An efficient particle-based method for maximum likelihood estimation in nonlinear state-space models

Thi Tuyet Trang Chau, Pierre Ailliot, Valérie Monbet, Pierre Tandeo

▶ **To cite this version:**

Thi Tuyet Trang Chau, Pierre Ailliot, Valérie Monbet, Pierre Tandeo. An efficient particle-based method for maximum likelihood estimation in nonlinear state-space models. 2018. hal-01708631v2

HAL Id: hal-01708631

https://inria.hal.science/hal-01708631v2

Preprint submitted on 14 Feb 2018 (v2), last revised 18 Apr 2018 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An efficient particle-based method for maximum likelihood estimation in nonlinear state-space models

T. T. T. Chau[a*], P. Ailliot[b,c], V. Monbet[a], and P. Tandeo[c]

[a] *IRMAR and INRIA, Rennes, France*
[b] *University of Western Brittany and LMBA, Brest, France*
[c] *IMT-Atlantique and Lab-STICC, Brest, France*

*Correspondence to: T. T. T. Chau , IRMAR, University of Rennes 1, 35042 Rennes, France.
Email: thi-tuyet-trang.chau@univ-rennes1.fr

Data assimilation methods aim at estimating the state of a system by combining observations with a physical model. When sequential data assimilation is considered, the joint distribution of the latent state and the observations is described mathematically using a state-space model, and filtering or smoothing algorithms are used to approximate the conditional distribution of the state given the observations. The most popular algorithms in the data assimilation community are based on the Ensemble Kalman Filter and Smoother (EnKF/EnKS) and its extensions. In this paper we investigate an alternative approach where a Conditional Particle Filter (CPF) is combined with Backward Simulation (BS). This allows to explore efficiently the latent space and simulate quickly relevant trajectories of the state conditionally to the observations. We also tackle the difficult problem of parameter estimation. Indeed, the models generally involve statistical parameters in the physical models and/or in the stochastic models for the errors. These parameters strongly impact the results of the data assimilation algorithm and there is a need for an efficient method to estimate them. Expectation-Maximization (EM) is the most classical algorithm in the statistical literature to estimate the parameters in models with latent variables. It consists in updating sequentially the parameters by maximizing a likelihood function where the state is approximated using a smoothing algorithm. In this paper, we propose an original Stochastic Expectation-Maximization (SEM) algorithm combined to the CPF-BS smoother to estimate the statistical parameters. We show on several toy models that this algorithm provides, with reasonable computational cost, accurate estimations of the statistical parameters and the state in highly nonlinear state-space models, where the application of EM algorithms using EnKS is limited. We also provide a Python source code of the algorithm.

Key Words: EM algorithm; conditional particle filtering; backward simulation; nonlinear models; data assimilation

Received . . .

1. Introduction

Data assimilation has been applied in various fields such as oceanography, meteorology or navigation (Ghil and Malanotte-Rizzoli 1991; Work *et al.* 2010; Carrassi *et al.* 2017) to reconstruct dynamical processes given observations. When sequential data assimilation is used, a state-space model is considered. It is defined sequentially for $t = 1 : T$ by

$$\begin{cases} x_t = \mathcal{M}_\theta\left(x_{t-1},\ \eta_t\right) \\ y_t = \mathcal{H}_\theta\left(x_t,\ \epsilon_t\right) \end{cases} \tag{1}$$

where (x_t, y_t) belong to the state and observation spaces $(\mathcal{X}, \mathcal{Y})$ and (η_t, ϵ_t) are independent white noise sequences with covariance matrices denoted respectively Q and R. The functions $\mathcal{M}_\theta$ and $\mathcal{H}_\theta$ describe respectively the evolution of the state (x_t) and the transformation between the state and the observations (y_t). We denote $\theta \in \Theta$ the vector of parameters. For instance, θ may contain physical parameters in the models $(\mathcal{M}_\theta, \mathcal{H}_\theta)$ and error covariances (Q, R).

Given a fixed vector θ and T measurements $y_{1:T} = (y_1, ..., y_T)$, data assimilation schemes relate to compute filtering distributions $\{p_\theta(x_t|y_{1:t})\}_{t=1:T}$ or smoothing distributions $\{p_\theta(x_t|y_{1:T})\}_{t=1:T}$. However, it is often difficult to identify a reasonable value of θ. This is due to the diversity of observation sources, the effect of physical terms and model complexity, or numerical failures (Dreano *et al.* 2017; Zhu *et al.* 2017). And incorrect values of θ may lead to bad reconstruction results. This is

© 0000

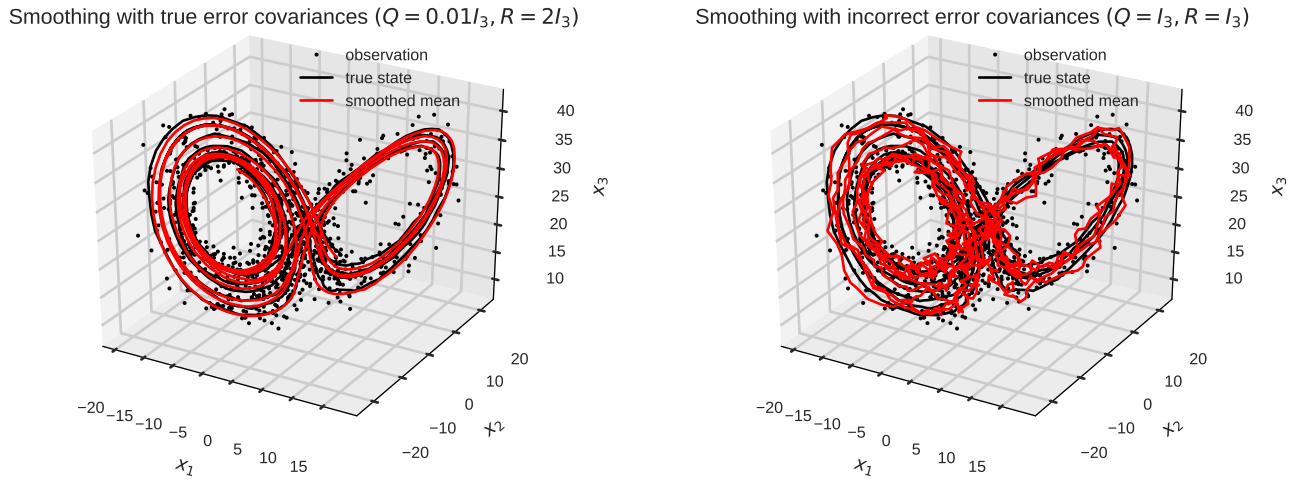

Figure 1. Impact of parameter values on smoothing distributions for the Lorenz-63 model. The true state (black curve) and observations (black points) have been simulated with $\theta = (Q, R) = (0.01I_3, 2I_3)$. The mean of the smoothing distributions (read curve) are computed using a standard particle smoother (Douc *et al.* 2009) with 100 particles. Results obtained with the true parameter values $\theta^* = (0.01I_3, 2I_3)$ (left panel) and wrong parameter values $\tilde{\theta} = (I_3, I_3)$ (right panel).

illustrated on Figure 1 using the Lorenz-63 model (see Section 3.3 for a formal definition). Smoothing with true parameter value provides a good approximation of the true trajectory (left panel) whereas the trajectory obtained with wrong parameter value is noisy and biased (right panel). This illustration emphasizes the role of parameter estimation in data assimilation context. A nice explanation of the problem is also given in Berry and Sauer (2013).

One common approach to estimate parameters in data assimilation community is based on innovation statistics (Miyoshi 2011; Berry and Sauer 2013; Zhen and Harlim 2015; Zhu *et al.* 2017) whose formulas were first given in Desroziers *et al.* (2005). Although these methods permit an adaptive estimation of the error covariances Q and R, physical parameters of nonlinear dynamical models are difficult to estimate with this approach. An alternative is to implement likelihood-based methods. A recent review, including Bayes inference and maximum likelihood estimate, can be found in Kantas *et al.* (2015). The Bayesian approach (Andrieu *et al.* 2010; Lindsten *et al.* 2013; Kantas *et al.* 2015) aims to infer arbitrary parameter by simulating from the joint distribution of the state and the parameter. Additionally it is able to describe the shape of parameter distribution which might be multi-modal. In data assimilation community this approach is used in Stroud and Bengtsson (2007); Ueno and Nakamura (2016); Stroud *et al.* (2017). But the Bayesian approaches still have some drawbacks. First, a very large number of iterations is required to get good approximations of the parameter distributions. Moreover simulating the distributions in high-dimensional state-space models is sometimes impractical. For example, it is difficult to simulate directly the full model covariance Q which involves a lot of parameters if the latent state has values in a high dimensional space. To simplify the problem, Q is typically supposed to have a predefined form, such as the multiplication of a scalar by a given matrix, and only the scale factor is estimated. In this paper we hence focus on maximum likelihood estimation.

There are two major approaches in the statistical literature to maximize numerically the likelihood in models with latent variables: Gradient ascent and Expectation-Maximization (EM) algorithms. As stated in Kantas *et al.* (2015) *gradient ascent algorithms can be numerically unstable as they require to scale carefully the components of the score vector* and thence the EM approach is generally favored when considering complicated models such as the one used in data assimilation. The first EM algorithm was suggested by Dempster *et al.* (1977). Various variants of the EM algorithm were proposed in the statistical literature (see e.g. Celeux *et al.* (1995); McLachlan and Krishnan (2007); Schön *et al.* (2011); Lindsten (2013); Kantas *et al.* (2015) and references therein) and in the data assimilation community (see Ueno and Nakamura (2014); Tandeo *et al.* (2015b); Pulido *et al.* (2017); Dreano *et al.* (2017)). The common idea of these algorithms is to run an iterative procedure where an auxiliary quantity which depends on the smoothing distribution is maximized at each iteration, until some convergence criteria are reached.

Within the EM machinery, the challenging issue is generally to compute the joint smoothing distribution $p_\theta(x_{0:T}|y_{1:T})$ of the latent state given the entire sequence of observations, where $x_{0:T} = (x_0, x_1, \cdots, x_T)$. For a linear Gaussian model, the Kalman smoother (KS) (Shumway and Stoffer 1982) based on Rauch-Tung-Streibel (RTS) provides an exact solution to this problem. The difficulty arises when the model is nonlinear and the state does not take its values in a finite state-space. In such situation the smoothing distribution is intractable. To tackle this issue, simulation-based methods were proposed. In data assimilation, Ensemble Kalman smoother (EnKS) (Evensen and Van Leeuwen 2000) is one of the most favorite choices. By implementing the best linear unbiased estimate strategy, this method is able to approximate the smoothing distribution using only a few simulations of the physical model (members) at each time step. Unfortunately the approximation does not converge to the exact distribution $p_\theta(x_{0:T}|y_{1:T})$ for non-linear state-space models (Le Gland *et al.* 2009). Particle smoothers have been proposed as an alternative in Doucet *et al.* (2001); Godsill *et al.* (2004); Cappé *et al.* (2007); Douc *et al.* (2009). However, they demand a huge amount of particles (and thus to run the physical many times) to get a good approximation of the target probability distribution. Since 2010, conditional particle smoothers (CPSs) (Lindsten and Schön 2011, 2012; Lindsten *et al.* 2012, 2013, 2014; Svensson *et al.* 2015), pioneered by Andrieu *et al.* (2010), have been developed as other strategies to simulate the smoothing distribution. Contrary to the more usual smoothing samplers discussed above, CPSs simulate realizations using an iterative algorithm. At each iteration, one conditioning trajectory is plugged in a standard particle smoothing scheme. It helps the sampler to explore interesting parts of the state space with only few particles. After a sufficient number of iterations, the

algorithm provides samples approximately distributed according to the joint smoothing distribution.

