An Introduction to Particle Filtering

Author: Lisa Turner
Supervisor: Dr. Christopher Sherlock

10th May 2013

Abstract

This report introduces the ideas behind particle filters, looking at the Kalman filter and the SIS and SIR filters to learn about the latent state of state space models. It then introduces particle MCMC as a way of learning about the parameters behind these models. Finally, the SIR filter and particle MCMC algorithms are applied to reaction networks, in particular the Lotka Volterra model.

1 Introduction

Particle filtering has become an established technique for solving state space models. The Monte Carlo techniques behind particle filters have existed since the 1950s (Hammersley and Morton, 1954) but a lack of computational power at the time and problems with degeneracy meant these methods were generally overlooked. Since the introduction of the bootstrap filter (Gordon et al., 1993) and more general resampling schemes, there has been a large increase in research in this area. Particle filters have been applied to a wide range of fields, such as economics (Kim et al., 1998; Johannes et al., 2009), signal processing (Arulampalam et al., 2002), target tracking (Ristic et al., 2004), neuroscience (Salimpour and Soltanian-Zadeh, 2009) and biochemical networks (Djuric and Bugallo, 2009) to name a few.

Before particle filtering methods became popular, the Kalman filter was the standard method for solving state space models. The Kalman filter can be applied to optimally solve a linear Gaussian state space model. When the linearity or Gaussian conditions do not hold, its variants, the extended Kalman filter and the unscented Kalman filter, can be used. However, for highly non linear and non-Gaussian problems they fail to provide a reasonable estimate.

Particle filtering techniques offer an alternative method. They work online to approximate the marginal distribution of the latent process as observations become available. Importance sampling is used at each time point to approximate the distribution with a set of discrete values, known as particles, each with a corresponding weight. There are several papers and books which give detailed reviews of particle filters and their applications, for example Ristic et al. (2004), Arulampalam et al. (2002), Doucet et al. (2000) and Pollock (2010). Furthermore, an overview of the more general sequential Monte Carlo methods is given by Doucet and Johansen (2008).

This report introduces state space models in section 1.1 before looking, in section 2, at the special cases of linear Gaussian state space models which can be solved optimally using the Kalman filter. The extended Kalman filter and unscented Kalman filter are also introduced in section 2. The sequential importance sampling (SIS) and sequential importance resampling (SIR) algorithms are introduced in sections 3 and 4 respectively. Section 4 also gives a comparison of the different resampling schemes commonly used in the SIR filter. A new area of research, particle Markov chain Monte Carlo (PMCMC) methods, introduced by Andrieu et al. (2010), is discussed as a way of determining the parameters of the latent process from the observations in section 5. Finally, section 6 looks at how particle filters have been applied to reaction networks, including the use of PMCMC to find the rate parameters of the Lotka Volterra model.

1.1 State Space Models

A state space model contains two sequences of random variables (Pollock, 2010):

1. A latent (hidden) state, X_t, which forms a Markov chain, so $p(x_t|x_{0:t-1}) = p(x_t|x_{t-1})$, where $x_{0:t-1} = \{x_0, \ldots, x_{t-1}\}$.
2. A series of observations, Y_t, which depends only on X_t. Hence, given X_t, Y_t is independent of X_s, Y_s with $s \neq t$.

The problem can be decomposed into a number of smaller, more manageable problems and online modification is possible. This means that the state estimates do not have to be completely recalculated at each time step. Consider the joint distribution of $x_{0|T}$ and $y_{0|T}$, given by:

$$p(x_{0:T}, y_{0:T}) = p(y_{0:T}|x_{0:T})p(x_{0:T}) = p(x_0)\prod_{t=0}^{T} p(y_t|x_t)\prod_{t=1}^{T} p(x_t|x_{t-1}).$$

The filtering problem of interest is how to incorporate all observed information to give an estimate of the latent process. This could be the joint distribution of all the latent variables up to time t given all observations up to this time point which can be written in a recursive form as:

$$p(x_{0:t}|y_{0:t}) = p(x_{0:t-1}|y_{0:t-1})\frac{p(x_t|x_{t-1})p(y_t|x_t)}{p(y_t|y_{0:t-1})}. \tag{1}$$

Or it could be the marginal distribution:

$$p(x_t|y_{0:t}) = \frac{p(x_t|y_{0:t-1})p(y_t|x_t)}{p(y_t|y_{0:t-1})} = p(y_t|x_t)\int p(x_{t-1}|y_{0:t-1})\frac{p(x_t|x_{t-1})}{p(y_t|y_{0:t-1})}dx_{t-1}, \tag{2}$$

where

$$p(y_t|y_{0:t-1}) = \int p(y_t|x_t)\left\{\int p(x_{t-1}|y_{0:t-1})p(x_t|x_{t-1})dx_{t-1}\right\}dx_t. \tag{3}$$

Unfortunately, it is not often possible to calculate the integral directly, because the normalising constant $p(y_t|y_{0:t-1})$ inhibits direct calculation of the integral. In a few cases direct implementation is possible. One of these is the Kalman filter, discussed in section 2.

2 The Kalman Filter

The Kalman filter is a special class of particle filter, also known as a linear Gaussian state space model, which is analytically tractable. It was named after Kalman (1960) who described the recursive relationship for solving the discrete data filtering problems as a way of detecting a signal of a known form in the presence of random noise within control theory.

The latent variables, $\{x_t\}_{t=0}^T$, and the observed values, $\{y_t\}_{t=0}^T$, have Gaussian distributions with a linear dependency between variables. Hence the joint distribution, $p(x_0, \ldots, x_T, y_0, \ldots, y_T)$, and the marginal distributions, $p(x_t|y_0, \ldots, y_t)$, are also Gaussian. The Kalman filter combines all available measurement data and prior knowledge of the system to produce an estimate of the desired variables in such a manner that the error is minimised statistically. It is an optimal filter with respect to many different criteria (Maybeck, 1982), for example the root mean squared error.

2.1 Computing Kalman filter equations

The transition equations for the Kalman filter can be written in the general form:

$$X_t = AX_{t-1} + c + u_t \ , \text{ where } u_t \sim N(0, \Sigma), \tag{4}$$

$$Y_t = BX_t + d + v_t \ , \text{ where } v_t \sim N(0, \Sigma_y). \tag{5}$$

The initial latent variable also has a Gaussian distribution:

$$X_0 = \mu_0 + u_0 \ , \text{ where } u_0 \sim N(0, \Sigma_0). \tag{6}$$

In principle, the inference problem could be solved by using standard results of multivariate Gaussian marginal and conditional distributions (Bishop, 2006). However, the idea of the Kalman filter is to provide a more efficient way to do these calculations. The aim is to find the mean and covariance of the latent variables at time t given all observations up to time t, and so define the marginal distribution at every time step. Let $x_{1:t} = \{x_1, \ldots, x_t\}$ and:

- $\mu_{t|t-1}$ and $\Sigma_{t|t-1}$ denote the mean and covariance of the marginal distribution of x at time t given observations $y_1, \ldots, y_{t-1}$; the *a priori* estimate.
- $\mu_{t|t}$ and $\Sigma_{t|t}$ denote the mean and covariance of the marginal distribution of x at time t given observations $y_1, \ldots, y_t$; the *a posteriori* estimate.

The Kalman filter is a recursive algorithm with two steps: a prediction step and an updating step.

