High-dimensional particle filters

Author: Srshti Putcha *Supervised by:* Dr. Chris Nemeth

1 Introduction

1.1 Motivating state space modelling

A time series is an ordered, sequence of data points (typically) measured at equally spaced time intervals. This type of data can be collected in many different disciplines, ranging from finance to meteorology. To model this data, we can use a stochastic process, described by the parameter vector, $\boldsymbol{\theta}$. More formally, time series models seek to explain the behaviour of a set of random variables, $\{X_t\}_{t\geq 1}$, with observed data points, $\{x_t\}_{t\geq 1}$.

In many applications, it is not possible to capture information about the true process, $\{X_t\}_{t\geq 1}$. Instead, we are only able to learn about the *hidden* or *latent* process via a second signal, $\{Y_t\}_{t\geq 1}$. A *state space model* (alternatively known as a *hidden Markov model*) seeks to specify the joint distribution of the latent variables, based on existing subject knowledge and the observed information (Fearnhead and Künsch, 2018). Figure 1.1 provides a graphical representation of a state-space model.

$$\begin{tikzcd}
\cdots \arrow[r] & x_{t-1} \arrow[r] \arrow[d] & x_t \arrow[r] \arrow[d] & x_{t+1} \arrow[r] \arrow[d] & \cdots \\
 & y_{t-1} & y_t & y_{t+1} &
\end{tikzcd}$$

Figure 1.1: A graphical representation of a *hidden Markov model*

The observations, $\{y_t\}_{t\geq 1}$, are assumed to be independent of each other, conditional on the latent process. If we further assume a Markovian structure, then X_t is only dependent on X_{t-1} at time t. This ensures that the history of the stochastic process before the previous time point is irrelevant.

Application 1: target tracking

There is a history of motivating state space modelling using target tracking problems. For instance, Gordon et al. (1993) simulate a *bearings-only* tracking scenario. In this context, the hidden process is the position and velocity of a tracked object, over time. The observed process is the bearing (i.e. angle) of the object with respect to the sensor. In practice, inference methods can be used to estimate the current position of a target and predict its future trajectory. This type of analysis necessitates the use of online, iterative algorithms, such as *particle filters*.

Application 2: systems biology

There are also many examples of state space models being used in systems biology. Previously, much of the mathematical modelling in systems biology was deterministic, using ODEs. It is now widely accepted that stochastic models are incredibly useful for understanding biochemical network dynamics at the single-cell level.

The Lotka-Volterra (predator-prey) reaction network model is often used to model the biochemical reactions taking place in a cell. Bayesian inference can be used to tune parameters quickly, so that the model output better emulates experimental data. Particle filtering algorithms can then be implemented to predict the true reaction system, given imprecise information.

Often, the true parameters of the Lotka-Volterra model are unknown and must be estimated from the observed data. Golightly and Wilkinson (2011) successfully use *particle-MCMC* techniques to conduct inference on reaction networks.

1.2 Outline of report

When using state space models, one of the main issues is being able to estimate the hidden state, x_t, at time t, given all observed information, $\{y_1, ..., y_t\}$. In the literature, this problem is known as *filtering*. We can only perform *exact filtering* for simple state space models. For instance, the *Kalman filter* (Kalman, 1960) can be used to optimally solve a linear Gaussian model. Certain variants, such as the *extended Kalman filter* (e.g. Welch and Bishop, 2006), can be used for nonlinear, non-Gaussian problems. However, for highly nonlinear and non-Gaussian problems, these methods cannot optimally recover the latent process.

An alternative approach is to use *particle filters*. While Kalman filters and their variants produce strictly deterministic approximations, particle filters are designed to produce stochastic approximations based on Monte Carlo sampling. As information becomes available over time, particle filters can be used to approximate the underlying latent process. There are several journal papers and books that are devoted to particle filtering and its applications, such as Fearnhead and Künsch (2018) and Doucet and Johansen (2009).

The aim of this report is to explore modelling approaches for high-dimensional particle filters. The remainder of the introduction is devoted to describing Monte Carlo methods for inference. In Section 2, we introduce particle filters. In Section 3, we discuss the challenges presented by high-dimensional particle filters and review some of the methods that have been developed to tackle the problem.

1.3 Monte Carlo methods

Suppose that we are interested in integrating a measurable function, $\psi : \mathcal{X} \to \mathbb{R}^d$, with respect to a well-defined probability density function, f, over some measurable space, $\mathcal{X}$. We define a random variable, X, such that $X \sim f(\cdot)$. Then, the expected value of $\psi(x)$ is

$$\mathbb{E}_f\{\psi(X)\} = \int_{\mathcal{X}} \psi(x) f(x)\, dx. \tag{1.1}$$

It can often be the case that integrals of the form of 1.1 are not analytically tractable. In other words, we cannot obtain a closed-form value. Numerical integration methods (i.e. quadrature-based), such as *Simpson's Rule*, can be easily applied to approximate the one-dimensional version of this problem. In general, however, these techniques do not scale up well to higher dimensions. For this reason, statisticians turn to Monte Carlo methods. In this section, we adopt the notational framework used in Robert and Casella (2004) and Nemeth (2014).

1.3.1 Perfect Monte Carlo

In classic Monte Carlo, we assume that is possible to generate identical and independently distributed (i.i.d) samples, $x^{(1)}, ..., x^{(N)}$, from $f(x)$. Using these samples, it is possible to empirically approximate the density function,

$$\widehat{f}(x) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x^{(i)}}(dx), \tag{1.2}$$

where $\delta(\cdot)$ is the Dirac delta function, applied with an empirical measure. We can use 1.2 to calculate

$$\widehat{f}(\psi) = \frac{1}{N}\sum_{i=1}^{N} \psi(x^{(i)}) = \int \psi(x^{(i)})\widehat{f}(dx). \tag{1.3}$$

It can be shown that 1.3 is an unbiased estimator of $\mathbb{E}_f\{\psi\}$ and that as $N \to \infty$, $\widehat{f}(\psi) \overset{a.s.}{\to} \mathbb{E}_f\{\psi\}$.

Properties of Perfect Monte Carlo

1. If $\text{Var}\{\psi(x)\} < \infty$, we know that $\text{Var}\{\widehat{f}(\psi)\} = \frac{1}{N^2}\sum_{i=1}^{N} \text{Var}\{\psi(x^{(i)})\} = \frac{1}{N}\text{Var}\{\psi(X)\}$.
2. *Central Limit Theorem:* if $\text{Var}\{\psi(x)\} < \infty$, then as $N \to \infty$,
$$\sqrt{N}\left(\widehat{f}(\psi) - \mathbb{E}(\psi(x))\right) \overset{D}{\to} N(0, \text{Var}\{\psi(X)\}).$$
3. The approximation error of Monte Carlo integration is $O(N^{-\frac{1}{2}})$, irrespective of the dimension of the integrand, d. In comparison, quadrature-based methods scale up in error as d grows (Doucet et al., 2001). The quality of the approximation improves as the number of samples increases.

Up until this point, we have assumed that is possible to directly sample from the target density, $f(\cdot)$. However, this might not always be the case and we must weaken our assumption to simply that $f(\cdot)$ can be evaluated pointwise. In this situation, we need to develop alternative methods to form our Monte Carlo estimate. This is usually done by sampling from an alternative proposal density, $q(\cdot)$. *Importance sampling* and *rejection sampling* are two such techniques.

The main premise of *rejection sampling* is simple. Let us consider a function, $f(\cdot)$, that we are interested in sampling from, whose functional form is well known. At each iteration, we generate a proposed value, $x \sim q(\cdot)$, and a corresponding uniform random number, $u \sim U(0, 1)$. The proposed candidate is then only accepted if $u \leq \frac{f(x)}{Mg(x)}$, where M is a suitably chosen large constant. In this way, those samples that are more likely to have come from the target density are kept and the others are discarded (von Neumann, 1951). Unfortunately, this approach leads to only a subset of the generated samples being used to calculate the Monte Carlo estimate. Computationally, this can be quite wasteful.

In contrast, *importance sampling* does not have this problem. All of the generated samples can be used to estimate the integrand. This method also adapts well to recursive estimation problems, as discussed further in Section 2.

