

Particle Filters and Data Assimilation

Paul Fearnhead¹ and Hans R. Künsch²

¹Department of Mathematics and Statistics, Lancaster University, Lancaster LA1 4YF, United Kingdom; email: p.fearnhead@lancaster.ac.uk

²Seminar for Statistics, Department of Mathematics, ETH Zurich, CH-8092 Zurich, Switzerland; email: kuensch@stat.math.ethz.ch

Annu. Rev. Stat. Appl. 2018. 5:421–49

First published as a Review in Advance on
December 8, 2017

The *Annual Review of Statistics and Its Application* is
online at statistics.annualreviews.org

<https://doi.org/10.1146/annurev-statistics-031017-100232>

Copyright © 2018 by Annual Reviews.
All rights reserved

Keywords

ensemble Kalman filter, particle filter, particle Markov chain Monte Carlo, particle smoother, sequential Monte Carlo, state-space model

Abstract

State-space models can be used to incorporate subject knowledge on the underlying dynamics of a time series by the introduction of a latent Markov state process. A user can specify the dynamics of this process together with how the state relates to partial and noisy observations that have been made. Inference and prediction then involve solving a challenging inverse problem: calculating the conditional distribution of quantities of interest given the observations. This article reviews Monte Carlo algorithms for solving this inverse problem, covering methods based on the particle filter and the ensemble Kalman filter. We discuss the challenges posed by models with high-dimensional states, joint estimation of parameters and the state, and inference for the history of the state process. We also point out some potential new developments that will be important for tackling cutting-edge filtering applications.



ANNUAL REVIEWS Further

Click [here](#) to view this article's
online features:

- Download figures as PPT slides
- Navigate linked references
- Download citations
- Explore related articles
- Search keywords

1. INTRODUCTION

This article gives an overview of Monte Carlo methods for estimating parameters and latent variables and for making predictions in state-space models. In some fields state-space models are known by the name of hidden Markov models, but we use the term state-space model throughout.

1.1. What Is a State-Space Model?

In order to predict a time series of observations, it is essential to take subject knowledge of the dynamics of the series into account. However, in many applications subject knowledge involves adding to the observed variables other variables that are hard or impossible to measure. A state-space model specifies the joint distribution of all the variables that are required for a dynamical model based on subject knowledge, and the variables that have been observed. The former are called the state variables and are denoted by $(\mathbf{X}_t; t \geq 0)$. The evolution of the state variables is assumed to be given either by a Markov process or deterministically by a system of ordinary or partial differential equations. The state variables are latent; we only have access to observations $(\mathbf{Y}_i; i \in \mathbb{N})$ that are partial and noisy functions of the state \mathbf{X}_{t_i} at observation times t_i .

In some applications, the state variables are not obtained by a detailed subject-based modeling, but rather represent dynamic random effects or unknown time-varying parameters that have simple dynamics, often a linear Gaussian autoregression. Combined with a generalized linear model for the observations given the states, this leads to what Cox (1981) calls parameter-driven models.

1.1.1. Example 1: tracking. The particle filter methods we review in this article were first motivated by tracking applications (e.g., Gordon et al. 1993, Stone et al. 2014). For these applications the state is the position and velocity of the target or targets being tracked. Observations are made of their location but can be partial (e.g., only measurement of the bearing), can be noisy, and can include clutter (spurious measurements that do not relate to any target). For these applications the key inference questions relate to estimating the current positions of targets and predicting their future movement. This requires online algorithms, such as particle filters, that can quickly update beliefs of the state as each new measurement is observed.

1.1.2. Example 2: numerical weather prediction. Advances in numerical weather prediction during the past 50 or 100 years have been termed a “quiet revolution” by Bauer et al. (2015, p. 47) in a “computational problem comparable to the simulation of the human brain and of the evolution of the early Universe.” These advances have been made possible by not only increased computing power, better measurements, and improved physical understanding, but also ensemble forecasts, which quantify uncertainty, and data assimilation methods, which sequentially integrate measurements into the forecasting process.

1.1.3. Example 3: ecology. A model for the evolution of a population usually needs information about the abundance in different age classes. The dynamics of the model relate abundances at the next time-point to current abundances while accounting for rates of fertility, mortality, catchment, and migration. The models thus have states that record population sizes within each age range, and those rates that are considered time varying. Observations, for example from capture-recapture experiments, will relate indirectly to these population sizes. Interest is often about future predictions about the population, which requires estimates of both the current state and the parameters. For more details, the reader is directed to Aeberhard et al. (2017) or Nielsen & Berg (2014).

1.2. What Are Filtering and Data Assimilation?

In order to apply state-space models, we need to be able to estimate unobserved states, future observations, and unknown parameters of the model from available data. For this, the key task is to compute the conditional distribution of the state \mathbf{X}_{t_i} at time t_i based on observations up to time t_i , the so-called filtering distribution. Once we know this filtering distribution, we can obtain predictive distributions of future states by letting the state process evolve with the filtering distribution as initial distribution at time t_i . From the predictive distribution of future states, the predictive distribution of observations follows immediately. All the relevant information about future states and observations is thus contained in the filtering distribution.

Computing the filtering distribution is, however, a difficult task. Some simplification occurs by exploiting a recursive scheme. Using the filtering distribution at time t_{i-1} , we first compute the predictive distribution of \mathbf{X}_{t_i} based on observations up to time t_{i-1} , using the dynamics of the state. The filtering distribution at time t_i then follows from Bayes' formula applied with the predictive distribution as the prior and using the likelihood of \mathbf{x}_{t_i} given \mathbf{y}_i . Hence recursive filtering proceeds by an alternation of prediction or propagation steps based on the dynamics of the state, and update steps based on the most recent observation.

Filtering is engineering terminology. In geophysics, the term data assimilation is used instead. The predictive distribution of \mathbf{X}_{t_i} given observations up to time t_{i-1} is usually called the background distribution, and the filter distribution is usually called the analysis distribution. The exchange of ideas and methods for filtering between statistics on the one hand and geophysics and applied mathematics on the other has only recently become more common, and one aim of this review is to bring the two communities closer together.

In geophysics, the state evolution is often deterministic but chaotic, that is, sensitive to initial conditions. In fact, the phenomenon of chaos was discovered in a toy atmospheric physics model by Lorenz (1963). Because of this sensitivity to initial conditions, new observations have to be assimilated frequently for good predictions.

Except in special cases, the propagation and the update steps cannot be computed analytically. As the steps involve integration over the often high-dimensional state space, Monte Carlo approximations are currently preferred. This review is limited to these approximations.

1.3. Outline of the Review

After giving some background on state-space models and a brief treatment of the basic recursions for the true filtering and predictive distributions in Section 2, in Section 3 we describe Monte Carlo methods to approximate these recursions, namely the particle filter, the ensemble Kalman filter, and their extensions. Section 4 briefly summarizes theoretical properties of the particle filter, and Section 5 discusses the challenges that arise when applying these filter methods to models with high-dimensional states.

We then focus on methods for smoothing and parameter estimation. Smoothing involves calculating the conditional distribution of historic values of the state given all observations to date. We show how particle filter ideas can be extended and applied to approximate these smoothing distributions in Section 6. Then, in Section 7, we look at particle filter methods for estimating parameters, with particular emphasis on recent particle MCMC methods. The review ends with a summary and outlook.

We do not make an attempt to give a complete overview of all aspects of filtering or to provide a comprehensive list of references. Recent other reviews are those of Doucet & Johansen (2011), Künsch (2013), and Kantas et al. (2015), while part III of Douc et al. (2014b) contains a detailed introduction with many examples and proofs. Majda & Harlim (2012) and Reich & Cotter (2015)

present the field from applied mathematics and geophysics perspectives. One area we view as important, but do not cover, is the increasing need to design filtering algorithms that can take advantage of modern computer architecture. This is an area we flag later as an important future issue, but readers are directed to Lee & Whiteley (2016) and Vergé et al. (2015) for some recent work.

Software in R for implementing some of the examples we consider in this article is available as online supplemental material. This is provided primarily to give the reader the opportunity to run the algorithms under different settings to build up a stronger intuition as to when and why any of these methods work well. Software for implementing some of the methods we describe in this review for generic applications is also available. We are aware of the following: SMCTC (Johansen 2016), LiBi (Murray 2015), the package `nimble` (Michaud et al. 2017) in R, the Robotics System Toolbox of MATLAB, and PDAF (Nerger & Hiller 2013) and DART (<http://www.image.ucar.edu/DAReS/DART/>) for geophysical applications.

Supplemental Material

2. STATE-SPACE MODELS AND FILTER RECURSIONS

2.1. State-Space Models

In order to simplify the notation, we assume that the observation times are equally spaced with $t_i = i$ and that the model is time-homogeneous. All results and methods can be easily extended to unequally spaced observations and time-inhomogeneous models. We also repeatedly use the notation that the subscript $s:t$ refers to the set of values at all times from time s to time t , so, for example, $\mathbf{X}_{0:n} = (\mathbf{X}_0, \dots, \mathbf{X}_n)$.

The state process (\mathbf{X}_t) is assumed to be Markovian, and the i th observation, \mathbf{Y}_i , depends only on the state at time i , \mathbf{X}_i , and is conditionally independent of all other observations. This means that

$$\mathbf{X}_t \mid (\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t-1}) \sim P(d\mathbf{x}_t \mid \mathbf{x}_{t-1}), \quad \mathbf{X}_0 \sim \pi_0(d\mathbf{x}_0) \quad 1.$$

$$\mathbf{Y}_t \mid (\mathbf{x}_{0:t}, \mathbf{y}_{1:t-1}) \sim g(\mathbf{y}_t \mid \mathbf{x}_t) d\nu(\mathbf{y}_t). \quad 2.$$

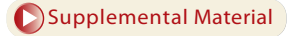
If the state evolution is given by a time-homogeneous (autonomous) differential equation, P becomes a point mass at the solution at time t with initial condition \mathbf{x}_{t-1} at time $t-1$. Similarly, other common models, such as state-space formulations of autoregressive models of higher order than 1, can mean that components of \mathbf{X}_t are deterministic functions of \mathbf{X}_{t-1} . We therefore do not want to assume that the state transitions have densities, but the conditional distribution of the observations should have densities so that we can use Bayes' formula (though extensions of filters to exact observation of part of the state is possible; see Section 3.2). The measure ν is usually either Lebesgue or counting measure. In practice either or both of the distributions that determine the state evolution or the measurement process can depend on parameters. Many particle filter methods assume such parameters are known. We will suppress the dependence of $P(d\mathbf{x}_t \mid \mathbf{x}_{t-1})$ and $g(\mathbf{y}_t \mid \mathbf{x}_t)$ in Equations 1 and 2 on such parameters in our notation except when we consider estimating the parameters in Section 7.

State-space models are directed graphical models (Lauritzen 1996) with the following graph

$$\begin{array}{ccccccc} \dots & \rightarrow & \mathbf{X}_{t-1} & \rightarrow & \mathbf{X}_t & \rightarrow & \mathbf{X}_{t+1} & \rightarrow & \dots \\ & & \downarrow & & \downarrow & & \downarrow & & \\ \dots & & \mathbf{Y}_{t-1} & & \mathbf{Y}_t & & \mathbf{Y}_{t+1} & & \dots \end{array}$$

Various conditional independence properties follow from this graph. For instance, \mathbf{X}_t is conditionally independent of $(\mathbf{Y}_{t+1}, \mathbf{X}_{t+2}, \mathbf{Y}_{t+2}, \dots)$ given \mathbf{X}_{t+1} . Such properties are used in Section 6.

2.1.1. Two examples. To make these ideas concrete, we give two simple examples of state-space models. These examples are used for illustration in the **Supplemental Appendix**.



2.1.1.1. Example: stochastic volatility. To help demonstrate the different algorithms clearly in the figures and animations of the **Supplemental Appendix**, we will use an example with a 1-dimensional state:

$$X_t | x_{t-1} \sim \mathcal{N}(\phi x_{t-1}, \sigma^2), \quad Y_t | x_t \sim \mathcal{N}(0, \beta^2 \exp\{x_t\}),$$

where $\mathcal{N}(0, \sigma^2)$ denotes the normal distribution with mean 0 and variance σ^2 . This is a simple stochastic volatility model (see e.g., Kim et al. 1998), with the state, X_t , being proportional to the log-volatility of the observation series. The model has three parameters, ϕ , σ , and β , which respectively govern the dependence and noise in the state process and the baseline variance of the observation process.

2.1.1.2. Example: Lorenz 96. This is a toy model of a one-dimensional atmosphere, popular as a test bed for data assimilation in atmospheric physics (Lorenz & Emanuel 1998). We use it to illustrate the ensemble Kalman filter. The state is 40-dimensional, with dynamics given by the differential equation

$$\frac{dX_{t,k}}{dt} = (X_{t,k+1} - X_{t,k-2})X_{t,k-1} - X_{t,k} + 8, \quad k = 1, \dots, 40, \quad X_{t,k} \equiv X_{t,k+40}.$$

At times $t = i\Delta$, every m th component of X_t is observed with independent additive Gaussian noise. This can be written as $\mathbf{Y}_t | \mathbf{x}_t \sim \mathcal{N}(\mathbf{H}\mathbf{x}_t, \sigma^2\mathbf{I})$ where \mathbf{H} is the appropriate matrix to select the observed components and \mathbf{I} is the identity matrix.

2.2. Prediction, Filter, and Smoothing Distributions

We collect here the basic formulae of conditional distributions and likelihoods that are needed for prediction, filtering, smoothing, and parameter estimation.

For $0 \leq s \leq u$ and $t \geq 1$, the conditional distribution of $\mathbf{X}_{s:u}$ given $\mathbf{Y}_{1:t} = \mathbf{y}_{1:t}$ is denoted by $\pi_{s:u|t}$, and we use $\pi_{s|t}$ instead of $\pi_{s:s|t}$. Hence, $\pi_{t|t-1}$ is the predictive distribution at time t based on observations up to time $t-1$, for short, the prediction distribution at time t . Finally, we denote the filtering distribution at time t by π_t instead of $\pi_{t|t}$.

With a slight abuse of notation, all other (conditional) densities are denoted by p : The arguments of p indicate which random variables are involved.