- **Prediction Step:** Responsible for projecting forward in time to obtain the *a priori* estimate at the next time step of the mean and variance.

$$\mu_{t|t-1} = A\mu_{t-1|t-1} + c. \tag{7}$$

$$\Sigma_{t|t-1} = A\Sigma_{t-1|t-1}A^T + \Sigma. \tag{8}$$

- **Updating Step:** Incorporates the new observation, y_t, into the *a priori* estimate to obtain an improved *a posteriori* estimate.

$$\mu_{t|t} = \mu_{t|t-1} + \Sigma_{t|t-1}B^T(B\Sigma_{t|t-1}B^T + \Sigma_y)^{-1}(y_t - B\mu_{t|t-1} - d). \tag{9}$$
$$\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{t|t-1}B^T(B\Sigma_{t|t-1}B^T + \Sigma_y)^{-1}B\Sigma_{t|t-1}. \tag{10}$$

2.1.1 Derivation of Recursion Equations

If two random variables, W and Z, are jointly Gaussian then (Bishop, 2006):

$$E(W|Z) = E(W) + \frac{Cov(W,Z)}{Var(Z)}(Z - E(Z)) \ , \ Cov(W|Z) = Cov(W) - \frac{Cov(W|Z)Cov(W|Z)^T}{Cov(Z)}. \tag{11}$$

Since the joint distribution of the latent variables and the observed variables at time t is Gaussian given $Y_{1:t}$, the recursion equations can be calculated using (11). From equations (4) and (5):

$$\begin{aligned} Cov(x_t, y_t|Y_{t-1}) &= Cov(x_t, Bx_t + d|Y_{t-1}) = \Sigma_{t|t-1}B^T, \\ Cov(y_t|Y_{t-1}) &= Cov(Bx_t + d + v_t|Y_{t-1}) = B\Sigma_{t|t-1}B^T + \Sigma_y, \\ E(x_t|Y_{t-1}) &= \mu_{t|t-1}, \ E(y_t|Y_{t-1}) = B^T\mu_{t|t-1} + d. \end{aligned}$$

Setting $W = x_t|Y_{t-1}$ and $Z = y_t|Y_{t-1}$, and substituting into the standard results in (11) gives the recursion equations (9) and (10).

2.2 The Kalman Filter Algorithm

Algorithm 1 The Kalman Filter Algorithm

Initialise:

1. $\mu_{0|0} = \mu_0 + \Sigma_0 B^T(B\Sigma_0 B^T + \Sigma_y)^{-1}(y_0 - B\mu_0)$
2. $\Sigma_{0|0} = \Sigma_0 - \Sigma_0 B^T(B\Sigma_0 B^T + \Sigma_y)^{-1}B\Sigma_0$

Iterate: For t in 1 to T

1. Prediction Step:
   (a) $\mu_{t|t-1} = A\mu_{t-1|t-1} + c$
   (b) $\Sigma_{t|t-1} = A\Sigma_{t-1|t-1}A^T + \Sigma$
2. Make observation y_t
3. Updating Step:
   (a) $\mu_{t|t} = \mu_{t|t-1} + \Sigma_{t|t-1}B^T(B\Sigma_{t|t-1}B^T + \Sigma_y)^{-1}(y_t - B\mu_{t|t-1} - d)$
   (b) $\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{t|t-1}B^T(B\Sigma_{t|t-1}B^T + \Sigma_y)^{-1}B\Sigma_{t|t-1}$

Output: $\mu_{t|t}$ and $\Sigma_{t|t}$ for $t \in \{0, T\}$

With each time step, the previous *a posteriori* estimate is used as the current *a priori* estimate. This

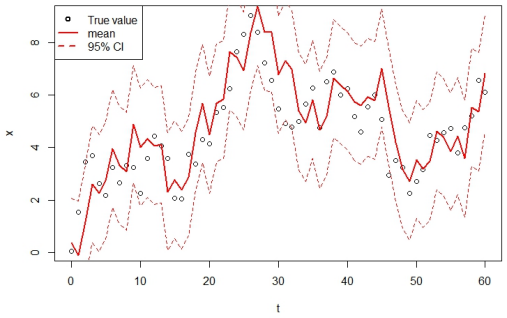


Figure 1: The Kalman filter estimate and 95% CI for a set of simulated latent values and observations from the linear Gaussian state space model given in (12).

recursive relationship is one of the big advantages of the Kalman filter. It does not require all previous data to be kept in storage and reprocessed every time an observation is made. The Kalman filter algorithm is given by Algorithm 1.

2.3 Example

To demonstrate the Kalman filter, consider the simple one dimensional linear Gaussian dynamical system:

$$X_t \sim N(X_{t-1} + 0.2, 1) \ , \ Y_t \sim N(X_t, 3) \ , \ X_0 \sim N(0, 1). \tag{12}$$

A latent process and a set of observations were simulated using (12) for times $t = 0, \ldots, 60$. These observations were then used to predict the latent distribution using the Kalman filter. Figure 1 shows the mean and 95% credible intervals (CI) of the Kalman filter, in red, along with the true latent values, given by the black points. Nearly all the latent values are within the 95% CI of the mean and the mean lies close to the true simulated latent values.

2.4 Extended Kalman Filter

In many situations the Kalman filter cannot be used to find an optimal solution, because the linearity and Gaussian assumptions do not hold. A possible alternative is to use the extended Kalman filter (EKF), which is a sub optimal algorithm. When the evolution of the latent variable and the observed process cannot be written as a linear model, a local linearisation of the equations can be applied, akin to a Taylor expansion (Welch and Bishop, 1996). This may be sufficient to describe the non-linearity. Once this approximation has been made, Kalman filter theory can be applied.

The marginal distribution, $p(x_t|y_{0:t})$, is approximated by a Gaussian distribution. For this to work, the latent and observed transitions need to be differentiable and of the form:

$$\begin{aligned} x_t &= f(x_{t-1}, u_t) \ , \text{ where } u_t \sim N(0, \Sigma), \\ y_t &= h(x_t, v_t) \ , \text{ where } v_t \sim N(0, \Sigma_y). \end{aligned}$$

The function, f, relates the current latent variables to those at the previous time step and has a noise parameter, u_t, which has zero mean and covariance Σ. The function h relates the observed variables to the latent variables and has a noise parameter, v_t, which has zero mean and covariance Σ_y. In practice, the value of the noise at each time step is not known so this value is set to zero in the estimation of the latent and observed process:

$$\mu_{t|t-1} = f(\mu_{t-1|t-1}, 0), \tag{13}$$

$$\tilde{y} = h(\mu_{t|t-1}, 0), \tag{14}$$

where $\mu_{t-1|t-1}$ is the *a posteriori* estimate at time $t-1$ and $\mu_{t|t-1}$ is the *a priori* estimate of the latent state at time t. The governing equations, which linearise the estimates about equations (13) and (14) are given by:

$$x_t \approx \mu_{t|t-1} + A(x_{t-1} - \mu_{t-1|t-1}) + Uu_t,$$
$$y_t \approx \tilde{y}_t + H(x_t - \mu_{t|t-1}) + Vv_t,$$

where x_t and y_t are the actual latent and observed vectors and:

- A is the Jacobian of partial derivatives of f with respect to x, evaluated at the $\mu_{t-1|t-1}$: $A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\mu_{t-1|t-1})$,
- U is the Jacobian of partial derivatives of f with respect to u: $U_{[i,j]} = \frac{\partial f_{[i]}}{\partial u_{[j]}}(\mu_{t-1|t-1})$,
- H is the Jacobian of partial derivatives with respect to x: $H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\mu_{t|t-1}, 0)$,
- V is the Jacobian of partial derivatives with respect to v: $V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\mu_{t|t-1}, 0)$.

These must be evaluated at each time step, but the subscript t has been left out for simplicity. The updated equations are analogous to those of the Kalman filter and are given by (Welch and Bishop, 1996):

$$\mu_{t|t-1} = f(\mu_{t-1|y-1}, 0) \ , \ \Sigma_{t|t-1} = A\Sigma_{t-1|t-1}A^T + U\Sigma U^T,$$

where $U\Sigma U^T$ represents the estimated covariance of the latent process. The prediction equations are given by:

$$\mu_{t|t} = \mu_{t|t-1} + \Sigma_{t|t-1}H^T(H\Sigma_{t|t-1}H^T + V\Sigma_y V^T)^{-1}(y_t - h(\mu_{t|t-1})),$$
$$\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{t|t-1}H^T(H\Sigma_{t|t-1}H^T + V\Sigma_y V^T)^{-1}H\Sigma_{t|t-1}.$$

For more information on the extended Kalman filter see Welch and Bishop (1996). The EKF algorithm is given in Ristic et al. (2004) along with applications of the EKF in target tracking.