In the data assimilation community, EM algorithms have been generally used in conjunction with EnKS (EnKS-EM algorithm). Recent contributions (Tandeo *et al.* 2015b; Pulido *et al.* 2017; Dreano *et al.* 2017) implement this approach using $20-100$ members and concentrate on estimating the initial state distribution and error covariances. In the statistical community, the combination of standard or approximate particle smoothers (PSs) with large amount of particles and EM algorithms (PS-EM) (Olsson *et al.* 2008; Schön *et al.* 2011; Kokkala *et al.* 2014; Kantas *et al.* 2015; Picchini and Samson 2018) is preferred. The number of particles is typically in the range 10^2-10^6 which would lead to unrealistic computational time for usual data assimilation problems (the number of particles corresponds to the number of time that the physical model needs to be run at each time step). In Lindsten (2013), the author proposed to use a CPS algorithm, named Conditional particle filtering-Ancestor sampling (CPF-AS, Lindsten *et al.* (2012)), within a stochastic EM algorithm (CPF-AS-SEM). The authors showed that the method can estimate Q and R using only 15 particles for univariate spate-space models. However CPF-AS suffers from degeneracy (the particle set reduce to a very few effective particles) and consequently the estimated parameters of CPF-AS have bias and/or large variance. In the present paper, we propose to combine another CPS, referred to as Conditional particle filtering-Backward Simulation (CPF-BS, Lindsten *et al.* (2013)), with the stochastic EM scheme. The novel proposed maximum likelihood estimate method, abbreviated as CPF-BS-SEM, aims at estimating the parameters with only a few particles and thus reasonable computational costs for data assimilation. In this article we show that our approach has better performances than the EM algorithms combined with standard PS (Schön *et al.* 2011), CPF-AS (Lindsten 2013) and EnKS (Dreano *et al.* 2017). Numerical illustrations are compared in terms of estimation quality and computational cost on highly nonlinear models. We also provide an open-source Python library of all mentioned algorithms which is available on-line at `https://github.com/tchau218/parEMDA`.

The article is organized as follows. In Section 2 we introduce the main methods used in the article, including smoothing with the conditional particle smoother CPF-BS and maximum likelihood estimation using CPF-BS-SEM. Section 3 is devoted to numerical experiments and Section 4 contains conclusions.

2. Methods

In this section, we first introduce the conditional particle smoother which is the key ingredient of the proposed method. This smoother is based on conditional particle filtering (CPF) which is described in Section 2.1.1. Standard particle filtering algorithm is also reminded and its performance is compared to the one of CPF. Section 2.1.2 presents iterative filtering/smoothing schemes which are the combinations of CPF and ancestor tracking algorithms. We also analyze benefits and drawbacks of these filters/smoothers. Then an iterative smoothing sampler based on CPF-BS is provided as an alternative to the CPF smoothers and theirs theoretical properties are quickly discussed in Section 2.1.3. Finally, the combination of CPF-BS with the EM machinery for maximum likelihood estimation is discussed in Section 2.2.

2.1. Filtering and smoothing using conditional particle-based methods

2.1.1. Particle Filtering (PF) and Conditional Particle Filtering (CPF)

In the state-space model defined by (1), the latent state $(x_t)_{t=0:T}$ is a Markov process defined by its background distribution $p_\theta(x_0)$ and transition kernel $p_\theta(x_t|x_{t-1})$. The observations $(y_t)_{t=1:T}$ are conditionally independent given the state process and we denote $p_\theta(y_t|x_t)$ as the conditional distribution of y_t given x_t. The transition kernel $p_\theta(x_t|x_{t-1})$ depends on both the dynamical model $\mathcal{M}_\theta$ and the distribution of the model error η_t whereas the conditional observation distribution $p_\theta(y_t|x_t)$ is a function of the observation model $\mathcal{H}_\theta$ and the distribution of the observation error ϵ_t. In this section we discuss algorithms to approximate the filtering distribution $p_\theta(x_t|y_{1:t})$ which represents the conditional distribution of the state at time t given the observations up to time t.

For linear Gaussian models, the filtering distributions are Gaussian distributions which means and covariances can be computed using the Kalman recursions. When state-space models are nonlinear, as it is the typical case for data assimilation applications, the filtering distributions do not admit a closed form and particle filtering (PF) methods have been proposed to compute approximations of these quantities (Doucet *et al.* 1998, 2001; Cappé *et al.* 2007). The general PF algorithm is based on the following relation between the filtering distributions at time $t-1$ and t

$$p_\theta(x_{0:t}|y_{1:t}) = \frac{p_\theta(y_t|x_t)\, p_\theta(x_t|x_{t-1})}{p_\theta(y_t|y_{1:t-1})}\, p_\theta(x_{0:t-1}|y_{1:t-1}) \quad (2)$$

where $p_\theta(y_t|y_{1:t-1})$ is the normalization term of $p_\theta(x_{0:t}|y_{1:t})$. Note that if we are able to compute the joint filtering distribution $p_\theta(x_{0:t}|y_{1:t})$ then it is possible to deduce the marginal filtering distribution $p_\theta(x_t|y_{1:t})$ by integrating over all variables $x_{0:t-1}$.

PF runs with N_f particles to approximate $p_\theta(x_{0:t}|y_{1:t})$ recursively in time. Let us suppose that initial particles $\{x_0^{(i)}\}_{i=1:N_f}$ have been drawn from the background $p_\theta(x_0)$ and the filtering process has been done up to time $t-1$. Since PF is based on importance sampling, we now have a system of particles and their corresponding weights $\{x_{0:t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1:N_f}$ which approximates the joint filtering distribution $p_\theta(x_{0:t-1}|y_{1:t-1})$. The next step of the algorithm consists in deriving an approximation

$$\hat{p}_\theta\left(x_{0:t}|y_{1:t}\right) = \sum_{i=1}^{N_f} \delta_{x_{0:t}^{(i)}}\left(x_t\right) w_t^{(i)} \quad (3)$$

of $p_\theta(x_{0:t}|y_{1:t})$ based on Eq. (2). It is carried out in three main steps (see left panel of Figure 2 for an illustration):

- **Resampling.** Systematic resampling method with multinomial distribution $\mathcal{M}\text{ul}(\cdot|w_{t-1}^{(1)}, w_{t-1}^{(2)}, \cdots, w_{t-1}^{(N_f)})$ (Douc and Cappé 2005; Hol *et al.* 2006) is used to reselect potential particles in $\{x_{0:t-1}^{(i)}\}_{i=1:N_f}$. In this step the filter duplicates particles with large weight and removes particles with very small weight.
- **Forecasting.** It consists in propagating the particles from time $t-1$ to time t with a proposal kernel $\pi_\theta(x_t|x_{0:t-1}, y_{1:t})$.
- **Weighting.** Importance weights $\{w_t^{(i)}\}_{i=1:N_f}$ of the particles $\{x_{0:t}^{(i)}\}_{i=1:N_f}$ are computed according to the formula

$$W\left(x_{0:t}\right) = \frac{p_\theta\left(x_{0:t}|y_{1:t}\right)}{\pi_\theta(x_t|x_{0:t-1}, y_{1:t})} \overset{(2)}{\propto} \frac{p_\theta\left(y_t|x_t\right) p_\theta(x_t|x_{t-1})}{\pi_\theta(x_t|x_{0:t-1}, y_{1:t})}\, p_\theta\left(x_{0:t-1}|y_{1:t-1}\right). \quad (4)$$

The entire algorithm of PF is presented in **Algorithm 0**. Notation $\{I_t^i\}_{t=1:T}^{i=1:N_f}$ in **Algorithm 0** is used to store the particle's indices

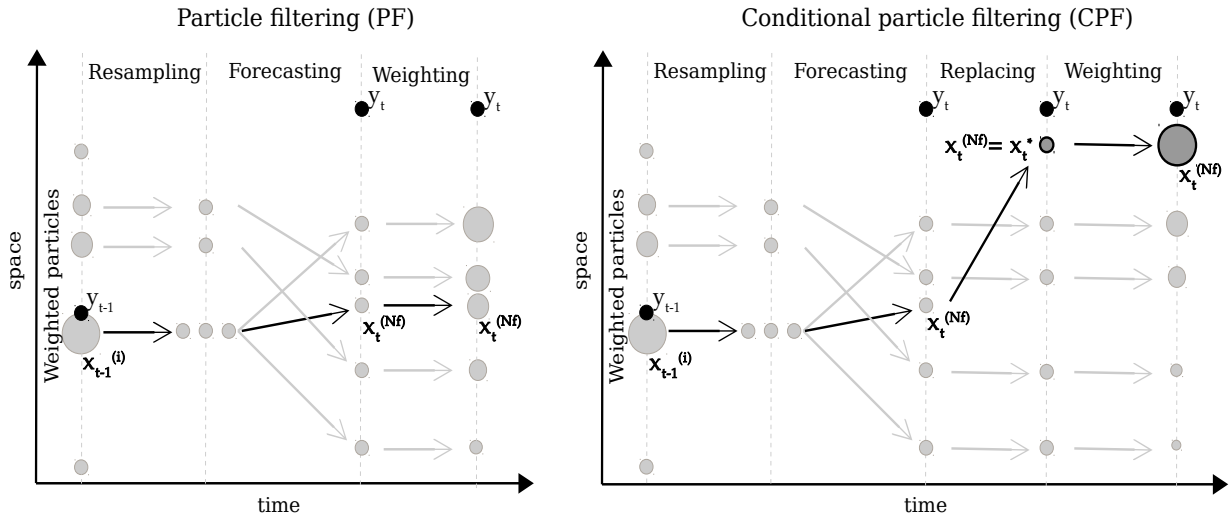

Figure 2. Comparison of PF and CPF schemes using $N_f = 5$ particles (light gray points) in time window $[t-1, t]$ on the SSM (1). The observation model is the identity function. The main difference is shown on black quivers as CPF replaces the particle $x_t^{(N_f)}$ with conditioning particle x_t^* (dark gray point).

accross time in order to be able to reconstruct trajectories. It is a key ingredient in the smoothing algorithms which are presented later. Note that, in a general PF algorithm, particles can be propagated according to any proposal distribution π_θ. If we choose $\pi_\theta(x_t|x_{0:t-1}, y_{1:t}) = p_\theta(x_t|x_{t-1})$ $p_\theta(x_{0:t-1}|y_{1:t-1})$ (see Doucet *et al.* (1998); Pitt and Shephard (1999); Cappé *et al.* (2007) and Snyder (2011) for discussions on the choice of π_θ), the importance weight function (4) can be simplified as $W(x_{0:t}) \propto p_\theta(y_t|x_t)$. With this choice, which is referred to as bootstrap filter in the literature, the forecasting step consists in sampling according to the dynamical model $\mathcal{M}_\theta$. It is the favorite choice for data assimilation applications (Van Leeuwen 2015; Poterjoy and Anderson 2016; Lguensat *et al.* 2017) and it is used in this article for numerical illustrations.