1.3.2 Importance sampling

Our goal is to calculate $\mathbb{E}_f\{\psi(x)\}$, as defined in 1.1. Since we cannot sample directly from the target density, $f(\cdot)$, we can instead assume that we can evaluate it pointwise. Then, we can choose an alternative proposal density, $q(\cdot)$, to sample from. The proposal must be chosen with appropriate support, i.e. $q(x) > 0$ must imply that $f(x) > 0$. The integral in 1.1 can be rewritten, such that,

$$\mathbb{E}_f\{\psi(x)\} = \int_{\mathcal{X}} \psi(x)f(x)\;dx = \int_{\mathcal{X}} \frac{\psi(x)f(x)}{q(x)}q(x)\;dx = \mathbb{E}_q\left\{\frac{\psi(x)f(x)}{q(x)}\right\} = \mathbb{E}_q\left\{\psi(x)w(x)\right\}, \tag{1.4}$$

where $w(x) = \frac{f(x)}{q(x)}$ is known as the importance weight. This reformulation is incredibly useful within computational statistics, since it provides greater flexibility in simulation. In practice, we choose a proposal density that is easy to sample from. Upon generating the samples, $\{x_i\}_{i=1}^N$, from $q(\cdot)$, we can approximate 1.4 as,

$$\widehat{f}(\psi) = \frac{1}{N}\sum_{i=1}^N w(x^{(i)})\psi(x^{(i)}).$$

We can rewrite 1.2 to accommodate weighted samples. So, the estimated target density is

$$\widehat{f}(x) = \frac{1}{N}\sum_{i=1}^N w(x^{(i)})\delta_{x^{(i)}}(dx).$$

In certain situations, it is possible that we may not know the normalising constant for the target density. In this case, the target density $f(x) = \frac{\tilde{f}(x)}{C}$, where C is unknown. Then, the corresponding empirical approximation of 1.4 is,

$$\widehat{f}(\psi) = \frac{1}{CN}\sum_{i=1}^N \tilde{w}^{(i)}\psi(x^{(i)}),$$

where $\tilde{w}^{(i)} = \frac{\tilde{f}(x^{(i)})}{q(x^{(i)})}$ $\forall i$. Unfortunately, the normalising constant now appears in the Monte Carlo estimate of the expectation. Luckily, it is possible to approximate the constant C. Using the samples $\{x^{(i)}\}_{i=1}^N$ from the proposal, we know that,

$$C = \int_{\mathcal{X}} \tilde{f}(x)dx \approx \frac{1}{N}\sum_{i=1}^N \tilde{w}^{(i)}.$$

We are now in the position to write down the normalised Monte Carlo estimate for 1.4,

$$\widehat{f}(\psi) = \frac{\sum_{i=1}^N \psi(x^{(i)})f(x^{(i)})/q(x^{(i)})}{\sum_{i=1}^N f(x^{(i)})/q(x^{(i)})} = \sum_{i=1}^N w^{(i)}\psi(x^{(i)}), \tag{1.5}$$

where $w^{(i)}$ is the normalised importance weight for sample i (such that $\sum_{j=1}^N w^{(i)} = 1$).

We know that as $N \to \infty$, $N^{-1}\sum_{i=1}^N \frac{f(x^{(i)})}{q(x^{(i)})} \to 1$. Therefore, by the Strong Law of Large Numbers, $\widehat{f}(\psi) \to \mathbb{E}_f\{\psi(X)\}$ almost surely (Robert and Casella, 2004). Although the estimator provided in 1.5 is biased, this bias is incredibly small. The importance sampler can be improved by choosing a proposal distribution, $q(\cdot)$, that minimises the variance. Please refer to Theorem 3.12 in Robert and Casella (2004) for further details. Full details of the importance sampler are provided below in Algorithm 1.

Algorithm 1: Importance Sampling

```
1. Draw i.i.d samples, $\{x^{(i)}\}_{i=1}^N$, from $q(\cdot)$.
2. Calculate the unnormalised importance weights, $\tilde{w}^{(i)} = \tilde{f}(x^{(i)})/q(x^{(i)})\ \forall i$ and normalise to obtain $\{w^{(i)}\}_{i=1}^N$.
3. Evaluate the Monte Carlo estimator, $\widehat{f}(\psi) = \sum_{i=1}^N \psi(x^{(i)})w^{(i)}$.
```

1.4 Markov chain Monte Carlo (MCMC) methods

Markov chain Monte Carlo methods offer an alternative approach to the Monte Carlo methods discussed above. The main idea of MCMC is to create a Markov chain whose stationary distribution is the target distribution. In general, the samples generated from a Markov chain are statistically dependent, which is not ideal. Fortunately if the chain is run long enough, we can easily apply a processing step, such as *thinning*, to eliminate as much of the dependence as possible. In this section, we adopt the notational framework used in Robert and Casella (2004) and Nemeth (2014).

Formally, we define a Markov chain to be a sequence of random variables, $\{X^{(n)}\}_{n\geq 0}$, on the state space $\mathcal{X}$, fully described by

$$\mathbb{P}(X^{(n+1)} \in A|x^{(n)}, x^{(n-1)}, ..., x^{(0)}) = \mathbb{P}(X^{(n+1)} \in A|x^{(n)}) = \int_A K(x^{(n)}, dx),$$

where $K(\cdot,\cdot)$ is the *transition kernel* and $A \in \mathcal{B}(\mathcal{X})$ (Robert and Casella, 2004). In other words, the chain is memoryless and depends only on its behaviour at the previous state.

When the state space is discrete, the transition kernel is known as the *transition matrix*, containing the elements, $P_{xy} = \mathbb{P}(X^{(n)} = y|X^{(n-1)} = x)\ \forall x, y \in \mathcal{X}$. When the state space is continuous, the kernel function can also be thought of as a conditional density function, $K(x, x')$ for $x, x' \in \mathcal{X}$. Then, $\mathbb{P}(X \in A|x) = \int_{\mathcal{X}} K(x, x')\ dx'$.

If a target distribution is *invariant*, we know that $x^{(n-1)} \sim f(\cdot)$ implies that $x^{(n)} \sim f(\cdot)$. The Markov kernel of an invariant distribution has the property that,

$$\int_{\mathcal{X}} K(x, x')f(x)\ dx = f(x').$$

A Markov chain Monte Carlo (MCMC) method for the simulation of the target density, $f(\cdot)$, is any approach that generates an *irreducible, aperiodic,* and *ergodic* Markov chain, $\{X^{(n)}\}_{n\geq 0}$, whose stationary distribution is $f(\cdot)$ (Robert and Casella, 2004). For a full discussion of the theory behind ergodic Markov chains, we refer the reader to Chapter 6 of Robert and Casella (2004).

Ergodicity guarantees that the empirical average, $\frac{1}{N}\sum_{i=1}^N \psi(x^{(i)})$, converges almost surely to $\mathbb{E}_f\{\psi(X)\}$. If $f(\cdot)$ is the invariant density of the Markov chain, we also know that $x^{(n)} \to f(\cdot)$ in distribution.

Upon convergence, it is important to verify that the MCMC method samples from the correct stationary distribution. The easiest way to do this is by proving *detailed balance*. That is, for a Markov chain, $\{X^{(n)}\}_{n\geq 0}$, with transition kernel, $K(x, x')$, we must show that

$$K(x, x')f(x) = K(x', x)f(x'), \text{ for } x, x' \in \mathcal{X}.$$

1.4.1 Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm was first developed by Metropolis et al. (1953) and was later modified by Hastings (1970). The main advantage of this method is that it creates a Markov chain which is guaranteed to converge to the target density, $f(\cdot)$.

Once again, we must assume that it is not possible to sample directly from the target density. Instead, a proposal density, $q(x'|x)$, is chosen with respect to the dominating measure. In practice, the proposal density is easy to sample from and is either explicitly defined (up to a constant of proportionality) or *symmetric*, such that $q(x|x') = q(x'|x)$. Given the current state x, a new sample, x', is drawn from $q(\cdot|x)$. Then, this proposal is either accepted with some probability, $\alpha(x, x')$, or rejected.

The full method is provided in Algorithm 2. Please see Chapter 7 of Robert and Casella (2004) for a comprehensive overview of Metropolis-Hastings and its theoretical validity.