The assumptions of Equations 1 and 2 imply the following joint distributions:

$$(\mathbf{X}_{0:s}, \mathbf{Y}_{1:t}) \sim \pi_0(d\mathbf{x}_0) \prod_{i=1}^s P(d\mathbf{x}_i | \mathbf{x}_{i-1}) \prod_{j=1}^t g(\mathbf{y}_j | \mathbf{x}_j) v(d\mathbf{y}_j), \quad s \geq t. \quad 3.$$

Integrating out the path of the state process, we obtain that $\mathbf{Y}_{1:t} \sim p(\mathbf{y}_{1:t}) \prod_j v(d\mathbf{y}_j)$, where

$$p(\mathbf{y}_{1:t}) = \int \pi_0(d\mathbf{x}_0) \prod_{i=1}^s P(d\mathbf{x}_i | \mathbf{x}_{i-1}) \prod_{j=1}^t g(\mathbf{y}_j | \mathbf{x}_j).$$

If the model contains unknown parameters θ , $p(\mathbf{y}_{1:t})$ becomes the likelihood of θ . By Bayes' formula, $\pi_{0:s|t}$ is the right-hand side of Equation 3 divided by $p(\mathbf{y}_{1:t})$. From this it is easy to see that the following recursion holds:

$$\pi_{0:t|t-1}(d\mathbf{x}_{0:t} | \mathbf{y}_{1:t-1}) = \pi_{0:t-1|t-1}(d\mathbf{x}_{0:t-1} | \mathbf{y}_{1:t-1}) P(d\mathbf{x}_t | \mathbf{x}_{t-1}), \quad 4.$$

$$\pi_{0:t|t}(d\mathbf{x}_{0:t} | \mathbf{y}_{1:t}) = \pi_{0:t|t-1}(d\mathbf{x}_{0:t} | \mathbf{y}_{1:t-1}) \frac{g(\mathbf{y}_t | \mathbf{x}_t)}{p(\mathbf{y}_t | \mathbf{y}_{1:t-1})}, \quad 5.$$

where

$$p(\mathbf{y}_t | \mathbf{y}_{1:t-1}) = \frac{p(\mathbf{y}_{1:t})}{p(\mathbf{y}_{1:t-1})} = \int \pi_{t|t-1}(\mathbf{d}\mathbf{x}_t | \mathbf{y}_{1:t-1}) g(\mathbf{y}_t | \mathbf{x}_t). \quad 6.$$

Integrating out the states $\mathbf{x}_{0:t-1}$ in Equations 4 and 5 leads to the recursion discussed in the Introduction:

$$\pi_{t|t-1}(\mathbf{d}\mathbf{x}_t | \mathbf{y}_{1:t-1}) = \int \pi_{t-1}(\mathbf{d}\mathbf{x}_{t-1} | \mathbf{y}_{1:t-1}) P(\mathbf{d}\mathbf{x}_t | \mathbf{x}_{t-1}), \quad 7.$$

$$\pi_t(\mathbf{d}\mathbf{x}_t | \mathbf{y}_{1:t}) = \pi_{t|t-1}(\mathbf{d}\mathbf{x}_t | \mathbf{y}_{1:t-1}) \frac{g(\mathbf{y}_t | \mathbf{x}_t)}{p_t(\mathbf{y}_t | \mathbf{y}_{1:t-1})}. \quad 8.$$

Both recursions consist of a propagation step, Equation 4 or 7, and an update or correction step, Equation 5 or 8. Making predictions more than one time-step ahead is simple, as we can apply the propagation update (Equation 7) without the correction step:

$$\pi_{t+s|t}(\mathbf{d}\mathbf{x}_{t+s} | \mathbf{y}_{1:t}) = \int \pi_{t+s-1|t}(\mathbf{d}\mathbf{x}_{t+s-1} | \mathbf{y}_{1:t}) P(\mathbf{d}\mathbf{x}_{t+s} | \mathbf{x}_{t+s-1}), \text{ for } s = 1, \dots$$

Because the observations, \mathbf{y}_t , are fixed, we often drop them from the notation, for example, writing $\pi_t(\mathbf{d}\mathbf{x}_t)$ rather than $\pi_t(\mathbf{d}\mathbf{x}_t | \mathbf{y}_{1:t})$.

3. MONTE CARLO FILTER ALGORITHMS

Monte Carlo filter algorithms approximate the filter distributions, π_t , by weighted samples $(\mathbf{x}_t^i, w_t^i) = (\mathbf{x}_t^i, w_t^i)_{i=1}^N$ of size N :

$$\pi_t(\mathbf{d}\mathbf{x}_t) \approx \hat{\pi}_t^N(\mathbf{d}\mathbf{x}_t) = \sum_{i=1}^N w_t^i \delta_{\mathbf{x}_t^i}(\mathbf{d}\mathbf{x}_t). \quad 9.$$

Here $\delta_{\mathbf{x}}$ denotes the point mass at \mathbf{x} . When we simultaneously consider approximations of π_t by weighted and evenly weighted samples, we denote the latter by $(\tilde{\mathbf{x}}_t^i)$. The sample members are called particles because in the algorithm they move in space and have offspring or die. In geophysics, samples are usually called ensembles.

3.1. The Bootstrap Filter

If we insert the weighted sample approximation (Equation 9) at time $t - 1$ into the propagation (Equation 7), we obtain

$$\pi_{t|t-1}(\mathbf{d}\mathbf{x}_t) \approx \sum_{i=1}^N w_{t-1}^i P(\mathbf{d}\mathbf{x}_t | \mathbf{x}_{t-1}^i).$$

Therefore, if $\mathbf{x}_t^i \sim P(\mathbf{d}\mathbf{x}_t | \mathbf{x}_{t-1}^i)$ independently for $i = 1, \dots, N$, the weighted sample $(\mathbf{x}_t^i, w_{t-1}^i)$ approximates the prediction distribution $\pi_{t|t-1}$:

$$\pi_{t|t-1}(\mathbf{d}\mathbf{x}_t) \approx \hat{\pi}_{t|t-1}^N(\mathbf{d}\mathbf{x}_t) = \sum_{i=1}^N w_{t-1}^i \delta_{\mathbf{x}_t^i}(\mathbf{d}\mathbf{x}_t).$$

Applying the Bayes' update (Equation 8) to this approximation gives

$$\pi_t(\mathbf{d}\mathbf{x}_t) \approx \hat{\pi}_t^N(\mathbf{d}\mathbf{x}_t) = \sum_{i=1}^N w_t^i \delta_{\mathbf{x}_t^i}(\mathbf{d}\mathbf{x}_t), \quad w_t^i \propto W_t^i = w_{t-1}^i g(\mathbf{y}_t | \mathbf{x}_t^i). \quad 10.$$

This closes the recursion of the sequential importance sampling algorithm. At time $t = 0$, we initialize it by drawing \mathbf{x}_0^i from $\pi_0(\mathbf{d}\mathbf{x}_0)$ and set $w_0^i = 1/N$. As a by-product, the normalizing

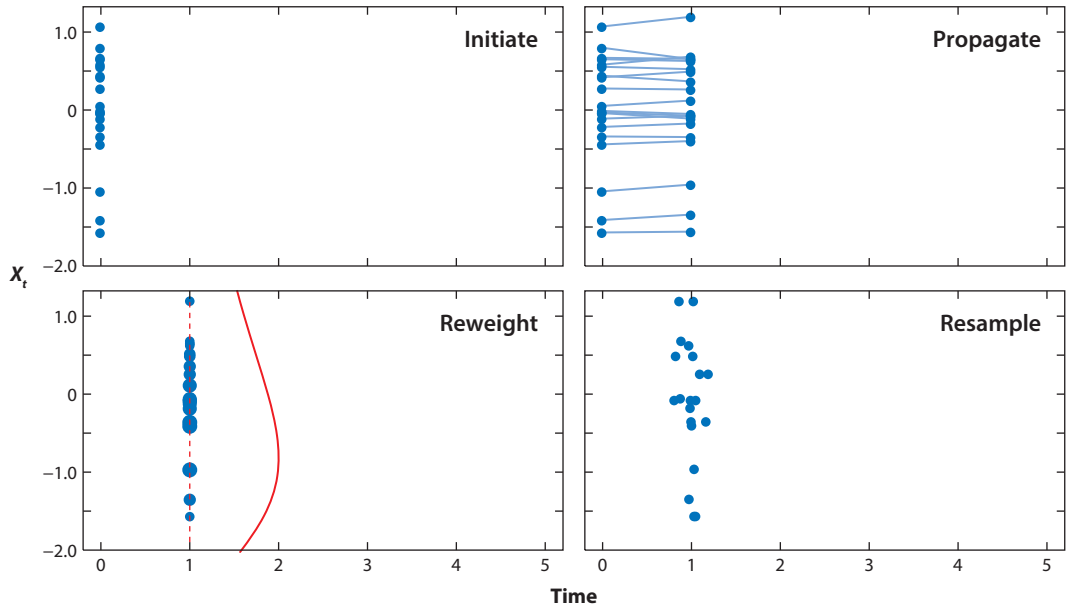


Figure 1

Plots of the bootstrap filter for the stochastic volatility model. We show the output after each of the three stages: propagate, reweight, and resample. For the reweight plot, the red line shows the likelihood function $g(x_t)$ (the distance of the red line from the red dashed line being proportional to the likelihood for the corresponding value of x_t), and we have rescaled the particles according to the weight they are given. For the resample plot, we have jittered the time component of the particles purely so that one can see when multiple copies of a particle are resampled. An animated version of the figure is available in the **Supplemental Appendix**.

constant for the weights w_t^i provides an approximation of $p(\mathbf{y}_t | \mathbf{y}_{1:t-1})$ because

$$\mathbb{E} \left(\sum_{i=1}^N W_t^i | \mathbf{x}_{t-1}^i \right) = \sum_{i=1}^N w_{t-1}^i \int g(\mathbf{y}_t | \mathbf{x}_t) P(d\mathbf{x}_t | \mathbf{x}_{t-1}^i) \approx \int g(\mathbf{y}_t | \mathbf{x}_t) \pi_{t|t-1}(d\mathbf{x}_t).$$

The sequential importance sampling algorithm has the drawback that after a few iterations, the weights are essentially concentrated on a few particles, and most or all particles are in regions where the true filter distribution has little mass. To avoid this, the basic bootstrap filter makes weights equal by resampling before propagating. It consists of the following steps:

1. Resample: Set $\tilde{\mathbf{x}}_{t-1}^i = \mathbf{x}_{t-1}^{A(i)}$ where $\mathbb{P}(A(i) = j) = w_{t-1}^j$ for $i = 1, \dots, N$.
2. Propagate: Draw \mathbf{x}_t^i from $P(d\mathbf{x}_t | \tilde{\mathbf{x}}_{t-1}^i)$, independently for $i = 1, \dots, N$.
3. Reweight: Set $w_t^i \propto W_t^i = g(\mathbf{y}_t | \mathbf{x}_t^i)/N$.
4. Likelihood estimation: Calculate $\hat{p}(\mathbf{y}_t | \mathbf{y}_{1:t-1}) = \sum_{i=1}^N W_t^i$, and set $\hat{p}(\mathbf{y}_{1:t}) = \hat{p}(\mathbf{y}_{1:t-1})\hat{p}(\mathbf{y}_t | \mathbf{y}_{1:t-1})$.

Figure 1 shows an example of the output of this recursion.

The computational complexity of one iteration of the bootstrap filter is $O(N)$. The observation likelihood, g , is required in closed form, whereas we need only to be able to simulate from the propagation distribution, P . Particles interact through the normalization in the reweighting step.

As a by-product of running the bootstrap filter, we get an approximation of the prediction distribution by the evenly weighted sample (\mathbf{x}_t^i) in step 2. Step 4 is optional but gives an estimate of the parameter likelihood. The estimate $\hat{p}(\mathbf{y}_{1:t})$ is unbiased [see theorem 7.4.2 in Del Moral (2004) or Pitt et al. (2012)], a property that is used in Section 7.2. However, $\hat{p}(\mathbf{y}_t | \mathbf{y}_{1:t-1})$ is biased in general.

Supplemental Material

We call $\mathbf{x}_{t-1}^{A(i)}$ the “ancestor” of particle \mathbf{x}_t^i and denote the number of offspring of particle \mathbf{x}_t^i at time t by N_t^i . Resampling replaces the weights w_t^i by random weights N_t^i/N and thus always increases the Monte Carlo error of the approximation to the current filtering distribution. However, resampling is beneficial when we propagate particles to future time-steps. It stochastically removes particles with low weight and produces multiple copies of particles with high weight. Multiple copies of a current particle are then able to independently explore the future of the state.

The benefit of resampling depends crucially on the level of stochasticity in the propagation distribution, P . If this is high relative to the filter variance, then even if the current particles lack diversity, this diversity is quickly regained as we propagate forward in time. If the state dynamics is deterministic, we will not regain diversity as we propagate the particles forward. To overcome this, it is possible to add random noise to the current particles prior to propagation, which can be justified as sampling from a kernel density approximation to the filter density (Hürzeler & Künsch 1998, Liu & West 2001). The issue of deterministic dynamics arises not only in many geophysical applications but also when we have unknown, fixed parameters in the model. We discuss this issue in more detail in Section 7.

The ancestors $A(i)$ do not have to be drawn independently for different i ; it is only required that $\mathbb{E}(N_t^i) = Nw_t^i$. Balanced sampling (also called stratified or systematic sampling; Kitagawa 1996, Carpenter et al. 1999) makes the resampling error as small as possible. The simplest method partitions the interval $[0, N]$ into subintervals of length Nw_t^i and counts how many points of the sequence $U, U + 1, \dots, U + N - 1$ where $U \sim \mathcal{U}(0, 1)$ fall in each subinterval. Crisan (2001) presents a different balanced sampling method.

If the weights, that is, the values of the observation likelihood, are very unbalanced, resampling risks losing too much diversity that cannot be restored correctly in the propagation step, even with stochastic dynamics. This problem, and ways to alleviate or overcome it, are considered in the rest of this section and in Section 5.