Conditional particle filtering (CPF) was introduced first time by Andrieu *et al.* (2010) and then discussed by many authors (Lindsten and Schön 2011, 2012; Lindsten *et al.* 2013, 2014; Svensson *et al.* 2015). The main difference with PF consists in plugging a conditioning trajectory $X^* = (x_1^*, \cdots, x_T^*) \in \mathcal{X}^T$ into a regular filtering scheme. In practice, CPF works in an iterative environment where the conditioning trajectory X^* is updated at each iteration. This is further discussed in the next section. In this section we assume that X^* is given. Due the conditioning, CPF algorithm differs from the PF algorithm in adding a replacing step between the forecasting and weighting steps. In this step, one of the particles is replaced by the conditioning trajectory. It is possible to set the conditioning particle as the particle number N_f and this leads to updating the position of the particles at time t according to

$$x_t^{(i)} = \begin{cases} x_t^{(i)} \sim \pi_\theta(x_t|x_{0:t-1}^{(I_t^i)}, y_{1:t}), & \forall i = 1:N_f-1 \\ x_t^*, \quad i = N_f. \end{cases} \tag{5}$$

Similarly to PF, the reset sample $\{x_t^{(i)}\}_{i=1:N_f}$ is next weighted according to Eq. (4). In **Algorithm 0** we present the differences between PF and CPF algorithms. The additional ingredients of CPF are highlighted using a gray color.

Algorithm 0: Particle Filtering (PF)/*Conditional Particle Filtering (CPF)* **given** ***the conditioning*** $X^* = (x_1^*, x_2^*, \cdots, x_T^*)$ ***(only for CPF)*** **and** T **observations** $y_{1:T}$**, for fixed parameter** θ.

- Initialization:
  + Sample $\{x_0^{(i)}\}_{i=1:N_f} \sim p_\theta(x_0)$.
  + Set weights: $w_0^{(i)} = 1/N_f, \forall i = 1:N_f$.
- For $t = 1:T$,
  + **Resampling**: draw indices

$$\{I_t^i\}_{i=1:N_f} \sim \mathcal{M}\text{ul}\left(j|w_{t-1}^{(1)}, w_{t-1}^{(2)}, \cdots, w_{t-1}^{(N_f)}\right)$$

  with $j \in \{1, 2, \cdots, N_f\}$.
  + **Forecasting**: sample new particle

$$x_t^{(i)} \sim \pi_\theta\left(x_t|x_{0:t-1}^{(I_t^i)}, y_{1:t}\right), \forall i = 1:N_f.$$

  + ***Replacing*** *(only for CPF): set* $x_t^{(N_f)} = x_t^*$ *and* $I_t^{N_f} = N_f$.
  + **Weighting**: compute $\tilde{w}_t^{(i)} = W\left(x_{0:t-1}^{(I_t^i)}, x_t^{(i)}\right)$ by using Eq. (4) then calculate its normalized weight $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^{N_f} \tilde{w}_t^{(i)}}, \forall i = 1:N_f$.

  end for.

The general principle of the CPF algorithm is also presented on Figure 2. CPF does a selection between particles sampled from the proposal kernel π_θ and the conditioning particle. We can imagine two opposite situations. If the conditioning particle is "bad" (i.e. far from the true state) then the filtering procedure will eliminate it by weighting and resampling. But if conditioning particle is "good" (i.e. close to the true state) then it will have a high weight and it will be duplicated and propagated at the next time step. This ensures that if an interesting sequence is used as conditioning trajectory, then the CPF algorithm will explore the state space in the neighborhood of this trajectory and thus, hopefully, an interesting part of the state space. This is also illustrated on Figure 3 which has been drawn using the Kitagawa state-space model (given later in Eq. 20). This univariate model was chosen because it is known that it is difficult to compute accurate approximations of the filtering distribution: the forecasting distribution $p_\theta(x_t|x_{t-1})$ can be bimodal due to the cos-term and the observation operator is quadratic. In addition, we use a large value of R to get unreliable observations. On the left panel of the figure, around time $t = 17$, PF starts to simulate trajectories which are far away from the true states. All the particles are close to 0 and the dynamical model provides unstable and inaccurate forecasts. At the same time, the observation y_t is unreliable and can not help to correct the forecasts. It leads to a bad approximation of the filtering distribution at time $t = 18$. It

© 0000

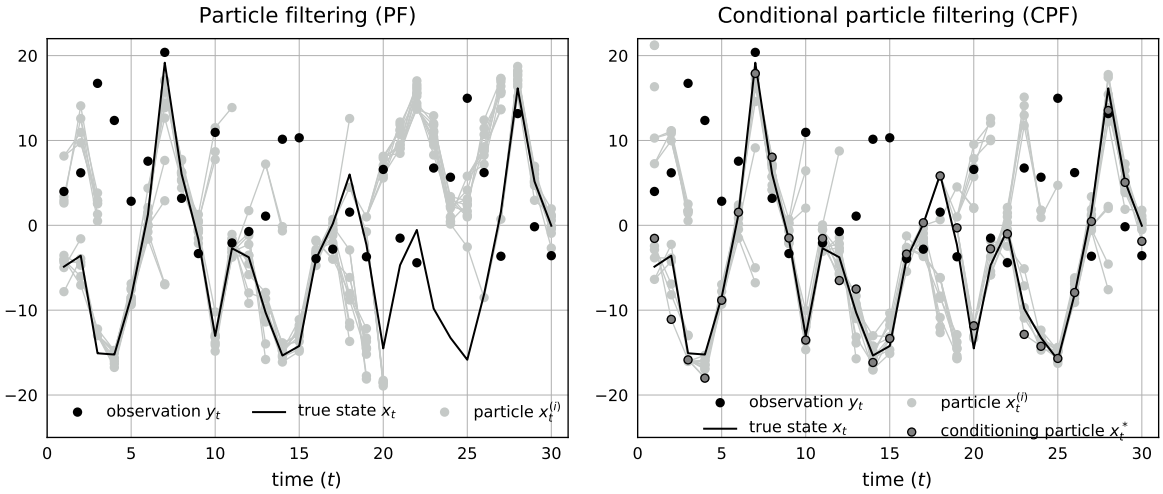

Figure 3. Comparisons of PF and CPF performances with 10 particles on Kitagawa model (20), where $T = 30$, $(Q, R) = (1, 10)$. Conditioning particles (dark gray points) are supposed to live around to the true state trajectory (black curve). Gray lines are the links among particles which have the same ancestor.

has consequences on the next time steps: the forecast distributions remain far from the true state and the filter gives bad results. CPF gives better results thanks to a good conditioning trajectory which helps generating relevant forecasts (see right panel of Figure 3).

When the number of particles N_f is big, the effect of the conditioning particles becomes negligible and the PF and CPF algorithms give similar results. However running a particle filter with large amount of particles is generally computationally impossible for data assimilation problems. Algorithms which can provide a good approximation of the filtering distributions using only a few particles (typically in the range $10 - 100$) are needed. An alternative strategy to PF/CPF with a large number of particles, based on iterating the CPF algorithm with a low number of particles, is discussed in the next section.

2.1.2. *Filtering and smoothing with conditional particle filter*

A key input to the CPF algorithm is the conditioning particles of the given trajectory X^*. As discussed in the previous Section, the "good" conditioning particles must be "close" to the true state in order to help the algorithm simulates interesting particles in the forecasting step with reasonable computational costs. Remark also that the distribution of the particles simulated by running one iteration of the CPF depends on the distribution of the conditioning trajectory X^*. The distribution of X^* must be chosen in such a way that the output of the CPF is precisely the filtering/smoothing distributions that we are targeting. One solution to this problem can be found in Andrieu *et al.* (2010) (see a summary in Theorem 2.1): if X^* is simulated according to the smoothing distribution then running the CPF algorithm with this conditioning particle will provide other sequences distributed according to the smoothing distributions. A more interesting result for the applications states that if the conditioning trajectory is "bad", then iterating the CPF algorithm after a certain number of iterations will provide "good" sequences for X^* which are distributed approximatively according to the smoothing distribution. At each iteration the conditioning trajectory X^* is updated using one of the trajectories simulated by the CPF algorithm at the previous iteration. The corresponding procedure is described more precisely below.

Running the CPF algorithm (**Algorithm 0**) until the final time step T gives a set of particles, weights and indices which define an empirical distribution on $\mathcal{X}^{T+1}$,

$$\hat{p}_\theta\left(x_{0:T}|y_{1:T}\right) = \sum_{i=1}^{N_f} \delta_{x_{0:T}^{(i)}}\left(x_{0:T}\right) w_T^{(i)} \qquad (6)$$

where $x_{0:T}^{(i)}$ is one particle path (realization) taken among particles (eg. one continuous gray link on Figure 3), $w_T^{(i)}$ is its corresponding weight and i is an index of its particle at the final time step. The simulation of one trajectory according to Eq. (6), is based on sampling its final particle with respect to the final weights $(w_T^{(i)})_{i=1:N_f}$ such that

$$p(x_{0:T}^s = x_{0:T}^{(i)}) \propto w_T^{(i)}.$$

Then, given the final particle, eg. $x_T^s = x_T^{(i)}$, the rest of the path is obtained by tracing the ancestors (parent, grandparent, etc) of the particle $(x_T^{(i)})$. The information on the genealogy of the particles is stored in the indices $(I_t^i)_{t=1:T}^{i=1:N_f}$ since I_t^i is the index of the parent of $x_t^{(i)}$. The technique is named ancestor tracking (also presented in statistical literature of standard PF such as Doucet and Johansen (2009)). It is illustrated on Figure 4. Given $i = 1$, the parent of particle $x_4^{(1)}$ is the particle $x_3^{(I_4^1)} = x_3^{(3)}$, its grandparent is the particle $x_2^{(I_3^3)} = x_2^{(3)}$ and its highest ancestor is $x_1^{(I_2^3)} = x_1^{(2)}$. At the end, we obtain one realization $x_{1:4}^s = x_{1:4}^{(1)} = (x_1^{(2)}, x_2^{(3)}, x_3^{(3)}, x_4^{(1)})$.