Algorithm 2: Metropolis-Hastings

```
1. Given $x^{(n)}$, generate $x' \sim q(\cdot|x^{(n)})$.
2. Calculate the acceptance probability, $\alpha(x^{(n)}, x') = \min\left\{\frac{f(x')q(x^{(n)}|x')}{f(x^{(n)})q(x'|x^{(n)})}, 1\right\}$.
3. Accept $x'$ with probability $\alpha$ and set $x^{(n+1)} = x'$. Otherwise, set $x^{(n+1)} = x^{(n)}$.
```

1.4.2 Gibbs sampler

The Gibbs sampler is a specific extension of the Metropolis-Hastings algorithm that is suitable for multivariate problems (Geman and Geman, 1984). We assume that the random variable of interest, $X = (X_1, ..., X_p)' \in \mathcal{X}$, is p-dimensional. Furthermore, it is assumed that we can sample from each of the univariate conditional distributions, $f_1, .., f_p$. In other words, we must know the mathematical form of

$$X_i|(x_1, ..., x_{i-1}, x_{i+1}, .., x_p) = X_i|x_{-i} \sim f_i(x_i|x_{-i}),$$

for $i = 1, .., p$. The conditional densities, $f_1, .., f_p$, are known as the ***full conditionals*** and are the only distributions required for sampling. At iteration n, a sample is drawn from each of the full conditional densities and each component of $x^{(n)}$ is updated. Unlike Metropolis-Hastings, the acceptance rate of the Gibbs sampler is always equal to 1. The full details of the Gibbs sampler are provided in Algorithm 3.

Algorithm 3: Gibbs sampler

```
1. Initialise $x^{(0)} = (x_1^{(0)}, ..., x_p^{(0)})$.
2. Then, for $i = 1, ..., N$,
    (a) Generate $x_1^{(i)} \sim f_i(x_1|x_2^{(i-1)}, ..., x_p^{(i-1)})$.
    (b) Generate $x_2^{(i)} \sim f_i(x_2|x_1^{(i)}, x_3^{(i-1)}, ..., x_p^{(i-1)})$.
    (c) ...
    (d) Generate $x_p^{(i)} \sim f_i(x_p|x_1^{(i)}, x_2^{(i)}, ..., x_{p-1}^{(i)})$.
```

2 Particle Filters

In this section, we provide an overview of the various particle filtering approaches available. Across the literature, there is some variation in the notation used for particle filters. For convenience, we adopt the mathematical notation used by Doucet and Johansen (2009) and also by Nemeth (2014).

2.1 State space models

Consider a discrete Markov process, $\{X_t\}_{t\geq 1} \subseteq \mathcal{X}$, fully described by the initial density $X_1 \sim \mu_\theta(x)$ and the probability density,

$$X_t|(X_{t-1} = x_{t-1}) \sim f_\theta(x_t|x_{t-1}), \tag{2.1}$$

where $\boldsymbol{\theta}$ is a parameter vector. In practice, $\boldsymbol{\theta}$, is unknown and it must be estimated. Here, $f_\theta(x'|x)$ represents the transition density of moving from x to x'. It is assumed that $\{X_t\}_{t\geq 1}$ is *latent* and that it cannot be observed directly. Instead, a set of observations, $\{Y_t\}_{t\geq 1} \subseteq \mathcal{Y}$, can be used to make inferences about the hidden process. Conditional on $\{X_t\}_{t\geq 1}$, the observations are independent to each other, such that,

$$Y_t|(X_t = x_t) \sim g_\theta(y_t|x_t). \tag{2.2}$$

+ The model specified by 2.1 and 2.2 is known as a *state space model* (or, alternatively a *hidden Markov model*) in the literature. Below, we describe two common state space models.

Finite state space HMM: Let $\mathcal{X} = \{1, ..., K\}$. Then,

$$\mathbb{P}(X_1 = k) = \mu(k), \qquad \mathbb{P}(X_t = k|X_{t-1} = l) = f(k|l).$$

The observations, $\{Y_t\}_{t\geq 1}$, are specified according to 2.2, using an arbitrary distribution, $g_\theta(.|x_t)\ \forall t$. Finite state space HMMs applied in a variety of different contexts. For instance, these models can be used by scientists to describe partially-observed, genetic sequences.

Linear Gaussian model: Let $\mathcal{X} = \mathbb{R}^{n_x}$, $\mathcal{Y} = \mathbb{R}^{n_y}$ and $X_1 \sim N(0, \Sigma)$. Here, the model is specified such that,

$$X_t = AX_{t-1} + BV_t, \tag{2.3}$$

$$Y_t = CX_t + DW_t, \tag{2.4}$$

where $V_t \overset{iid}{\sim} N(0, I_{n_y})$, $W_n \overset{iid}{\sim} N(0, I_{n_w})$ and A, B, C and D are well-defined matrices. The linear Gaussian model is often applied to target tracking and signal processing problems because of its tractability.

2.2 Bayesian filtering

Let $x_{1:t} = \{x_1, ..., x_t\}$ denote the hidden states up to time t and $y_{1:t} = \{y_1, .., y_t\}$ denote the observations up to time t. Then, using 2.1 and 2.2, we can define the prior distribution of $\{X_t\}_{t\geq 1}$ and the likelihood functions of the observations, $\{Y_t\}_{t\geq 1}$, respectively:

$$p(x_{1:t}) = \mu_\theta(x_1)\prod_{k=2}^{t} f_\theta(x_k|x_{k-1}), \tag{2.5}$$

$$p(y_{1:t}|x_{1:t}) = \prod_{k=1}^{t} g_\theta(y_k|x_k). \tag{2.6}$$

Thus, using 2.5 and 2.6, the unnormalised posterior distribution is

$$p(x_{1:t}, y_{1:t}, \theta) = \mu_\theta(x_1)\prod_{k=2}^{t} f_\theta(x_k|x_{k-1})\prod_{k=1}^{t} g_\theta(y_k|x_k). \tag{2.7}$$

Using Bayes theorem, we can express the joint posterior distribution as

$$p(x_{1:t}|y_{1:t}, \theta) = \frac{p(x_{1:t}, y_{1:t}, \theta)}{p(y_{1:t}, \theta)} = \frac{p(x_{1:t}, y_{1:t}, \theta)}{\int p(x_{1:t}, y_{1:t}, \theta)\ dx_{1:n}}. \tag{2.8}$$

When applying a finite-state HMM, we are able to fully evaluate 2.8. The integrals can be rewritten as sums and the probabilities can be calculated exactly. Similarly, for the linear Gaussian model (2.3, 2.4), it is a simple exercise to verify that $p(x_{1:t}|y_{1:t})$ is also Gaussian. In this context, it is normal practice to use Kalman filter techniques (Kalman, 1960) to evaluate the mean and covariance matrix.

For nonlinear, non-Gaussian models, it becomes very difficult to find analytical expressions for these distributions. Particle filters are a flexible family of numerical methods that can be used to approximate these intractable distributions. There are four main inference aims associated with particle filter methods: filtering, prediction, smoothing, and parameter estimation Fearnhead and Künsch (2018). In this report, we focus on reviewing methods for filtering.

Deriving the filtered density

Much of the filtering literature concentrates on approximating the *filtered density* (also known as the *marginal posterior*), $p(x_t|y_{1:t}, \theta)$. In other words, we are interested in determining the latent state, x_t, given all previous observations, $y_{1:t}$. The filtered density can be derived as follows.

The unnormalised posterior from equation 2.7 can be written recursively as,

$$p(x_{1:t}, y_{1:t}, \theta) = p(x_{1:t-1}, y_{1:t-1}, \theta) g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}). \tag{2.9}$$

In a similar manner, the joint posterior from equation 2.8 can be rewritten as,

$$p(x_{1:t}|y_{1:t}, \theta) = \frac{p(x_{1:t-1}, y_{1:t-1}, \theta) g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1})}{p(y_{1:t-1}, \theta) p(y_t|y_{1:t-1})} = p(x_{1:t-1}|y_{1:t-1}, \theta)\frac{g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1})}{p(y_t|y_{1:t-1})}, \tag{2.10}$$

where the predictive likelihood can be written as,

$$p(y_t|y_{1:t-1}, \theta) = \int g_\theta(y_t|x_t) \int f_\theta(x_t|x_{t-1}) p(x_{t-1}|y_{1:t-1}, \theta)\ dx_{t-1}\ dx_t.$$

By integrating out $x_{1:t-1}$ in equation 2.10, we can obtain the marginal posterior,

$$p(x_t|y_{1:t},\theta) = \frac{g_\theta(y_t|x_t)p(x_t|y_{1:t-1},\theta)}{p(y_t|y_{1:t-1},\theta)} = g_\theta(y_t|x_t)\int \frac{f_\theta(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1},\theta)}{p(y_t|y_{1:t-1},\theta)}\, dx_{t-1}. \tag{2.11}$$

So, up to a constant of proportionality, the filtered density can be written as,

$$p(x_t|y_{1:t},\theta) \propto g_\theta(y_t|x_t)\int f_\theta(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1},\theta)\, dx_{t-1}. \tag{2.12}$$

2.3 Sequential importance sampling (SIS)

It is widely agreed that the normalising constant in 2.11 is intractable. Our goal is to construct a Monte Carlo approximation of the filtered density of the form

$$\widehat{p}(x_t|y_{1:t},\theta) = \sum_{i=1}^{N} w_t^{(i)}\delta_{x_t^{(i)}}(dx_t),$$

where $\delta(\cdot)$ is the Dirac delta function, $\{x_t^{(i)}\}_{i=1}^N$ is the set of *particles* or samples at time t and $\{w_t^{(i)}\}_{i=1}^N$ is the corresponding set of importance weights.

