Particle Filtering

Michael Johannes and Nicholas Polson

Abstract This chapter provides an overview of particle filters. Particle filters generate approximations to filtering distributions and are commonly used in non-linear and/or non-Gaussian state space models. We discuss general concepts associated with particle filtering, provide an overview of the main particle filtering algorithms, and provide an empirical example of filtering volatility from noisy asset price data.

1 Introduction

Filtering generally refers to an extraction process, and statistical filtering refers to an algorithm for extracting a latent state variable from noisy data using a statistical model. The original filtering applications were in physical systems, for example, tracking the location of an airplane or missile using noisy radar signals, but filtering quickly became a crucial theoretical and empirical tool in economics and finance due to widespread use of models incorporating latent variables.

Latent variables capture unobserved changes in the economic environment. In many cases, there is clear evidence for time-variation, but the underlying causes are unknown or difficult to quantify. For example, in finance, it is clear that asset return volatility time-varies, but it is difficult to identify factors generating the variation. To capture the variation, common models assume that volatility is unobserved but evolves through time as a stochastic

Michael Johannes
Graduate School of Business, Columbia University, 3022 Broadway, NY, 10027, e-mail: mj335@columbia.edu

Nicholas Polson
Graduate School of Business, University of Chicago, 5807 S. Woodlawn, Chicago IL 60637, e-mail: ngp@chicagogsb.edu

T.G. Anderson et al., *Handbook of Financial Time Series*,
DOI: 10.1007/978-3-540-71297-8_44,

process. In macroeconomics, a similar challenge occurs when modeling time-varying components that drive aggregates. For example, the key component in "long-run risk" models is a highly persistent unobserved variable driving consumption and dividend growth rates.

Given the widespread use of latent variables in economic models, a central challenge is to develop statistical methods for estimating the latent variables. This chapter discusses a computational approach for filtering known as the particle filter.

To understand the nature of the filtering problem and why particle filters are useful, the formal problem is defined as follows. A statistical model generates the observed data, a vector y_t, and the conditional distribution of y_t depends on a latent state variable, x_t. Formally, the data is generated by the state space model, which consists of the observation and state evolution equations,

$$\begin{aligned} \text{Observation equation: } y_t &= f\left(x_t, \varepsilon_t^y\right) \\ \text{State evolution: } x_{t+1} &= g\left(x_t, \varepsilon_{t+1}^x\right), \end{aligned}$$

where ε_{t+1}^y is the observation error or "noise," and ε_{t+1}^x are state shocks. The observation equation is often written as a conditional likelihood, $p\left(y_t|x_t\right)$, and the state evolution as $p\left(x_{t+1}|x_t\right)$. Both of these distributions typically depend on static parameters, θ, whose dependence is suppressed, except where explicitly noted.

The posterior distribution of x_t given the observed data, $p\left(x_t|y^t\right)$, solves the filtering problem, where $y^t = (y_1, ..., y_t)$ is the observed data. Beginning with Kalman's filter, computing $p\left(x_t|y^t\right)$ uses a two-step procedure of prediction and Bayesian updating. The prediction step combines the current filtering distribution with the state evolution,

$$p\left(x_{t+1}|y^t\right) = \int p\left(x_{t+1}|x_t\right) p\left(x_t|y^t\right) dx_t, \tag{1}$$

providing a forecast of next period's state. Next, given a new observation, y_{t+1}, the predictive or "prior" views are updated by Bayes rule

$$\underbrace{p\left(x_{t+1}|y^{t+1}\right)}_{\text{Posterior}} \propto \underbrace{p\left(y_{t+1}|x_{t+1}\right)}_{\text{Likelihood}} \underbrace{p\left(x_{t+1}|y^t\right)}_{\text{Prior}}. \tag{2}$$

The problem is that $p\left(x_t|y^t\right)$ is known analytically only in a limited number of settings, such as a linear, Gaussian model where $p\left(y_t|x_t\right) \sim \mathcal{N}\left(x_t, \sigma^2\right)$ and $p\left(x_{t+1}|x_t\right) \sim \mathcal{N}\left(x_t, \sigma_x^2\right)$. In this case, the Kalman filter implies that $p(x_t|y^t) \sim N\left(\mu_t, \sigma_t^2\right)$, where μ_t and σ_t^2 solve the Kalman recursions. In nonlinear or non-normal models, it is not possible to analytically compute $p(x_t|y^t)$. In these settings, $p\left(x_t|y^t\right)$ is a complicated function of y^t, and simulation methods are typically required to characterize $p\left(x_t|y^t\right)$.

The particle filter is the most popular approach. A particle filter simulates approximate samples from $p(x_t|y^t)$, which are used for Monte Carlo integration to estimates moments of interest such as $E[f(x_t)|y^t]$. Particle filters use a discrete approximation to $p(x_t|y^t)$ consisting of states or "particles", $\left\{x_t^{(i)}\right\}_{i=1}^N$, and weights associated with those particles, $\left\{\pi_t^{(i)}\right\}_{i=1}^N$. A particle approximation is just a random histogram.