3.2. Auxiliary Particle Filters

The bootstrap filter considers the target proportional to $\sum_i P(\mathbf{dx}_t | \tilde{\mathbf{x}}_{t-1}^i) \cdot g(\mathbf{y}_t | \mathbf{x}_t)$ and uses importance sampling with the proposal $N^{-1} \sum_i P(\mathbf{dx}_t | \tilde{\mathbf{x}}_{t-1}^i)$. If the observation likelihood is informative, the target and the proposal are not close enough and the weights become unbalanced. In this case, we can try to find a better proposal, such as $\sum \tilde{w}_{t-1}^{(i)} \tilde{P}(\mathbf{dx}_t | \tilde{\mathbf{x}}_{t-1}^i)$, where the weights, $\tilde{w}_{t-1}^{(i)}$, give preference to current particles most consistent with the next observation, \mathbf{y}_t , and where the transition, \tilde{P} , moves particles to places that are compatible with \mathbf{y}_t . If the transitions $P(\mathbf{dx}_t | \mathbf{x}_{t-1})$ have densities with respect to $\tilde{P}(\mathbf{dx}_t | \mathbf{x}_{t-1})$, the importance weights are well defined, but the algorithm has complexity $O(N^2)$ because each unnormalized weight involves a summation over N terms.

The auxiliary particle filter of Pitt & Shephard (1999) avoids this increase in complexity by considering the target distribution on the product space of the state at times $t - 1$ and t , which is proportional to

$$\sum_{i=1}^N w_{t-1}^i \delta_{\mathbf{x}_{t-1}^i}(\mathbf{dx}_{t-1}) P(\mathbf{dx}_t | \mathbf{x}_{t-1}) g(\mathbf{y}_t | \mathbf{x}_t).$$

This is an approximation of $\pi_{t-1:t|t}$. If we use a proposal of the form

$$\sum_{i=1}^N \tilde{w}_{t-1}^i \delta_{\mathbf{x}_{t-1}^i}(\mathbf{dx}_{t-1}) \tilde{P}(\mathbf{dx}_t | \mathbf{x}_{t-1}),$$

the proposed pairs are obtained by first resampling the particles (\mathbf{x}_{t-1}^i) with probabilities (\tilde{w}_{t-1}^i) and then propagating the resampled particles with the transition \tilde{P} . If we draw the sample ($\mathbf{x}_{t-1}^{A(i)}, \mathbf{x}_t^i$) from this proposal, then its importance weight is

$$w_t^i \propto W_t^i = \frac{w_{t-1}^{A(i)} P(d\mathbf{x}_t^i | \mathbf{x}_{t-1}^{A(i)}) g(\mathbf{y}_t | \mathbf{x}_t^i)}{\tilde{w}_{t-1}^{A(i)} \tilde{P}(d\mathbf{x}_t^i | \mathbf{x}_{t-1}^{A(i)})}. \quad 11.$$

In contrast to the bootstrap filter, the weights depend on the particles at both times t and $t - 1$. The average of the unnormalized weights again provides an approximation of $p(\mathbf{y}_t | \mathbf{y}_{1:t-1})$ and can be used to obtain an unbiased estimate of the likelihood.

The second ratio in the equation for W_t^i has to be understood as a Radon–Nikodym derivative. If densities exist, it is simply the ratio of these densities. The auxiliary particle filter can be applied also to models where the observation distribution does not have a density, because we observe part of the state without error. The formula for the weights W_t^i still makes sense, provided we use a proposal density $\tilde{P}(d\mathbf{x}_t | \mathbf{x}_{t-1})$ that simulates states consistent with the new observation \mathbf{y}_t .

If $\tilde{w}_t^i = w_{t-1}^i$ and $\tilde{P} = P$, we recover the bootstrap filter, but both the weights and the transition in the proposal can depend on the new observation \mathbf{y}_t . The optimal proposal in the sense of making the weights from Equation 11 constant is (Doucet et al. 2000)

$$\tilde{w}_{t-1}^i \propto w_{t-1}^i p(\mathbf{y}_t | \mathbf{x}_{t-1}^i), \quad \tilde{P}(d\mathbf{x}_t | \mathbf{x}_{t-1}) = P(d\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{y}_t).$$

These quantities are usually not tractable, but often one can obtain good approximations with reasonable computing complexity.

Even when the weights w_t^i are constant, resampling will occur at the beginning of the next iteration. The weights are then proportional to $p(\mathbf{y}_{t+1} | \mathbf{x}_t^i)$, whereas the bootstrap filter has weights proportional to $g(\mathbf{y}_t | \mathbf{x}_t^i) = p(\mathbf{y}_t | \mathbf{x}_t^i)$. Since the former likelihood is flatter, the auxiliary particle filter has more equal weights, but the difference is substantial only if the dependence in the state dynamics is weak compared with the information from the observations.

3.3. Quasi–Monte Carlo Filters

Quasi–Monte Carlo methods achieve faster convergence rates than standard Monte Carlo by replacing random draws by more regular samples. They start with low discrepancy points (\mathbf{u}^i) in the unit cube $[0, 1]^d$ and transform these into low discrepancy points from a general distribution of interest. The advantage of quasi–Monte Carlo is that the error decays at a rate close to $1/N$, rather than the $1/\sqrt{N}$ of standard Monte Carlo (Niederreiter 1978). Randomized versions of quasi–Monte Carlo can even achieve rates close to $N^{-3/2}$ (Owen 1998), though for high-dimensional applications, we need large N to see any benefit from these quicker convergence rates (Caflisch et al. 1997).

The first use of quasi–Monte Carlo for particle filters was by Fearnhead (2005), though the computational cost was $O(N^2)$. More recently, Gerber & Chopin (2015) have shown how quasi–Monte Carlo can be applied within a particle filter while still retaining the $O(N)$ computational complexity. Their idea is to transform the state so that it is in the d -dimensional unit cube and write the state transition as

$$\mathbf{X}_t = \Gamma_t(\mathbf{X}_{t-1}, \mathbf{U}_t) \quad \mathbf{U}_t \sim \mathcal{U}([0, 1]^d),$$

where Γ_t is an appropriate smooth function. Assuming that (\mathbf{x}_{t-1}^i) is a quasi–Monte Carlo sample, one wants to modify the bootstrap filter so that (\mathbf{x}_t^i) is still a quasi–Monte Carlo sample. For this, particles should not be resampled or propagated independently: If \mathbf{x}_{t-1}^i and \mathbf{x}_{t-1}^j are close, then

the number of times they are resampled, $N_t^i + N_t^j$, should be as close as possible to $N(w_t^i + w_t^j)$, and they should be spread out in the propagation step.

In the one-dimensional case, $d = 1$, this is easy to achieve: We can assume that the x_{t-1}^i are in increasing order and that the innovations u_t^i are numbered such that the first k points $u_t^{1:k}$ have low discrepancy for any $k \leq N$. Then the propagated particles

$$x_t^i = \Gamma_t(x_{t-1}^i, u_t^i) \quad (1 \leq i \leq N)$$

have the desired features. Similarly, for the balanced resampling method discussed above, we arrange the subintervals in the same order as the particles.

The difficulty in extending these ideas to higher dimensions is how to define a suitable total order of points in the unit cube. Gerber & Chopin (2015) use the Hilbert curve, which is a space-filling fractal curve $H: [0, 1] \mapsto [0, 1]^d$ that preserves locality and low discrepancy.

3.4. Sequential Monte Carlo

Particle filters have many applications outside of time series analysis. They are then usually called sequential Monte Carlo algorithms and produce samples from a complicated target distribution π by recursive sampling from a sequence of n intermediate distributions $\pi_0, \pi_1, \dots, \pi_n = \pi$. Here π_0 is a distribution from which one can sample easily, and we choose the sequence of distributions so that any two consecutive distributions π_{t-1} and π_t are close in some appropriate sense. A prime example is tempering, where

$$\pi(\mathbf{dx}) \propto \phi(\mathbf{x})\pi_0(\mathbf{dx}), \quad \pi_t(\mathbf{dx}) \propto \phi(\mathbf{x})^{t/n}\pi_0(\mathbf{dx}).$$

As in Monte Carlo filtering, sequential Monte Carlo produces a sequence of particles by resampling, propagation with transitions P_t , and reweighting with weight functions g_t . If $(\mathbf{x}_{t-1}^i, w_{t-1}^i)$ is a weighted sample from π_{t-1} , then the propagated particles (\mathbf{x}_t^i, w_t^i) are a weighted sample from

$$\pi_{t|t-1}(\mathbf{dx}) = \int P_{t-1}(\mathbf{dx} \mid \mathbf{x}')\pi_{t-1}(\mathbf{dx}').$$

Therefore, the correct weight function g_t is the density of π_t with respect to $\pi_{t|t-1}$. In situations other than filtering, this g_t is intractable unless P_t leaves π_t invariant. Sequential Monte Carlo methods overcome this intractability by working on the joint space of $(\mathbf{x}_t, \mathbf{x}_{t-1})$. They construct a joint distribution whose marginal for \mathbf{X}_t is $\pi_t(\mathbf{x}_t)$, and for which it is then possible to calculate appropriate importance sampling weights. Del Moral et al. (2006) provide more details.

3.5. Ensemble Kalman Filter

The ensemble Kalman filter has been developed in geophysics (Evensen 1994, 2007) and is used frequently in atmospheric physics, oceanography, and reservoir modeling. The propagation step is the same as in the bootstrap filter. For the update, the observations are assumed to be linear combinations of the state with additive Gaussian noise: $\mathbf{Y}_t \mid \mathbf{x}_t \sim \mathcal{N}(\mathbf{H}\mathbf{x}_t, \mathbf{R})$. If the prediction distribution is normal, $\pi_{t|t-1} = \mathcal{N}(\mathbf{m}_{t|t-1}, \mathbf{P}_{t|t-1})$, then the filter distribution is also normal, $\pi_t = \mathcal{N}(\mathbf{m}_t, \mathbf{P}_t)$, with

$$\mathbf{m}_t = \mathbf{m}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \mathbf{H}\mathbf{m}_{t|t-1}), \quad \mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_{t|t-1},$$

where

$$\mathbf{K}_t = \mathbf{K}(\mathbf{P}_{t|t-1}, \mathbf{R}) = \mathbf{P}_{t|t-1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^T + \mathbf{R})^{-1}$$

is the Kalman gain. In the ensemble Kalman filter, both $\pi_{t|t-1}$ and π_t are approximated by evenly weighted samples that we denote by (\mathbf{x}_t^i) and $(\tilde{\mathbf{x}}_t^i)$. The update step in the ensemble Kalman filter estimates $\mathbf{m}_{t|t-1}$ and $\mathbf{P}_{t|t-1}$ from the prediction sample and then constructs the filter sample by transforming the prediction sample such that it has the mean and covariance given above. This can be done in different ways.

The stochastic ensemble Kalman filter updates each particle in the same way as the mean, but after adding an artificial noise to the observation:

$$\tilde{\mathbf{x}}_t^i = \mathbf{x}_t^i + \hat{\mathbf{K}}_t(\mathbf{y}_t - \mathbf{H}\mathbf{x}_t^i + \boldsymbol{\varepsilon}_t^i), \quad \boldsymbol{\varepsilon}_t^i \sim \mathcal{N}(0, \mathbf{R}), \quad 12.$$

where $\hat{\mathbf{K}}_t$ is the Kalman gain for the estimated prediction covariance. Square root filters use a deterministic affine transformation of the sample (\mathbf{x}_t^i) . To define it, we introduce the matrix $\Delta\mathbf{x}_t$, whose columns contain the centered particles $\mathbf{x}_t^i - \hat{\mathbf{m}}_{t|t-1}$, and similarly, we introduce $\Delta\tilde{\mathbf{x}}_t$. Then the mean is updated by

$$\hat{\mathbf{m}}_t = \hat{\mathbf{m}}_{t|t-1} + \hat{\mathbf{K}}_t(\mathbf{y}_t - \mathbf{H}\hat{\mathbf{m}}_{t|t-1})$$

and the centered particles are updated by either pre- or postmultiplication:

$$\Delta\tilde{\mathbf{x}}_t = \mathbf{A}_t \cdot \Delta\mathbf{x}_t, \quad \text{or} \quad \Delta\tilde{\mathbf{x}}_t = \Delta\mathbf{x}_t \cdot \mathbf{W}_t.$$

The matrices \mathbf{A}_t and \mathbf{W}_t are obtained by requiring that the sample $(\tilde{\mathbf{x}}_{t|t}^i)$ has the desired covariance

$$\Delta\tilde{\mathbf{x}}_t(\Delta\tilde{\mathbf{x}}_t)^T = (N-1)\hat{\mathbf{P}}_t = (N-1)(\mathbf{I} - \hat{\mathbf{K}}_t\mathbf{H})\hat{\mathbf{P}}_{t|t-1}.$$

This results in quadratic equations for \mathbf{A}_t and \mathbf{W}_t that can be solved (see Tippett et al. 2003). Postmultiplication is preferred for computational reasons if the sample size N is smaller than the dimension of the state.

Figure 2 shows an update step by the ensemble Kalman filter in an example from numerical weather prediction.

For stability of the ensemble Kalman filter, the estimation of the prediction covariance $\mathbf{P}_{t|t-1}$ is crucial. We come back to this point briefly in Section 5.2. There are various methods to compute the update efficiently, depending on which version is used and how $\mathbf{P}_{t|t-1}$ is estimated (see, e.g., Evensen 2003, Tippett et al. 2003). If the state is high-dimensional, $\hat{\mathbf{P}}_{t|t-1}$ need not be computed or stored, and the ensemble Kalman filter has computational advantages over the population Kalman filter (see Butala et al. 2009).

The ensemble Kalman filter updates the particles by moving them in space instead of weighting and resampling. It therefore does not suffer from sample depletion like particle filters. It is, however, biased in general, and can be viewed as reducing the variance by allowing a bias.

3.6. Particle Filter Updates Using Sequential Monte Carlo

If the observations are informative about the state, the two distributions $\pi_{t|t-1}$ and π_t are not close enough to make importance sampling efficient. Using sequential Monte Carlo to go from $\pi_{t|t-1}$ to π_t in a sequence of intermediate steps is therefore an attractive idea. Intermediate steps can be defined either by tempering the likelihood, $\pi_{t,\gamma} \propto \pi_{t|t-1}(\mathrm{d}\mathbf{x}_t)g(\mathbf{y}_t | \mathbf{x}_t)^\gamma$, or—if the dimension d of \mathbf{y}_t is large and the components of \mathbf{Y}_t are independent given \mathbf{x}_t —by the posterior given the first $k < d$ components of \mathbf{y}_t . However, in order to obtain an algorithm that differs from a bootstrap filter, the sequential Monte Carlo algorithm must include some propagation, and the choice of the transition kernels used in propagating the particles is difficult, in particular when the dynamics of the state is not tractable and there is no analytic expression for $\pi_{t|t-1}$.