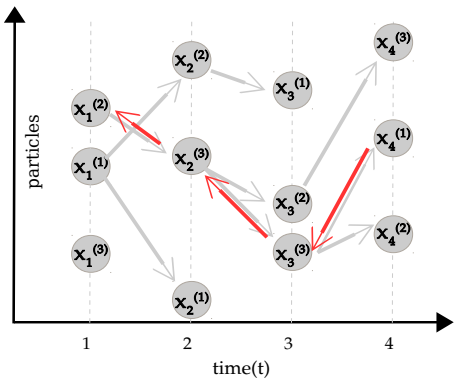

Figure 4. An example of ancestor tracking one smoothing trajectory (backward quiver) based on ancestral links of filtering particles (forward quivers). Particles (gray balls) are assumed to be obtained by a filtering algorithm with $T = 4$ and $N_f = 3$.

In practice the following procedure can be implemented to generate a path $x_{0:T}^s = x_{0:T}^{(J_T)} = (x_0^{(J_0)}, x_1^{(J_1)}, \cdots, x_T^{(J_T)})$ according to Eq. (6)

- For $t = T$, draw index J_T with $p(J_T = i) \propto w_T^{(i)}$ and set $x_T^s = x_T^{(J_T)}$.

- For $t < T$, set index $J_t = I_{t+1}^{J_{t+1}}$ and $x_t^s = x_t^{(J_t)}$.

Finally the iterative smoothing algorithm using CPF can be described as follows,

Algorithm 1: Smoothing with Conditional Particle Filtering (CPF) given the conditioning $X^* = (x_1^*, x_2^*, \cdots, x_T^*)$ and T observations $y_{1:T}$, for fixed parameter θ.

- Run CPF (**Algorithm 0**) given X^* and observations $y_{1:T}$, with fixed parameter θ and N_f particles.
- Run ancestor tracking procedure N_s times to simulate N_s trajectories according to Eq. (6).
- Update the conditioning particle X^* with one of these trajectories.

According to Theorem 2.1 given in Andrieu *et al.* (2010), this algorithm will generate trajectories which are approximatively distributed according to the smoothing distribution after a certain number of iterations, even if a low number of particles is used at each iteration. However, in practice running one iteration of the CPF algorithm leads to generating trajectories which are generally almost identical to the conditioning particle (Lindsten *et al.* 2013; Svensson *et al.* 2015). The main reason for this is the so-called degeneracy issue: all the particles present at the final time step T share the same ancestors after a few generations. This is illustrated on Figure 4: all the particles present at time $t = 4$ have the same grandparent at time $t = 2$. This is also visible on the left panel of Figure 5. The resampling makes disappear many particles whereas other particles have many children. As a consequence, all 10 particles at the final time step $T = 30$ have the same ancestors for $t < 20$. This degeneracy issue clearly favors the conditioning particle which is warranted to survive and reproduce at each time step. When iterating the CPF algorithm, the next conditioning sequence is thus very likely to be identical to the previous conditioning sequence, except maybe for the last time steps. This leads to an algorithm which has a poor mixing and lots of iterations are needed before converging to the smoothing distribution.

To improve the mixing, Lindsten *et al.* (2012, 2013, 2014) proposed to modify the replacing step of **Algorithm 0** as follows. After setting the final particle $x_t^{(N_f)} = x_t^* \in X^*$ to the conditioning particle, the index of its parent $I_t^{(N_f)}$ is drawn following the Bayes' rule

$$p_\theta(I_t^{N_f} = i | x_t^*, y_{1:t}) \propto p_\theta(x_t^* | x_{t-1}^{(i)})\, w_{t-1}^{(i)}. \tag{7}$$

Resampling $I_t^{N_f}$ helps to break the conditioning trajectory X^* into pieces so that the algorithm is less likely to simulate trajectories which are close to X^*. The different steps of a smoother using this algorithm, referred to as Conditional Particle Filtering-Ancestor Sampling (CPF-AS) algorithm, are given below.

Algorithm 2: Smoothing with Conditional Particle Filtering-Ancestor Sampling (CPF-AS) given the conditioning $X^* = (x_1^*, x_2^*, \cdots, x_T^*)$ and T observations $y_{1:T}$, for fixed parameter θ.

- Run CPF-AS, **Algorithm 0** wherein indices of conditional particles $(I_t^{N_f})_{t=1:T}$ are resampled with the rule (7), given X^* and observations $y_{1:T}$, with fixed parameter θ and N_f particles.
- Run ancestor tracking procedure N_s times to get N_s trajectories among particles of the CPF-AS algorithm.
- Update the conditioning particle X^* with one of these trajectories.

In the above-mentioned references, it is shown empirically that this algorithm is efficient to simulate trajectories of the smoothing distribution with only $5 - 20$ particles. It is also proven that it has the same good theoretical properties (see Theorem 2.1) as the original CPF algorithm and that running enough iterations of the CPF-AS algorithm, starting from any conditioning particle X^*, permits to generate trajectories which are approximatively distributed according to the smoothing distribution.

The comparison of the left and middle panels of Figure 5 shows that resampling the indices permits to obtain ancestor tracks which are different from the conditioning particles. However, like CPF smoother (**Algorithm 1**), tracking ancestral paths in the CPF-AS smoother (**Algorithm 2**) still surfers from the degeneracy problem mentioned above. It implies that the N_s trajectories simulated at one iteration of the CPF-AS generally coincide, except for the last time steps, and thus give a poor description of the smoothing distribution. This is illustrated on Figure 5: all the trajectories simulated with the CPF-AS coincide for $t < 20$ and thus can not describe the spread of the smoothing distribution. In practice, many particles which are simulated with the physical model in the forecasting step are forgotten when running the ancestor tracking and it leads to waste information and computing resources for data-assimilation applications. In the next section we propose to replace ancestor tracking by backward simulation in order to better use the information contained in the particles.

2.1.3. Smoothing with Conditional particle filtering- Backward simulation (CPF-BS)

Backward simulation (BS) was first proposed in the statistical literature in association with the regular particle filter (Godsill *et al.* 2004; Douc *et al.* 2009; Doucet and Johansen 2009). Recently BS was combined with conditional smoothers Lindsten and Schön (2011, 2012); Lindsten *et al.* (2013). In the framework of these smoothers, the smoothing distribution $p_\theta(x_{0:T}|y_{1:T})$ is decomposed as

$$p_\theta\left(x_{0:T}|y_{1:T}\right) = p_\theta\left(x_T|y_{1:T}\right) \prod_{t=0}^{T-1} p_\theta\left(x_t|x_{t+1}, y_{1:t}\right), \tag{8}$$

where

$$p_\theta(x_t|x_{t+1}, y_{1:t}) \propto p_\theta\left(x_{t+1}|x_t\right)\; p_\theta\left(x_t|y_{1:t}\right) \tag{9}$$

is the so-called backward kernel. Given the particles $(x_t^{(i)})_{t=0:T}^{i=1:N_f}$ and the weights $(w_t^{(i)})_{t=0:T}^{i=1:N_f}$ of the CPF algorithm (**Algorithm 0**) we obtain an estimate (3) of the filtering distribution $p_\theta\left(x_t|y_{1:t}\right)$. By plugging this estimate in (9), we deduce the following estimate of the backward kernel

$$\hat{p}_\theta\left(x_t|x_{t+1}, y_{1:t}\right) \propto \sum_{i=1}^{N_f} p_\theta(x_{t+1}|x_t^{(i)}) w_t^{(i)} \delta_{x_t^{(i)}}\left(x_t\right) \tag{10}$$

Using the relation (8) and the estimate (10), one smoothing trajectory $x_{0:T}^s = x_{0:T}^{J_{0:T}} = (x_0^{(J_0)}, x_1^{(J_1)}, \cdots, x_{T-1}^{(J_{T-1})}, x_T^{(J_T)})$ can be simulated recursively backward in time as follows.

- For $t = T$, draw J_T with $p(J_T = i) \propto w_T^{(i)}$.
- For $t < T$,
  + Compute weights $w_t^{s,(i)} = p_\theta(x_{t+1}^{(J_{t+1})}|x_t^{(i)})\; w_t^{(i)}$ using (10), for all $i = 1 : N_f$.
  + Sample J_t with $p(J_t = i) \propto w_t^{s,(i)}$.
  end for

To draw N_s distinct realizations we just need to repeat N_s times the procedure. The performance of BS given outputs of one run of the CPF algorithm is displayed on Figure 5 and the complete smoother using CPF-BS is described below.

© 0000

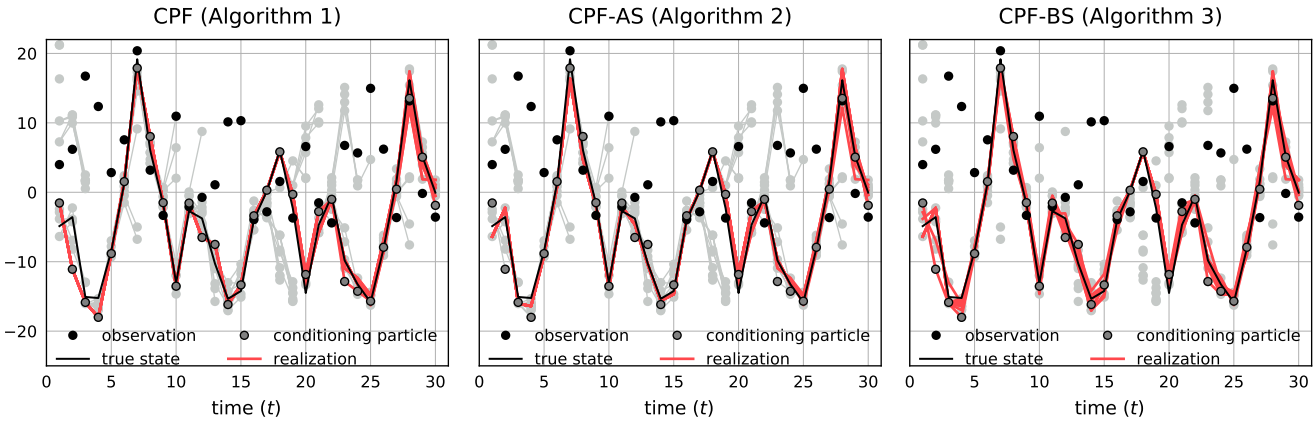

Figure 5. Comparison for simulating $N_s = 10$ realizations by using CPF smoother (**Algorithm 1**), CPF-AS smoother (**Algorithm 2**) (both based on particle genealogy-light gray links) and CPF-BS smoother (**Algorithm 3**) (based on backward kernel (10)) given the same forward filtering pattern with $N_f = 10$ particles (light gray points). The experiment is run on Kitagawa model (20) where $T = 30$, $(Q, R) = (1, 10)$.

Algorithm 3: Smoothing with Conditional Particle Filtering-Backward Simulation (CPF-BS) given the conditioning $X^* = (x_1^*, x_2^*, \cdots, x_T^*)$ and T observations $y_{1:T}$, for fixed parameter θ.

- Run CPF (**Algorithm 0**) given (X^*, Y) with N_f particles and fixed parameter θ.
- Run BS procedure N_s times provided the forward filtering outputs to sample N_s trajectories.
- Update the conditioning trajectory X^* with one of these trajectories.