Recursive sampling is the central challenge in particle filtering: given a sample from $p^N(x_t|y^t)$, how to generate a random sample from the particle approximation to $p(x_{t+1}|y^{t+1})$ after receiving a new data point y_{t+1}? Essentially this is a problem of using a discrete approximation to the integral in (1) and then sampling from $p^N\left(x_{t+1}|y^{t+1}\right)$. Since there are multiple ways to sample from a given distribution, there are many different particle filters. For example, importance sampling is commonly used to sample from non-standard distributions, and different importance densities generate different particle filters.

This chapter provides an introduction to these particle filters, and outlines a number of the most common algorithms. Before delving into details, it is important to understand the two main reasons why particle filters are so popular. First, particle filters are very flexible and adaptable. Like all Monte Carlo methods, particle filters can be adapted to the particular model specification under consideration. In particular, it is possible to develop accurate filters for non-linear models with fat-tailed and asymmetric error distributions. These are particularly important for applications where errors are often not normally distributed. Second, particle filters are easy to program and computationally very fast to run, in terms of computing time. For these reasons, particle filters provide an attractive filtering methodology.

2 A Motivating Example

The following provides a common setting in which particle filters are useful. Consider a simple log-stochastic volatility model

$$\begin{aligned} y_{t+1} &= \sqrt{V_{t+1}}\varepsilon_{t+1}^y \\ \log(V_{t+1}) &= \alpha_v + \beta_v \log(V_t) + \sigma_v \varepsilon_{t+1}^v, \end{aligned}$$

where $\varepsilon_{t+1}^y$and ε_{t+1}^y are independent standard normal and V_t is the conditional variance. Again, the parameters are assumed to be known. This is a benchmark specification for modeling time-varying volatility. The top panel of Figure 1 provides a simulated sample path of returns from the specification with $\alpha_v = 0$, $\beta_v = 0.95$, and $\sigma_v = 0.10$. By merely observing the data, it is clear that the conditional volatility time varies, as the amplitude of the fluctuations vary over time.

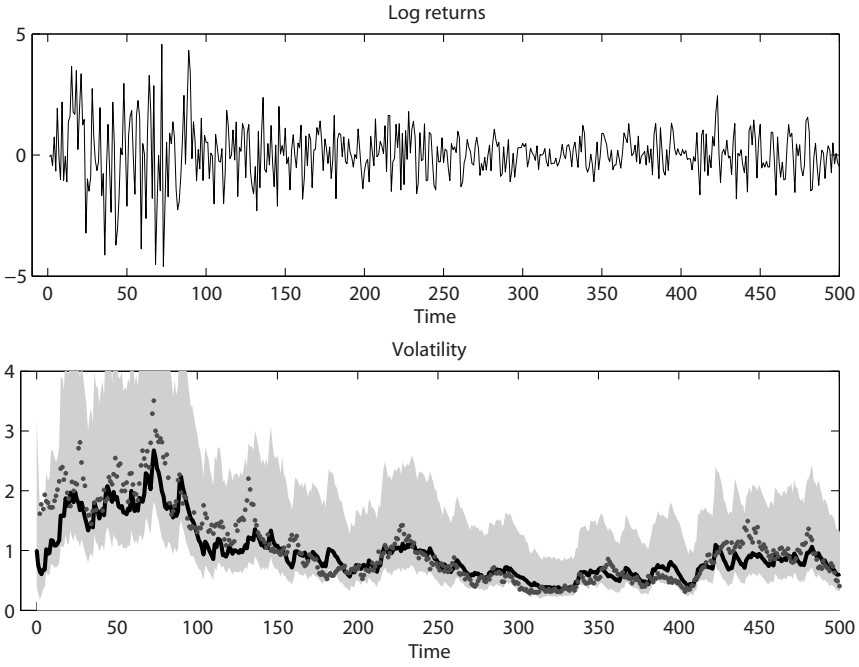

Fig. 1 Simulated returns (top panel) and summaries of the filtering distribution through time (bottom panel) for a logarithmic stochastic volatility model.

This model has conditionally normal errors, but is nonlinear, because of the term $\sqrt{V_{t+1}}\varepsilon_{t+1}^{y}$. Alternatively, the observation equation can be written as

$$\log\left(y_{t+1}\right)^{2} = \log\left(V_{t+1}\right) + \log\left(\varepsilon_{t+1}^{y}\right)^{2},$$

which leads to a model that is linear in $log\left(V_{t+1}\right)$, but is now non-normal because $\log\left(\varepsilon_{t+1}^{y}\right)^{2}$ is $\mathcal{X}^{2}$ with one degree of freedom.