2.5 The Unscented Kalman Filter

The unscented Kalman filter can be applied to general non linear Gaussian problems of the form:

$$x_t = f(x_{t-1}) + u_t,$$
$$y_t = h(x_t) + v_t.$$

It is a recursive minimum mean squared error estimator which addresses some of the issues with the EKF. The EKF only uses first order terms of the Taylor expansion and so large errors can be introduced, especially when the function is highly non linear and so higher order terms have a significant effect. The UKF follows the intuition that, with a fixed number of parameters, it should be easier to approximate a Gaussian distribution than it is to approximate an arbitrary, non linear function (Julier and Uhlmann, 2004). The Gaussian random variable is specified by a minimal set of deterministically chosen sample points, known as sigma points. These capture the mean and covariance so errors are only introduced in the third moment. More information on the UKF can be found in Merwe et al. (2000) and Julier et al. (1995). The unscented Kalman filter has been shown to have superior performance to the EKF in both accuracy and robustness without any additional computational costs (Merwe et al., 2000).

3 The Sequential Importance Sampling Algorithm

The basis of particle filtering methods lies in sequentially updating a distribution using importance sampling techniques. One particle filtering method is the sequential importance sampling (SIS) method, introduced by Kong et al. (1994). SIS involves using importance sampling to solve the recursion equation.

3.1 Importance Sampling

Importance sampling forms the basis for the sequential Monte Carlo methods used in particle filtering to solve the recursion equation. In a Bayesian context, a probability distribution is known up to a normalising constant. The integral $I = \frac{\int h(x)p(x)dx}{\int p(x)dx}$, where h(x) is an arbitrary density, cannot be calculated directly. Introduce an importance density, $q(x)$, which is on the same support at $p(x)$, that is if $p(x) > 0$ then $q(x) > 0$. Then the integral can be rewritten as (Pollock, 2010):

$$I = \frac{\int h(x)\frac{p(x)}{q(x)}q(x)dx}{\int \frac{p(x)}{q(x)}dx}, \tag{15}$$

and thought of as a ratio of expectations with respect to $q(x)$. Thus, it can be estimated by:

1. Sampling x_i, $i = 1, \ldots, N$, iid from $q(x)$.

2. Estimating the integral by:

$$\hat{I} = \frac{\frac{1}{N}\sum_{i=1}^{N} h(x_i)\frac{p(x_i)}{q(x_i)}}{\frac{1}{N}\sum_{j=1}^{N}\frac{p(x_j)}{q(x_j)}}.$$

Define the unnormalised weights to be $\tilde{w}_i = \frac{p(x_i)}{q(x_i)}$ and normalised weights to be:

$$w_i = \frac{\tilde{w}_i}{\sum \tilde{w}_j} = \frac{\frac{p(x_i)}{q(x_i)}}{\sum_{j=1}^{N} \frac{p(x_j)}{q(x_j)}},$$

and so this integral is given by the estimation, $\hat{I} = \sum_{i=1}^{N} w_i h(x_i)$.

The set of particles (samples) and their associated weights, $\{x_i, w_i\}_{i=1}^{N}$, can be viewed as an approximation to the probability distribution $p(x)$. This method of approximation is known as importance sampling (IS) and removes the need for the normalising constant to be calculated.

3.2 Sequential Importance Sampling

In particle filtering, the distribution of interest is the marginal or joint distribution of the latent variables at time t, given all observations up to that point. However, the intractability of the normalising constant, $p(y_t|y_{0:t-1})$, often inhibits direct calculation so IS must be used. The aim is to be able to sequentially update the posterior distribution at time t without modifying the previously simulated states $x_{0:t-1}$. To do this, suppose there is a importance function, $q(x_{0:t}|y_{0:t})$, which is easy to sample from and that $p(x_{0:t}|y_{0:t}) > 0 \;\Rightarrow\; q(x_{0:t}|y_{0:t}) > 0$. Furthermore, assume that the importance function is chosen so that it updates recursively in time when the next observation becomes available and is of the form(Ristic et al., 2004):

$$q(x_{0:t}|y_{0:t}) = q(x_{0:t-1}|y_{0:t-1})q(x_t|x_{0:t-1}, y_{0:t}).$$

This means the estimate at time $t-1$ can be propagated to time t without modifying the past simulated trajectories $\{x_{0:i}, i = 1, \ldots, t-1\}$.

The recursive equation of the joint distribution is given by:

$$\begin{aligned}
p(x_t|y_{0:t}) &= p(y_t|x_t) \int p(x_{t-1}|y_{0:t-1}) \frac{p(x_t|x_{t-1})}{p(y_t|y_{0:t-1})} dx_{t-1}, \\
&= p(y_t|x_t) \int p(x_{t-1}|y_{0:t-1}) \frac{p(x_t|x_{t-1}) q(x_{0:t-1}|y_{0:t-1}) q(x_t|x_{0:t-1}, y_{0:t})}{p(y_t|y_{0:t-1}) q(x_{0:t-1}|y_{0:t-1}) q(x_t|x_{0:t-1}, y_{0:t})} dx_{t-1}, \\
&= p(y_t|x_t) \int \frac{p(x_{t-1}|y_{0:t-1}) q(x_{0:t-1}|y_{0:t-1})}{q(x_{0:t-1}|y_{0:t-1})} \frac{p(x_t|x_{t-1}) q(x_t|x_{0:t-1}, y_{0:t})}{p(y_t|y_{0:t-1}) q(x_t|x_{0:t-1}, y_{0:t})} dx_{t-1}.
\end{aligned}$$

Then, assuming a weighted particle approximation $\left\{w_{t-1}^{(i)}, x_{t-1}^{(i)}\right\}$ exists for $p(x_{t-1}|y_{0:t-1})$, IS can be used to give an approximation to $p(x_t|y_{0:t})$ where the unnormalised weights are:

$$\tilde{w}_t^{(i)} = \tilde{w}_{t-1}^{(i)} \frac{p(y_t|x_t) p(x_t|x_{t-1})}{q(x_t|x_{0:t-1}, y_{0:t})} \;, \text{ for } i = 1, \ldots, N,$$

and the normalised weights are given by

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}},$$

which provides a weighted particle approximation $\left\{w_t^{(i)}, x_t^{(i)}\right\}_{i=1}^{N}$ to the marginal distribution at time t. Using this type of importance function, the weights of the particles can be updated sequentially in time as the next observation becomes available. The choice of the importance function is discussed more in section 3.4. Since it is possible to sample from the importance function, and the likelihood and transitional probabilities are both known, all that is needed is for an initial set of samples to be generated, and for the importance weights to be calculated iteratively. This process is known as the sequential importance sampling (SIS) method (Kong et al., 1994) and provides the basis for estimating the marginal distribution, $p(x_t|y_{0:t})$, with a weighted approximation $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}$. The SIS algorithm recursively propagates weights and particles as each measurement is sequentially received; it is given by Algorithm 2.

3.3 Degeneracy of the SIS Algorithm

Ideally, the posterior distribution would be the importance density function but this is not possible. For an importance function of the form:

$$q(x_{0:t}|y_{0:t}) = q(x_t|x_{0:t-1}, y_{0:t})q(x_{0:t-1}|y_{0:t}), \tag{16}$$

the variance of the importance weights can only increase over time (Kong et al., 1994). This has a harmful effect on the accuracy and leads to the degeneracy phenomenon. After only a few steps, nearly all the particles will have negligible weight. As a result, a large amount of computational effort will be devoted to updating a contribution which has almost zero weight. The degeneracy phenomonon is a big problem in particle filtering. One way to reduce it is to increase the number of samples, N, which may be impractical. Degeneracy can also be minimised through a good choice of importance function and by including a resampling step in the SIS algorithm (see section 4.1).