Figure 6 illustrates how the iterative CPF-BS smoother works and performs on the Kitagawa model. The smoothing procedure is initialized with a "bad" conditioning trajectory ($x_t^* = 0$ for $t \in \{1, ..., T\}$). This impacts the quality of the simulated trajectories which are far from the true state at the first iteration. Similar issues usually occurs when running regular particle smoothers (such as Particle Filtering-Backward Simulation, PF-BS, see Godsill *et al.* (2004); Douc *et al.* (2009)) with a small number of particles. The conditioning trajectory is then updated and it helps to drive the particles to interesting parts of the state space. After only 3 iterations, the simulated trajectories stay close to the true trajectory. Note that only 10 particles are used at each iteration.

Algorithm 1, **2** and **3** generate a new conditioning trajectory at each iteration and this defines a Markov kernel on $\mathcal{X}^T$ since the conditioning trajectory obtained at one iteration only depends on the conditioning particle at the previous iteration. Theorem 2.1 shows that these Markov kernels have interesting theoretical properties (see also Chopin *et al.* (2015) for more results). This theorem was first proven for the CPF smoother in Andrieu *et al.* (2010). These results were then extended to CPF-BS in Lindsten and Schön (2011) and to CPF-AS in Lindsten *et al.* (2014) with some extensions to solving inverse problems in non-Markovian models.

Theorem 2.1 *For any number of particles $(N_f \geq 2)$ and a fixed parameter $\theta \in \Theta$,*

*i. Markov kernel $\mathcal{K}_\theta$ defined by one of conditional smoothers (CPF: **Algorithm 1**, CPF-AS: **Algorithm 2** and CPF-BS: **Algorithm 3**) leaves the invariant smoothing distribution $p_\theta(x_{0:T}|y_{1:T})$. That is, for all $X^* \in \mathcal{X}^T$ and $A \subset \mathcal{X}^{T+1}$,*

$$p_\theta(A|y_{1:T}) = \int \mathcal{K}_\theta(X^*, A)\, p_\theta(X^*|y_{1:T})\, dX^*$$

where $\mathcal{K}_\theta(X^*, A) = \mathbb{E}_{\theta, X^*}\left[\mathbb{1}_A(x_{0:T}^{J_{0:T}})\right]$, *and* $x_{0:T}^{J_{0:T}} = \{x_0^{(J_0)}, \cdots, x_T^{(J_T)}\}$.

ii. The kernel $\mathcal{K}_\theta$ has p_θ- irreducible and aperiodic. It hence converges to $p_\theta(x_{1:T}|y_{1:T})$ for any starting point X^*. Consequently,

$$\|\mathcal{K}_\theta^r(X^*, \cdot) - p_\theta(\cdot|y_{1:T})\|_{TV} \overset{r\to\infty}{\longrightarrow}^{\text{as}} 0.$$

where $\|\cdot\|_{TV}$ is the total variation norm.

The second property of this theorem implies that running the algorithm with any initial conditioning trajectory will permit to simulate samples distributed approximatively according to the smoothing distribution after a sufficient number of iterations. However, in practice, the choice of a good initial trajectory is very important, in particular when the considered state space is complex (high non-linearities, partly observed components,...). If we set an initial conditioning trajectory far from the truth, then lots of iterations are needed before exploring a space relevant to the true state. In such situations it may be useful to provide an estimate of the true state using an alternative method (e.g. running another smoothing algorithm such as EnKS).

Despite of sharing the same theoretical properties as the CPF and CPF-AS smoothers, we will show in Section 3 that CPF-BS algorithm gives better results in practice. This is due to its ability to avoid the degeneracy problem and hence provide better descriptions of the smoothing distribution. At the first glance, the computational cost of the backward technique seems to be higher than the one of ancestor tracking. Nevertheless, for data assimilation applications, the computational complexity mainly comes from the numerical model which is used to propagate the N_f particles in the forecasting step. In addition the transition probability in the backward kernel (10) can be computed by reusing the forecast information and do not require extra runs of the physical model. The computational cost of the CPF-BS algorithm is thus similar to the ones of CPF or CPF-AS algorithms and grows linearly with N_f.

Recently the CPF-BS with few particles $(5 - 20)$ has been employed to sample θ and simulate the latent state in a Bayesian framework (Lindsten and Schön 2011, 2012; Lindsten *et al.* 2013, 2014; Svensson *et al.* 2015). In the next section, we propose to use the CPF-BS smoother to perform maximum likelihood estimation which is the main contribution of this paper.

2.2. *Maximum likelihood estimate (MLE) using CPF-BS*

In this section we discuss the estimation of the unknown parameter θ given a sequence of measurements $y_{1:T}$ of the SSM (1). The inference will be based on maximizing the incomplete likelihood

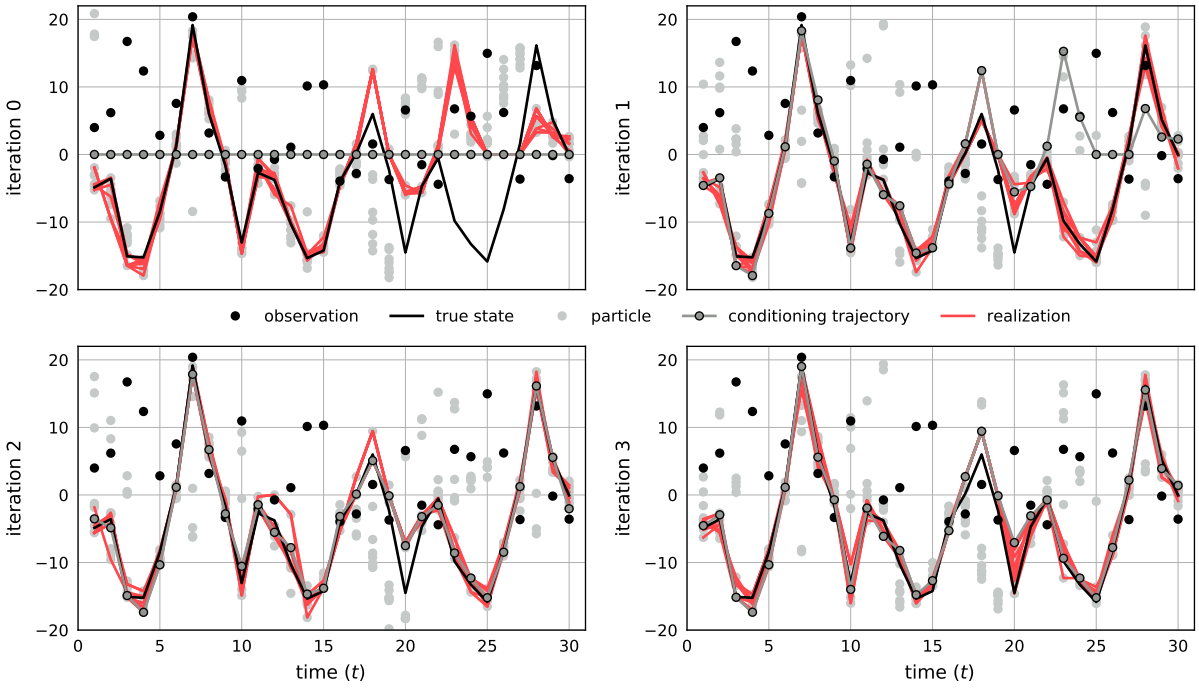

Figure 6. Performance of an iterative CPF-BS smoother (**Algorithm 3**) with $N_f = 10$ particles in simulating $N_s = 10$ realizations. The experiment is on Kitagawa model (20) where $(Q, R) = (1, 10), T = 30$. The smoother given a zero-initial conditioning ($X^* = \mathbf{0} \in \mathbb{R}^T$) is run within 3 iterations. For each iteration the conditioning trajectory X^* is one of realizations obtained from the previous.

of the observations,

$$L(\theta) = p_\theta(y_{1:T}) = \int p_\theta\left(x_{0:T}, y_{1:T}\right) dx_{0:T}. \tag{11}$$

The EM algorithm is the most classical numerical method to maximize the likelihood function in models with latent variables (Dempster *et al.* 1977; Celeux *et al.* 1995). It works following the auxiliary function

$$G(\theta, \theta') = \mathbb{E}_{\theta'}\left[\ln\ p_\theta\left(x_{0:T}, y_{1:T}\right)\right] \tag{12}$$

$$\triangleq \int \ln p_\theta\left(x_{0:T}, y_{1:T}\right)\ p_{\theta'}\left(x_{0:T}|y_{1:T}\right) dx_{0:T} \tag{13}$$

where θ and θ' denote two possible values for the parameters. Due to Markovian assumption of the SSM (1) and independence properties of noises (ϵ_t, η_t) and the initial state x_0, the complete likelihood $p_\theta(x_{0:T}, y_{1:T})$ which appears in (12) can be decomposed as

$$p_\theta(x_{0:T}, y_{1:T}) = p_\theta\left(x_0\right) \prod_{t=1}^{T} p_\theta\left(x_t|x_{t-1}\right) \prod_{t=1}^{T} p_\theta\left(y_t|x_t\right). \tag{14}$$

The auxiliary function $G(.|\theta')$ is typically much more simpler to optimize than the incomplete likelihood function and the EM algorithm consists in maximizing iteratively this function. Starting from an initial parameter θ_0 an iteration r of the EM algorithm has two main steps:

- **E-step**: compute the auxiliary quantity $G(\theta, \theta_{r-1})$
- **M-step**: compute $\theta_r = \arg\max_\theta G(\theta, \theta_{r-1})$.

It can be shown that it leads to increasing the likelihood function at each iteration and gives a sequence which converges a local maximum of L.

The EM algorithm combined with Kalman smoothing (KS-EM, Shumway and Stoffer (1982)) has been the dominant approach to estimate parameters in linear Gaussian models. In nonlinear and/or non-Gaussian models, the expectation (12) under the distribution $p_{\theta'}(x_{0:T}|y_{1:T})$ is generally intractable and the EM algorithm can not work in such situation. An alternative, originally proposed in Celeux *et al.* (1995) and Chan and Ledolter (1995), is to use a Monte Carlo approximation of (12)

$$\hat{G}(\theta, \theta') \triangleq \frac{1}{N_s} \sum_{j=1}^{N_s} \ln p_\theta\left(x_{0:T}^j, y_{1:T}\right), \tag{15}$$

where $(x_{0:T}^j)_{j=1,\ldots,N_s}$ are N_s trajectories simulated according to the smoothing distribution $p_{\theta'}\left(x_{0:T}|y_{1:T}\right)$. This algorithm is generally named Stochastic EM (SEM) algorithm in the literature.

To implement such procedure it is necessary to generate samples of the smoothing distribution. In the literature (Olsson *et al.* 2008; Schön *et al.* 2011; Kokkala *et al.* 2014; Kantas *et al.* 2015; Picchini and Samson 2018), standard or approximate particle smoothing methods are generally used. As discussed, it is generally computationally intractable for data assimilation applications. A classical alternative in data assimilation consists in using the EnKS algorithm (Evensen and Van Leeuwen 2000) leading to the EnKS-EM algorithm (Tandeo *et al.* 2015b; Dreano *et al.* 2017). Note that this procedure does not necessarily lead to increasing the likelihood function at each iteration and may not converge. Here we explore alternative procedures based on the smoothers introduced in the previous section.