The main idea behind the *sequential importance sampling* (SIS) filter is simple. At time $t-1$, we simulate N samples (particles) from $p(x_{t-1}|y_{1:t-1},\theta)$ with their respective importance weights, $\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^N$. At time t, the filtered density in 2.12 can then be approximated by,

$$\widehat{p}(x_t|y_{1:t},\theta) \propto g_\theta(y_t|x_t)\sum_{i=1}^{N} w_{t-1}^{(i)}\; f_\theta(x_t|x_{t-1}^{(i)}).$$

Since we cannot sample directly from the filtered density, we can instead sample $x_t^{(i)}$ from an appropriate proposal distribution, $q(x_t|x_{1:t-1},y_t,\theta)$. We can recursively calculate the importance weights as

$$\tilde{w}_t^{(i)} = \tilde{w}_{t-1}^{(i)}\frac{g_\theta(y_t|x_t^{(i)})f_\theta(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|x_{1:t-1}^{(i)},y_t,\theta)}. \tag{2.13}$$

As discussed in Section 1.3.2, we must normalise the importance weights. That is, $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}}$.

The filter can easily be modified to approximate the joint density, $p(x_{1:t}|y_{1:t},\theta)$, as specified in 2.10. Then,

$$\widehat{p}(x_{1:t}|y_{1:t},\theta) = \sum_{i=1}^{N} w_t^{(i)}\delta_{x_{1:t}^{(i)}}(dx_{1:t}).$$

In this case, we store the particle path, $x_{1:t}^{(i)} = \{x_{1:t-1}^{(i)}, x_t^{(i)}\}$ and modify the importance weights accordingly,

$$w_t^{(i)} \propto p(x_{1:t-1}^{(i)}|y_{1:t-1},\theta)\frac{g_\theta(y_t|x_t^{(i)})f_\theta(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_{1:t}^{(i)}|x_{1:t-1}^{(i)},y_{1:t},\theta)}.$$

The importance function for each particle can be written as

$$q(x_{1:t}|x_{1:t-1},y_{1:t},\theta) = q(x_t|x_{1:t-1},y_t,\theta)\sum_{i=1}^{N} v_{t-1}^{(i)}\delta_{x_{1:t-1}^{(i)}}(dx_{1:t-1}),$$

where $v_{t-1}^{(i)}$ are normalised weights. Then, we can recursively update the importance weights for the joint distribution in the following way,

$$w_t^{(i)} \propto \frac{w_{t-1}^{(i)}}{v_{t-1}^{(i)}} \frac{g_\theta(y_t|x_t^{(i)}) f_\theta(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t|x_{1:t-1}^{(i)}, y_{1:t}, \theta)},$$

where $x_{1:t}^{(i)} = \{x_{1:t-1}^{(i)}, x_t^{(i)}\}$ is drawn from $q(x_{1:t}|x_{1:t-1}^{(i)}, y_{1:t}, \theta)$ (Godsill and Clapp, 2001).

Using this framework, the SIS algorithm iteratively propagates the particles and their associated weights through time. However, the accuracy of the approximation deteriorates as t increases, due to the phenomenon of *particle degeneracy*. Over time, the variance of importance weights grows and consequently, more and more particles have negligible weights. This means that only a few of the particles are propogated through time. See Kong et al. (1994) for a full discussion of particle degeneracy.

2.4 Sequential importance resampling (SIR)

To control the variance of the importance weights, a resampling step (with replacement) can be introduced to the SIS algorithm. In practice, the resampling probability of each particle is proportional to its importance weight (e.g. Gordon et al., 1993). In this way, the particles with negligible contributions are dropped and only those that improve the approximation of the filtered density are replicated. After the resampling has been performed, the weights are reset to $\frac{1}{N}$.

Figure 2.1 provides a visual representation of the method. In this example, we start at time $t-1$ with N equally weighted particles, $\{\tilde{x}_{t-1}^{(i)}, N^{-1}\}_{i=1}^N$, which can be used to approximate $p(x_{t-1}|y_{1:t-2}, \theta)$. We can calculate the importance weight of each particle using the recursion outlined in 2.13 and then normalise, yielding $\{\tilde{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^N$. At this point, we can conduct a resampling step to discard negligible particles and retain the set $\{x_{t-1}^{(i)}, N^{-1}\}_{i=1}^N$, which can be used to approximate $p(x_{t-1}|y_{1:t-1}, \theta)$. Finally, at time t, we can propagate the particles forward, resulting in $\{\tilde{x}_t^{(i)}, N^{-1}\}_{i=1}^N$.

While resampling improves the long-term stability of the algorithm, it introduces unwanted dependence between the particle paths. In this case, we cannot use the standard convergence results for importance sampling. Furthermore, the replication of particles results in several identical paths, reducing the quality of the approximation as a whole. However, these side-effects can be mitigated by carefully choosing when and how we resample. A common approach is to monitor the *effective sample size* (ESS),

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^N (w_t^{(i)})^2},$$

over time. A low effective sample size indicates severe particle degeneracy (Kong et al., 1994).

Bootstrap filter

One of the first particle filters to introduce a resampling step was proposed by Gordon et al. (1993). The *sequential importance resampling* or *bootstrap* filter introduces a *multinomial resampling scheme*, where the resampling probability of a particle is proportional to its importance weight. For convenience, the proposal density is $q(x_t|x_{1:t-1}, y_t, \theta) = f_\theta(x_t|x_{t-1})$.

By using the transition density, the importance weight recursion in 2.13 simplifies to $\tilde{w}_t^{(i)} = g_\theta(y_t|x_t^{(i)})$.

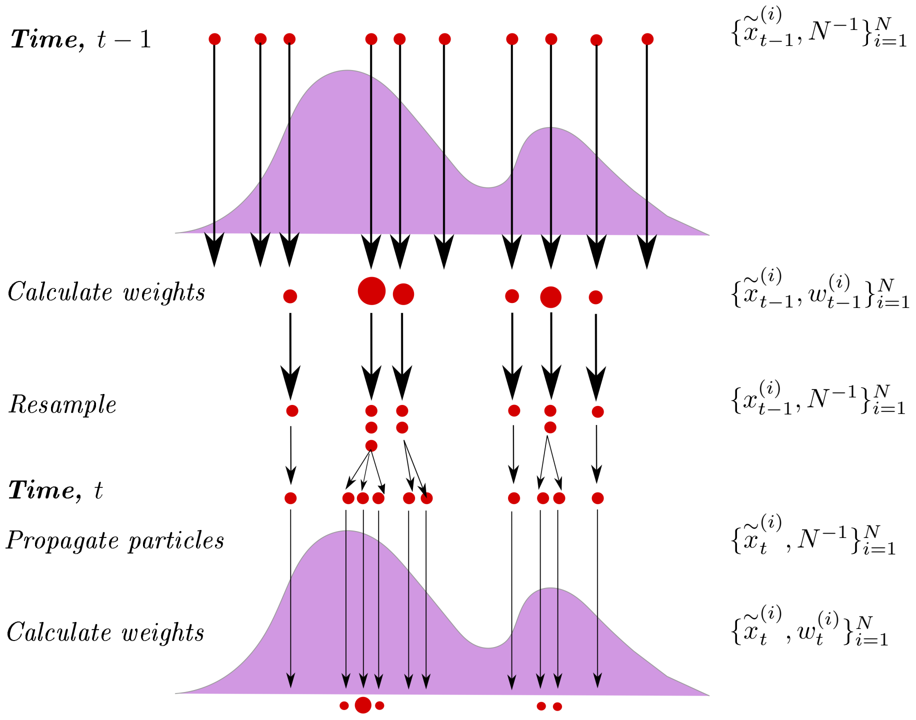
Time, $t-1$
$\{\tilde{x}_{t-1}^{(i)}, N^{-1}\}_{i=1}^{N}$
Calculate weights
$\{\tilde{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N}$
Resample
$\{x_{t-1}^{(i)}, N^{-1}\}_{i=1}^{N}$
Time, t
Propagate particles
$\{\tilde{x}_{t}^{(i)}, N^{-1}\}_{i=1}^{N}$
Calculate weights
$\{\tilde{x}_{t}^{(i)}, w_{t}^{(i)}\}_{i=1}^{N}$

Figure 2.1: Sequential importance resampling

The full details of the bootstrap filter are provided in Algorithm 4.