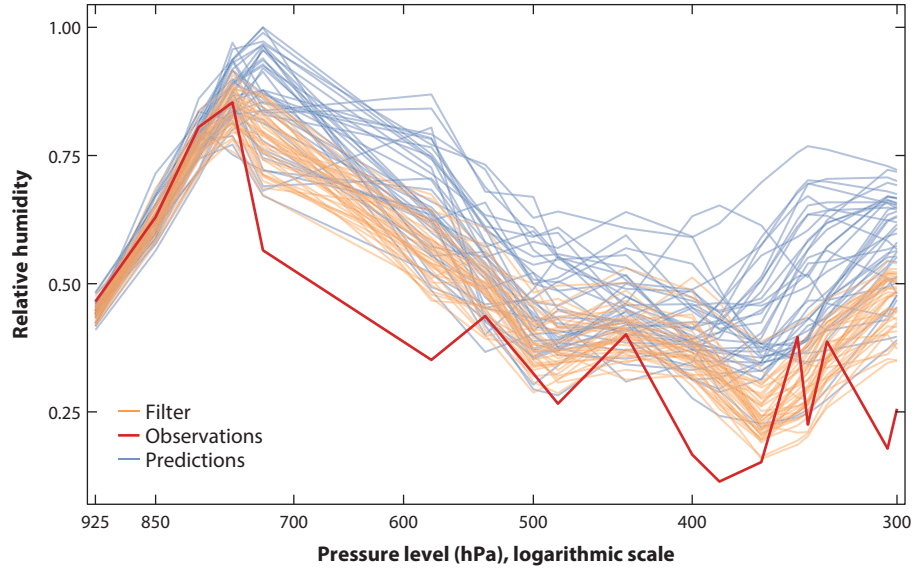


Figure 2

Update of relative humidity as a function of pressure by the ensemble Kalman filter using measurements from one radiosonde launched on June 14, 2015, in K  mmersbruck, Bavaria. The values of the state are averaged over different pressure levels and a region near K  mmersbruck. The update moves the prediction particles closer to the observations and reduces the spread. As other observations are also used in this update, the filter mean does not match the observed relative humidity at some pressure levels. Adapted with permission from Robert (2017).

The simplest implementation of this idea is presented in Frei & K  nsch (2013), where there is just one intermediate tempering of the likelihood and the ensemble Kalman filter is used for the first step while the particle filter is used for the second. Both steps can be done analytically, and sampling is only required for the next propagation step. Bunch & Godsill (2016) and Beskos et al. (2017) present alternative approaches.

3.7. Other Monte Carlo Filtering Algorithms

Here we briefly discuss a few other filtering algorithms. Most of them try to reduce the sample depletion problem of the particle filter and simultaneously improve on the ability of the ensemble Kalman filter in non-Gaussian situations.

3.7.1. Transport filters. Reich (2013) proposes a linear deterministic update that replaces resampling by averaging: Instead of $\tilde{\mathbf{x}}_t^i = \mathbf{x}_t^{A(i)}$, he uses

$$\tilde{\mathbf{x}}_t^i = \sum_{j=1}^N \mathbb{P}(A(i) = j) \mathbf{x}_t^j$$

for some chosen resampling scheme that specifies the probabilities $\mathbb{P}(A(i) = j)$ for all i and j . This approximation preserves the mean but can be shown to have a reduced spread by a simple application of Jensen's inequality.

The approximation error depends on the chosen resampling scheme. The stochastic matrix $(p_{ij}) = (\mathbb{P}(A(i) = j))$ must satisfy $\sum_i p_{ij} = N w_t^j$ but is otherwise arbitrary. Reich (2013) chooses

(p_{ij}) such that on average the resampled particles are as close as possible to the original particles. For a given distance matrix $d_{ij} = d(\mathbf{x}_t^i, \mathbf{x}_t^j)$ between prediction particles, this means we minimize $\sum_{i,j} d_{ij} p_{ij}$ subject to $p_{ij} \geq 0$, $\sum_j p_{ij} = 1$, and $\sum_i p_{ij} = N w_{it}^j$. This is a famous linear programming problem (see, e.g., Reich & Cotter 2015, section 7.4). If (p_{ij}) is the solution of this minimization problem, the approximate update is then consistent as the sample size N tends to infinity. Heuristically this is true because most $p_{ij} = 0$ in the optimal solution.

3.7.2. Hybrid filters. Hybrid filters combine particle and ensemble Kalman filters with the goal of exploiting the advantages of both methods. Chustagulprom et al. (2016) follow the idea of Frei & Künsch (2013), described above, with a two-step update. They use the deterministic transport filter of Reich (2013) in the first step and the ensemble Kalman filter in the second.

Van Leeuwen (2010) proposes a method to obtain equal weights by an ensemble Kalman filter–type proposal density.

3.7.3. Robust filters. Calvet et al. (2015) propose a robust filter to address the issue of observation outliers. They modify the likelihood $g(\mathbf{y}_t | \mathbf{x}_t)$ in order to reduce the impact of an observation \mathbf{y}_t that comes from a distribution other than the nominal $g(\mathbf{y}_t | \mathbf{x}_t) d\nu(\mathbf{y}_t)$. This has the additional benefit of reducing sample depletion in the particle filter.

3.7.4. Rao–Blackwellization. For some models we can partition the state, $\mathbf{X}_t = (\mathbf{X}_t^{(1)}, \mathbf{X}_t^{(2)})$, such that $p(\mathbf{x}_{1:t}^{(1)} | \mathbf{x}_{1:t}^{(2)}, \mathbf{y}_{1:t})$ is tractable. This most often happens if, conditional on $\mathbf{x}_{1:t}^{(2)}$, the model is linear-Gaussian. In this case $p(\mathbf{x}_{1:t}^{(1)} | \mathbf{x}_{1:t}^{(2)}, \mathbf{y}_{1:t})$ is Gaussian, with a mean and covariance that will depend on $\mathbf{x}_{1:t}^{(2)}$ but that can be calculated using the Kalman filter. In such cases we can “Rao–Blackwellize” out the $\mathbf{X}_t^{(1)}$ component of the state and implement a particle filter that targets $p(\mathbf{x}_{1:t}^{(2)} | \mathbf{y}_{1:t})$. By reducing the dimension of the state space, this latter particle filter can be much more efficient. Doucet et al. (2000) and Chen & Liu (2000) present examples and further details.

4. BASIC CONVERGENCE RESULTS

There is now a substantial literature describing whether and how fast the particle filter approximation converges to the filtering distribution as the number of particles increases. Comprehensive results can be found in Crisan & Doucet (2002), Del Moral (2004), or Cappé et al. (2005). Chopin (2004) and Künsch (2005) use a less general but more direct approach. For brevity, we limit ourselves here to an intuitive discussion of some key ideas behind these convergence results.

Let $\psi(\mathbf{x})$ be a suitable test function, and let \mathbf{X}_t^i be the evenly weighted particles of our particle filter at time t . Then the goal is to prove an L^p -bound or a central limit theorem for the filter error at time t ,

$$\frac{1}{N} \sum_{i=1}^N \psi(\mathbf{X}_t^i) - \int \psi(\mathbf{x}_t) \pi_t(d\mathbf{x}_t),$$

as the number of particles, N , goes to infinity and the observations $\mathbf{y}_{1:t}$ are fixed.

For fixed time t , such results hold under weak conditions on the evolution and observation models, and typically one has the standard $1/\sqrt{N}$ error of a Monte Carlo procedure. But for applications, one would like to know whether the bounds or the convergence are uniform in t . If the number of particles required for a given accuracy of the estimated filter mean needs to increase with the number of time steps, particle filters will be of limited use.

In addition to a sampling error at time t , the filter error has a second component that occurs because the particles \mathbf{X}_t^i are not sampled from the exact filter distribution π_t , but from the

distribution proportional to $\sum_i P(d\mathbf{x}_t | \mathbf{x}_{t-1}^i)g(\mathbf{y}_t | \mathbf{x}_t)$. The second component of the filter error is thus the difference of the expectation of $\psi(\mathbf{x}_t)$ with respect to these two distributions. Both distributions are obtained through a propagation step (Equation 7) and an update step (Equation 8), applied to π_{t-1} and $\hat{\pi}_{t-1}^N$, respectively. Therefore, the second component of the filter error is due to the error present at time $t-1$. Uniform-in-time bounds require a control of error accumulation as t increases or, equivalently, that the filtering distribution π_t forgets the initial distribution π_0 as $t \rightarrow \infty$.

Intuitively, if the state process mixes well, then the error at time $t-1$ will be reduced when we go forward one time step using a propagation and an update step. However, the update step can make this intuition invalid, and there are examples of state-space models with ergodic dynamics for which the filter does not forget the initial distribution. Forgetting of the initial distribution at a sufficiently fast rate does hold under an unrealistically strong condition of uniform mixing of the state process. This assumption has been used in most uniform-in-time convergence results for particle filters. Recently however, Douc et al. (2014a) have been able to prove such results under substantially weaker conditions. Atar (2011) gives a review of results about forgetting of the initial distribution by the filter.

By contrast, if there is strong dependence in the model, particle filters can have poor Monte Carlo properties. This is most clearly seen when some components do not change at all in the state evolution, something that occurs when we perform smoothing or when there are unknown fixed parameters—these are discussed in Sections 6 and 7, respectively. In these cases, the particle filter variance will increase with t , and thus we need an increasing Monte Carlo sample size as we analyze longer time series. Often the Monte Carlo sample size may need to increase exponentially with t .

5. FILTER COLLAPSE

5.1. Importance Weights in High Dimensions

In many examples, one observes that the maximal weight in the bootstrap filter is very close to one, leading to the collapse of the filter. Bengtsson et al. (2008) provide theoretical insight into why this occurs by analyzing cases where the dimension d of the observation \mathbf{y}_t and the number of particles N both go to infinity. Conditionally on \mathbf{y}_t , the log likelihood values $\log g(\mathbf{y}_t | \mathbf{x}_t^i)$ then typically behave like a sample from $\mathcal{N}(\mu_d, \sigma_d^2)$, where $\sigma_d = O(\sqrt{d})$. If this holds, the ratio of the largest and the second-largest weights can be approximated in distribution by

$$\exp(\sigma_d(Z_{(N)} - Z_{(N-1)})),$$

where $Z_{(N-1)}$ and $Z_{(N)}$ are the two largest values of N independent $\mathcal{N}(0, 1)$ -variables. By standard results from extreme value theory (e.g., Embrechts et al. 1997, theorem 4.2.8), the difference $Z_{(N)} - Z_{(N-1)}$ is of the order $O_P(1/\sqrt{2 \log N})$. Therefore, the maximal weight converges to one in probability unless N grows exponentially with d . For the two largest weights to be asymptotically equal, one needs the even stronger condition $\log(N)/d \rightarrow \infty$.


For related results on the required Monte Carlo sample size for importance sampling, readers are directed to Chatterjee & Diaconis (2017), Agapiou et al. (2017) and Sanz-Alonso (2016).

5.2. Stability of the Ensemble Kalman Filter

Le Gland et al. (2011) and Frei (2013) have studied the asymptotics of ensemble Kalman filters in the standard setting where we fix the dimension of the states and observations and increase the number of particles, $N \rightarrow \infty$. For practical applications, the more relevant question is whether the filter does not lose track of the state when N is smaller than the dimension of the state.

Some modifications of the method are necessary to achieve this stability, and we still lack a full understanding of the problem.

One reason for instability is sampling errors in the estimated prediction covariance $\hat{\mathbf{P}}_{t|t-1}$. Particularly harmful are underestimation of the diagonal elements, leading to overconfidence in the prediction sample, and spurious nonzero off-diagonal elements in cases with sparse observations. This is because the ensemble Kalman filter works by first updating the observed components and then regressing unobserved components on these updates. Regularization of estimated covariances is used to mitigate these problems, either by inflation of the diagonal elements or by tapering, that is, elementwise multiplication of $\hat{\mathbf{P}}_{t|t-1}$ with a band-limited correlation matrix. The animations in the **Supplemental Appendix** illustrate covariance tapering in the Lorenz 96 model.

 [Supplemental Material](#)

Regularization techniques are effective, but the choice of the tuning constants is difficult. A method that is similar, but not equivalent, to covariance tapering is localization. Since localization is used also for particle filters, we discuss it separately in the next subsection.

Kelly et al. (2015) present a rigorous analysis of how the unmodified ensemble Kalman filter can diverge to infinity even when the dynamics of the state is deterministic with a stable fixed point. Tong et al. (2016) propose an adaptive inflation of the diagonal of $\hat{\mathbf{P}}_{t|t-1}$ such that the Markov process consisting of the state \mathbf{X}_t and the filter ensemble (\mathbf{X}_t^i) is geometrically ergodic provided the state dynamics has a Lyapunov function. Hence the ensemble cannot diverge to infinity, but there is no information about how close the state \mathbf{X}_t and the filter mean are.

5.3. Preventing Collapse by Localization

In many applications with high-dimensional states or observations, the components of \mathbf{X}_t and \mathbf{Y}_t are associated with positions in space. Usually in such cases the observation distribution is local,

$$g(\mathbf{y}_t \mid \mathbf{x}_t) = \prod_v g(y_{t,v} \mid x_{t,N(v)}),$$

where v is used to indicate components of \mathbf{y}_t and $N(v)$ is a set of components of \mathbf{x}_t whose positions are close to v . If dependence in the predictive distribution $\pi_{t|t-1}$ is small between regions that are far apart, then intuitively it seems reasonable to use local updates where any given component of \mathbf{x}_t is only affected by nearby components of \mathbf{y}_t . However, the exact update is typically not local, as can be seen in the example where under $\pi_{t|t-1}$, the state is a circular Gaussian moving average of order 1, and $\mathbf{Y}_t \mid \mathbf{x}_t \sim \mathcal{N}(\mathbf{x}_t, \mathbf{I})$. Still, a local update should be close to optimal and more stable because it combines updates in much lower dimensions. In addition, local updates can take advantage of modern, highly parallel computer architectures.