Lindsten (2013) proposed to use the CPF-AS smoother, leading to the CPF-AS-SEM algorithm. Given an initial parameter $\hat{\theta}_0$ and the first conditioning X_0^*, the algorithm is summed up as follows

- **E-step**:
  i. Draw N_s realizations by using the CPF-AS smoother (**Algorithm 2**) once with fixed parameter $\hat{\theta}_{r-1}$, the conditioning X_{r-1}^* and the given observations $y_{1:T}$, wherein X_r^* is new conditioning trajectory obtained after updating.
  ii. Compute the quantity $\hat{G}(\theta, \hat{\theta}_{r-1})$ via Eq. (14) and Eq. (15).
- **M-step**: Compute $\hat{\theta}_r = \arg\max_\theta\ \hat{G}(\theta, \hat{\theta}_{r-1})$,

For each iteration r, N_s smoothing trajectories are sampled given the previous conditioning trajectory X^*_{r-1}. It creates some (stochastic) dependence between the successive steps of the algorithms. This leads to such algorithm slightly different from regular EM algorithms. In Lindsten (2013) the author applied a similar algorithm to univariate models. Numerical results showed that this approach can give reasonable estimates with only few particles. Unfortunately, the degeneracy issue in the CPF-AS sampler may lead to estimates with some bias and large variance.

As discussed in the previous section, the CPF-BS smoother (**Algorithm 3**) outperforms the CPF-AS in producing better descriptions of the smoothing distribution. We hence propose a new method, CPF-BS-SEM, as an alternative to the CPF-AS-SEM for parameter estimation. The complete algorithm of the CPF-BS-SEM is presented as

Algorithm 4: Stochastic EM algorithm using Conditional Particle Filtering-Backward Simulation (CPF-BS-SEM) given T observations $y_{1:T}$.

- Initial setting: $\hat{\theta}_0$, X^*_0.
- For iteration $r \geq 1$,
  + **E-step**:
    i. Simulate N_s samples by running CPF-BS smoother (**Algorithm 3**) once with fixed parameter $\hat{\theta}_{r-1}$, the conditioning X^*_{r-1} and the given observations $y_{1:T}$, wherein X^*_r is new conditioning trajectory obtained after updating.
    ii. Compute the quantity $\hat{G}(\theta, \hat{\theta}_{r-1})$ via Eq. (14) and Eq. (15).

  + **M-step**: compute $\hat{\theta}_r = \arg\max_{\theta} \hat{G}(\theta, \hat{\theta}_{r-1})$.

  end for.

The **E-step** of this algorithm permits to get several samples at the same computational cost that the one of CPF-AS-SEM which suffers from degeneracy. That is expected to give better estimates of the quantity G in Eq. (15). Depending on the complexity of the SSM (1), analytical or numerical procedure may be applied in the **M-step** to maximize $\hat{G}$. For Gaussian state-space models the explicit expressions of estimators can be obtained directly as in the following example. Such models have popularly been considered in data assimilation context and are thus used to validate the algorithms in this article.

Example: Estimate parameter $\theta = \{Q, R\}$ in a Gaussian model

$$\begin{cases} x_t = m(x_{t-1}) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, Q) \\ y_t = h(x_t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, R). \end{cases} \tag{16}$$

where m and h can be linear or nonlinear functions.
Through Eq. (14) and (15), an estimate of the function G of this Gaussian model is express by

$$\begin{aligned} \hat{G}(\theta, \hat{\theta}_{r-1}) = &-\frac{T}{2}\ln|Q| - \frac{T}{2}\ln|R| + C \\ &- \frac{1}{2N_s}\sum_{t=1}^{T}\sum_{j=1}^{N_s}\left[x_t^{s,(j)} - m\left(x_{t-1}^j\right)\right]' Q^{-1}\left[x_t^j - m\left(x_{t-1}^j\right)\right] \\ &- \frac{1}{2N_s}\sum_{t=1}^{T}\sum_{j=1}^{N_s}\left[y_t - h\left(x_t^j\right)\right]' R^{-1}\left[y_t - h\left(x_t^j\right)\right] \end{aligned} \tag{17}$$

where C is independent to θ and $(x_t^j)_{t=0:T}^{j=1:N_s}$ are sampled from the CPF-BS smoother with respect to $\hat{\theta}_{r-1}$.
Hence, an analytical expression of the estimator $\hat{\theta}_r = \{\hat{Q}_r, \hat{R}_r\}$ of θ which maximizes (17) is

$$\begin{aligned} \hat{Q}_r &= \frac{1}{TN_s}\sum_{t=1}^{T}\sum_{j=1}^{N_s}\left[x_t^j - m\left(x_{t-1}^j\right)\right]\left[x_t^j - m\left(x_{t-1}^j\right)\right]' \\ \hat{R}_r &= \frac{1}{TN_s}\sum_{t=1}^{T}\sum_{j=1}^{N_s}\left[y_t - h\left(x_t^j\right)\right]\left[y_t - h\left(x_t^j\right)\right]'. \end{aligned} \tag{18}$$

Different strategies have been proposed in the literature for choosing the number N_s of simulated trajectories in the **E-step**. If N_s is large, then the law of large numbers implies that $\hat{G}$ is a good approximation of G and the SEM algorithm is close to the EM algorithm. It is generally not possible to run the SEM algorithm with a large value of N_s. In such situation, it has been proposed to increase the value of N_s at each iteration of the EM (Monte Carlo EM algorithm, MCEM, see Celeux *et al.* (1995)) or to re-use the smoothing trajectories simulated in the previous iterations (stochastic approximation EM algorithm, SAEM, see Delyon *et al.* (1999); Kuhn and Lavielle (2004)). It permits to decrease the variance of the estimates obtained with the SEM algorithms. For data assimilation applications, it is generally computationally infeasible to increase significantly the value of N_s but the SAEM strategy could be explored. In this article, we only consider the combination of SEM and CPF-BS to facilitate the reading.

3. Numerical illustrations

Now we aim at validating the CPF-BS-SEM algorithm and comparing it with other EM algorithms including CPF-AS-SEM, PF-BS-SEM and EnKS-EM (such algorithms are presented in the mentioned references: Lindsten (2013); Schön *et al.* (2011) and Dreano *et al.* (2017) respectively). This is done through numerical experiments on three state-space models. An univariate linear Gaussian model (19) is first considered. For this model the KS-EM algorithm can be run to provide an exact numerical approximation to the MLE and check the accuracy of the estimates derived from the SEM algorithms. Next more complicated nonlinear models (Kitagawa (20) and a three-dimensional Lorenz-63 (21)) are considered. We focus on showing the comparisons in terms of parameter and state estimation of the CPF-BS-SEM and CPF-AS-SEM algorithms with few particles on these highly nonlinear models, where we also point out the inefficiency of the EnKS-EM algorithm.

3.1. *Linear model*

A linear Gaussian SSM is defined as

$$\begin{cases} x_t = Ax_{t-1} + \eta_t, \quad \eta_t \sim \mathcal{N}(0, Q) \\ y_t = x_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, R), \end{cases} \tag{19}$$

where $(x_t, y_t)_{t=1:T} \in \mathbb{R} \times \mathbb{R}$. $\theta = (A, Q, R)$ denotes the vector of unknown parameters where A is the autoregressive coefficient and (Q, R) are the error variances. Implementations of stochastic version of the EM algorithms for this model are discussed in Olsson *et al.* (2008); Lindsten (2013); Kantas *et al.* (2015). A sequence of measurements $y_{1:T}$ is obtained by running (19) with true parameter value $\theta^* = (0.9, 1, 1)$ and $T = 100$ (shown on Figure 9). We set up the initial conditioning trajectory X^*_0 (only for the CPF-BS-SEM and CPF-AS-SEM algorithms) as the constant sequence equal to 0 (the same choice is done for the models considered in Sections 3.2 and 3.3) and the initial parameter $\hat{\theta}_0$ is sampled from a uniform distribution $\mathcal{U}([0.5,\ 1.5]^3)$.

For the first experiment, the CPF-BS-SEM and CPF-AS-SEM algorithms are run with $N_f = N_s = 10$ particles/realizations.

© 0000

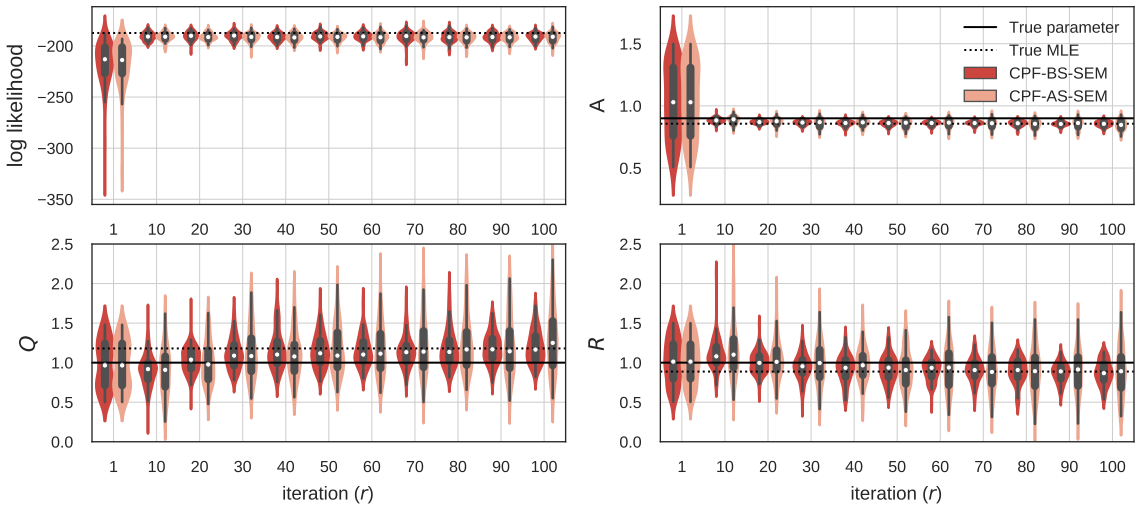

Figure 7. Comparison between CPF-BS-SEM and CPF-AS-SEM in estimating $\theta = (A, Q, R)$ and log likelihood function for the linear Gaussian SSM model (19) with true parameter $\theta^* = (0.9, 1, 1)$ and $T = 100$. The results are obtained by running 100 repetitions of the two methods with 10 particles/realizations and 100 iterations. The empirical distribution of the log-likelihood and parameter estimates is represented every 10 iterations using one violin object with (black) quantile box and (white) median point inside. The true MLE (dotted line) is computed using KS-EM with 10^4 iterations.