Algorithm 4: Sequential importance resampling

```
1. Initialise: draw samples, $\{\tilde{x}_1^{(i)}\}_{i=1}^N$, from the prior $p(x_1|\theta)$, with weights $w_1^{(i)} = g_\theta(y_1|x_1)$ $\forall i$.
2. Iterate for $t \geq 2$,
    (a) Resample according to the importance weights, $w_{t-1}^{(i)}$ $\forall i$, to obtain $\{x_{t-1}^{(i)}, N^{-1}\}_{i=1}^N$.
    (b) Propagate particles, $\tilde{x}_t^{(i)} \sim f_\theta(x_t^{(i)}|x_{t-1}^{(i)})$ $\forall i$.
    (c) Calculate weights, $\tilde{w}_t^{(i)} = g_\theta(y_t|x_t^{(i)})$ and normalise to obtain $w_t^{(i)}$.
```

Example: one-dimensional, nonlinear model (Gordon et al., 1993)

We consider the one-dimensional, nonlinear model,

$$x_t = 0.5x_{t-1} + \frac{25x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2(t-1)) + w_t, \tag{2.14}$$

$$\text{and, } y_t = \frac{x_t^2}{20} + v_t, \tag{2.15}$$

where v_t and w_t are zero-mean Gaussian i.i.d noise, with variances $\sigma_w^2 = 10$ and $\sigma_v^2 = 1$, respectively. The initial density is $\mu_\theta(x_1) = N(0, 2)$. The left-hand plot of Figure 2.2 shows a simulation of length $T = 100$ of the nonlinear model. This simulation example is very challenging because of the squared term in the observed signal. The filter needs to be able to distinguish the magnitude of the signal and its direction at each time point. Naturally, there is a large amount of noise generated by the system.

The right-hand plot in Figure 2.2 shows the result of implementing the bootstrap filter on the simulated

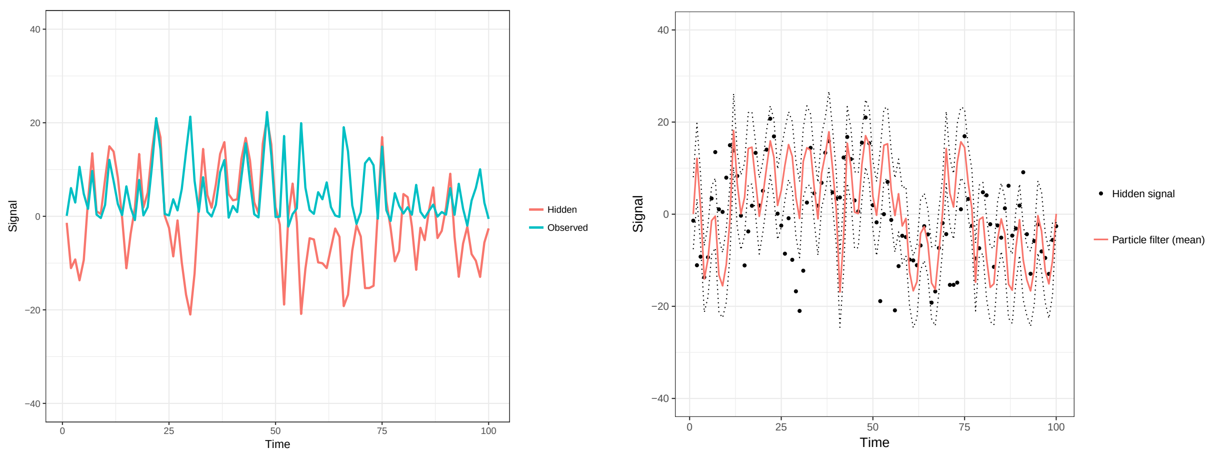

Figure 2.2: (L) $T = 100$ realisations of the nonlinear model; (R) Bootstrap filter estimate of posterior mean

data, with $N = 500$ particles. The initial prior is set to be $p(x_1|\theta) = N(0, \sigma_w^2)$. Within each iteration, a multinomial resampling method is used. The solid red line represents the posterior mean estimate over time and the dashed lines estimate the 99% probability region. At certain time points, the hidden state lies outside of the percentile estimates. However, most of the hidden states lie within or very close to the probability region.

2.5 Auxiliary particle filter

In general, resampling goes a long way to mitigate against the impact of particle degeneracy. However, the introduction of a resampling step temporarily increases the Monte Carlo variance. Sophisticated resampling procedures, such as residual resampling (Liu and Chen, 1998), tend to only control the variance introduced by the resampling step itself.

As mentioned in Section 1.3.2, the variance of the importance sampler is linked to the choice of proposal distribution. The particle approximation can be improved by choosing a proposal that resembles the filtered density. In this case, the importance weights (2.13) would be approximately equal and their variance would be nearly zero. The *optimal proposal* distribution that minimises the variance of the importance weights is

$$q(x_t|x_{1:t-1}, y_t, \theta) = p(x_t|x_{t-1}, y_t, \theta). \tag{2.16}$$

See Proposition 3.4.1 in Nemeth (2014) for a proof of this result. We note that

$$p(x_t|x_{t-1}, y_t, \theta) = \frac{g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1})}{p(y_t|x_{t-1}, \theta)} = \frac{g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1})}{\int g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}) dx_t}. \tag{2.17}$$

Unfortunately, the denominator of 2.17 is difficult to evaluate for many state space models. Therefore, the optimal proposal density may not necessarily be available in closed form. The main exception to this is the linear Gaussian model (2.3, 2.4). Instead, we can try to approximate the optimal proposal distribution as closely as possible.

Pitt and Shephard (1999) propose the *auxiliary particle filter* as an alternative. To motivate the method, we first rewrite the filtered density from 2.11 as

$$\begin{aligned}\widehat{p}(x_t|y_{1:t},\theta) &\propto \sum_{i=1}^{N} w_{t-1}^{(i)} g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}^{(i)}) \\ &= \sum_{i=1}^{N} \frac{g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}^{(i)})}{\int g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}^{(i)})\, dx_t} \int g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}^{(i)})\, dx_t.\end{aligned}$$

So,

$$\widehat{p}(x_t|y_{1:t},\theta) \propto \sum_{i=1}^{N} p(x_t|x_{t-1}^{(i)}, y_t, \theta) w_{t-1}^{(i)} \int g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}^{(i)})\, dx_t = \sum_{i=1}^{N} \zeta_t^{(i)} p(x_t|x_{t-1}^{(i)}, y_t, \theta), \tag{2.18}$$

where $\zeta_t^{(i)} = w_{t-1}^{(i)} \int g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}^{(i)})\, dx_t$. The filtered density has now been reinterpreted as a mixture of $p(x_t|x_{t-1}^{(i)}, y_t, \theta)$ for $i = 1,..,N$, each with weight $\zeta_t^{(i)}$. Pitt and Shephard create a new auxiliary variable, i, to denote the i-th density in the mixture. For the i-th component, the joint distribution is

$$\widehat{p}(x_t, i|y_{1:t},\theta) = \zeta_t^{(i)} p(x_t|x_{t-1}^{(i)}, y_t, \theta). \tag{2.19}$$

Then, the following proposal density can be used to sample from 2.19,

$$q(x_t^{(i)}|y_{1:t},\theta) = \xi_t^{(i)} q(x_t|x_{t-1}^{(i)}, y_t, \theta). \tag{2.20}$$

Here, $q(x_t|x_{t-1}, y_t, \theta)$ is a suitably chosen approximation of the optimal proposal (2.17) and $\{\xi_t^{(i)}\}_{i=1}^N$ are probabilities which are approximately equal to $\{\zeta_t^{(i)}\}_{i=1}^N$.