For the ensemble Kalman filter update, localization was introduced early on (see, e.g., Evensen 2003, Ott et al. 2004) and is now a well-established technique. It can be achieved by imposing sparsity on the estimated Kalman gain, $\hat{\mathbf{K}}_t$. Instead of simply setting most entries of the gain equal to zero, updates with better smoothness properties can be obtained by artificially increasing the observation error variance to infinity as the distance to the position to be updated increases. Hunt et al. (2007) provide details and further discussion.

For particle filters or hybrid methods, localization is much more difficult because any kind of resampling that does not occur globally for all components introduces artificial discontinuities in the particles. Such discontinuities can have drastic consequences in the next propagation step if the transition depends on differences between neighboring components, which is typical in many applications. Robert & Künsch (2017) propose a method that introduces a smooth transition between regions with different resampling by making partial Gaussian assumptions for π_t . It is similar but not identical to the idea presented by Bengtsson et al. (2003). Rebeschini & van

Handel (2015) analyze theoretical properties of a particle that partitions the set of positions into blocks and applies independent resampling for state variables in each block. Their error bounds are independent of dimension and small at positions away from boundaries between blocks. It would be interesting to extend these results to methods that also reduce the discontinuities between blocks.

6. PARTICLE SMOOTHING

We now turn to the related problem of smoothing. This involves calculating, or approximating, the distribution of past values of the state. We consider three related smoothing problems. The first is full smoothing, calculating the conditional distribution of the complete trajectory of the state, $\pi_{0:t|t}(\mathbf{dx}_{0:t})$. The others are fixed-lag smoothing, where our interest is only in the trajectory of the state at the most recent $L + 1$ time-points, $\pi_{t-L:t|t}(\mathbf{dx}_{t-L:t})$, and marginal smoothing, where our interest concerns the state at a fixed time-point $\pi_{s|t}(\mathbf{dx}_s)$ for some $s < t$. If we can solve the full smoothing problem well, then this immediately gives us a solution to the fixed-lag and marginal smoothing problems. However, as we will see, there are approaches that can work well for the latter two problems but not the former.

The auxiliary particle filter of Section 3.2 gives a solution to the fixed-lag smoothing problem for a lag of $L = 1$. When calculating the filtering distribution at time t , it generates weighted particles that consist of states at both times t and $t - 1$. These weighted particles approximate the joint density $\pi_{t-1:t|t}(\mathbf{dx}_{t-1:t})$. This idea has been extended by Doucet et al. (2006), though for larger L this approach suffers from the need to define an efficient proposal distribution for $\mathbf{x}_{t-L+1:t}$, which can be difficult.

Kitagawa (1996) noted that we can trivially adapt a particle filter to solve the full smoothing problem by just storing the trajectory associated with each particle. Thus, at time $t - 1$, a particle will consist of a realization of the full state trajectory $\mathbf{x}_{0:t-1}$. When we perform the propagation step of the particle filter at time t , the new particle will be the concatenation of its ancestor particle at time $t - 1$ and the simulated value for the state at time t . The resulting algorithm can be viewed as a particle filter approximation to the recursions in Equations 4 and 5 rather than those in Equations 7 and 8. It incurs an additional storage cost over the basic particle filter, but otherwise shares the same computational properties.

However, the algorithm is impracticable in most applications. When we redefine our particle filter so that its particles are the full trajectory of the state, the value of $\mathbf{x}_{0:s}$ of any particle at time $t > s$ will necessarily be equal to one of the particles from time s . Furthermore, the number of distinct paths for $\mathbf{x}_{0:s}$ will decrease monotonically as t increases. If $t - s$ is sufficiently large, all particles will share the same value of $\mathbf{x}_{0:s}$; Jacob et al. (2015) show that this will almost surely happen for a time t such that $t - s = O(N \log N)$.

Thus, we observe particle degeneracy in earlier parts of the particle trajectories. This can be seen, for the stochastic volatility model, in the left column of plots in **Figure 3**. In each case the particle approximation for the smoothing distribution of \mathbf{X}_s given $\mathbf{y}_{1:t}$ degenerates to a single distinct value for $s \ll t$.

While this simple algorithm of Kitagawa (1996) is in practice not suitable for the full smoothing problem, in some situations it can give good results for fixed-lag smoothing. There is some indication of this from the bottom-left plot of **Figure 3**, with the smoothed coverage intervals for \mathbf{X}_s demonstrating reasonable particle diversity for the most recent time-points.

Furthermore, the simple smoother can be used to approximately solve the marginal smoothing problem if we are willing to assume that observations sufficiently far in the future have little information about the current state. This motivates an approximation whereby, when estimating the state at time s , we ignore any observations after a time $s + L$ for some suitably chosen lag L .

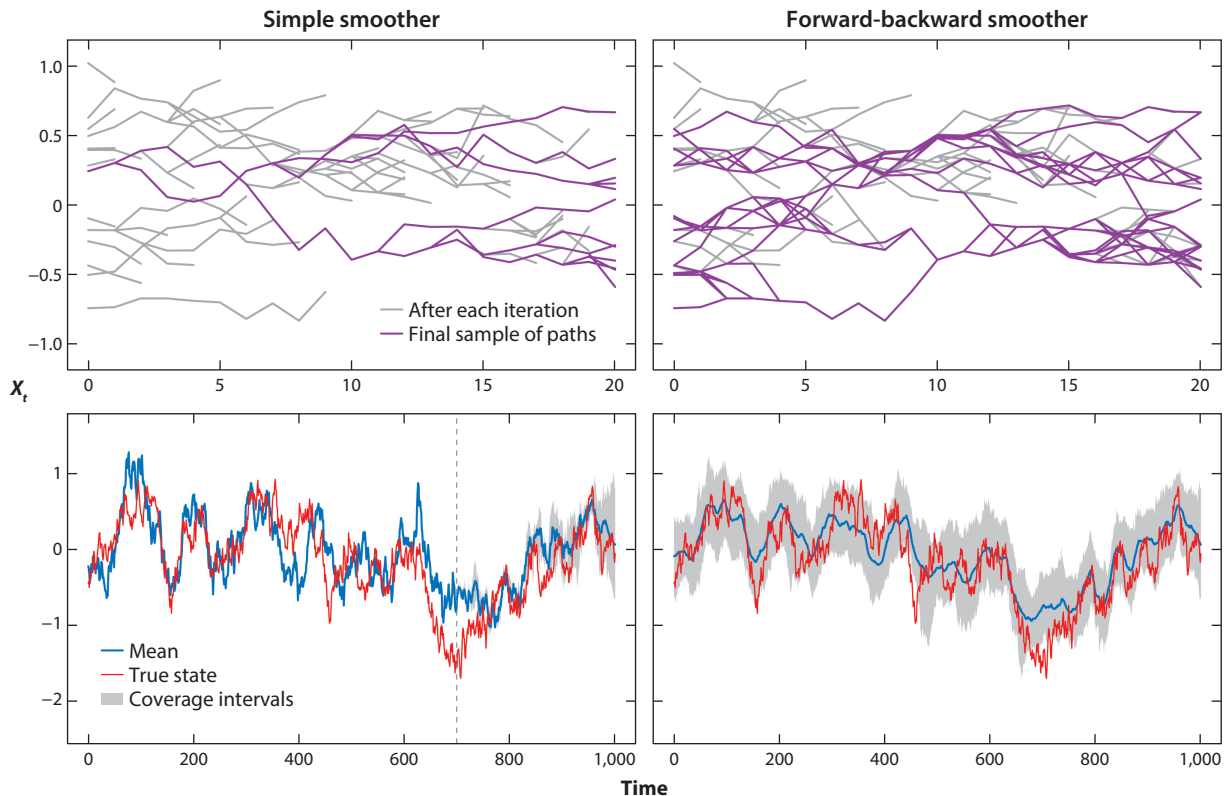


Figure 3

(Left column) Results from the simple smoother of Kitagawa (1996) for the stochastic volatility model. (Right column) Results from the forward-backward smoother. Top plots show the paths stored after each iteration of the Kitagawa (1996) smoother (gray) and the final sample of paths (purple). For the simple smoother, the latter coincide with all particles at the final time-point, but their diversity is reduced as we go back toward the start. The forward-backward smoother is able to sample new paths during the backward simulation step, and thus maintains sample diversity throughout. The bottom plots show estimated mean and 95% coverage intervals for the state given data up to time 1,000. For each plot the mean is in blue, coverage intervals are in gray, and the true state is in red. The gray dashed vertical line shows the point at which all particles at time 1,000 have converged to single value for the state. We used 20 and 200 particles for the top and bottom plots, respectively. In the **Supplemental Appendix**, the top left plot is available as an animation.

[Supplemental Material](#)

Mathematically, this equates to assuming $\pi_{s|n}(\mathbf{dx}_s) \approx \pi_{s|s+L}(\mathbf{dx}_s)$ for $n > s + L$ and large enough L (see, e.g., Polson et al. 2008). The algorithm of Kitagawa (1996) can be used to approximate $\pi_{s|s+L}(\mathbf{dx}_s)$ for a suitable value L , and then this approximation is used as an approximation to $\pi_{s|n}(\mathbf{dx}_s)$ for subsequent times $n > s + L$. While the assumption underlying this approach is often reasonable, choosing an appropriate L can be difficult in practice.

6.1. Forward-Backward Particle Smoother

Improvements on the simple smoother of Kitagawa (1996) are possible. One approach involves an additional backwards smoothing recursion to postprocess the output of the particle filter. For this method we require that the state-transition distribution has a density, denoted by $p(\mathbf{x}_{s+1} | \mathbf{x}_s)$, and that this density is analytically tractable. If state-transition densities exist, then all prediction and filter distributions also have densities, denoted by $\pi_s(\mathbf{x}_s)$ and $\pi_{s|s-1}(\mathbf{x}_s)$.

The forward-backward particle smoother is based on the same ideas that are used in the forward-backward algorithm for discrete-state hidden Markov models and linear Gaussian state-space models (Baum et al. 1970, Durbin & Koopman 2001). It applies to all smoothing problems, and we begin by describing how it is used to simulate from an approximation to the full smoothing distribution (Godsill et al. 2004).

By the conditional independence of $(\mathbf{x}_{s+2:t}, \mathbf{y}_{s+1:t})$ and $(\mathbf{x}_s, \mathbf{y}_{1:s})$ given \mathbf{x}_{s+1} and Bayes' formula,

$$p(\mathbf{x}_s | \mathbf{x}_{s+1:t}, \mathbf{y}_{1:t}) = p(\mathbf{x}_s | \mathbf{x}_{s+1}, \mathbf{y}_{1:s}) = \frac{p(\mathbf{x}_{s+1} | \mathbf{x}_s) \pi_s(\mathbf{x}_s)}{\pi_{s+1|s}(\mathbf{x}_{s+1})}. \quad 13.$$

This shows that under the smoothing distribution, the state is still a Markov process with backward transitions that are proportional to the product of the marginal forward transitions and the filter densities. Thus, given a particle approximation to π_s , (\mathbf{x}_s^i, w_s^i) , we can construct a particle approximation to $p(\mathbf{x}_s | \mathbf{x}_{s+1}, \mathbf{y}_{1:s})$. This will have the same particles as the filter approximation, but with modified weights that are proportional $w_s^i p(\mathbf{x}_{s+1} | \mathbf{x}_s^i)$. If we have run the particle filter forward in time and stored the particle approximations to all filtering distributions, we then have a backward simulation step:

1. Simulate $\mathbf{x}_{t|t}$ from the particle approximation to $\pi_t(d\mathbf{x}_t)$.
2. For $s = t - 1, \dots, 0$, simulate $\mathbf{x}_{s|t}$ conditional on $\mathbf{x}_{s+1|t}$ from the discrete distribution that assigns probability proportional to $w_s^i p(\mathbf{x}_{s+1|t} | \mathbf{x}_s^i)$ to value \mathbf{x}_s^i .

The cost of simulating one realization of the trajectory is $O(tN)$, and the cost of simulating a sample of N trajectories is $O(tN^2)$.

To obtain a weighted approximation of the marginal smoothing density, we integrate out the state at time $s + 1$:

$$\pi_{s|t}(\mathbf{x}_s) = \pi_s(\mathbf{x}_s) \int \frac{p(\mathbf{x}_{s+1} | \mathbf{x}_s)}{\pi_{s+1|s}(\mathbf{x}_{s+1})} \pi_{s+1|t}(\mathbf{x}_{s+1}) d\mathbf{x}_{s+1}. \quad 14.$$

Thus, given a particle approximation to π_s , (\mathbf{x}_s^i, w_s^i) , and one to $\pi_{s+1|t}$, $(\mathbf{x}_{s+1|t}^i, w_{s+1|t}^i)$, we can construct a particle approximation to $\pi_{s|t}$ (Hürzeler & Künsch 1998, Godsill et al. 2004). As in the above algorithm, it has the same particles as the filter approximation, but with new weights

$$w_{s|t}^i = w_s^i \sum_{j=1}^N w_{s+1|t}^j \left(\frac{p(\mathbf{x}_{s+1|t}^j | \mathbf{x}_s^i)}{\sum_{k=1}^N w_s^k p(\mathbf{x}_{s+1|t}^j | \mathbf{x}_s^k)} \right) \text{ for } i = 1, \dots, N.$$

The calculation of smoothing weights at time s involves considering all pairs of particles at times s and $s + 1$, and hence has an $O(N^2)$ cost. If we wish to calculate the smoothing distribution at time $t - L$, then the cost of the smoothing iterations will be $O(LN^2)$. We get for free all smoothing distributions from time t to time $t - L$.

We can see the improvement that the forward-backward smoother gives by comparing the plots from the right column of **Figure 3** to those in the left column. In particular these show how the forward-backward smoother is able to maintain particle diversity, and thus give a reasonable approximation to the smoothing distribution, at all time-points.

A closely related particle smoothing algorithm is the two-filter smoother (Kitagawa 1996). This involves running two independent particle filters, one forward in time and one backward in time. The output of these filters at any time-point s can then be combined to obtain a particle approximation to the smoothing distribution. Readers are directed to Briers et al. (2010) for more detail and to Hürzeler & Künsch (1998), Fearnhead et al. (2010), and Douc et al. (2011) for importance sampling and rejection sampling approaches to reduce the complexity of either the forward-backward smoother or the two-filter smoother to linear, rather than quadratic, in the number of particles.