Since the considered algorithms are stochastic, each of them is run 100 times to show the estimators distributions. Note that in the **M-step**, the coefficient A can be easily computed using Eq. (17) before computing estimates of (Q, R) with Eq. (18). Figure 7 shows the distribution of the corresponding estimator of θ and log likelihood every 10 iterations. Because the model is linear and Gaussian, we can also run the KS-EM (Shumway and Stoffer 1982) algorithm a get an accurate approximation of the true MLE of θ. The estimate given by the KS-EM algorithm is shown on Figure 7. The differences with the true parameters values are mainly due to the sampling error of the MLE which is relatively important here because of the small sample size (only 100 observations to estimate 3 parameters). In the experiment the CPF-BS-SEM and CPF-AS-SEM algorithms start to stabilize after only 10 iterations. Even with few particles, both algorithms provide estimates which have mean values close to the true MLE. As expected, CPF-BS-SEM is clearly better than CPF-AS-SEM in terms of variance.

Then we compare the CPF-BS-SEM, CPF-AS-SEM and PF-BS-SEM algorithms varying the number of particles/realizations, $N_f = N_s \in \{10, 100, 1000\}$. The empirical distribution of the final estimators $\hat{\theta}_{100}$ obtained by the various algorithms are shown on Figure 8. The PF-BS-SEM algorithm with $N_f = N_s = 10$ or even $N_f = N_s = 100$ particles/realizations leads to estimates with a significant bias which is much bigger than the ones of other algorithms. It illustrates that the PF-BS-SEM algorithm based on the usual particle filter needs much more particles than the two other algorithms which use the idea of conditional particle filter. With $N_f = 1000$ particles, the PF-BS-SEM and CPF-BS-SEM give similar results since the effect of conditioning becomes negligible. Then comparing the performances of the CPF-BS-SEM and CPF-AS-SEM algorithms shows again that CPF-BS-SEM is better in terms of variance. The experiment was done on different T-sequences of measurements and similar results were obtained.

The reconstruction ability of the CPF-BS-SEM algorithm is displayed on Figure 9. 100 iterations of the algorithm is run once and the $N_s = 10$ trajectories simulated in each **E-step** of the last 10 iterations are stored. This produces 100 trajectories. Then empirical mean and 95% confidence interval (CI) of these 100-samples are computed and plotted on Figure 9. The root of mean square error (RMSE) between the smoothed mean and the true state is 0.6996 and the empirical coverage probability (percentage of the true states falling in the 95% CIs denoted CP hereafter) is

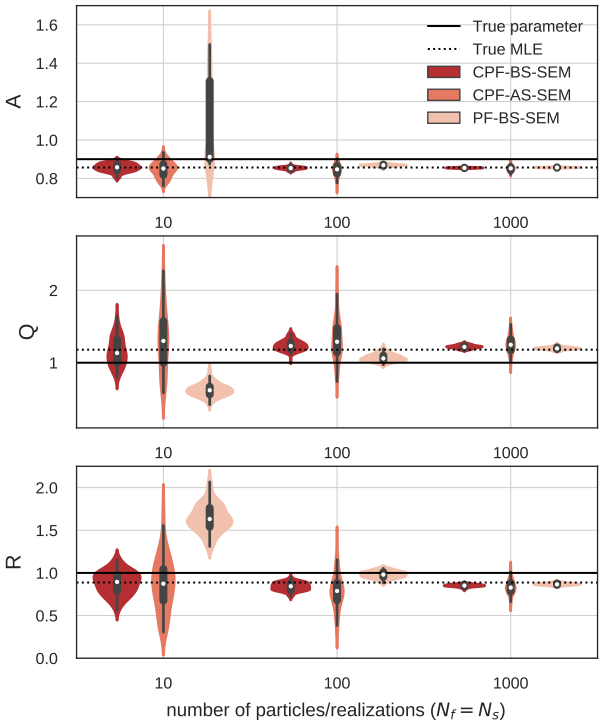

Figure 8. Comparison of the estimates of $\theta = (A, Q, R)$ at iteration 100 of CPF-BS-SEM, CPF-AS-SEM, and PF-BS-EM for the linear Gaussian SSM model (19) with true parameter $\theta^* = (0.9, 1, 1)$ and $T = 100$. These algorithms are run with different number of particles/trajectories ($N_f = N_s \in \{10, 100, 1000\}$). The true MLE (dotted line) is computed using KS-EM with 10^4 iterations.

86%. In theory the value should be close to 95%, here, the CPF-BS-SEM algorithm with non-large samples run on the short-fixed sequence of observations may give a smaller estimate of the score. An experiment to get the expected percentage is presented later (Table 1).

3.2. *Kitagawa model*

The algorithms are now applied on a highly nonlinear system widely considered in the literature to perform numerical illustrations on SSM (Kitagawa 1998; Doucet *et al.* 1998; Godsill *et al.* 2004; Schön *et al.* 2011; Kokkala *et al.* 2014). Both m and

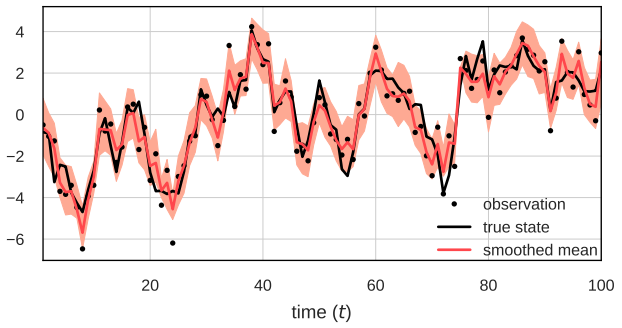

Figure 9. Reconstruction of the true state for the linear Gaussian SSM model (19) given $T = 100$ observations using the CPF-BS-SEM algorithm with 10 particles/realizations. Smoothed mean and 95% confidence interval are computed from realizations, which are simulated from last 10 iterations of the algorithm.

h of the model are nonlinear and defined as follows

$$\begin{cases} x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, Q) \\ y_t = 0.05x_t^2 + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, R) \end{cases} \tag{20}$$

where $(x_t, y_t)_{t=1:T} \in \mathbb{R} \times \mathbb{R}$. We denote $\theta = (Q, R)$ the unknown parameter. One sequence of $T = 100$ observations generated with true parameter value $\theta^* = (1, 10)$ is shown on Figure 12. Similar values are used in Godsill *et al.* (2004). The large value of the observation variance R leads to generate low quality observations and thus complicate the inference. Using only these 100 observations $y_{1:100}$, the target is to estimate θ and the true state $x_{1:100}$. The initial parameter value is simulated according to the uniform distribution $\hat{\theta}_0 \sim \mathcal{U}([1, 10]^2)$.

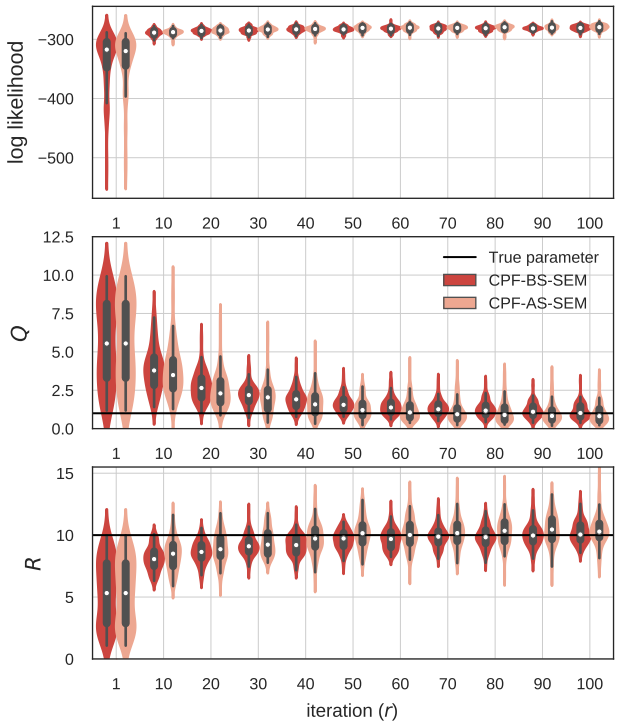

Figure 10. Comparison of the CPF-BS-SEM and CPF-AS-SEM algorithms on the Kitagawa model (20), where true parameter is $\theta^* = (1, 10)$ and number of observations is $T = 100$. The results are obtained by running 100 times of these methods with 10 particles/realizations and 100 iterations. The empirical distribution of the log-likelihood and parameter estimates is represented every 10 iterations using one violin object with (black) quantile box and (white) median point inside.

In this section, we only compare the CPF-BS-SEM and CPF-AS-SEM algorithms since PF-BS-SEM can not work with a small number of particles (as shown in the linear case) and Lindsten (2013) also illustrated that CPF-AS-SEM using $N_f = 15$ particles outperforms PF-BS-SEM using $N_f = 1500$ particles and $N_s = 300$ realizations on the Kitagawa model. In the first experiment CPF-BS-SEM and CPF-AS-SEM are run with $N_f = N_s = 10$ particles/realizations. A comparison of the two methods in terms of estimates of log likelihood and parameter $\theta = (Q, R)$ is shown in Figure 10. Even with few particles the estimates obtained with the two methods seem to stabilize after 50 iterations and again the CPF-BS-SEM algorithm permits to reduce the variance of the estimates compared to the CPF-AS-SEM algorithm .

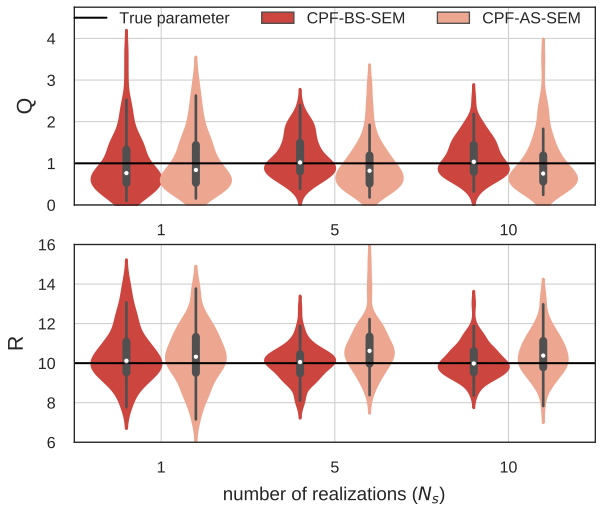

Figure 11. Comparison of the estimates of $\theta = (Q, R)$ at iteration 100 of the CPF-BS-SEM and CPF-AS-SEM algorithm on the Kitagawa model (20), where true parameter is $\theta^* = (1, 10)$ and number of observations is $T = 100$. The algorithms are run with fixed number of particles ($N_f = 10$) and different number of trajectories ($N_s \in \{1, 5, 10\}$).