We are now in a position to discuss the method used to estimate the full filtered density. For each component i, an index number, k_i, is sampled from the set $\{1, ..., N\}$ with probability $\xi_t^{(i)}$. Then, conditional on k_i, a new sample, $x_t^{(i)}$, is generated from $q(x_t|x_{t-1}^{(k_i)}, y_t, \theta)$. Each sample-index pair $\{x_t^{(i)}, k_i\}$ has importance weight,

$$\tilde{w}_t^{(i)} \propto \frac{w_{t-1}^{(k_i)} g_\theta(y_t|x_t^{(i)}) f_\theta(x_t^{(i)}|x_{t-1}^{(k_i)})}{\xi_t^{(k_i)} q(x_t^{(i)}|x_{t-1}^{(k_i)}, y_t, \theta)}.$$

If we drop the indices, we now have a full approximation of the filtered density, $\{w_t^{(i)}, x_t^{(i)}\}_{i=1}^N$. The full auxiliary particle filter algorithm can be found in Section 4.2 of Doucet and Johansen (2009) or Section 3.4.2 of Nemeth (2014).

3 High-dimensional particle filtering

We now turn our attention to the issues that are encountered in particle filtering when the system dimension, d, grows in size. There have been several methodological advances that have been proposed in the literature to try to tackle this problem. Each of these so-called *high-dimensional particle filters* make specific, structured assumptions about the state-space models they can be applied to. In many ways, high-dimensional particle filtering is still an unsolved problem. In this section, we endeavour to provide a comparative discussion of the existing methods and highlight future avenues of research.

3.1 Problems in high dimensions

3.1.1 Filter instability

If d is small, *filter stability* guarantees that the approximation error (i.e the error between the estimated filtered density and the truth) generated by a particle filter does not accumulate over time. One of the most influential papers on filter stability is by Del Moral and Guionnet (2001), which outlines the key properties of time-uniform convergence.

It can be established that the approximation error can be bounded by some constant, $\frac{C}{\sqrt{N}}$, over time. As the system dimension grows, however, it is often the case that C is exponential in d. Please see Rebeschini et al. (2015) for a more detailed discussion of instability in high dimensions.

3.1.2 Filter collapse

Bickel et al. (2008) are often credited with triggering the study of high-dimensional particle filters. In their paper, it is theoretically shown that the bootstrap filter is unable to scale up properly to a high-dimensional setting.

Specifically, Bickel et al. prove that the maximal importance weight in a particle filter converges to 1 in probability as d grows. Unless the sample size, N, grows exponentially in the system dimension, the filter collapses. This phenomenon is similar to that of particle degeneracy for SIS. In high dimensions, weight degeneracy occurs quickly and can be seen within one iteration. The sequential application of importance sampling increases the variance of the importance weights and reduces the effective sample size obtained.

Example: sensor network simulation (Septier and Peters, 2016)

We demonstrate filter collapse using a simulation example from Septier and Peters (2016). Let d be the number of sensors that are uniformly deployed on a network, $\{1, ..., , d\} \times \{1, .., d\}$. The location of each sensor, k, is $\mathcal{S}_k \in \mathbb{R}^2$. It is assumed that each sensor collects some noisy information about some underlying process, such that $\forall k = 1, .., d$, $Y_t(k)|(X_t(k) = x_t(k)) \sim g(y_t(k)|x_t(k))$. We set,

$$f_\theta(x_t|x_{t-1}) = N(x_t; \alpha x_t, \Sigma), \qquad g_\theta(y_t|x_t) = N(y_t; x_t, \Sigma_y).$$

Here, $\Sigma_y = \sigma_y^2 I_{d\times d}$ and $\Sigma_{ij} = \alpha_0 \exp(-\frac{||\mathcal{S}_i - \mathcal{S}_j||_2^2}{\beta}) + \alpha_1 \delta_{ij}$ for $i, j \in 1, ..., d$. The Kronecker delta function is defined such that $\delta_{ij} = 1$ if $i = j$, and 0 otherwise. As suggested by Septier and Peters, the model parameters are fixed as $\alpha_0 = 3, \alpha_1 = 0.01, \alpha = 0.9, \beta = 20$, and $\sigma_y^2 = 2$.

Several replications of the bootstrap filter are implemented for $d = 4, 9$ and 16. Below, the variance of the importance weights and the effective sample size (see Figure 3.1) are averaged and plotted over time. To illustrate the effects of degeneracy, the sample size is fixed at $N = 10,000$. As the number of sensors grows, the variance of the importance weights increases and the effective sample size reduces substantially.

3.2 Resample-Move and Bridging Densities

In the past, filter collapse has been commonly addressed using the *Resample-Move* algorithm, which was first proposed by Gilks and Berzuini (2001). This method consists of applying an MCMC transition kernel, $K_t(x'_{1:t}|x_{1:t})$, one or more times to a set of particles, after the resampling stage. For instance, it is quite common for a kernel

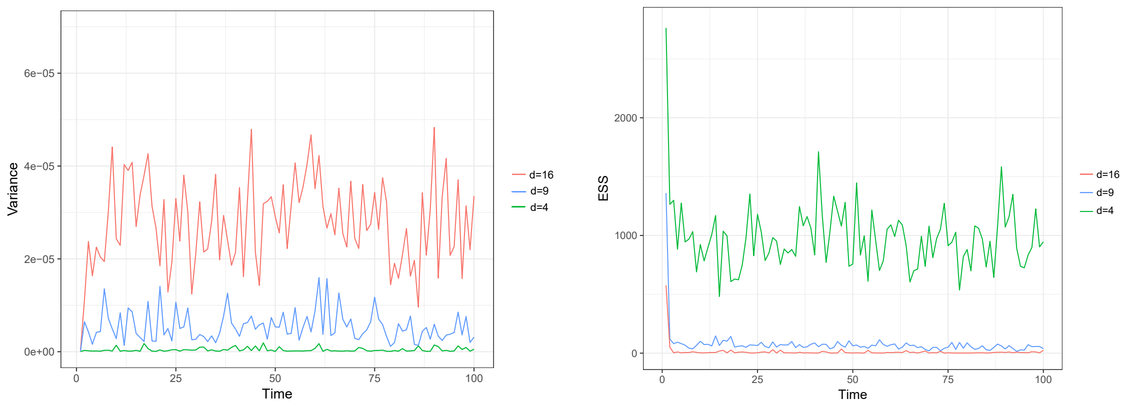

Figure 3.1: (L) Variance of importance weights over time; (R) Effective sample size (ESS) over time

based upon the Gibbs sampler or the Metropolis-Hasting algorithm to be used. The introduction of MCMC moves makes it possible to perturb the particle locations, thereby reducing degeneracy and improving the quality of the approximation.

The invariance property of MCMC methods is incredibly important. That is, we want to ensure that,

$$\int p(x_{1:t}|y_{1:t})K_t(x'_{1:t}|x_{1:t})\ dx_{1:t} = p(x'_{1:t}|y_{1:t}).$$

This ensures that if the particles, $\{x_{1:t}^{(i)}\}_{i=1}^{N}$, are truly generated from $p(x_{1:t}|y_{1:t})$, we know that applying the transition kernel to any of the samples will produce new sequences from the same distribution. Furthermore, even if the particles are not correctly sampled from the target distribution, the use of a transition kernel will shift the particles so that their new distribution is more accurate, with respect to the total variation norm (Septier and Peters, 2015).

In practice, Markov transition kernels are not applied to the entire path history, as this would not be achievable in linear computational time. To retain an online algorithm, we must restrict ourselves to only updating the particles up to some fixed lag, L. The Resample-Move method proceeds as follows in Algorithm 5, with K_n defined as the Markov transition kernel of the invariant distribution, $p(x_{1:t}|y_{1:t}, \theta)$. For further details about the Resample-Move algorithm, please refer to Septier and Peters (2015) and Doucet and Johansen (2009).