3.4 Choice of Importance Function

A good choice of importance function is key to slowing down the degeneracy of the filter and this is done by minimising the variance of the importance weights. The optimal importance function, first given by Zaritskii et al. (1976) is:

$$\begin{aligned} q(x_t|x_{t-1}^{(i)}, y_{0:t})_{opt} &= p(x_t|x_{t-1}^{(i)}, y_t), \\ &= \frac{p(y_t|x_t, x_{t-1}^{(i)})p(x_t|x_{t-1}^{(i)})}{p(y_t|x_{t-1}^{(i)})}. \end{aligned} \tag{17}$$

Substituting (17) into the weight gives $w_t^{(i)} \propto w_{t-1}^{(i)} p(y_t|x_{t-1}^{(i)})$. This means that the importance weights at time t can be calculated, and possibly resampled, before the particles are propagated to time t. However,

Algorithm 2 The Sequential Importance Sampling Algorithm

Initialise: At time $t = 0$

1. For $i = 1, \ldots, N$
   (a) Sample $x_0^{(i)} \sim p(x_0)$
   (b) Evaluate the importance weights up to a normalising constant:

$$\tilde{w}_0^{(i)} = p(y_0|x_0^{(i)})$$

2. For $i = 1, \ldots, N$ normalise the importance weights:

$$w_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{j=1}^N \tilde{w}_0^{(j)}}$$

Iterate: For t in 1 to T

1. For $i = 1, \ldots, N$
   (a) Sample $x_t^{(i)} \sim q(x_t|x_{t-1}^{(i)}, y_{0:t})$ and $x_{0:t}^{(i)} = (x_{0:t-1}^{(i)}, x_t^{(i)})$
   (b) Evaluate the importance weights up to a normalising constant:

$$\tilde{w}_t^{(i)} = \tilde{w}_{t-1}^{(i)} \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{0:t})}$$

2. For $i = 1, \ldots, N$ normalise the importance weights:

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^N \tilde{w}_t^{(j)}}$$

Output: $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^N$ for $t \in \{0, T\}$

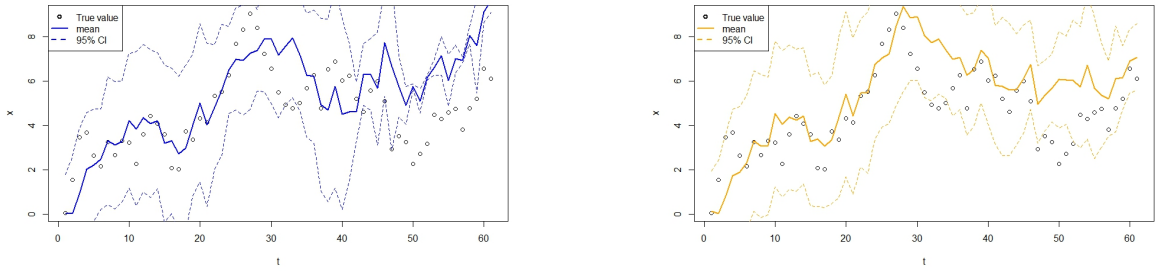
to use (17) it must be possible to sample from $p(x_t|x_{t-1}^{(i)}, y_t)$ and calculate $p(y_t|x_{t-1}^{(i)})$, one or both of which are often not possible (Doucet et al., 2000). A suboptimal choice of importance function must be used. The most popular choice is the transitional prior, $p(x_t|x_{t-1})$, which gives a weight $w_t^{(i)} \propto w_{t-1}^{(i)} p(y_t|x_t)$. It is no longer possible to calculate the importance weights before the particles have been propagated at time t.

3.5 Example

The SIS filter can be applied to the latent values and observations used in the Kalman filter example, see section 2.3. These values were simulated from the following linear Gaussian state space model:

$$X_t \sim N(X_{t-1} + 0.2, 1) \ , \ Y_t \sim N(X_t, 3) \ , \ X_0 \sim N(0, 1).$$

The SIS filter estimates, with $N = 100$ particles and $N = 500$ particles and an importance function $p(x_t|x_{t-1})$, are shown in Figure 2a and Figure 2b respectively along with the 95% credible intervals.



(a) N=100 (b) $N = 500$

Figure 2: The SIS filter for the Linear Gaussian model with N particles where the observations are simulated from the linear Gaussian state space model, given by (12).

Although initially this particle filters may work well, the means begins to drift away from the latent process. Furthermore, the credible intervals reduce. These effects are both due to degeneracy of the particle filter. With $N = 100$ particles, the degeneracy happens quicker than with $N = 500$ showing that an increase in the number of particles helps reduce the problem. When comparing with the Kalman filter estimate, shown in Figure 1, it is clear the Kalman filter performs a lot better.

4 The Sequential Importance Resampling Algorithm

The sequential importance resampling (SIR) algorithm, developed separately from the SIS algorithm, was first introduced by Gordon et al. (1993) and contains a resampling step at each iteration of the sequential importance sampling algorithm. Since then, resampling has been shown to have both major practical and theoretical benefits (Doucet and Johansen, 2008).

4.1 Resampling

An IS approximation of the target distribution is based on weighted samples from $\pi(x_t|y_{0:t})$ and hence is not distributed according to $p(x_t|y_{0:t})$. Resampling provides a way to give an approximation from the target distribution by resampling N particles from the IS approximation. Each particle is picked with a probability proportional to the weight associated with it. Resampling means that a particles is removed with a high probability if their corresponding weight is low and multiple copies of particles with high weights are produced; the future of these can be explored independently. This can be seen to provide stability in the future at the cost of an increase in the immediate Monte Carlo variance (Doucet and Johansen, 2008).

The sequential importance resampling algorithm is given by Algorithm 3. Often, a natural choice for the proposal distribution is $p(x_t|x_{t-1})$ which was the choice for the original SIR filter given by Gordon et al. (1993). This gives weights in the resampling algorithm of $w_t^{(i)} \propto p(y_t|x_t^{(i)})$. Although resampling reduces the problem of degeneracy, it introduces other problems which did not exist with the SIS algorithm. The

Algorithm 3 The Sequential Importance Resampling Algorithm

Initialise: At time $t = 0$

1. For $i = 1, \ldots, N$
   (a) Sample $x_0^{(i)} \sim p(x_0)$
   (b) Assign weights $\tilde{w}_0^{(i)} = p(y_0|x_0^{(i)})$
2. For $i = 1, \ldots, N$
   (a) Normalise weights $w_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{j=1}^{N} \tilde{w}_0^{(j)}}$

Iterate: For t in 1 to T

1. For $i = 1, \ldots, N$
   (a) Resample $\tilde{x}_{t-1}^{(i)}$ by resampling from $\left\{x_{t-1}^{(i)}\right\}_{i=1}^{N}$ with probabilities $\left\{w_{t-1}^{(i)}\right\}_{i=1}^{N}$
   (b) Set $\left\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\right\}_{i=1}^{N} \longleftarrow \left\{\tilde{x}_{t-1}^{(i)}, \frac{1}{N}, \right\}_{i=1}^{N}$
   (c) Propagate weights $\tilde{w}_t^{(i)} = \frac{p(y_t|x_t^{(i)})p(x_t^{(i)})}{\pi(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{0:t})}$
2. For $i = 1, \ldots, N$
   (a) Normalise weights $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}}$

Output: $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}$ for $t \in \{0, T\}$

simulated particles interact and hence they are no longer statistically independent. Furthermore, there is a loss of diversity in the particles which results from resampling. A common approach is to monitor the effective sample size (ESS) which can be estimated from the the normalised weights by $\hat{N}_{eff}$, as (Kong et al., 1994):

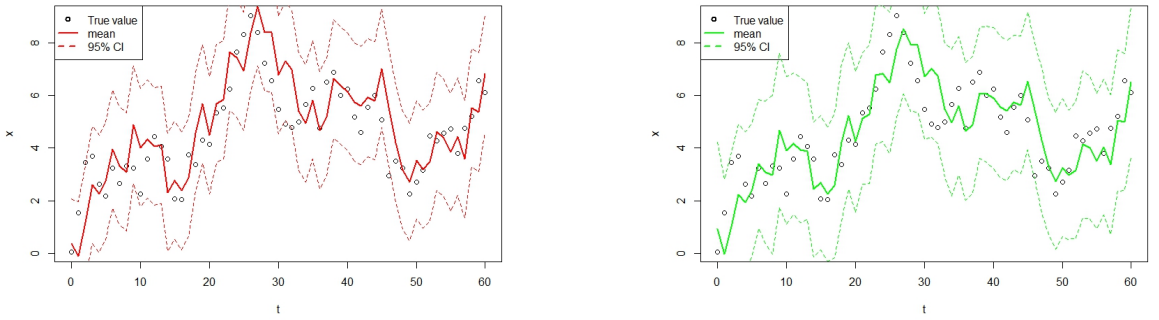
$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2}. \tag{18}$$

A small value of $\hat{N}_{eff}$ indicates severe degeneracy. Hence, a resampling step is only used when this value drops below a certain threshold. The algorithm is given in Doucet et al. (2000).