6.2. Ensemble Kalman Smoothing

The trivial adaptation of the particle filter that uses the trajectory $\mathbf{x}_{0:t}$ instead of the value \mathbf{x}_t at time t also works for the ensemble Kalman filter (see Evensen 2003, appendix D). The stochastic version uses, in addition to Equation 12, the following update for the particles $\mathbf{x}_{s|t}^i$ at times $s < t$:

$$\mathbf{x}_{s|t}^i = \mathbf{x}_{s|t-1}^i + \widehat{\mathbf{K}}_{s,t}(\mathbf{y}_t - \mathbf{H}\mathbf{x}_t^i + \boldsymbol{\varepsilon}_t^i),$$

where the cross-gain $\widehat{\mathbf{K}}_{s,t}$ is based on an estimate of the cross-covariance between $(\mathbf{x}_{s|t}^i)$ and (\mathbf{x}_t^i) , and $\boldsymbol{\varepsilon}_t^i$ is the same artificial noise that occurs in Equation 12. The weak point of this method is the estimation of the large number of cross-covariances.

There is also an analogue of the forward-backward marginal particle smoother (see Stroud et al. 2010). In a linear Gaussian model, the recursion (Equation 14) allows us to express the first and second moments of $\pi_{s|t}$ in terms of the first and second moments of $\pi_{s+1|t}$ and π_s . Using this, one can derive an approximate sample $(\mathbf{x}_{s|t}^i)$ by a transformation of the samples (\mathbf{x}_s^i) and $(\mathbf{x}_{s+1|t}^i)$.

In numerical weather prediction, so-called four-dimensional variational data assimilation is used frequently. It computes the posterior mode of $\pi_{t-L:t}$ numerically, but it lacks uncertainty quantification. Hybrid methods that combine variational data assimilation with ensemble Kalman filters have also been developed (see Bannister 2017).

7. PARAMETER ESTIMATION

So far we have ignored the issue of parameter estimation in the filtering and smoothing problems. The algorithms we have presented have been suitable for inference conditional on knowing the parameter values of the underlying model. We now turn to problems in which the parameters are unknown. We denote the vector of parameters by $\boldsymbol{\theta}$, and we write $P_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\theta}}$ for the transition distribution of the state model and the likelihood function, respectively. We can still apply the filtering recursions, for example, Equations 7 and 8, but these will be conditional on a parameter value. As such we write, for example, $\pi_t(d\mathbf{x}_t | \boldsymbol{\theta})$ as the filtering distribution conditional on a given parameter value $\boldsymbol{\theta}$.

We consider two different situations where we wish to estimate parameters. The first is on-line parameter estimation and the second is batch estimation. For the latter we only consider a relatively recent class of algorithms, particle Markov chain Monte Carlo (MCMC), which embeds a particle filter within an MCMC algorithm. Our overview of online parameter estimation methods is deliberately brief, and the reader is referred to Kantas et al. (2015) for a recent article length review of both online approaches and related approaches to batch estimation. Kantas et al. (2015) discuss particle MCMC methods only briefly, hence our stronger focus on those methods here.

7.1. Online Parameter Estimation

Issues and methods for online parameter estimation within a particle filter are closely linked to those for particle smoothing. This stems from the fact that many quantities, such as the likelihood or score function, or even the posterior distribution for the parameters, can be written as expectations with respect to the smoothing distribution. The first link between the two problems is that, just as for smoothing, there is a trivial extension to a particle filter that can deal directly with estimating the parameters, but that is rarely useful in practice.

This extension applies when we have a prior distribution for the parameter. In this case we can extend the state of our model to incorporate the parameter vector. So we have a new state,

$\mathbf{x}'_t = (\mathbf{x}_t, \boldsymbol{\theta})$, say. We can trivially write down the state evolution and observation model for \mathbf{x}'_t and apply a particle filter to approximate the filtering distribution $\pi_t(d\mathbf{x}'_t)$. However, the dynamics of this particle filter will leave the parameter component unchanged at each iteration. As a consequence of this deterministic update, the number of distinct parameter values of the particles can only decrease at each iteration, and often the filter's approximation to the posterior for $\boldsymbol{\theta}$ reduces very quickly to having a handful, or even just one, distinct particle value. There have been three main approaches to overcome the particle degeneracy that necessarily occurs in this simple method.

The simplest way to avoid this degeneracy is to break the ties by adding small random noise to the parameter component of the particles after resampling. This effectively means that we use a kernel density approximation of the filter distribution instead of a mixture of point masses. The most effective version of this idea is that of Liu & West (2001), which shrinks the filter particles toward their mean before adding the noise. This ensures that the kernel density approximation has both the same mean and variance as the original particle approximation. Without this shrinkage the approximation of the kernel density estimation leads to an increase in the variance of our approximation of the posterior at each iteration. These can accumulate and lead to substantial overestimation of the parameter uncertainty. The algorithm of Liu & West (2001) has been shown to perform well in some applications, but it lacks any theoretical guarantees and has tuning parameters, such as the kernel bandwidth, that can be hard to choose.

An alternative approach is to use MCMC moves within the particle filter to sample new parameter values for each particle using an MCMC kernel that has the current posterior distribution as its invariant distribution. The most common choice of MCMC kernel is a Gibbs kernel that samples a new $\boldsymbol{\theta}$ for a particle from the parameter's conditional distribution given the particle's stored trajectory. In many situations this distribution depends on the trajectory through some low-dimensional summary statistics, and we need only store and update these summaries, as opposed to storing the full trajectory. The initial idea of using MCMC with a particle filter comes from Fearnhead (1998), though the original use of such MCMC moves to update parameters was in Gilks & Berzuini (2001), and the use of summary statistics was suggested by Storvik (2002) and Fearnhead (2002). Recently the general idea of using MCMC to update parameter values for models where sufficient statistics exist has been termed particle learning (Carvalho et al. 2010). While using MCMC steps within the filter does reduce the problem of degeneracy within the particle filter, it does not remove it, because the updates for the parameters depend on summaries of the trajectory of the state. As mentioned above, the particle filter's approximation to the smoothing distribution of the trajectory also degenerates (see section 7.2 of Kantas et al. 2015 for a thorough, empirical evaluation of this method).

The third approach is to use some form of stochastic approximation method. The idea is to have a current estimate of the parameter at each iteration. The particle filter update at iteration t is performed conditional on the current parameter estimate, $\hat{\boldsymbol{\theta}}_t$. Simultaneously, the particle filter is used to estimate the score function, that is, the gradient of the log-likelihood, at $\hat{\boldsymbol{\theta}}_t$, and this gradient information is used to update the estimate of the parameter. Poyiadjis et al. (2011), Nemeth et al. (2016a) and Olsson & Westerborn (2017) provide further details.

7.2. Particle MCMC

We now consider batch estimation of parameters. That is, we assume we have been given a fixed data set $\mathbf{y}_{1:n}$, from which we wish to estimate parameters of our model. We are no longer constrained to methods that are sequential or online, though methods for online parameter estimation,

together with simple extensions of them, can still be applied (see Poyiadjis et al. 2011, Kantas et al. 2015).

We focus on one specific class of methods, particle MCMC (Andrieu et al. 2010). These are MCMC methods that target the joint posterior of the parameter and the state and that use particle filter methods to develop novel, and hopefully efficient, proposal distributions. The most basic particle MCMC algorithm is a form of pseudomarginal MCMC algorithm (Andrieu & Roberts 2009) that leverages the fact that a particle filter gives an unbiased estimate of the likelihood (see Section 3.1). However, a particle filter approximation to a Gibbs sampler has also been developed. We describe these two approaches in turn. While particle MCMC was initially derived based on using particle filters to improve MCMC algorithms, it is also possible to embed particle MCMC methods within particle filters (Chopin et al. 2013, Fulop & Li 2013).

7.2.1. Particle Metropolis–Hastings. Assume we have a prior density $p(\theta)$ for our parameter vector. The posterior density $p(\theta \mid \mathbf{y}_{1:n})$ is then proportional to $p(\theta)L(\theta)$, where $L(\theta) = p(\mathbf{y}_{1:n} \mid \theta)$ is the likelihood. If we run a particle filter conditional on parameter θ , we can obtain an unbiased estimate $\hat{L}(\theta)$ of $L(\theta)$. The particle marginal Metropolis–Hastings algorithm simulates a Markov chain with state, $(\theta, \hat{L}(\theta))$, which consists of the parameter vector and an estimate of the likelihood. Given a proposal distribution for the parameter, with density $q(\theta' \mid \theta)$, the algorithm iterates the following steps:

0. Assume the current state at iteration i is (θ_i, \hat{L}_i) , where \hat{L}_i is an unbiased estimate of $L(\theta_i)$.
1. Propose a new parameter vector $\theta' \sim q(\theta' \mid \theta_i)$.
2. Run a particle filter, conditional on parameter vector θ' , to get \hat{L}' , an unbiased estimate of $L(\theta')$.
3. With probability

$$\min \left\{ 1, \frac{p(\theta')q(\theta_i \mid \theta')\hat{L}'}{p(\theta_i)q(\theta' \mid \theta_i)\hat{L}_i} \right\},$$

set $(\theta_{i+1}, \hat{L}_{i+1}) = (\theta', \hat{L}')$, otherwise $(\theta_{i+1}, \hat{L}_{i+1}) = (\theta_i, \hat{L}_i)$.

The acceptance probability in step 3 is just the standard Metropolis–Hastings acceptance probability, except the true likelihood values are replaced by their unbiased estimates. Perhaps surprisingly, the resulting algorithm still has the posterior for θ as the θ -marginal of its stationary distribution. To see this, denote by \mathbf{U} the set of random variables used within the particle filter, and let $L(\mathbf{U}, \theta)$ be the estimator of the likelihood we get from the particle filter run using random variables, \mathbf{U} , with parameter θ . Then the particle marginal Metropolis–Hastings algorithm is a standard Metropolis–Hastings algorithm with target density proportional to $p(\theta)\mathbb{P}(\mathrm{d}\mathbf{u})L(\mathbf{u}, \theta)$ and with proposal density $q(\theta' \mid \theta)\mathbb{P}(\mathrm{d}\mathbf{u})$, but that stores $(\theta, L(\mathbf{u}, \theta))$ rather than (θ, \mathbf{u}) . The marginal density of this target is the posterior density for θ as

$$\int p(\theta)\mathbb{P}(\mathrm{d}\mathbf{u})L(\mathbf{u}, \theta) = p(\theta)\mathbb{E}(L(\mathbf{U}, \theta)) = p(\theta)L(\theta)$$

by the unbiasedness of the estimator of the likelihood from the particle filter.

The quality of the approximation to the likelihood affects how well the resulting algorithm mixes (Andrieu & Vihola 2015, 2016). The benefit of particle MCMC is that, for well-mixing models, there is evidence that as the number of observations increases, we only need the number of particles used to increase linearly to maintain a similar level of mixing of the MCMC algorithm. This results in a computational complexity that is quadratic in the number of observations.

It is straightforward to extend the above particle marginal Metropolis–Hastings algorithm so as to obtain samples from the joint posterior for the parameter and the state, $\mathbf{x}_{0:n}$. In step 2, we

run a particle filter that stores the state trajectory for each particle and outputs both our unbiased likelihood estimate, \hat{L}' , and a sample trajectory. We then accept or reject the new parameter value, the unbiased estimate and the trajectory in step 3. Andrieu et al. (2010) provide further details.

There is flexibility in the above particle MCMC algorithm in terms of the choice of proposal distribution for the parameter, the number of particles to use in the particle filter algorithm, and the version of particle filter used. As we get better estimates of the likelihood, we may expect the particle MCMC algorithm to behave increasingly like an MCMC algorithm using the true likelihood values, so our choice for proposal distribution can be informed by experience from implementing standard MCMC algorithms. Better proposal distributions for the underlying exact MCMC algorithm should lead to better mixing of the particle MCMC algorithm. In particular, this has led to particle versions of Langevin algorithms that leverage the particle filter's ability to estimate gradient information so as to improve the proposal distribution (Dahlin et al. 2015, Nemeth et al. 2016b).

Theory also shows that the better the estimate of the likelihood, the more efficient the particle MCMC algorithm will be. This suggests using the most efficient particle filter algorithm available for a given problem. It also shows that increasing the number of particles will improve mixing. However, this comes with an increased computational cost of running the filter. There have now been a number of theoretical studies linked to choosing the optimal number of particles to trade off better mixing with increased computational cost. The first results for this were from Pitt et al. (2012), and these have been extended by Sherlock et al. (2015), Doucet et al. (2015), and Nemeth et al. (2016b). The main conclusion is that we should tune the number of particles so that the variance of our estimate of the log-likelihood is between 1 and 3 (this choice differs substantially from the optimal choice for general pseudomarginal MCMC algorithms; see Sherlock et al. 2017).

Recent theoretical work has shown that introducing correlation into the estimates of the likelihood across successive iterations can substantially improve mixing (Deligiannidis et al. 2015, Murray & Graham 2017). For particle MCMC the idea would then be to couple the randomness in the resampling and propagation steps of the particle filter, so that two successive runs of the filter, with similar parameter values, would generate similar particles and trajectories and hence similar estimates of the likelihood. This would reduce the variance in the ratio of likelihood estimates that appear in the acceptance probability, and hence improve the acceptance rate. Simulating such coupled particle filters is challenging, but Sen et al. (2017) and Jacob et al. (2016) present recent approaches.

7.2.2. Particle Gibbs. An alternative particle MCMC algorithm is based around using a particle filter to approximate a Gibbs update. Our target distribution is the joint posterior for the parameter, θ , and the trajectory of the state, $\mathbf{x}_{0:n}$. A Gibbs sampler would iterate between updating θ from its full conditional given $\mathbf{x}_{0:n}$ and then simulating $\mathbf{x}_{0:n}$ given θ . For many models the former update is relatively simple to perform, whereas the latter is intractable. A Gibbs algorithm that updates only one component \mathbf{x}_t at a time by Metropolis–Hastings steps would be feasible, but convergence is usually slow. For some models, data augmentation has been used successfully (see, e.g., Frühwirth-Schnatter et al. 2013), but it is often restricted to models with specific structure. The idea of particle Gibbs is to use a particle filter as a generic way of updating the whole path, $\mathbf{x}_{0:n}$. While the particle filter only samples from an approximation to the true conditional distribution of the path given θ and $\mathbf{y}_{1:n}$, the particle Gibbs algorithm is designed such that the resulting MCMC algorithm still has the true posterior as its stationary distribution.