Given this structure, how can V_t be filtered from the observed data? One approach would be to ignore the nonlinearity/non-normality and use the Kalman filter. Alternatively, the model could be linearized using the extended Kalman filter. Both of these approaches are inefficient and biased. Another alternative would be to use deterministic numerical integration to compute the integral in equation (1) and characterize $p\left(V_t|y^t\right)$. This is computationally more difficult, but may work satisfactorily in some settings. In higher dimensional settings, these deterministic numerical integration schemes suffer from Bellman's curse of dimensionality.

Alternatively, the particle filter, as described in the following sections, can be used. The results of the particle filter are displayed in the bottom panel of Figure 1. Here, the true states are given by the dotted line, filtered estimates,

$E\left(\sqrt{V_t}|y^t\right)$, by the solid line, and estimates of the $(5\%, 95\%)$ quantiles in the shades. These are all naturally computed as output from the particle filter.

3 Particle Filters

A particle filter is a discrete approximation, $p^N\left(x_t|y^t\right)$, to $p\left(x_t|y^t\right)$, generally written as $\left\{\pi_t^{(i)}, x_t^{(i)}\right\}_{i=1}^N$, where the weights sum to one, $\sum_{i=1}^N \pi_t^{(i)} = 1$. The support of the discrete approximation, the $x_t^{(i)}$'s, is not preset as would be the case in deterministic approximation schemes, but rather is the stochastic outcome of a simulation algorithm. Thus, the support of the distribution, the $x_t^{(i)}$'s, change from period to period. Thus, a generic particle approximation is given by

$$p^N\left(x_t|y^t\right) = \sum\nolimits_{i=1}^N \pi_t^{(i)} \delta_{x_t^{(i)}},$$

where δ is the Dirac function.

The particle approximation can be transformed into an equally weighted random sample from $p^N\left(x_t|y^t\right)$ by sampling, with replacement, from the discrete distribution, $\left\{\pi_t^{(i)}, x_t^{(i)}\right\}_{i=1}^N$. This procedure, called *resampling*, produces a new sample with uniformly distributed weights, $\pi_t^{(i)} = 1/N$. Resampling can be done in many ways, but the simplest is multinomial sampling. Other methods include stratified sampling, residual resampling, and systematic resampling.

The discrete support of particle filters makes numerical integration easy because integrals becomes sums. For example,

$$p^N\left(x_{t+1}|y^{t+1}\right) \propto \int p\left(y_{t+1}|x_{t+1}\right) p\left(x_{t+1}|x_t\right) p^N\left(x_t|y^t\right) dx_t \tag{3}$$

$$\propto \sum_{i=1}^N p\left(y_{t+1}|x_{t+1}\right) p\left(x_{t+1}|x_t^{(i)}\right) \pi_t^{(i)}, \tag{4}$$

where the proportionality sign, '$\propto$,' signifies that the normalizing constant does not depend on x_{t+1}. Given the discretization, the central problem in particle filtering is how to generate a sample from $p^N\left(x_{t+1}|y^{t+1}\right)$. Equation (3) implies that $p^N\left(x_{t+1}|y^{t+1}\right)$ is a finite mixture distribution, and different sampling methods generate alternative particle filtering algorithms, each with their own strengths and weaknesses.

In general, there are two sources of approximation errors in particle filtering algorithms. Approximating $p\left(x_t|y^t\right)$ by $p^N\left(x_t|y^t\right)$ generates the first source of error. This is inherent in all particle filtering algorithms, but can be mitigated by choosing N large. Importance sampling or other approximate

sampling methods generate the other source of error, which is present in some particle filtering algorithms. Importance sampling generates an approximate sample from $^{N}(x_t|y^t)$, which in turn approximates $p(x_t|y^t)$. This leads to a second layer of approximation errors.

It is useful to briefly review common uses of the output of particle filtering algorithms. The main use is to estimate latent states. This is done via Monte Carlo. A particle approximation of $E(f(x_t)|y^t)$ is

$$E^N\left(f(x_t)|y^t\right) = \int f(x_t)\, p^N\left(x_t|y^t\right) dx_t = \sum_{i=1}^{N} f\left(x_t^{(i)}\right)\pi_t^{(i)}.$$

As N becomes large, the particle estimates converge by the law of large numbers and a central limit theorem is typically available, both using standard Monte Carlo convergence results.

The filtering distribution is useful for a likelihood based parameter estimation and model comparison. Although in the rest of the chapter we assume parameters are known, a central problem in state space models is estimating the parameters, θ. In the case when parameters are unknown, the particle filter can be used to compute the likelihood function. The likelihood of the observed sample is denoted as $\mathcal{L}\left(\theta|y^T\right)$. In time series models, the likelihood is given by

$$\mathcal{L}\left(\theta|y^T\right) = \prod_{t=0}^{T-1} p\left(y_{t+1}|\theta, y^t\right).$$

In latent variable models, the predictive distribution of the data $p(y_{t+1}|y^t,\theta)$ is not generally known, but rather given as an integral against the filtering distribution:

$$p\left(y_{t+1}|y^t,\theta\right) = \int p(y_{t+1}|\theta,x_{t+1})\, p(x_{t+1}|\theta,x_t)\, p\left(x_t|\theta,y^t\right) dx_t dx_{t+1},$$

where $p(y_{t+1}|\theta,x_{t+1})$ is the conditional likelihood, $p(x_{t+1}|\theta,x_t)$ is the state evolution, and $p(x_t|\theta,y^t)$ is the filtering distribution, all conditional on the unknown parameters. Given a particle approximation to $p(x_t|y^t,\theta)$, it is straightforward to approximate the predictive likelihoods, and therefore to estimate parameters or compare models with likelihood ratios. For the rest of the chapter, we suppress the dependence on the parameters.