4.2 Example

The SIR filter can be applied to the same set of simulated latent process values and observations from the linear Gaussian model which was used for the Kalman filter example (section 2.3). Figure 3b shows the expected value and 95% credible intervals for the SIR filter with 500 particles, and where the transitional prior was used as the importance function. For comparison, the Kalman filter has also been included in Figure 3a. Figure 3 shows that the Kalman filter and the SIR filter have different means and credible intervals. However, both contain nearly all true latent values within the 95% credible intervals. The

Kalman filter is the optimal filter. This means the root mean squared error (RMSE) will on average be minimal for the Kalman Filter. With 500 particles, the SIR filter produces an equally low RMSE value for this simple example.



(a) The Kalman Filter output

(b) An SIR filter with $N = 500$

Figure 3: The mean and 95% credible intervals for the Kalman Filter and SIR filter with 500 particles where the observations are simulated from the linear Gaussian state space model, given by (12).

4.3 Resampling Schemes

Resampling removes particles with low importance weights and produces multiple copies of those with high importance weights. The approximated distribution produced by the particle filter before resampling is given by (Douc, 2005)

$$\hat{p}_N(x) = \sum_{i=1}^{N} w_i \delta(x - x_i),$$

and after resampling the approximated distribution is given by:

$$\tilde{p}_N(x) = \sum_{j=1}^{N} \frac{1}{N} \delta(x - \tilde{x}_j) = \sum_{i=1}^{N} \frac{n_i}{N} \delta(x - x_i),$$

where n_i is the number of copies of particle x_i in the new set, $\{\tilde{x}_j\}_{j=1}^{N}$. It has been proven (Douc, 2005) that convergence holds as long as the resampling scheme is 'close' to the original density. That is:

$$E\left[\left(\int g(x)\hat{p}_N(x)dx - \int g(x)\tilde{p}_N(x)dx\right)^2\right] \overset{N\to\infty}{\to} 0.$$

There are four commonly used resampling schemes within particle filtering literature. All these are unbiased, have $O(n)$ implementation and are based on a multinomial selection of N particles with replacement from the original particle set $\{x_i\}_{i=1}^{N}$ (Hol et al., 2006). The probability of selecting particle x_i is equal to w_i, that is $p(\tilde{x}_j = x_i) = w_i$ for $i, j = 1, \ldots, N$. The four methods can be summarised as (Hol et al., 2006):

1. **Multinomial resampling:** Call the selection of particle $\tilde{x}_j$ an event, which is the result of the j^{th}

experiment, $I_j = i$. Then the generation of I_j, with probability $p(I_j = i) = w_i$, is carried out by:

(a) Simulate a standard uniform random number $u_j \sim U(0,1)$

(b) Assign I_j value i according to $Q_{i-1} \leq u_j < Q_i$ where $Q_i = \sum_{s=1}^{i} w_s$.

This method uses the generalised inverse of the cumulative density function to map a standard uniform random number to the event.

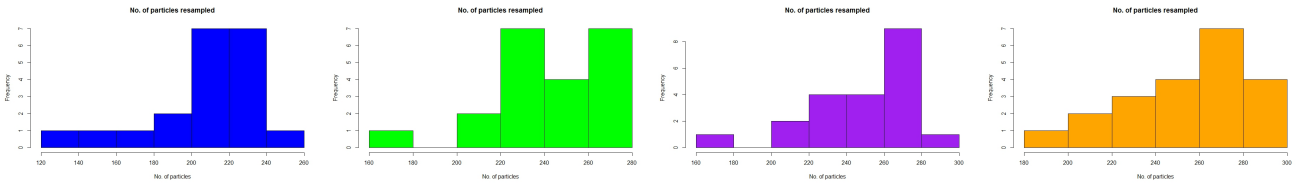
2. **Stratified resampling:** Generate N ordered random numbers, $u_j = \frac{(j-1)+\tilde{u}_j}{N}$ where $\tilde{u}_j \sim U(0,1)$, and then select the $\tilde{x}_j$ according to a multinomial distribution.

3. **Systematic resampling:** Generate N ordered numbers, $u_j = \frac{(j-1)+u}{N}$ where $u \sim U(0,1)$, and then select the $\tilde{x}_j$ according to a multinomial distribution.

4. **Residual resampling:** Allocate $\lfloor Nw_i \rfloor = n_i'$ copies of particle x_i to the new set of particles. The remaining $m = N - \sum_{i=1}^{N} n_i$ particles are then selected by one of the other resampling schemes, where the probability of selecting particle x_i is proportional to $Nw_i - n_i'$.

4.3.1 Comparison of Resampling Schemes

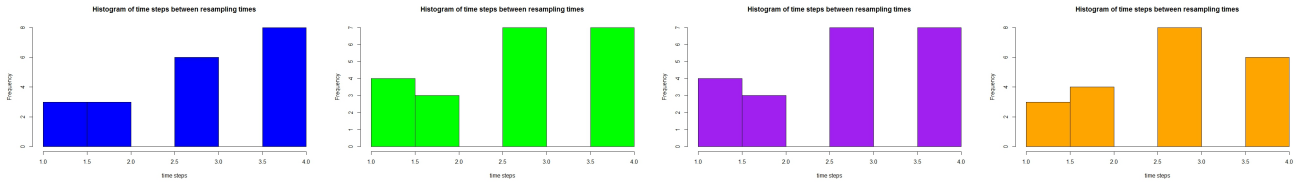
There are several factors which will lead to a good resampling scheme such as the number of unique particles selected at each resampling step or how often resampling needs to be carried out. The SIR filter with 500 particles and resampling when the effective sample size fell below 250 was run with different resampling schemes for the linear Gaussian model and simulated latent variables and observations given in section 2.3. Figure 4 shows the number of particles, $\{x_i\}$, which were chosen in each resampling step. Multinomial resampling, see Figure 4a, selected fewest particles with the majority between 180 and 240. The best performing was systematic resampling, see Figure 4d, where the majority of resampling steps selected were between 260 and 300 particles. Selecting more particles ensures that there is less particle degeneration and so the particle filter will perform better.



(a) Multinomial Resampling (b) Residual Resampling (c) Stratified Resampling (d) Systematic Resampling

Figure 4: The number of different particles that are selected at each resampling step, for the SIR filter applied to the linear Gaussian model given in (12) with 500 particles and different resampling schemes.

Another factor to consider when selecting a scheme is the number of steps between resampling. Figure 5 shows the time between resampling in the SIR filter for the four resampling schemes considered. The less often the resampling step takes place, the less Monte Carlo variation will be added. In all resampling schemes, the time between resampling is very similar.



(a) Multinomial Resampling (b) Residual Resampling (c) Stratified Resampling (d) Systematic Resampling

Figure 5: The time between resampling steps for the SIR filter applied to the linear Gaussian model given in (12) with 500 particles and different resampling schemes.

Furthermore, Figure 6 suggests that, for this model, all the resampling schemes have very similar effective sample sizes over time. Hol et al. (2006) shows that multinomial resampling involves the most computational effort of the resampling schemes, with the other three performing better. Considering all these results, multinomial resampling appears to perform worst whilst the other three are not easily distinguishable. The superior performance of systematic resampling for the number of particles carried forward suggests this may be the best resampling scheme of the four. A more extensive review of the resampling schemes can be found in Pollock (2010).

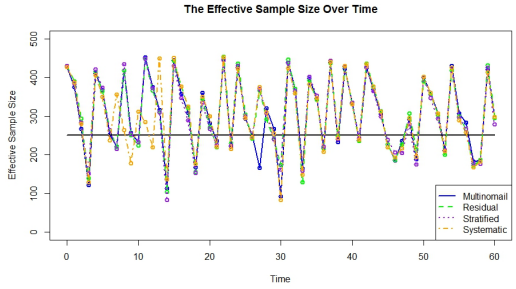


Figure 6: The effective sample size of the different resampling schemes over time for the SIR filter applied to the linear Gaussian model given in (12) with 500 particles and different resampling schemes.

4.4 The Auxiliary Particle Filter

One way to try and minimise degeneracy is to choose an importance function which is as close as possible to the optimal importance function, given by equation (17). A number of techniques have been developed to provide a good approximation. One of these is the auxiliary particle filter (APF). Any knowledge of the observation at time t can be incorporated into a well chosen importance function. Hence, particles are not just moving blindly into regions of the state space which are extremely unlikely given the observation, y_t.