The particle Gibbs sampler updates $\mathbf{x}_{0:n}$ by drawing from a distribution that depends not only on the current value θ but also on the current value of the state sequence, denoted $\mathbf{x}_{0:n}^{\text{cur}}$. Viewed as

such, the term particle Gibbs is, in fact, a slight misnomer—though it can be viewed as a Gibbs sampler on an extended state space. Drawing $\mathbf{x}_{0:n}$ given θ and $\mathbf{x}_{0:n}^{\text{cur}}$ is based on running a conditional particle filter algorithm such that one of the final particles has a trajectory that is identical to $\mathbf{x}_{0:n}^{\text{cur}}$. In particular this means that we only ever simulate $N - 1$ particles at each iteration of the conditional particle filter, as the remaining particle is set to the corresponding entry of $\mathbf{x}_{0:n}^{\text{cur}}$.

The conditional particle filter algorithm is as follows:

0. Condition on the current state trajectory $\mathbf{x}_{0:n}^{\text{cur}}$, and parameter value, θ . Set $\mathbf{x}_0^1 = \mathbf{x}_0^{\text{cur}}$ and independently simulate \mathbf{x}_i^j from $\pi_0(d\mathbf{x}_i | \theta)$ for $i = 2, \dots, N$. Set $t = 1$.
1. Resample: If $t > 1$, set $\tilde{\mathbf{x}}_{t-1}^i = \mathbf{x}_{t-1}^{A(i)}$ where $A(1) = 1$ and $\mathbb{P}(A(i) = j) = w_{t-1}^j$ for $i = 2, \dots, N$.
2. Propagate: Set $\mathbf{x}_t^1 = \mathbf{x}_t^{\text{cur}}$ and draw \mathbf{x}_t^i from $P_\theta(d\mathbf{x}_t | \tilde{\mathbf{x}}_{t-1}^i)$ for $i = 2, \dots, N$.
3. Reweight: Set $w_t^i \propto g_\theta(\mathbf{y}_t | \mathbf{x}_t^i)$ for $i = 1, \dots, N$. If $t < n$, set $t = t + 1$ and go to step 1.
4. Sample and output a particle and its associated trajectory at time n , with the probability of sampling particle i being w_n^i for $i = 1 \dots, N$.

Steps 0–3 are similar to the bootstrap filter of Section 3.1, except that at each iteration we fix the first particle to be the corresponding part of the trajectory we are conditioning on. Thus, at time n the trajectory of the first particle will be $\mathbf{x}_{0:n}^{\text{cur}}$, and there is a nonzero probability that the current trajectory does not change in the update. The dynamics of the filter depends only on the parameter value, and this is made explicit within the notation for steps 2 and 3. Andrieu et al. (2010) provide a proof that updating the state trajectory using a conditional particle filter update leaves $p(\theta, \mathbf{x}_{0:n} | \mathbf{y}_{1:n})$ invariant. Example output from the conditional particle filter is given in **Figure 4**.

As with particle Metropolis–Hastings algorithms, empirical and theoretical results suggest that as the number of observations, n , increases, we would need to increase the number of particles, N , linearly to maintain a fixed level of mixing of the resulting MCMC algorithm. The conditional particle filter update is more efficient at updating later values of the state than earlier ones, a property that is linked to the sample impoverishment of the simple smoothing algorithm where we store the trajectory of the particles in the filter (see the discussion in Section 7), and that can be seen in the top row of **Figure 4**. However, there are very strong results on the mixing of particle Gibbs if sufficiently many particles are used (Chopin & Singh 2015, Lindsten et al. 2015, Del Moral et al. 2016, Andrieu et al. 2018).

There have been a number of extensions to the particle Gibbs sampler. First, most improvements on the bootstrap filter can be applied to the conditional particle filter sampler. For example, balanced resampling can be used instead of multinomial resampling, and this improves mixing (Chopin & Singh 2015). However, care is needed, as resampling in step 1 above needs to be from the conditional distribution of the balanced resampling scheme, given that particle 1 must have at least one offspring (see Andrieu et al. 2010, appendix A for more details). It is also possible to extend the conditional particle filter update so as to use better proposal distributions, as in the auxiliary particle filter.

Second, it is possible to employ ideas from the forward-backward smoother to increase the mixing of the trajectory at early time-points (Whiteley 2010). The idea here is that, in step 4, rather than simulating a trajectory associated with one of the N particles at time n , we can use backward simulation to obtain a new trajectory given all the particles that have been stored at time 0 to n . As we simulate a single trajectory, the cost of the backward simulation is linear in N . A particular implementation of this idea has been termed ancestor sampling (Lindsten et al. 2014). After each iteration of the conditional particle filter algorithm, they sample a new ancestor for the first particle, that is, the particle from the conditioned path. This means that while there is still degeneracy of the paths in the conditional particle filter, this degenerate path is different from the conditioned path, and hence we get better mixing in the actual MCMC algorithm (see **Figure 4**

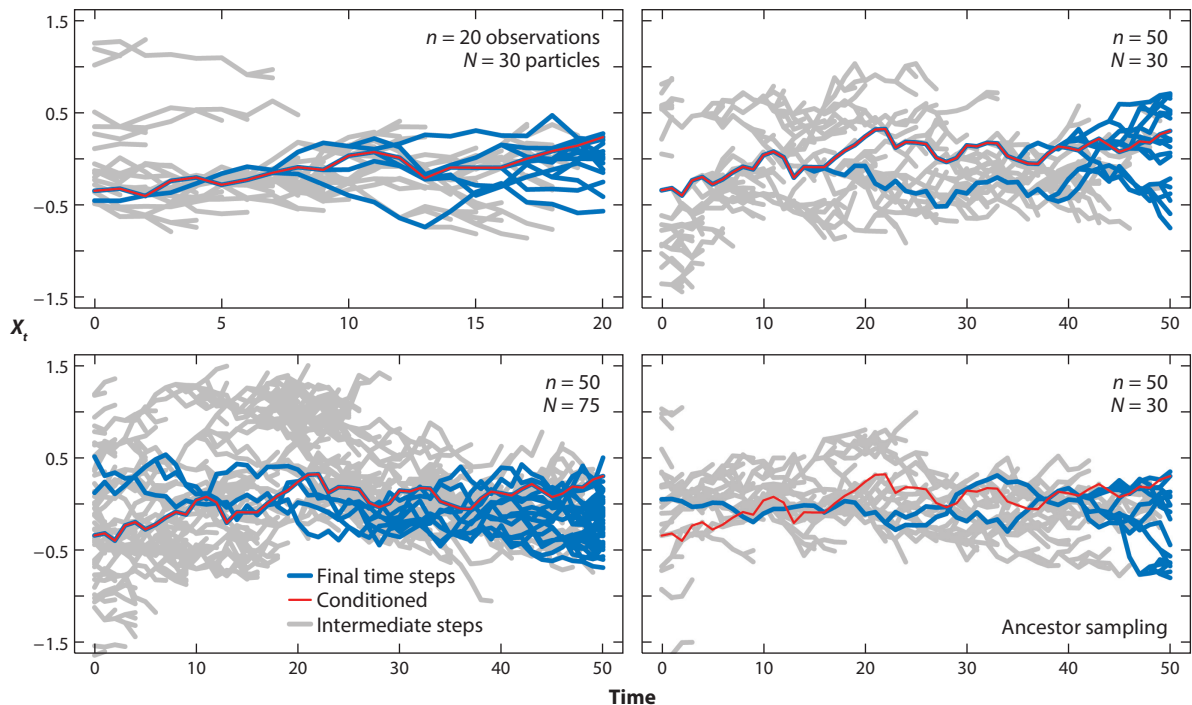


Figure 4

Output from the conditional particle filter for the stochastic volatility model. For each figure the conditioned path is in red, the paths of particles from intermediate steps of the filter are in gray, and the paths associated with the particles at the final time-step are in blue. The main issue with the conditional particle filter is that it can struggle to change the start of the state trajectory unless sufficient particles are used—compare the top right and bottom left plots. A rule of thumb is that we need to increase N linearly with any increase in n , hence the similar behavior we obtain for the two left plots. The use of ancestor sampling can overcome this problem—compare the two right plots. In particular, the sample path of the particles in the bottom right plot is different from that of the path we conditioned on because of the ancestor sampling.

for an example; for theoretical support for such algorithms, see Chopin & Singh 2015). It is also possible to use a conditional particle filter to update blocks of the trajectory, rather than the whole trajectory, and this can lead to an algorithm whose computational cost scales linearly, rather than quadratically, with the number of observations (Singh et al. 2017).

Third, even if the conditional particle filter update mixes well, the resulting particle Gibbs algorithm can be poor if there is strong dependence between the parameter and the trajectory. It is possible to overcome this by performing partial updates of the parameters within the conditional particle filter update (see Fearnhead & Meligkotsidou 2016).

8. SUMMARY AND OUTLOOK

State-space models provide a flexible framework for inference and prediction of complex, partially observed systems. We have given only a few examples in this review, and there are many more that could have been mentioned. In order to apply state-space models, one has to be able to solve challenging computational problems. We have reviewed the currently available Monte Carlo methods to solve these problems, including methods that originated in geophysical applications. The following points contain the main messages of this review.

SUMMARY POINTS

1. Filtering and data assimilation combine partial observation with a dynamical model to estimate latent states of a system.
2. Particle filters are completely general, but often suffer from sample depletion. Ensemble Kalman filters are more robust but rely on Gaussian assumptions.
3. Combinations of particle and MCMC methods are promising new developments for joint estimation of parameters and states.

FUTURE ISSUES


1. How do we best exploit the different strengths of particle and ensemble Kalman filters to improve filtering of high-dimensional system with nonlinear and non-Gaussian features? Are there versions of particle filters that can move particles, like the ensemble Kalman filter does, instead of reweighting them?
2. There is a need to develop theory for better understanding of localized ensemble Kalman filters in realistic settings where the number of particles is much smaller than the dimension of the state.
3. How do we design particle filter and related algorithms to best take advantage of modern computer architectures?

DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

ACKNOWLEDGMENTS

We are grateful to Sylvain Robert for the permission to reproduce **Figure 2** from his thesis and for his help producing the animations of the Lorenz 96 model in the **Supplemental Appendix**. We would like to thank Christophe Andrieu, Nicolas Chopin, Sylvain Robert, Chris Sherlock, and an anonymous reviewer for helpful comments on an earlier version of this review article. Paul Fearnhead was funded by the EPSRC program grant EP/K014463 (*i-like*).

 [Supplemental Material](#)

LITERATURE CITED

- Aeberhard WH, Mills Flemming JM, Nielsen A. 2017. Review of state-space models for fisheries science. *Annu. Rev. Stat. Appl.* 5:215–35
- Agapiou S, Papaspiliopoulos O, Sanz-Alonso D, Stuart A. 2017. Importance sampling: computational complexity and intrinsic dimension. *Stat. Sci.* 32:405–31
- Andrieu C, Doucet A, Holenstein R. 2010. Particle Markov chain Monte Carlo (with discussion). *J. R. Stat. Soc. B* 62:269–342
- Andrieu C, Lee A, Vihola M. 2018. Uniform ergodicity of the iterated conditional SMC and geometric ergodicity of particle Gibbs samplers. *Bernoulli* 24:842–72
- Andrieu C, Roberts GO. 2009. The pseudo-marginal approach for efficient Monte Carlo computations. *Ann. Stat.* 37:697–725

- Andrieu C, Vihola M. 2015. Convergence properties of pseudo-marginal Markov chain Monte Carlo algorithms. *Ann. Appl. Probab.* 25:1030–77
- Andrieu C, Vihola M. 2016. Establishing some order amongst exact approximations of MCMCs. *Ann. Appl. Probab.* 26:2661–96
- Atar R. 2011. Exponential decay rate of the filter's dependence on the initial distribution. In *The Oxford Handbook of Nonlinear Filtering*, ed. D Crisan, B Rozovskiĭ, pp. 299–318. Oxford, UK: Oxford Univ. Press
- Bannister RN. 2017. A review of operational methods of variational and ensemble-variational data assimilation. *Q. J. R. Meteorol. Soc.* 143:607–33
- Bauer P, Thorpe A, Brunet G. 2015. The quiet revolution of numerical weather prediction. *Nature* 525:47–55
- Baum LE, Petrie T, Soules G, Weiss N. 1970. A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.* 41:164–71
- Bengtsson T, Bickel P, Li B. 2008. Curse-of-dimensionality revisited: collapse of the particle filter in very large scale systems. In *Probability and Statistics: Essays in Honor of David A. Freedman*, ed. D Nolan, T Speed, pp. 316–34. Bethesda, MD: Inst. Math. Stat.
- Bengtsson T, Snyder C, Nychka D. 2003. Toward a nonlinear ensemble filter for high-dimensional systems. *J. Geophys. Res.* 108:8775
- Beskos A, Crisan D, Jasra A, Kamatani K, Zhou Y. 2017. A stable particle filter for a class of high-dimensional state-space models. *Appl. Probab.* 49:24–48
- Briers M, Doucet A, Maskell S. 2010. Smoothing algorithms for state-space models. *Ann. Inst. Stat. Math.* 62:61–89
- Bunch P, Godsill S. 2016. Approximations of the optimal importance density using Gaussian particle flow importance sampling. *J. Am. Stat. Assoc.* 111:748–62
- Butala MD, Frazin RA, Chen Y, Kamalabadi F. 2009. Tomographic imaging of dynamic objects with the ensemble Kalman filter. *IEEE Trans. Image Proc.* 18:1573–87
- Caflich RE, Morokoff W, Owen A. 1997. Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension. *J. Comput. Finance* 1:27–46
- Calvet LE, Czellar V, Ronchetti E. 2015. Robust filtering. *J. Am. Stat. Assoc.* 110:1591–606
- Cappé O, Moulines E, Rydén T. 2005. *Inference in Hidden Markov Models*. Berlin: Springer
- Carpenter J, Clifford P, Fearnhead P. 1999. An improved particle filter for non-linear problems. *IEEE Proc. Radar Sonar Navig.* 146:2–7
- Carvalho CM, Johannes MS, Lopes HF, Polson NG. 2010. Particle learning and smoothing. *Stat. Sci.* 25:88–106
- Chatterjee S, Diaconis P. 2017. The sample size required in importance sampling. *Ann. Appl. Probab.* In press
- Chen R, Liu JS. 2000. Mixture Kalman filters. *J. R. Stat. Soc. B* 62:493–508
- Chopin N. 2004. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *Ann. Stat.* 32:2385–411
- Chopin N, Jacob PE, Papaspiliopoulos O. 2013. SMC²: an efficient algorithm for sequential analysis of state space models. *J. R. Stat. Soc. B* 75:397–426
- Chopin N, Singh SS. 2015. On particle Gibbs sampling. *Bernoulli* 21:1855–83
- Chustagulprom N, Reich S, Reinhardt M. 2016. A hybrid ensemble transform filter for nonlinear and spatially extended dynamical systems. *SIAM/ASA J. Uncertain. Quantif.* 4:592–608
- Cox D. 1981. Statistical analysis of time series: some recent developments. *Scand. J. Stat.* 8:93–115
- Crisan D. 2001. Particle filters—a theoretical perspective. In *Sequential Monte Carlo Methods in Practice*, ed. A Doucet, N de Freitas, N Gordon, pp. 17–41. New York: Springer-Verlag
- Crisan D, Doucet A. 2002. A survey of convergence results on particle filtering methods for practitioners. *IEEE Trans. Signal Proc.* 50:736–46
- Dahlin J, Lindsten F, Schön TB. 2015. Particle Metropolis-Hastings using gradient and Hessian information. *Stat. Comput.* 25:81–92
- Del Moral P. 2004. Feynman-Kac formulae: genealogical and interacting particle systems with applications. New York: Springer
- Del Moral P, Doucet A, Jasra A. 2006. Sequential Monte Carlo samplers. *J. R. Stat. Soc. B* 68:411–36
- Del Moral P, Kohn R, Patras F. 2016. On Feynman-Kac and particle Markov chain Monte Carlo models. *Ann. Inst. Henri Poincaré Probab. Stat.* 52:1687–733