Godsill and Clapp (2001) propose a general framework for incorporating MCMC moves into particle filtering methodology, in which a sequence of *bridging densities* is constructed between the initial sampling distribution at time $t-1$ and the target distribution at time t. By introducing a series of gradual transitions, the idea is to reduce the variance of the importance weights and improve MCMC convergence.

The initial distribution is $\pi_0(x_{1:t}) = q(x_{1:t}|x_{1:t-1}, y_{1:t}, \theta)$ and the final distribution is $\pi_{M+1} = p(x_{1:t}|y_{1:t}, \theta)$, where $M \geq 1$. At time t, the following sequence of bridging densities, $\pi_1, .., \pi_M$, is introduced,

$$\pi_m(x_{1:t}) \propto q(x_{1:t}|x_{1:t-1}, y_{1:t}, \theta)^{\alpha_m} p(x_{1:t-1}|y_{1:t-1}, \theta)^{1-\alpha_m},$$

where $0 < \alpha_M < ... < \alpha_1 = 1$. In order to move the particles through these densities, Godsill and Clapp propose the following algorithm. At time t, the particles can be propagated using the proposal distribution, $q(x_{1:t}|x_{1:t-1}^{(i)}, y_{1:t}, \theta)$, to obtain $\{x_{1:t}^{(i)}, w_t^{(i)(0)}\}_{i=1}^{N}$, with $w_t^{(i)(0)} = w_{t-1}^{(i)}$. Then, at step $m = 1, ..., M$, the importance

Algorithm 5: Resample-Move particle filter

```
1. Initialise at $t = 1$
    • Sample $\tilde{x}_1^{(i)} \sim q(x_1|y_1, \theta)$ and compute weights $w_1^{(i)}$ $\forall i$.
    • Resample $\{\tilde{x}_1^{(i)}, w_1^{(i)}\}_{i=1}^N$ to obtain $N$ equally-weighted particles, $\{x_1^{(i)}, \frac{1}{N}\}_{i=1}^N$.
    • Sample $x_1^{'(i)} \sim K_1(x_1|x_1^{(i)})$ $\forall i$.
2. Iterate for $2 \leq t < L$,
    • Sample $\tilde{x}_t^{(i)} \sim q(x_t|x_{1:t-1}^{'(i)}, y_t, \theta)$ and set $x_{1:t}^{(i)} \leftarrow (x_{1:t-1}^{'(i)}, x_t^{'(i)})$ $\forall i$.
    • Compute weights $w_t^{(i)}$ $\forall i$.
    • Resample $\{\tilde{x}_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$ to obtain $N$ equally-weighted particles, $\{x_{1:t}^{(i)}, \frac{1}{N}\}_{i=1}^N$.
    • Sample $x_{1:t}^{'(i)} \sim K_t(x_{1:t}|x_{1:t}^{(i)})$ $\forall i$.
3. Iterate for $t \geq L$,
    • Sample $\tilde{x}_t^{(i)} \sim q(x_t|x_{1:t-1}^{'(i)}, y_t, \theta)$ and set $x_{1:t}^{(i)} \leftarrow (x_{1:t-1}^{'(i)}, x_t^{'(i)})$ $\forall i$.
    • Compute weights $w_t^{(i)}$ $\forall i$.
    • Resample $\{\tilde{x}_{1:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$ to obtain $N$ equally-weighted particles, $\{x_{1:t}^{(i)}, \frac{1}{N}\}_{i=1}^N$.
    • Sample $x_{t-L+1:t}^{'(i)} \sim K_t(x_{t-L+1:t}|x_{1:t}^{(i)})$ and set $x_{1:t}^{'(i)} \leftarrow (x_{1:t-L}^{(i)}, x_{t-L+1:t}^{'(i)})$ $\forall i$.
```

weights are computed as,

$$w_t^{(i)(m)} \propto w_t^{(i)(m-1)} \frac{\pi_m(x_{1:t}^{(i)(m-1)})}{\pi_{m-1}(x_{1:t}^{(i)(m-1)})}.$$

If the weights are degenerate, an additional resampling step can be performed. Finally, each particle is moved forward independently. We generate $x_{1:t,m}^{(i)}$ using the Markov transition kernel $K(x_{1:t,m-1}^{(i)}, \cdot)$ with stationary distribution $\pi_m(x_{1:t})$. If $M = 0$, we recover the standard MCMC resampling algorithm of Gilks and Berzuini (2001). Please see Section 2.3.3 of Septier and Peters (2015) for the full algorithm.

3.3 Block particle filter

An alternative approach to prevent filter collapse is to use *localisation* techniques, such as the *block particle filter* proposed by Rebeschini et al. (2015). In many applications, the components of X_t and Y_t are associated with positions in space. Rebeschini et al. propose a local state space model, constructed over a graph $G = (V, E)$. At time t, the latent state $X_t = (X_t^\nu)_{\nu \in V}$ and the observed state $Y_t = (Y_t^\nu)_{\nu \in V}$. Within this framework, the transition and observation densities (for convenience, we have dropped the θ-notation) are given by,

$$f(x|z) = \prod_{\nu \in V} f^\nu(x^\nu|z) \qquad\qquad g(y|x) = \prod_{v \in V} g^v(y^\nu|x^\nu),$$

respectively. The spatial graph, G, is equipped with a natural distance, d, such that $d(\nu, \nu')$ is the length of the shortest path in G between $\nu, \nu' \in V$. For simplicity, it is assumed that $\{X_t\}_{t \geq 1}$ is *local* to the extent that $f^\nu(x^\nu|z)$ depends only on $x^{N(\nu)}$. Here, the neighbourhood is defined to be $N(\nu) = \{\nu' \in V | d(\nu, \nu') \leq r\}$, with $r \in \mathbb{N}$. Similarly, the observations are local to extent that $Y_t^\nu | X_t$ depends only on X_t^ν.

In many cases, these graphical models exhibit the *decay of correlations* property, where (X_t^ν, Y_t^ν) and

(X_t^w, Y_t^w) are approximately independent when $d(\nu, w)$ is sufficiently large. In other words, the graph has a limited spatial memory. Rebeschini et al. exploit this property to localise their particle filter and guarantee stability.

To define the block particle filter, the vertex set, V, is partitioned into non-overlapping blocks, $\mathcal{K}$. That is, $V = \bigcup_{K \in \mathcal{K}} K$ and $K \cap K' = \emptyset$ for $K, K' \in \mathcal{K}$. The algorithm divides the variables into separate blocks and updates the filtered density approximation independently in each block. The estimated filtered density is given by,

$$\widehat{p}(x_t|y_{1:t}) = \prod_{K \in \mathcal{K}} \sum_{i=1}^{N} w_{t,K}^{(i)} \delta_{x_{t,K}^{(i)}}(dx_t).$$

In the case where $\mathcal{K} = \{V\}$, then the block particle filter reduces to the bootstrap particle filter.

3.4 Space-time particle filter

Beskos et al. (2017) have recently proposed the *space-time particle filter* (STPF). In their method, the authors assume that there exists an increasing sequence of sets, $\{A_{t,j}\}_{j=1}^{B_t}$, such that $A_{t,1} \subset A_{t,2} \subset \ldots \subset A_{t,B_t} = \{1 : d\}$. The integer B_t, where $0 < B_t \leq d$, is set so that,

$$g_\theta(y_t|x_t) f_\theta(x_t|x_{t-1}) = \prod_{j=1}^{B_t} \alpha_{t,j}(y_t, x_{t-1}, x_t(A_{t,j})), \tag{3.1}$$

where $\alpha_{t,j}(\cdot)$ are pre-specified functions and $x_t(A) = \{x_t(j) : j \in A\} \in \mathbb{R}^{|A|}$.

It should be noted that the structure specified above is not unusual in state space modelling. For example, a product factorisation could be devised for the transition density, $f_\theta(x_t|x_{t-1})$, by marginalising over subsets of coordinates. Then, a factorisation for the observation density, $g_\theta(y_t|x_t)$, could be obtained if a local dependence structure is assumed for the model. Additionally, the structure in 3.1 is not strictly necessary for the STPF, but in such scenarios the overall performance of the method would be affected, as extra reweighting and sampling steps would need to be included.