The SIR filter resamples particles at the end of iteration $t-1$ before the observation y_t is seen, which is wasteful. The auxiliary particle filter, first introduced by Pitt and Shephard (1999), employs knowledge about the observation, y_t, before resampling at time $t-1$ so that particles, likely to be compatible with the observation, have a good chance of surviving. The idea of the APF is that the resampling step can be changed so that an auxiliary variable is sampled for each particle, which is weighted in terms of its

compatibility with coming observations. This weighting is given by:

$$p(y|x_{t-1}) = \int p(y_t|x_t)p(x_t|x_{t-1})dx_t, \tag{19}$$

or an approximation, $\hat{p}(y_t|x_{t-1})$, if (19) is not available directly (Whitely and Johansen, 2011). The new states are then resampled from the particles indicated by the auxiliary variables. This is the same as, at each iteration, resampling according to these weights and then carrying out the standard sampling and resampling steps. A similar approach, where the auxiliary weights are combined with standard weighting was proposed (Carpenter et al., 1999), which only involved a single resampling step in each iteration. The APF algorithm can be found in Whitely and Johansen (2011).

Other variance reduction methods are also available, for example Rao-Blackwellisation. Doucet et al. (2000) successfully applies Rao-Blackwellisation to state space models to obtain hybrid filters and Casella and Robert (1996) gives more general information on the Rao-Blackwellisation method.

5 Particle MCMC

So far, it has been assumed that the parameters of the distributions of the latent and observed processes are known. The joint distribution of the parameters, θ, the latent process and the observations, is given by:

$$p(\theta, x, y) = p(\theta)p(x|\theta)p(y|x, \theta). \tag{20}$$

Suppose θ is not known but is of interest; the required distribution is the posterior $p(\theta|y)$. Ideally, the latent process could be integrated out to calculate the marginal distribution:

$$p(y|\theta) = \int p(x|\theta)p(y|x, \theta)dx, \tag{21}$$

and then an Markov chain Monte Carlo(MCMC) algorithm can be used on the parameter space only. A new value, θ', can be proposed from the proposal distribution, $q(\theta'|\theta)$, and accepted with probability:

$$\min\left\{1, \frac{q(\theta|\theta')p(\theta')p(y|\theta')}{q(\theta'|\theta)p(\theta)p(y|\theta)}\right\}. \tag{22}$$

To implement this, the distribution $p(y|\theta)$ is needed, which is often intractable. It can be approximated by using IS and this approximation, $\hat{p}(y|\theta)$, is used instead. There are two natural ways of implementing this, given by O'Neill et al. (2000) and Beaumont (2003). If an unbiased estimate of $p(y|\theta)$ is used, the second of these algorithms was shown by Andrieu and Roberts (2009) to sample from the true posterior and was called the pseudo marginal method. These methods are known as 'likelihood free' MCMC. When the model is a state space model, the particle filter can be used to estimate the distribution, $\hat{p}(y|\theta)$, which is unbiased.

Andrieu et al. (2010) showed that there is a lot more flexibility for using particle filters and MCMC, called particle MCMC algorithms (PMCMC). They give exact approximations to the idealised method given by (21) and (22). Furthermore, they can be interpreted as standard MCMC updates and so lead to convergent algorithms under mild assumptions, see Andrieu et al. (2010). With an infinite number of particles, PMCMC can be seen as the same as the idealised method, whereas with one particle, it can be viewed as a likelihood free method (Wilkinson, 2011). Hence providing a compromise between the idealised scheme and the 'likelihood free' scheme.

The simplest of these PMCMC algorithms is the particle independent Metropolis-Hastings (PIMH) update which approximates the distribution of $p(x_{0:T}|y_{0:T},\theta)$ and hence $p(y_{0:T}|\theta)$. However, the PIMH algorithm alone does not prove to be a serious competitor for the standard sequential Monte Carlo methods used to find them (Andrieu et al., 2010). The particle marginal Metropolis Hastings (PMMH) algorithm, which can be thought of as an exact approximation to the marginal Metropolis Hastings (MMH) update, can be used when the distribution of interest is $p(\theta, x_{0:T}|y_{0:T})$. The PMMH algorithm jointly updates θ and $x_{0:T}$. Assume sampling from the conditional density $p(x_{0:T}|y_{0:T},\theta)$ is feasible for any $\theta \in \Theta$ and:

$$p(\theta, x_{0:T}|y_{0:T}) = p(\theta|y_{0:T})p(x_{0:T}|y_{0:T}). \tag{23}$$

It is natural to suggest a proposal density for the MH update of:

$$q(\{x'_{0:T}, \theta'\} \mid \{x_{0:T}, \theta\}) = q(\theta'|\theta)p(x_{0:T}|y_{0:T},\theta'), \tag{24}$$

so that the only degree of freedom in the algorithm is $q(\theta'|\theta)$. The MH acceptance ratio is therefore given by:

$$\min\left\{1, \frac{p(y_{0:T}|\theta')p(\theta')q(\theta|\theta')}{p(y_{0:T}|\theta)p(\theta)q(\theta'|\theta)}\right\}. \tag{25}$$

The PMMH algorithm, see Algorithm 4, leaves $p(\theta, x_{0:T}|y_{0:T})$ invariant and, under weak assumptions, the PMMH sampler is ergodic (Andrieu et al., 2010). An alternative to the PMMH algorithm is the particle Gibbs sampler, which iterates between sampling a new θ from $p(\theta|x_{0:T}, y_{0:T})$ and $x_{0:T}$ from $p(x_{0:T}|y_{0:T},\theta)$. This removes the need to design a proposal distribution for θ, which is necessary in the PMMH sampler.

Since the introduction of particle MCMC methods, they have been applied to a range of fields including financial econometrics (Lopes and Tsay, 2011), graphics processing units (Henriksen et al., 2012), multitarget tracking (Bocquel et al., 2012) and population dynamics (Gao et al., 2012). Section 6.2 looks at applying the PMMH algorithm to reaction networks, to learn about the rate parameters of the latent process.

However, these methods are still computationally intensive. For each iteration of the algorithm a particle filter must be run and the performance of the particle MCMC algorithm depends heavily on the variance of the estimate to the normalising constant, $p(y|\theta)$. It is still an open question as to how many particles should be used in the particle filter for PMCMC. It affects the performance and so choosing the right

Algorithm 4 The Particle MCMC Algorithm

Initialise: for $i = 0$

1. Set $\theta(0)$ arbitrarily
2. Run a particle filter to give $\hat{p}(x_{0:T}|y_{0:T}, \theta(0))$ an unbiased estimate to $p(x_{0:T}|y_{0:T}, \theta(0))$
   (a) Sample $X_{0:T}(0) \sim \hat{p}(\cdot|y_{0:T}, \theta(0))$
   (b) Calculate $Z(0) = \hat{p}(y_{0:T}|\theta(0))$

Iterate: For $i \geq 1$

1. Sample $\theta' \sim q(\cdot|\theta(i-1))$
2. Run particle filter
   (a) Sample $X'_{0:T} \sim \hat{p}(\cdot|y_{0:T}, \theta')$
   (b) $Z' = \hat{p}(y_{0:T}|\theta)$
3. With probability

$$\min \left\{1, \frac{p(y_{0:T}|\theta')p(\theta')q(\theta|\theta')}{p(y_{0:T}|\theta)p(\theta)q(\theta'|\theta)}\right\}$$

Accept $(x_{0:T}(i), \theta(i)) = (x'_{0:T}, \theta')$ and $Z(i) = Z'$;
Else $(x_{0:T}(i), \theta(i)) = (x_{0:T}(i-1), \theta(i-1))$ and $Z(i) = Z(i-1)$.

number of particles requires a balance between having a reasonable acceptance rate and minimising the computational time. In section 6.2.1, the best number of particles to use is discussed within an example of PMCMC in reaction networks. Many other techniques which have been applied to SMC methods could also be applied to PMCMC as a way of improving their efficiency (Jacob et al., 2009).