The rest of the chapter discusses three common prominent particle filtering algorithms. For parsimony, focus is restricted to particle methods for approximating the filtering distribution, $p(x_t|y^t)$, and we do not discuss methods such as sequential importance sampling (SIS), that generate samples sequentially from the smoothing distribution, $p(x^t|y^t)$, where $x^t = (x_1, ..., x_t)$.

3.1 Exact particle filtering

The easiest way to understand particle filtering is to consider situations in which importance sampling is not required, because direct i.i.d. sampling from $p^N\left(x_{t+1}|y^{t+1}\right)$ is feasible. This is called exact sampling, and leads to an exact particle filter. To see how this works, first note that

$$p\left(y_{t+1}, x_{t+1}|x_t\right) \propto p\left(y_{t+1}|x_t\right) p\left(x_{t+1}|x_t, y_{t+1}\right), \tag{5}$$

which implies that the filtering recursion can be expressed as

$$p\left(x_{t+1}|y^{t+1}\right) \propto \int p\left(y_{t+1}|x_t\right) p\left(x_{t+1}|x_t, y_{t+1}\right) p\left(x_t|y^t\right) dx_t, \tag{6}$$

where $p\left(y_{t+1}|x_t\right)$ is the predictive likelihood and $p\left(x_{t+1}|x_t, y_{t+1}\right)$ is the posterior distribution of the new state given the previous state and the new observation.

This representation generates a different mixture distribution for $p^N\left(x_{t+1}|\right.$ $\left.y^{t+1}\right)$ than the one commonly used in particle filtering algorithms, which is given in 3. Given a particle approximation to $p^N\left(x_t|y^t\right)$,

$$\begin{aligned} p^N\left(x_{t+1}|y^{t+1}\right) &\propto \sum_{i=1}^{N} p\left(y_{t+1}|x_t^{(i)}\right) p\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right) \\ &= \sum_{i=1}^{N} w_t^{(i)} p\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right), \end{aligned} \tag{7}$$

where the normalized first stage weights are

$$w_t^{(i)} = \frac{p\left(y_{t+1}|x_t^{(i)}\right)}{\sum_{i=1}^{N} p\left(y_{t+1}|x_t^{(i)}\right)}.$$

Since $p\left(y_{t+1}|x_t\right)$ is a function of only x_t and y_{t+1}, these weights are known upon receipt of the new observation, which implies that $p^N\left(x_{t+1}|y^{t+1}\right)$ is a standard discrete mixture distribution. Sampling from a discrete mixture distribution is straightforward by first selecting the mixture index and then simulating from that mixture component. This simple procedure leads to exact particle filtering algorithm is

Step 1. Draw $z^{(i)} \sim Mult_N\left(\left\{w_t^{(i)}\right\}_{i=1}^{N}\right)$ for $i = 1, ..., N$ and set $x_t^{(i)} = x_t^{z(i)}$

Step 2. Draw $x_{t+1}^{(i)} \sim p\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right)$ for $i = 1, ..., N$,

where $Mult_N$ denotes an $N-$component multinomial distribution. Since this generates an i.i.d. sample, the second stage weights in the particle approximation $\pi_t^{(i)}$ are all $1/N$.

The intuition of the algorithm is instructive. At Step 1, upon observation of y_{t+1}, the resampling step selects the particles that were most likely, in terms of the predictive likelihood, $p\left(y_{t+1}|x_t^{(i)}\right)$, to have generated y_{t+1}. After this selection step, the algorithm simulates new particles from the component distribution $p\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right)$. The advantage of this algorithm is that it eliminates the second source of errors that can arise in particle filters. By directly sampling from $p^N\left(x_t|y^t\right)$, there are no importance sampling errors. Any remaining Monte Carlo errors can be minimized by choosing N large. It is also possible to sample the discrete distribution in other ways, such as residual or systematic sampling.