- Deligiannidis G, Doucet A, Pitt MK. 2015. The correlated pseudo-marginal method. arXiv:1511.04992 [stat.CO]
- Douc R, Garivier A, Moulines E, Olsson J. 2011. Sequential Monte Carlo smoothing for general state space hidden Markov models. *Ann. Appl. Probab.* 21:2109–45
- Douc R, Moulines E, Olsson J. 2014a. Long-term stability of sequential Monte Carlo methods under verifiable conditions. *Ann. Appl. Probab.* 24:1767–802
- Douc R, Moulines E, Stoffer DS. 2014b. *Nonlinear Time Series: Theory, Methods and Applications with R Examples*. Boca Raton, FL: Chapman and Hall/CRC
- Doucet A, Briers M, Sénécal S. 2006. Efficient block sampling strategies for sequential Monte Carlo Methods. *J. Comput. Gr. Stat.* 15:693–711
- Doucet A, Godsill SJ, Andrieu C. 2000. On sequential Monte Carlo sampling methods for Bayesian filtering. *Stat. Comput.* 10:197–208
- Doucet A, Johansen AM. 2011. A tutorial on particle filtering and smoothing: fifteen years later. In *The Oxford Handbook of Nonlinear Filtering*, ed. D Crisan, Rozovski, pp. 656–704. Oxford, UK: Oxford Univ. Press
- Doucet A, Pitt MK, Deligiannidis G, Kohn R. 2015. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika* 102:295–313
- Durbin J, Koopman SJ. 2001. *Time Series Analysis by State Space Methods*. Oxford, UK: Clarendon
- Embrechts P, Klüppelberg C, Mikosch T. 1997. *Modelling Extremal Events: For Insurance and Finance*. Berlin: Springer
- Evensen G. 1994. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *J. Geophys. Res.* 99:10143–62
- Evensen G. 2003. The ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dyn.* 53:343–67
- Evensen G. 2007. *Data Assimilation: The Ensemble Kalman Filter*. New York: Springer
- Fearnhead P. 1998. *Sequential Monte Carlo methods in filter theory*. PhD Thesis, Oxford Univ. <http://www.maths.lancs.ac.uk/~fearnhea/thesis.ps>
- Fearnhead P. 2002. MCMC, sufficient statistics and particle filters. *J. Comput. Gr. Stat.* 11:848–62
- Fearnhead P. 2005. Using random quasi-Monte Carlo within particle filters, with application to financial time series. *J. Comput. Gr. Stat.* 14:751–69
- Fearnhead P, Meligkotsidou L. 2016. Augmentation schemes for particle MCMC. *Stat. Comput.* 26:1293–306
- Fearnhead P, Wyncoll D, Tawn J. 2010. A sequential smoothing algorithm with linear computational cost. *Biometrika* 97:447–64
- Frei M. 2013. *Ensemble Kalman filtering and generalizations*. PhD Thesis, ETH Zurich
- Frei M, Künsch HR. 2013. Bridging the ensemble Kalman and particle filter. *Biometrika* 100:781–800
- Frühwirth-Schnatter S, Frühwirth R, Held L, Rue H. 2009. Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data. *Stat. Comput.* 19:479–92
- Fulop A, Li J. 2013. Efficient learning via simulation: a marginalized resample-move approach. *J. Econ.* 176:146–61
- Gerber M, Chopin N. 2015. Sequential quasi Monte Carlo. *J. R. Stat. Soc. B* 77:509–79
- Gilks WR, Berzuini C. 2001. Following a moving target—Monte Carlo inference for dynamic Bayesian models. *J. R. Stat. Soc. B* 63:127–46
- Godsill SJ, Doucet A, West M. 2004. Monte Carlo smoothing for non-linear time series. *J. Am. Stat. Assoc.* 99:156–68
- Gordon N, Salmond D, Smith AFM. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proc. F* 140:107–13
- Hunt BR, Kostelich EJ, Szunyogh I. 2007. Efficient data assimilation for spatiotemporal chaos: a local ensemble transform Kalman filter. *Physica D* 230:112–26
- Hürzeler M, Künsch HR. 1998. Monte Carlo approximations for general state-space models. *J. Comput. Gr. Stat.* 7:175–93
- Jacob PE, Lindsten F, Schön TB. 2016. Coupling of particle filters. arXiv:1606.01156 [stat.ME]
- Jacob PE, Murray LM, Rubenthaler S. 2015. Path storage in the particle filter. *Stat. Comput.* 25:487–96
- Johansen AM. 2009. SMCTC: sequential Monte Carlo in C++. *J. Stat. Softw.* 30:1–41

- Kantas N, Doucet A, Singh SS, Maciejowski J, Chopin N. 2015. On particle methods for parameter estimation in state-space models. *Stat. Sci.* 30:328–51
- Kelly D, Majda AJ, Tong X. 2015. Concrete ensemble Kalman filters with rigorous catastrophic filter divergence. *PNAS* 112:10589–94
- Kim S, Shephard N, Chib S. 1998. Stochastic volatility: likelihood inference and comparison with ARCH models. *Rev. Econ. Stud.* 65:361–93
- Kitagawa G. 1996. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *J. Comput. Gr. Stat.* 5:1–25
- Künsch HR. 2005. Recursive Monte Carlo filters: algorithms and theoretical analysis. *Ann. Stat.* 33:1983–2021
- Künsch HR. 2013. Particle filters. *Bernoulli* 19:1391–403
- Lauritzen S. 1996. *Graphical Models*. Oxford, UK: Clarendon
- Le Gland F, Monbet V, Tran V. 2011. Large sample asymptotics for the ensemble Kalman filter. In *The Oxford Handbook of Nonlinear Filtering*, ed. D Crisan, B Rozovski, pp. 598–634. Oxford, UK: Oxford Univ. Press
- Lee A, Whiteley N. 2016. Forest resampling for distributed sequential Monte Carlo. *Stat. Anal. Data Min. ASA Data Sci. J.* 9:230–48
- Lindsten F, Douc R, Moulines E. 2015. Uniform ergodicity of the particle Gibbs sampler. *Scand. J. Stat.* 42:775–97
- Lindsten F, Jordan MI, Schön TB. 2014. Particle Gibbs with ancestor sampling. *J. Mach. Learn. Res.* 15:2145–84
- Liu J, West M. 2001. Combined parameter and state estimation in simulation based filtering. In *Sequential Monte Carlo in Practice*, ed. A Doucet, JFG de Freitas, NJ Gordon. New York: Springer
- Lorenz EN. 1963. Deterministic non-periodic flows. *J. Atmos. Sci.* 20:130–41
- Lorenz EN, Emanuel KA. 1998. Optimal sites for supplementary weather observations: simulations with a small model. *J. Atmos. Sci.* 55:399–414
- Majda AJ, Harlim J. 2012. *Filtering Complex Turbulent Systems*. Cambridge, UK: Cambridge Univ. Press
- Michaud N, de Valpine P, Turek D, Paciorek CJ. 2017. Sequential Monte Carlo methods in the nimble R package. arXiv:1703.06206 [stat.CO]
- Murray I, Graham M. 2016. Pseudo-marginal slice sampling. *Proc. 19th Int. Conf. Artif. Intell. Stat.*, pp. 911–19
- Murray LM. 2015. Bayesian state-space modelling on high-performance hardware using LibBi. *J. Stat. Softw.* 67
- Nemeth C, Fearnhead P, Mihaylova L. 2016a. Particle approximations of the score and observed information matrix for parameter estimation in state-space models with linear computational cost. *J. Comput. Gr. Stat.* 25:1138–57
- Nemeth C, Sherlock C, Fearnhead P. 2016b. Particle Metropolis adjusted Langevin algorithms. *Biometrika* 103:701–17
- Nerger L, Hiller W. 2013. Software for ensemble-based data assimilation systems—implementation strategies and scalability. *Comput. Geosci.* 55:110–18
- Niederreiter H. 1978. Quasi-Monte Carlo methods and pseudo-random numbers. *Bull. Am. Math. Soc.* 84:957–1041
- Nielsen A, Berg CW. 2014. Estimation of time-varying selectivity in stock assessments using state space models. *Fisheries Res.* 158:96–101
- Olsson J, Westerborn J. 2017. Efficient particle-based online smoothing in general hidden Markov models: the PaRIS algorithm. *Bernoulli* 23:1951–96
- Ott E, Hunt BR, Szunyogh I, Zimin AV, Kostelich E, et al. 2004. A local ensemble Kalman filter for atmospheric data assimilation. *Tellus A* 56:415–28
- Owen AB. 1998. Monte Carlo extensions of quasi-Monte Carlo. *WSC '98 Proc. 30th Conf. Winter Sim., Washington, DC, Dec. 13–16*, pp. 571–78. Los Alamitos, CA: IEEE Comput. Soc.
- Pitt MK, dos Santos Silva R, Giordani P, Kohn R. 2012. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *J. Econom.* 171:134–51
- Pitt MK, Shephard N. 1999. Filtering via simulation: auxiliary particle filters. *J. Am. Stat. Assoc.* 94:590–99
- Polson NG, Stroud JR, Müller P. 2008. Practical filtering with sequential parameter learning. *J. R. Stat. Soc. B* 70:413–28

- Poyiadjis G, Doucet A, Singh SS. 2011. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika* 98:65–80
- Rebeschini P, van Handel R. 2015. Can local particle filters beat the curse of dimensionality? *Ann. Appl. Probab.* 25:2809–66
- Reich S. 2013. A nonparametric ensemble transform method for Bayesian inference. *SIAM J. Sci. Comput.* 35:A2013–24
- Reich S, Cotter C. 2015. *Probabilistic Forecasting and Bayesian Data Assimilation*. Cambridge, UK: Cambridge Univ. Press
- Robert S. 2017. *Ensemble Kalman particle filters for high-dimensional data assimilation*. PhD Thesis, ETH Zurich
- Robert S, Künsch HR. 2017. Localizing the ensemble Kalman particle filter. *Tellus A* 69:1
- Sanz-Alonso D. 2016. Importance sampling and necessary sample size: an information theory approach. arXiv:1608.08814 [stat.CO]
- Sen D, Thiery A, Jasra A. 2017. On coupling particle filter trajectories. *Stat. Comput.* <https://doi.org/10.1007/s11222-017-9740-z>
- Sherlock C, Thiery A, Lee A. 2017. Pseudo-marginal Metropolis–Hastings using averages of unbiased estimators. *Biometrika* 104:727–34
- Sherlock C, Thiery AH, Roberts GO. 2015. On the efficiency of pseudo marginal random walk Metropolis algorithms. *Ann. Stat.* 43:238–75
- Singh SS, Lindsten F, Moulines E. 2017. Blocking strategies and stability of particle Gibbs samplers. *Biometrika* 104:971–86
- Stone LD, Streit RL, Corwin TL, Bell KL. 2014. *Bayesian Multiple Target Tracking*. Boston: Artech House
- Storvik G. 2002. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Trans. Signal Proc.* 50:281–89
- Stroud JR, Stein ML, Lesht BM, Schwab DJ, Beletsky D. 2010. An ensemble Kalman filter and smoother for satellite data assimilation. *J. Am. Stat. Assoc.* 105:978–99
- Tippett MK, Anderson JL, Bishop CH, Hamill TM, Whitaker JS. 2003. Ensemble square root filters. *Mon. Weather Rev.* 131:1485–90
- Tong X, Majda AJ, Kelly D. 2016. Nonlinear stability of the ensemble Kalman filter with adaptive covariance inflation. *Commun. Math. Sci.* 14:1283–313
- Van Leeuwen P. 2010. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Q. J. R. Meteorol. Soc.* 136:1991–99
- Vergé C, Dubarry C, Del Moral P, Moulines E. 2015. On parallel implementation of sequential Monte Carlo methods: the island particle model. *Stat. Comput.* 25:243–60
- Whiteley N. 2010. Discussion on particle Markov chain Monte Carlo methods. *J. R. Stat. Soc. B* 72:306–7