6 Particle Filters and Reaction Networks

In mass action stochastic kinetics, the state of the system at a given time is given by the number of each chemical species $X_t = (X_1, \ldots, X_u)$ present at that time, where there are u species in the system. The state of the system is changed at discrete times according to the set of reactions $(R_1, \ldots, R_v)$. If reaction R_i takes place, the state is updated according to the i^{th} column of the stoichiometry matrix, S. Where the stoichiometry matrix gives the difference between the number of each species going into a reaction compared with the number of each leaving. Each reaction has a rate constant c_i for, $i = 1, \ldots, v$, associated with it, and a corresponding hazard function, $h_i(x_t, c_i)$, which gives the overall hazard of reaction i taking place. Given an initial value x_0 of the system and the hazard functions, discrete event simulation methods can be used to forward simulate exact realisations of the Markov process, often using the Gillespie algorithm (Gillespie, 1977). For more information of mass action stochastic kinetics see Wilkinson (2006).

The Lotka Volterra model is a simple reaction network which looks at the interaction between two species. These species are considered to represent prey, X_1 and predators, X_2. It consists of three reactions, which

can be modelled by the equations:

$$X_1 \xrightarrow{c_1} 2X_1, \tag{26}$$

$$X_1 + X_2 \xrightarrow{c_2} 2X_2, \tag{27}$$

$$X_2 \xrightarrow{c_3} 0. \tag{28}$$

Reaction (26) represents prey reproduction, reaction (27) represents predator-prey interaction, where the loss of a prey results in the reproduction of a predator and finally reaction (28) represents the death of a predator. Each reaction has a constant reaction rate associated with it, c_i, $i = 1, 2, 3$.

6.1 The SIR filter and Lotka Volterra Model

The SIR particle filter can be used to estimate the true states of the reaction system, given imprecise observations of predators and prey at discrete time points. The true latent process has been simulated using the Gillespie algorithm with initial values $x_0 = (50, 100)$ and rate constants $c = (1, 0.005, 0.6)$, given in Wilkinson (2006) and the value recorded at integer time points. These true values are shown by the black and red points on Figure 7, for the prey and predators respectively. The observations, Y_t for $t = 1, \ldots, 60$, were taken to be:

$$Y_{t,i} \sim \begin{cases} \text{Poisson}(X_{t,i}) & \text{for } i = 1, 2,\ t = 0, \ldots, 60 \text{ for } X_t \neq 0 \\ \text{Bernoulli}(1/2) & \text{for } i = 1, 2,\ t = 0, \ldots, 60 \text{ for } X_t = 0 \end{cases},$$

where $Y_{t,i}$ is the number of species i observed at time t and $X_{t,i}$ is the true number of species i at time t.

An SIR filter, with 100 particles, can then be used to estimate the latent process from these observations where the importance function is taken as the transitional prior and resampling is carried out if the effective sample size drops below 50. The particles are propagated forward to the next time step using the Gillespie algorithm. This estimated value is shown by the black and red lines for the predators and prey respectively in Figure 7. For the set of observations, the true latent value is very close to the value estimated using the SIR filter with 100 particles.

6.2 Particle MCMC for Inference in Reaction Networks

In section 6.1, the SIR filter was used to predict the true values of the LV output, given noisy observations. The rate constants, $c_i,\ i = 1, 2, 3$, were assumed to be known. Often, these parameters are unknown and the aim is to learn about them given the observations. Particle MCMC methods have been successfully applied to reaction network inference to solve these types of problems (Golightly and Wilkinson, 2011). Furthermore, they do not suffer from the curse of dimensionality or the particle degeneracy problems that sequential likelihood free MCMC methods, an alternative method for finding the rate parameters, do.

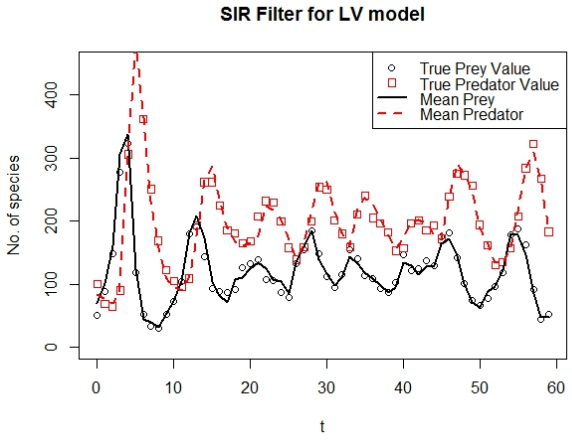


Figure 7: The SIR filter with 100 particles and resampling when $\hat{N}_{eff} < 50$ for the Lotka Volterra model. The true species numbers, simulated using the Gillespie algorithm, are given by the black and red points for prey and predators respectively and their estimated values are shown by the black and red lines.

6.2.1 Lotka Volterra Example

Consider the Lotka Volterra model, discussed in section 6. Given a set of observations, the aim is to determine the rate parameters which produced these observations. To do this, a PMMH algorithm can be used. The Gillespie algorithm was used to simulate 20 observations from the LV model, at integer time points, with $x_0 = (50, 100)$ and $c = (1, 0.005, 0.6)$. The observations were given by a Gaussian distribution (Golightly and Wilkinson, 2011) but with mean $X_{t,i}$ and standard deviation 15:

$$Y_{t,i} \sim N(X_{t,i}, 15^2).$$

Figure 8 shows the observed values and the true values of the Lotka Volterra model used. A PMMH algorithm using a simple Metropolis Hastings update for the parameters, from the smfsb R package (Wilkinson, 2012), was then run on the observations. In this example, the standard deviation was taken as known, although the package does contain an option to include this as one of the parameters to be estimated.

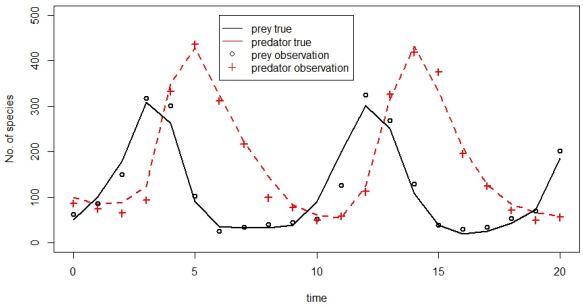
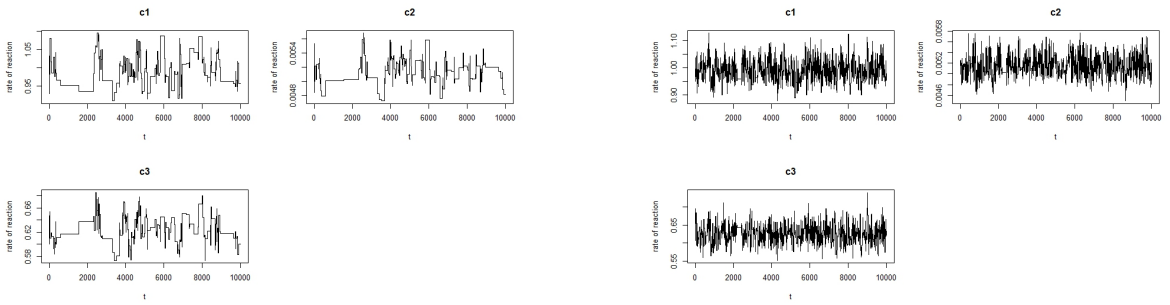


Figure 8: The true latent variables and observations used in the PMMH algorithm to find the rate parameters. The true rate parameters are $c = (1, 0.005, 0.6)$ and initial condition $x_0 = c(50, 100)$.

Figure 9 shows the output for the three rate parameters, when a particle filter with 25 particles was used, see Figure 9a, and when 150 particles was used, see Figure 9b. The initial values of the PMMH algorithm were set to the true parameters values so burn in was not necessary. Comparing the output from these two results, it is clear that with only 25 particles the chain is sticky, whereas, with 150, mixing is good. For the PMMH run with 25 particles, the estimates of the parameters are given by $\bar{c} = (0.9857, 0.005117, 0.6238)$ and the ESS ≈ 40 whilst for the PMMH run with 150 particles, the estimates of the parameters are given by $\bar{c} = (0.9900, 0.005147, 0.6255)$ and ESS ≈ 320. Although both PMMH schemes give reasonable estimates of the true parameter values, the PMMH algorithm run with 150 particles has a far larger effective sample size.