The exact particle filtering approach requires that the predictive likelihood

$$p\left(y_{t+1}|x_t\right) = \int p\left(y_{t+1}|x_{t+1}\right) p\left(x_{t+1}|x_t\right) dx_{t+1}$$

can be computed and that

$$p\left(x_{t+1}|x_t, y_{t+1}\right) \propto p\left(y_{t+1}|x_{t+1}\right) p\left(x_{t+1}|x_t\right)$$

can be sampled. In many models, these distributions are known or straightforward modifications of the general algorithm can be used to generate exact samples from p^N. One such example is given below. In general, exact particle filtering is possible in models with (a) linear observation equations, (b) non-Gaussian errors that can be represented as a discrete or scale mixture of normal distribution, and (c) models with state evolutions that have additive errors, but nonlinear conditional means. This would occur when

$$x_{t+1} = f\left(x_t\right) + \varepsilon_{t+1}^x$$

and $f\left(x_t\right)$ is a known analytical function of x_t. In particular, this allows for a wide range of observation or state errors, including finite mixtures of normal error distributions, t-distributed errors, or double exponential errors. Thus, the class of models in which exact sampling is possible is quite broad.

3.1.1 Example: Linear, Gaussian filtering

To see how exact sampling operates, consider the simple case of filtering in linear Gaussian models: $p\left(y_t|x_t\right) \sim \mathcal{N}\left(x_t, \sigma^2\right)$ and $p\left(x_{t+1}|x_t\right) \sim \mathcal{N}\left(x_t, \sigma_x^2\right)$. The exact or optimal filtering algorithm requires two distributions, $p\left(y_{t+1}|x_t\right)$ and $p\left(x_{t+1}|x_t, y_{t+1}\right)$, which are both easy to characterize:

$$p\left(y_{t+1}|x_t\right) \sim \mathcal{N}\left(x_t, \sigma^2 + \sigma_x^2\right) \text{ and}$$
$$p\left(x_{t+1}|x_t, y_{t+1}\right) \propto p\left(y_{t+1}|x_{t+1}\right) p\left(x_{t+1}|x_t\right) \sim \mathcal{N}\left(\mu_{t+1}, \sigma_{t+1}^2\right),$$

where

$$\frac{\mu_{t+1}}{\sigma_{t+1}^2} = \frac{y_{t+1}}{\sigma_y^2} + \frac{x_t}{\sigma_x^2} \text{ and } \frac{1}{\sigma_{t+1}^2} = \frac{1}{\sigma_y^2} + \frac{1}{\sigma_x^2}.$$

An exact particle filtering algorithm is

$$\text{Step 1: Sample } z^{(i)} \sim Mult_N\left(\left\{w\left(x_t^{(i)}\right)\right\}_{i=1}^N\right)$$
$$\text{and set } x_t^{(i)} = x_t^{z(i)} \text{ for } i = 1, ..., N$$
$$\text{Step 2: Draw } x_{t+1}^{(i)} \sim \mathcal{N}\left(\mu_{t+1}^{(i)}, \sigma_{t+1}^2\right) \text{ for } i = 1, ..., N$$

where

$$w\left(x_t^{(i)}\right) = \exp\left(-\frac{\left(y_{t+1} - x_t^{(i)}\right)^2}{2\left(\sigma^2 + \sigma_x^2\right)}\right) \Bigg/ \sum_{i=1}^N \exp\left(-\frac{\left(y_{t+1} - x_t^{(i)}\right)^2}{2\left(\sigma^2 + \sigma_x^2\right)}\right)$$

and $\mu_{t+1}^{(i)}$ displays the dependence on $x_t^{(i)}$. This generates an equally weighted sample $\left\{x_{t+1}^{(i)}\right\}_{i=1}^N$ from $p^N\left(x_{t+1}|y^{t+1}\right)$, thus $\pi_{t+1}^{(i)} = 1/N$.

3.1.2 Example: Log-stochastic volatility model

A more interesting example is the log-stochastic volatility model, as described in Section 2. The model can be written as

$$\log\left(y_{t+1}\right)^2 = x_{t+1} + \varepsilon_{t+1}$$
$$x_{t+1} = \alpha_v + \beta_v x_t + \sigma_v \varepsilon_{t+1}^v,$$

where ε_{t+1} has a $log\left(\chi_1^2\right)$ distribution. To develop the particle filtering algorithm, it is useful to approximate the $log\left(\chi_1^2\right)$ distribution with a discrete mixture of normals with fixed weights, $\sum_{j=1}^K p_j Z_{t+1}^j$ where $Z_{t+1}^j \sim \mathcal{N}(\mu_j, \sigma_j^2)$ and μ_j and σ_j are known. This approximation is can be made arbitrarily accurate, and in practice 10 mixture components is sufficient.

The key to an efficient particle filter is the introduction of an auxiliary indicator variable, s_{t+1}, that tracks the mixture components. For example, if $s_{t+1} = j$, then

$$p\left(\log\left(y_{t+1}\right)^2 | x_{t+1}, s_{t+1} = j\right) = \mathcal{N}(x_{t+1} + \mu_j, \sigma_j^2).$$

The introduction of an additional latent variable is called data augmentation and is commonly used when developing Markov Chain Monte Carlo algorithms. Given the additional auxiliary variable, the exact sampling algorithm consists of two steps: (1) resampling persistent state variables, in this case, $x_t = log\,(V_t)$, and (2) simulating x_{t+1} and s_{t+1}.