The main idea behind the STPF is to exploit the product structure in 3.1 and construct a particle filter that can progress simultaneously along the time index and the space index (in contrast to standard particle filters which move only along the time index). At $t = k$, the STPF breaks the k-th time step into B_k space steps and locally runs N independent particle filters, each with M particles. It is shown that the algorithm is asymptotically consistent and has a sub-exponential cost in d, the state dimension. Please see Section 2.3.3 of Septier and Peters (2015) for the full algorithm.

3.5 Discussion

Broadly speaking, there are three categories of high-dimensional particle filtering approaches available in the literature: MCMC methods, localisation methods, and other more experimental techniques, like the space-time particle filter. We consider each approach in turn and assess its strengths and limitations.

In general, the introduction of Markov chain moves to the standard particle filter is powerful, as it helps us to mitigate the effects of degeneracy. Arguably, the most well-known MCMC method is the Resample-Move algorithm (Gilks and Berzuini, 2001). This algorithm is easy to implement and has a simple interpretation.

However, the Resample-Move method suffers from a few drawbacks.

First, although the MCMC moves reintroduce some diversity amongst the particles, the importance weights have the same analytical form as the standard particle filter. So, this method does not significantly reduce the number of resampling steps applied. Consequently, Resample-Move only partially mitigates the problem associated with repeated sampling (Doucet and Johansen, 2009).

Second, there are often situations where a very large number of MCMC iterations are required to reach the target density. For instance, this can happen when the likelihood for a new data point is centered far away from the samples generated from the proposal distribution. In this case, the MCMC sampler must work harder to converge from a subset of atypical data points lying in the tails of the target distribution and there may be no definitive way of telling if convergence has actually been obtained. To prevent these issues, we can alternatively make use of bridging densities (Godsill and Clapp, 2001). It must be noted, however, that high-dimensional MCMC methods in any capacity are almost always computationally expensive. Their inclusion could make online filtering nearly impossible.

Localisation methods are growing in popularity within the particle filter community. In particular, Rebeschini et al. (2015) make great strides forward, by showing that when a local state space model exhibits decaying correlations in space, localisation can provide stability to the filter. It is also shown that the approximation error is independent of dimension and is almost negligible at positions away from the boundaries between blocks. Although this is a promising start, the theoretical justification for localisation needs further work, especially for other model structures. For example, in its current form, this method is not applicable to highly interconnected graphical models.

The major drawback of the block particle filter is that the blocking operation introduces a bias. Since resampling does not occur globally for all components, localisation introduces artificial discontinuities in the particles. This means that the filter estimates are not reliable on the vertices near the boundaries of the blocks (this may make up a considerable proportion of the total number of vertices). Fearnhead and Künsch (2018) point out that these discontinuities could have severe consequences in the next propagation step if the transition depends on differences between neigbouring regions. It would be interesting to see if the theoretical guarantees of localisation could be extended to methods that seek to reduce the discontinuities between blocks.

We now turn our attention to the space-time particle filter (Beskos et al., 2017). Fortunately, the factorising assumption made by the STPF is weaker than the assumptions made for the block particle filter, since some dependencies between different components of x_t (given x_{t-1}) from different subsets are permitted. Beskos et al. also demonstrate that the filter provides a consistent estimate as the number of particles, N, grows. Although these ideas are promising, the need to identify a complete factorisation means that this method is not generally applicable to complex models.

3.6 Conclusion

In this report, we have provided an introduction to classical particle filtering and illustrated some of the approaches that could be used in a high-dimensional setting. In many ways, high-dimensional particle filtering is still very much an open area of research and there is a lot that could be done. The main concerns that are driving research in this area are filter stability, the prevention of filter collapse and computational efficiency.

As it stands, there are several different research philosophies being explored, each making different assumptions about the structure of state space models. It would be beneficial to unify the current literature and devise a general framework to evaluate filtering methods. Advances in this area would open up new possibilities in applied fields such as geophysics and meteorology, especially in numerical weather prediction, where high-dimensional filtering problems need to solved quickly. Further extensions of high-dimensional particle filtering include, but are not limited to, the following.

- **Scalable particle filter methods without the use of importance weights**
  There is a general consensus that most of the problems encountered by particle filters in high dimensions result from the repeated application of importance sampling. So, it stands to reason that a particle filter method without importance weights could work well.
- **Parallelisation of localisation methods**
  By definition, local particle filter methods are well-suited to parallel computation. Since all of the updates are conducted locally, it would be possible to distribute each local region to a different processor. In general, parallelised particle filters have only just begun to appear within the literature. Alongside new methodology, it would be interesting to see how existing methods like the block particle filter would fit into a parallelised framework.
- **Use of the optimal proposal distribution in high dimensions**
  In standard particle filtering, approximations of the optimal proposal can be used to control the variance of the importance weights. It would be interesting to see if these tools could be modified for high dimensions.

References

Beskos, A., Crisan, D., Jasra, A., Kamatani, K. and Zhou, Y. (2017), 'A stable particle filter for a class of high-dimensional state-space models', **49**, 24–48.

Bickel, P., Li, B., Bengtsson, T. et al. (2008), Sharp failure rates for the bootstrap particle filter in high dimensions, *in* 'Pushing the limits of contemporary statistics: Contributions in honor of Jayanta K. Ghosh', Institute of Mathematical Statistics, pp. 318–329.

Del Moral, P. and Guionnet, A. (2001), On the stability of interacting processes with applications to filtering and genetic algorithms, *in* 'Annales de l'Institut Henri Poincaré (B) Probability and Statistics', Vol. 37, Elsevier, pp. 155–194.

Doucet, A., De Freitas, N. and Gordon, N. (2001), An introduction to sequential Monte Carlo methods, *in* 'Sequential Monte Carlo methods in practice', Springer, pp. 3–14.

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.

Geman, S. and Geman, D. (1984), 'Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images', *IEEE Transactions on Pattern Analyis and Machine Intelligence* **6**(6), 721–741.

Gilks, W. R. and Berzuini, C. (2001), 'Following a moving target - Monte Carlo inference for dynamic Bayesian models', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **63**(1), 127–146.

Godsill, S. and Clapp, T. (2001), Improvement strategies for Monte Carlo particle filters, *in* 'Sequential Monte Carlo methods in practice', Springer, pp. 139–158.

Golightly, A. and Wilkinson, D. J. (2011), 'Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo', *Interface focus* **1**(6), 807–820.

Gordon, N. J., Salmond, D. J. and Smith, A. F. (1993), Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *in* 'IEE Proceedings F (Radar and Signal Processing)', Vol. 140, IET, pp. 107–113.

Hastings, W. K. (1970), 'Monte Carlo sampling methods using Markov chains and their applications', *Biometrika* **57**(1), 97–109.

Kalman, R. E. (1960), 'A new approach to linear filtering and prediction problems', *Journal of basic Engineering* **82**(1), 35–45.

Kong, A., Liu, J. S. and Wong, W. H. (1994), 'Sequential Imputations and Bayesian Missing Data Problems', *Journal of the American Statistical Association* **89**(425), 278–288.

Liu, J. S. and Chen, R. (1998), 'Sequential Monte Carlo methods for dynamic systems', *Journal of the American statistical association* **93**(443), 1032–1044.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953), 'Equation of state calculations by fast computing machines', *The journal of chemical physics* **21**(6), 1087–1092.

Nemeth, C. (2014), Parameter estimation for state space models using Sequential Monte Carlo algorithms, PhD thesis.

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

Rebeschini, P., Van Handel, R. et al. (2015), 'Can local particle filters beat the curse of dimensionality?', *The Annals of Applied Probability* **25**(5), 2809–2866.

Robert, C. P. and Casella, G. (2004), *Monte Carlo Statistical Methods (Springer Texts in Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Septier, F. and Peters, G. W. (2015), An overview of recent advances in Monte-Carlo methods for Bayesian filtering in high-dimensional spaces, *in* 'Theoretical Aspects of Spatial-Temporal Modeling', Springer, pp. 31–61.

Septier, F. and Peters, G. W. (2016), 'Langevin and Hamiltonian based Sequential MCMC for efficient Bayesian filtering in high-dimensional spaces', *IEEE Journal of Selected Topics in Signal Processing* **10**(2), 312–327.

von Neumann, J. (1951), 'Various techniques used in connection with random digits', *Journal of Research of the National Bureau of Standards, Applied Math Series* **12**(3), 36–38.

Welch, G. and Bishop, G. (2006), *An Introduction to the Kalman Filter*, Technical report, University of North Carolina.