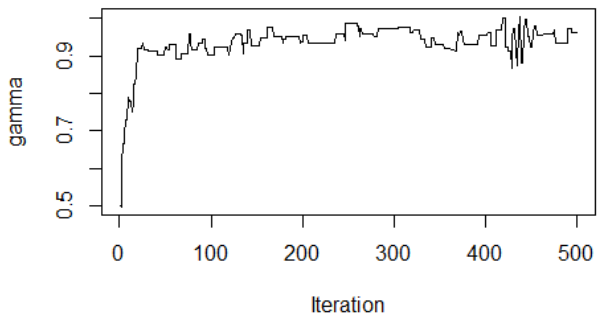

Figure 7: Trace plot for γ

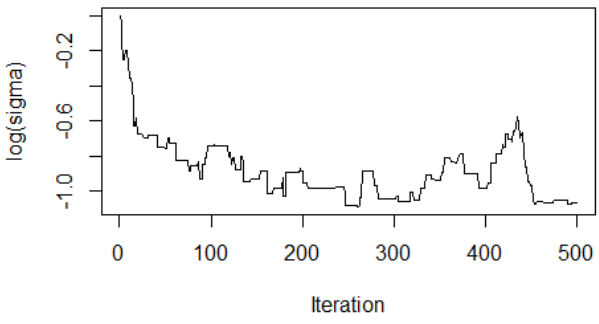

Figure 8: Trace plot for $\log(\sigma)$

Now this main step is complete, we can set up the output plots. In figure 9 we find the joint posterior distribution for θ, where we see the red lines denoting the true values of θ which shows well that the centre of the plot is approximately the true parameter value. Figure 10 shows the trajectory output with mean in blue line and 95% credible interval in the shaded regions. We see that the true latent x almost never leaves this credible region.

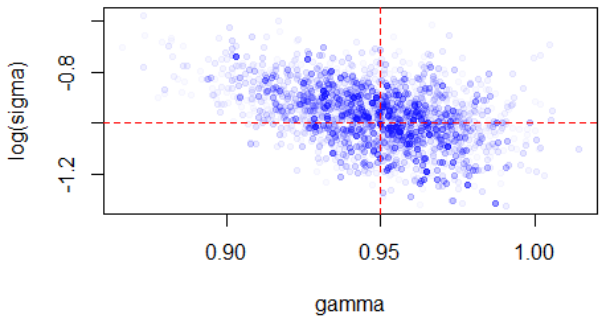

Figure 9: Joint Distribution of θ.

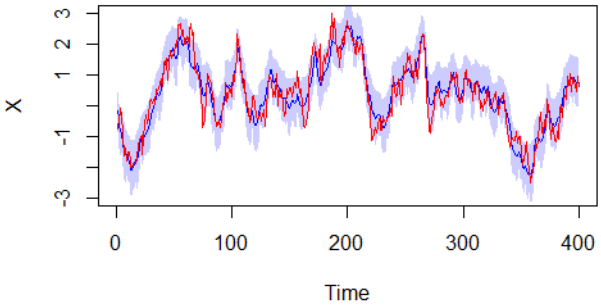

Figure 10: $\hat{x}_{1:T}$ (blue) and true $x_{1:T}$ (red).

6 Conclusion and Further Discussion

Our report covers two of the main problems within state-space modelling, filtering and parameter inference. We chose to approach both these problems with Monte Carlo methods so that we are able to deal with problems of intractability, where the relationships in the state-space models are not necessarily linear, Gaussian or stationary. One problem we did not explore in depth is *smoothing*, where we find $p(x_{1:T}|y_{1:T})$ looking at the full latent trajectory given all the observations. Here, filtering approaches are not preferred as we can infer latent variables from future observations as well as current observations. PMMH however can do smoothing well, but in simpler models we may prefer something that is quicker.

When we approached the filtering problem $p(x_t|y_{1:t})$ using bootstrap filter methods, which far outperformed sequential importance sampling by reducing particle degeneracy. However, the performance of both is equally poor when the observations is more heavily dependent on the latent state. One method one can explore is the auxiliary particle filters (Pitt and Shephard, 1999) which alters the proposal distribution q to one which is close to the posterior, which involves the new observations. This can guide the directions of the particles when propagating.

The problem within parameter inference, finding θ, we approached with particle marginal Metropolis Hastings, which allows us to use the MCMC methodologies to sample from the posterior $p(\theta, x_{1:T}|y_{1:T})$, despite our $\pi(\theta)$ being unbiased estimators $\hat{\pi}(\theta)$. The issue with this method is its large amount of tunable variables, which include particles, number of iterations, tunable h and proposal jump distribution for q. We tested this method on the stochastic volatility example and performed very strongly with a random walk proposal, however it may be of interest if this proposal perform similarly for varying non-Gaussian distributions. This method is also very computationally heavy, and oftentimes, using a particle variation of Gibbs sampler (Andrieu and Roberts, 2009) could be a solid alternative approach.

References

Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *Annals of Statistics*, 37(2):697–725.

Andrieu, C. and Vihola, M. (2015). Convergence properties of pseudo-marginal Markov chain Monte Carlo algorithms. *Annals of Applied Probability*, 25(2):1030–1077.

Chopin, N. and Papaspiliopoulos, O. (2020). *An introduction to sequential Monte Carlo*, volume 4. Springer.

Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3.

Fearnhead, P. and Künsch, H. R. (2018). Particle filters and data assimilation. *Annual Review of Statistics and Its Application*, 5(1):421–449.

Fearnhead, P. and Sherlock, C. (2025). MCMC for State Space Models. In *Handbook of Markov Chain Monte Carlo Volume 2*.

Liu, J. S. (1996). Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and computing*, 6:113–119.

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

Sherlock, C., Thiery, A. H., Roberts, G. O., and Rosenthal, J. S. (2015). On the efficiency of pseudo-marginal random walk Metropolis algorithms. *Annals of Statistics*, 43(1):238–275.