For the first step, the predictive density is

$$p(y_{t+1}|x_t) = \sum_{j=1}^{K} p_j \mathcal{N}(\alpha_v + \beta_v x_t + \mu_j, \sigma_j^2 + \sigma_v^2),$$

and thus

$$w\,(x_t) = \frac{p\,(y_{t+1}|x_t)}{\sum_{i=1}^{N} p\,(y_{t+1}|x_t)}.$$

Note that s_{t+1} is i.i.d., so there is no information available at time t to forecast its value. The second step requires drawing from

$$p\,(x_{t+1}, s_{t+1}|x_t, y_{t+1}) \propto p\,(x_{t+1}|s_{t+1}, x_t, y_{t+1})\,p\,(s_{t+1}|x_t, y_{t+1})\,.$$

The distribution $p\,(s_{t+1}|x_t, y_{t+1})$ is a known discrete distribution since

$$p\,(s_{t+1} = j|x_t, y_{t+1}) \propto p\,(y_{t+1}|x_t, s_{t+1} = j)\,p_j,$$

where

$$p\,(y_{t+1}|x_t, s_{t+1} = j) = \mathcal{N}\left(\alpha_v + \beta_v x_t + \mu_j, \sigma_v^2 + \sigma_j^2\right).$$

Similarly,

$$p\,(x_{t+1}|s_{t+1}, x_t, y_{t+1}) \propto p\,(y_{t+1}|s_{t+1}, x_t)\,p\,(x_{t+1}|x_t)$$

is a convolution of two normal distributions, which is also a normal distribution. Together, these two steps can be used to provide an exact sample from $p^N\left(s_{t+1}, x_{t+1}|y^{t+1}\right)$.

Figure 1 displays a summary of the output of the algorithm, for the simulated path of returns discussed earlier. The bottom panel displays the true simulated volatilities, $\sqrt{V_t}$, in red dots, the posterior mean, $E^N\left(\sqrt{V_t}|y^t\right)$, is the solid line, and the shaded area displays the $(5\%, 95\%)$ quantiles of $p^N\left(\sqrt{V_t}|y^t\right)$.

3.2 SIR

In settings in which exact sampling is not possible, importance sampling is typically used. One of the first, most popular, and most general particle filtering algorithm is known as the sampling importance resampling (SIR) algorithm. The algorithm is simplicity itself, relying only on two steps: given

samples from $p^N(x_t|y^t)$,

$$\text{Step 1. Draw } x_{t+1}^{(i)} \sim p\left(x_{t+1}|x_t^{(i)}\right) \text{ for } i = 1, ..., N$$

$$\text{Step 2. Draw } z^{(i)} \sim Mult_N\left(\left\{w_{t+1}^{(i)}\right\}_{i=1}^N\right) \text{ and set } x_{t+1}^{(i)} = x_{t+1}^{z^{(i)}},$$

where the importance sampling weights are given by

$$w_{t+1}^{(i)} = \frac{p(y_{t+1}|x_{t+1}^{(i)})}{\sum_{i=1}^N p(y_{t+1}|x_{t+1}^{(i)})}.$$

Prior to resampling, each particle had weight $w_{t+1}^{(i)}$. After resampling, the weights are equal, by the definition of resampling. The SIR algorithm has only two mild requirements: that the likelihood function can be evaluated and that the states can be simulated. Virtually every model used in practice satisfies these mild assumptions.

The justification for the algorithm is the weighted bootstrap algorithm or SIR algorithm, which was first developed to simulate posterior distributions, of the form $L(x)\,p(x)$, where L is the likelihood and p the prior. The algorithm first draws an independent sample $x^{(i)} \sim p(x)$ for $i = 1, ...N$, and then computes normalized importance weights $w^{(i)} = L\left(x^{(i)}\right) / \sum_{i=1}^N L\left(x^{(i)}\right)$. The sample drawn from the discrete distribution $\left\{x^{(i)}, w^{(i)}\right\}_{i=1}^N$ tends in distribution to a sample from the product density $L(x)\,p(x)$ as N increases.

In the case of the particle filter, the target density is

$$p\left(x_{t+1}|y^{t+1}\right) \propto p\left(y_{t+1}|x_{t+1}\right) p\left(x_{t+1}|y^t\right). \tag{8}$$

Given an independent sample from $p^N(x_t|y^t)$, the algorithm samples from

$$p^N\left(x_{t+1}|y^t\right) = \int p\left(x_{t+1}|x_t\right) p^N\left(x_t|y^t\right) dx_t,$$

by drawing $x_{t+1}^{(i)} \sim p\left(x_{t+1}|x_t^{(i)}\right)$ for $i = 1, ..., N$. Since $p^N(x_t|y^t)$ is a discrete distribution, this implies that $\left\{x_{t+1}^{(i)}\right\}_{i=1}^N$ is an independent sample from $p^N(x_{t+1}|y^t)$. Resampling with the appropriate weights provides an approximate sample from $p\left(x_{t+1}|y^{t+1}\right)$.