Ideally for the PMMH, it would be beneficial to use the number of particles which ensures that the effective sample size (ESS) is as large as possible, but which minimises the computational time. The computational time will be directly related to the number of particles used. Therefore, to find the optimal number of particles, the effective sample size divided by the number of particles (ESS/N) was considered. Figure 10 shows this value for different numbers of particle, where the ESS is averaged over three runs of the PMMH with $10,000$ iterations as well as the three rate parameters. Figure 10 suggests the optimal number of particles is around $N = 75$, although $N = 50$ and $N = 100$ particles also work well. Beyond this, the increases in N does not have as significant effect on the effective sample size compared to the increase in computational time.



(a) $N = 25$ (b) $N = 150$

Figure 9: The output from the PMMH algorithm with $10,000$ iterations and N particles. The initial values were set to the true rate parameter values, $c = (1, 0.005, 0.6)$.

One way the computational time could be minimised further would be to used delayed acceptance PM-CMC. This has been applied to reaction networks (Golightly et al., 2012) where the marginal likelihood is first estimated using the chemical Langevin equation diffusion approximation or the linear noise approximation, and only if the proposal is accepted under this estimation, is the particle filter run.

7 Conclusion

This report has looked at introducing particles filters and particle MCMC. Section 2 looks at a special class of particle filter, the Kalman filter, which can be used to optimally solve linear Gaussian state space models. Before the introduction of the bootstrap filter (Gordon et al., 1993) the Kalman filter and its variants were used, but these provide poor estimates for highly non linear and non Gaussian problems

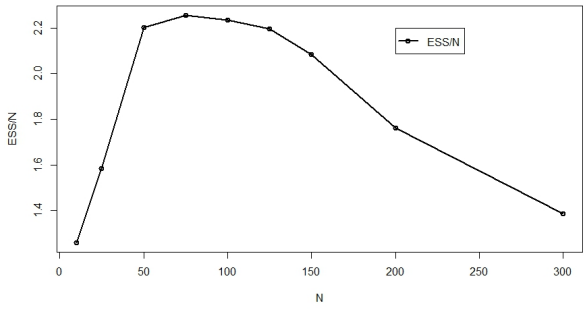


Figure 10: A graph showing the average effective sample size divided by the number of particles of the 3 rate parameters given 3 runs of the PMMH algorithm with 10000 iterations on the LV model shown in Figure 8.

and SMC methods are needed, including sequential importance sampling and sequential importance resampling. The SIS algorithm discussed in section 3 uses importance sampling to provide a weighted particle approximation to marginal distribution of the latent variables given the observations. Degeneracy is a major problem in the SIS algorithm, and within even a few time steps nearly all particles will have insignificant weights. Both a good choice of importance function and adding a resampling step into the algorithm can negate degeneracy. The SIR algorithm provides stability to the particle filtering algorithm. However, resampling can lead to sample impoverishment and so only sampling when the effective sample size drops below a threshold is recommended. Results from section 4.3 suggest that systematic resampling is the best resampling scheme for the SIR filter, out of the commonly used resampling schemes.

Section 5 introduces particle MCMC methods, which combine particle filtering and MCMC. They provide a way of estimating the latent parameters given the observations. First introduced by Andrieu et al. (2010), they have since been applied to a range of problems. Section 6.2 looks at how the PMMH can be applied to reaction networks to learn about the rate parameters. Although PMCMC has successfully been applied to several different problems they are still computationally intensive methods. Most of this computational time comes from applying a particle filter at each step of the algorithm. Further work is still needed in particle MCMC to minimise this computational time, a play off between acceptance rate and number of particles. Other methods, for example good proposal distribution and delayed acceptance particle MCMC, can be used.

References

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342.

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

Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188.

Beaumont, M. A. (2003). Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bocquel, M., Driessen, H., and Bagchi, A. (2012). Multitarget tracking with interacting population-based MCMC-PF. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 74–81. IEEE.

Carpenter, J., Clifford, P., and Fearnhead, P. (1999). Improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation*, 146(1):2–7.

Casella, G. and Robert, C. P. (1996). Rao-blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94.

Djuric, P. M. and Bugallo, M. F. (2009). Estimation of stochastic rate constants and tracking of species in biochemical networks with second-order reactions.

Douc, R. (2005). Comparison of resampling schemes for particle filtering. In *In 4th International Symposium on Image and Signal Processing and Analysis (ISPA*, pages 64–69.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statisitcs and Computing*, 10(3):197–208.

Doucet, A. and Johansen, A. M. (2008). A Tutorial on Particle Filtering and Smoothing: Fifteen years later. Technical report, Institute of Statistical Mathematics, Japan and Department of Statistics, University of Warwick.

Gao, M., Chang, X., and Wang, X. (2012). Bayesian parameter estimation in dynamic population model via particle Markov chain Monte Carlo. *Computational Ecology and Software*, 2(4):181–197.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361.

Golightly, A., Henderson, D. A., and Sherlock, C. (2012). Efficient particle MCMC for exact inference in stochastic biochemical network models through approximation of expensive likelihoods.

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., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113.

Hammersley, J. M. and Morton, K. W. (1954). Poor Man's Monte Carlo. *Journal of the Royal Statistical Society. Series B (Methodological)*, 16(1):23–38.

Henriksen, S., Wills, A., Schön, T. B., and Ninness, B. (2012). Parallel Implementation of Particle MCMC Methods on a GPU. In *System Identification*, volume 16, pages 1143–1148.

Hol, J., Schon, T., and Gustafsson, F. (2006). On Resampling Algorithms for Particle Filters. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 79–82.

Jacob, P., Chopin, N., Robert, C. P., and Rue, H. (2009). Comments on "Particle Markov chain Monte Carlo" by C. Andrieu, A. Doucet, and R. Hollenstein. *arXiv preprint arXiv:0911.0985*.

Johannes, M. S., Polson, N. G., and Stroud, J. R. (2009). Optimal filtering of jump diffusions: Extracting latent states from asset prices. *Review of Financial Studies*, 22(7):2759–2799.

Julier, S. and Uhlmann, J. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Julier, S., Uhlmann, J., and Durrant-Whyte, H. (1995). A new approach for filtering nonlinear systems. *Proceedings of 1995 American Control Conference ACC95*, 3(3):1628–1632.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Kim, S., Shephard, N., and Chib, S. (1998). Stochastic volatility: likelihood inference and comparison with ARCH models. *The Review of Economic Studies*, 65(3):361–393.

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.

Lopes, H. F. and Tsay, R. S. (2011). Particle filters and Bayesian inference in financial econometrics. *Journal of Forecasting*, 30(1):168–209.

Maybeck, P. S. (1982). *Stochastic models, estimation, and control*. Academic press.

Merwe, R., Doucet, A., de Freitas, N., and Wan, E. (2000). The Unscented Particle Filter. Technical report, Cambridge University Engineering Department.

O'Neill, P. D., Balding, D. J., Becker, N. G., Eerola, M., and Mollison, D. (2000). Analyses of Infectious Disease Data from Household Outbreaks by Markov chain Monte Carlo Methods. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 49(4):517–542.

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

Pollock, M. (2010). Introduction to Particle Filtering Discussion.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyond the Kalman Filter: Particle filters for tracking applications*. Artech House.

Salimpour, Y. and Soltanian-Zadeh, H. (2009). Particle filtering of point processes observation with application on the modeling of visual cortex neural spiking activity. In *Neural Engineering, 2009. NER '09. 4th International IEEE/EMBS Conference on*, pages 718–721.

Welch, G. and Bishop, G. (1996). An Introduction to the Kalman Filter. Technical report, University of North Carolina.

Whitely, N. and Johansen, A. M. (2011). Auxillary Particle Filtering: Recent Developments. In *Bayesian Time Series Models*. Cambridge University Press.

Wilkinson, D. (2006). *Stochastic Modelling for Systems Biology*. Taylor and Francis, London, 1st edition.

Wilkinson, D. (2011). Darren wilkinson's research blog: The particle marginal metropolis-hastings (pmmh) particle mcmc algorithm. `http://darrenjw.wordpress.com/2011/05/17/the-particle-marginal-metropolis-hastings-pmmh-particle-mcmc-algorithm/`. Accessed: 09/05/2013.

Wilkinson, D. (2012). *smfsb: SMfSB 2e: Stochastic Modelling for Systems Biology, second edition*. R package version 1.1/r62.

Zaritskii, V., Svetnik, V., and Shimelevich, L. (1976). Monte-carlo technique in problems of optimal information processing. *Automation and Remote Control*, 36(12):2015–2022.