To see the simplicity of the algorithm, consider the benchmark case of filtering in the linear Gaussian model considered earlier. The SIR algorithm requires simulating from $p\left(x_{t+1}|x_t\right) \sim \mathcal{N}\left(x_t, \sigma_x^2\right)$ and evaluating unnormalized weights, which take the form

$$p\left(y_{t+1}|x_{t+1}\right) \propto \exp\left(-\frac{1}{2}\frac{\left(y_{t+1} - x_{t+1}\right)^2}{\sigma^2}\right).$$

3.2.1 Problems with the SIR algorithm

There are a number of problems with SIR. One problem is sample impoverishment or weight degeneracy, which occurs when a vast majority of the weight is placed on a single particle. When this occurs, the resampling step results in a single particle being sampled multiple times. Thus resampling does not fix the sample impoverishment/weight degeneracy problem, it just hides it. Another related problem is that the states are drawn from the prior distribution, $p(x_{t+1}|x_t)$, without accounting for the next period's observation, y_{t+1}. This implies that the simulated states may not be in important or high likelihood, $p(y_{t+1}|x_{t+1})$, regions. In models with outliers, a large y_{t+1} is observed, but the SIR algorithm draws samples from $p\left(x_{t+1}|x_t^{(i)}\right)$ ignoring the new observation.

To mitigate this problem, it is possible to choose an alternative importance density. Instead of drawing from $p(x_{t+1}|x_t)$, it is possible to draw from an importance density that depends on y_{t+1},

$$x_{t+1}^{(i)} \sim q\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right).$$

In this case the unnormalized weights are

$$w_{t+1}^{(i)} \propto \frac{p(y_{t+1}|x_{t+1}^{(i)})p\left(x_{t+1}^{(i)}|x_t^{(i)}\right)}{q\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right)}.$$

The "optimal" importance sampling density, in terms of minimizing the variance of the importance weights, is $p(x_{t+1}|x_t, y_{t+1})$.

3.3 Auxiliary particle filtering algorithms

An alternative when the SIR algorithm performs poorly is the auxiliary particle filter (APF). The original description of the APF used the idea of auxiliary variables. The algorithm we provide motivates the APF as an importance sampling version of the exact sampling algorithm given in the previous section.

Like exact sampling, the APF consists of two steps: resampling old particles and propagating states. Unlike the exact sampling algorithm, the APF uses importance sampling when it is not possible to evaluate $p(y_{t+1}|x_t)$ or sample directly from $p(x_{t+1}|x_t, y_{t+1})$. The exact mixture weights $p(y_{t+1}|x_t)$ are approximated by an importance weight $q(y_{t+1}|x_t)$ and the posterior distribution $p(x_{t+1}|x_t, y_{t+1})$ is approximated by the importance distribution $q(x_{t+1}|x_t, y_{t+1})$. The APF algorithm is given by:

$$\text{Step 1: Compute } w\left(x_t^{(i)}\right) = \frac{q\left(y_{t+1}|x_t^{(i)}\right)}{\sum_{i=1}^{N} q\left(y_{t+1}|x_t^{(i)}\right)} \text{for } i = 1, ..., N$$

$$\text{Step 2: Draw } z\left(i\right) \sim Mult_N\left(\left\{w\left(x_t^{(i)}\right)\right\}\right) \text{ and set } x_t^{(i)} = x_t^{z(i)}$$

$$\text{Step 3: Draw } x_{t+1}^{(i)} \sim q\left(x_{t+1}|x_t^{(i)}, y_{t+1}\right) \text{ for } i = 1, ..., N$$

$$\text{Step 4: Reweight: } \pi\left(x_{t+1}^{(i)}\right) \propto \frac{\text{target}}{\text{proposal}} = \frac{p\left(y_{t+1}|x_{t+1}^{(i)}\right) p\left(x_{t+1}^{(i)}|x_t^{(i)}\right)}{q\left(y_{t+1}|x_t^{(i)}\right) q\left(x_{t+1}^{(i)}|x_t^{(i)}, y_{t+1}\right)}.$$

The weights at the end of the algorithm are the importance sampling weights. There is no need to resampling additionally using these weights, in fact, this introduces additional Monte Carlo error.

Like exact sampling, the APF resamples first, which is important to insure that high likelihood states are propagated forward. The performance of the APF is driven by the accuracy of the importance densities. If these are poor approximations, the APF may not perform much better than the SIR algorithm, and in some extreme cases, could even perform worse. A final advantage of the APF algorithm is its flexibility, as it allows for two importance densities. This allows the algorithm to be tailored to the specific application at hand.

4 Further Reading

Research on particle filtering methods has exploded recently over the past 10 years. It is impossible to cite all of the relevant work, and we will instead focus on the initial theoretical contributions, important review papers, and applications. For textbook discussions, see the monographs by Doucet, de Freitas, and Gordon (2001) and Ristic, Arulampalam, and Gordon (2004). These books provide more details, numerous alternative algorithms and extensions to improve performance, and extensive lists of references. Cappe, Godsill, and Moulines (2007) provide a very readable and up to date review article.

The sampling/importance resampling algorithm appears in Rubin (1987) and Smith and Gelfand (1992). The foundational particle filtering algorithm appears in Gordon, Salmond, and Smith (1993). Liu and Chen (1995, 1998) provide key contributions to sequential importance sampling. Pitt and Shephard (1999) developed the auxiliary particle filter and discuss various extensions and applications. Other important contributions are in Kitigawa (1996), Hurzeler and Kunsch (1998), Carpenter, Clifford, and Fearnhead (1999), and Kunsch (2005)

For applications in economics and finance, Kim, Shephard, and Chib (1998) and Chib, Nardari and Shephard (2006) apply particle filters to univariate and multivariate stochastic volatility models. Johannes, Polson, and Stroud (2008) develop particle filters for continuous-time jump-diffusion models, with option pricing applications. Pitt (2005) discusses particle filtering approaches for maximum likelihood estimation. Fernandez-Villaverde and Rubio-Ramirez (2005) use particle filters for parameter estimation in general equilibrium macroeconomics models.

There is also a growing literature applying particle filters for sequential parameter learning and state filtering, see, for example Liu and West (2001), Storvik (2002), Fearnhead (2002), Johannes, Polson and Stroud (2005, 2006), and Johannes and Polson (2006).

References

Cappe, O., Godsill, S. and Moulines, E. (2007): An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceedings of the IEEE* **95**, 899–924.

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

Chib, S., Nardari, F. and Shephard, N. (2006): Analysis of high dimensional multivariate stochastic volatility models. *Journal of Econometrics* **134**, 341–371.

Doucet, A., de Freitas, N. and Gordon, N. (2001): *Sequential Monte Carlo Methods in Practice. Statistics for Engineering and Information Science*. Springer, New York.

Fearnhead, P. (2002): MCMC, sufficient statistics and particle filter. *Journal of Computuational and Graphical Statistics* **11**, 848–862.

Fernandez-Villaverde, J. and Rubio-Ramirez, J. (2005): Estimating Macroeconomic Models: A Likelihood Approach. *Working paper, University of Pennsylvania.*

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

Kitagawa, G. (1996): Monte Carlo Filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics* **5**, 1–25.

Hurzeler, M. and Kunsch, H. (1998): Monte Carlo approximations for general state space models. *Journal of Computational and Graphical Statistics* **7**, 175–193.

Johannes, M. and Polson, N.G. (2006): Exact Bayesian particle filtering and parameter learning. *Working paper, University of Chicago.*

Johannes, M., Polson, N.G. and Stroud, J.R. (2008): Optimal filtering of Jump Diffusions: Extracting latent states from asset prices. *Review of Financial Studies* forthcoming.

Johannes, M., Polson, N.G. and Stroud, J.R. (2005): Sequential parameter estimation in stochastic volatility models with jumps. *Working paper, University of Chicago.*

Johannes, M., Polson, N.G. and Stroud, J.R (2006): Interacting particle systems for sequential parameter learning. *Working paper, University of Chicago.*

Kim, S., Shephard, N. and Chib, S. (1998): Stochastic volatility: likelihood inference and comparison with ARCH models. *Review of Economic Studies* **65**, 361–93.

Kunsch, H. (2005): Recursive Monte Carlo filters: Algorithms and theoretical analysis. *Annals of Statistics* **33**, 1983–2021.

Liu, J. and West, M. (2001): Combined parameter and state estimation in simulation-based filtering. *In: Doucet, A. et al. (Eds.): Sequential Monte Carlo Methods in Practice* Springer, New York.

Liu, J. and Chen, R. (1995): Blind deconvolution via sequential imputations. *Journal of American Statistical Association.* **89**, 278–288.

Liu, J. and Chen, R. (1998): Sequential Monte Carlo Methods for Dynamical Systems. *Journal of the American Statistical Association* **93**, 1032–1044.

Pitt, M. (2005): Smooth particle filters for likelihood evaluation and maximisation. *Working paper, University of Warwick.*

Pitt, M. and Shephard, N. (1999): Filtering via simulation: auxiliary particle filter. *Journal of the American Statistical Association* **94**, 590–599.

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

Rubin, D. B. (1987): A noniterative sampling/impotance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithm, comment to a paper by Tanner and Wong. *Journal of the American Statistical Association* **82**, 543–546.

Smith, A.F.M. and Gelfand, A. E. (1992): Bayesian statistics without tears. *American Statistician* **46**, 84–88.

Storvik, G. (2002): Particle filters in state space models with the presence of unknown static parameters. *IEEE Trans. on Signal Processing* **50**, 281–289.