In the second experiment we run the two algorithms with fixed number of particles ($N_f = 10$) but different numbers of realizations ($N_s \in \{1, 5, 10\}$). Figure 11 displays the corresponding empirical distributions of $\hat{\theta}_{100}$. It shows that CPF-AS-SEM gives almost the same distributions of estimates as CPF-BS-SEM with $N_s = 1$. Moreover CPF-AS-SEM could not improve the estimate when we increase N_s because of the degeneracy issue. CPF-BS-SEM with $N_s = 5$ and $N_s = 10$ gives better estimates in terms of bias and variance. In practice it seems useless to use a large value of N_s when using BS given forward filtering information. Here CPF-BS-SEM with $N_s = 5$ has similar performance as CPF-BS-SEM with $N_s = 10$ (see also Godsill *et al.* (2004); Lindsten *et al.* (2013); Svensson *et al.* (2015)).

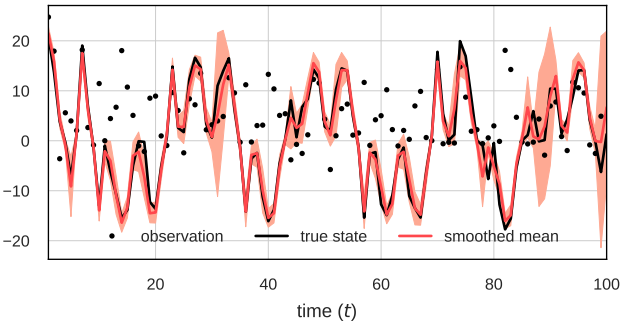

Figure 12. Reconstruction of the true state using CPF-BS-SEM with 10 particles/realizations on the Kitagawa model (20) given $T = 100$ observations. Smoothed means and 95% confidence intervals of all realizations simulated from last 10 iterations of the algorithm are presented.

Figure 12 shows the results obtained when reconstructing the latent space using the CPF-BS-SEM algorithm (using the same approach than for the linear model, based on storing the sequences simulated in the last 10 iterations of the algorithm). The mean of

the empirical smoothing distribution seems to be close to the true state. The width of the confidence intervals varies in time and is larger (eg. at $t \in [85, 90]$) when the true state is more difficult to retrieve from the observations. The RMSE and the empirical CP with respect to the empirical smoothing distribution are 2.2478 and 84%.

3.3. *Lorenz-63 model*

In this section we consider the Lorenz-63 model where only the first and last components are observed. This is one of the favorite toy models in data assimilation because it is a complex (nonlinear, non-periodic, chaotic) but low-dimensional (3 components) dynamical system. More precisely, the considered state-space model is defined as

$$\begin{cases} x_t = m(x_{t-1}) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, Q) \\ y_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} x_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, R), \end{cases} \tag{21}$$

with $(x_t, y_t)_{t=1:T} \in \mathbb{R}^3 \times \mathbb{R}^2$. The dynamical model m is related to the Lorenz (1963) model defined through the ordinary differential equation (ODE) system,

$$\frac{dx_\tau}{d\tau} = g(x_\tau), \tag{22}$$

where $g(x) = (10(x_2 - x_1),\ x_1(28 - x_3) - x_2,\ x_1 x_2 - 8/3 x_3)$, $\forall x = (x_1, x_2, x_3)' \in \mathbb{R}^3$. In order to compute $m(x_{t-1})$, we run a Runge-Kutta scheme (order 5) to integrate the system (22) on the time interval $[0, \triangle]$ with initial condition x_{t-1}. The value of $\triangle$ affects the non-linearity of the dynamical model m. According to Figure 14 (see top panels), when $\triangle = 0.01$ the relation between x_{t-1} and x_t is well described by a linear model whereas when $\triangle = 0.15$ the dynamical model has more time to evolve between successive observations and non-linearities are more pronounced. The intermediate value $\triangle = .08$ corresponding to 6-hour recorded-observed data in atmospheric application was considered in the works of Tandeo *et al.* (2015a); Lguensat *et al.* (2017) and Dreano *et al.* (2017).

For the sake of simplifying illustrations, error covariances are assumed to be diagonal. More precisely we assume that $Q = \sigma_Q^2 I_3$ and $R = \sigma_R^2 I_2$ and the unknown parameter to be estimated is $\theta = (\sigma_Q^2, \sigma_R^2) \in \mathbb{R}^2$. Note that an analytical solution can be derived for the **M-step** of the EM algorithm in this constrained model. It leads to the following expression for updating the parameters in the iteration r of the EM algorithm

$$\hat{\theta}_r = \left(\hat{\sigma}^2_{Q,r}, \hat{\sigma}^2_{R,r}\right) = \left(\frac{\mathrm{Tr}[\hat{Q}_r]}{3}, \frac{\mathrm{Tr}[\hat{R}_r]}{2}\right)$$

where $\hat{Q}_r$ and $\hat{R}_r$ come from (18). The initial parameter value of the EM algorithm is drawn using a uniform distribution $\hat{\theta}_0 \sim \mathcal{U}([0.5, 2] \times [1, 4])$.

For the first experiment we simulate $T = 100$ observations of the Lorenz-63 model (21) with the model time step $\triangle = 0.15$ (it corresponds to around 20 loops of the Lorenz-63 system) and true parameter $\theta^* = (1, 2)$ (shown on Figure 15). The CPF-BS-SEM and CPF-AS-SEM algorithms are compared on Figure 13. With only $N_f = N_s = 20$ particles/realizations, the two methods provide reasonable estimates of the parameters. The comparison has been done in different scenarios, with varying true parameter values θ^*, and similar results were obtained. A lower number of particles and realizations (eg. $N_f = N_s = 10$) can be used in these SEM algorithms but more iterations are needed (eg. 200) to obtain appropriate conditioning trajectories.

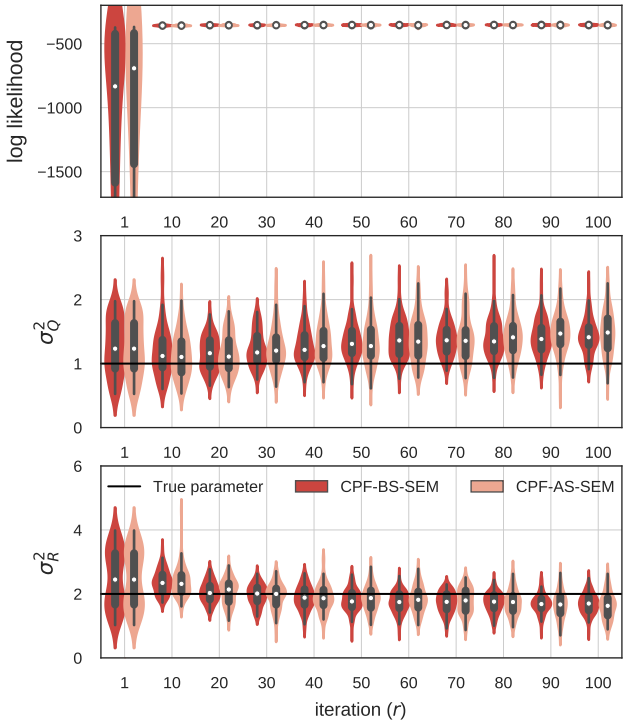

Figure 13. Comparison between CPF-BS-SEM and CPF-AS-SEM on Lorenz-63 models (21) with model time step $\triangle = 0.15$, true parameter $\theta^* = (1, 2)$ and $T = 100$ observations. Results obtained by running 100 repetitions of these methods with 20 particles/realizations and 100 iterations. The empirical distribution of the log-likelihood and parameter estimates is represented every 10 iterations using one violin object with (black) quantile box and (white) median point inside.

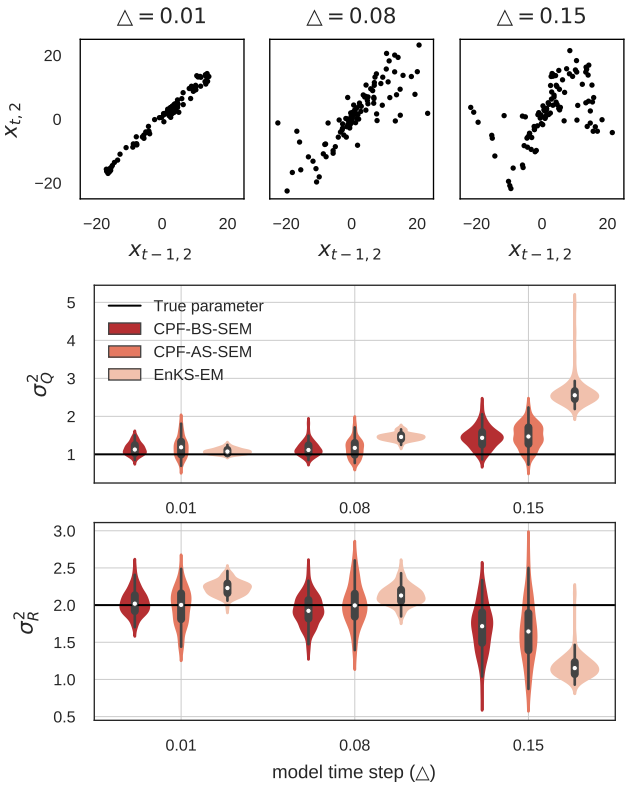

Figure 14. Comparison of the estimates of $\theta = (\sigma_Q^2, \sigma_R^2)$ for the CPF-BS-SEM, CPF-AS-SEM and EnKS-EM algorithms with 20 members/particles for the Lorenz-63 models (21) with varying model time step $\triangle \in \{0.01, 0.08, 0.15\}$, true parameter $\theta^* = (1, 2)$ and number of observations is $T = 100$. First row: scatter-plot $(x_{t-1,2}, x_{t,2})$ of the unobserved variable with respect to each model step $(\triangle)$. Last two rows: empirical distribution of the estimates of θ computed using 100 repetitions of each algorithm at the final iteration $r = 100$.

In the second experiment, we also compare the results obtained with the ones of the EnKS-EM algorithm. The EnKS-EM

algorithm with a low number N of members often gets numerical issues when computing empirical covariances. Values of N in the range $[20, 1000]$ has been chosen in lots of data assimilation schemes using EnKS (Evensen and Van Leeuwen 2000; Ueno *et al.* 2010; Ueno and Nakamura 2014; Tandeo *et al.* 2015b; Lguensat *et al.* 2017; Dreano *et al.* 2017; Pulido *et al.* 2017). We have chosen to run the three algorithms with 20 members/particles to have comparable computational costs. The experiment is run on different simulated sequences of length $T = 100$, where the model time step in (21) varies $\triangle \in \{0.01, 0.08, 0.15\}$. According to Figure 14, the CPF-BS-SEM algorithm gives better estimates compared to the CPF-AS-SEM and EnKS-EM algorithms. The bias and variance of the estimates obtained with the three algorithms increase with $\triangle$ and the non-linearity of the dynamic model. Note that the discrepancy increases quicker for the EnKS-EM algorithm. We found that it completely fail when $\triangle = 0.25$ whereas the CPF-BS-SEM and CPF-AS-SEM algorithms still give reasonable estimates (not shown; the Python library is available for such tests). This illustrates that the EnKS-EM algorithm is less robust to non-linearities compared to the two algorithms based on the conditional particle filter. Figure 15 shows

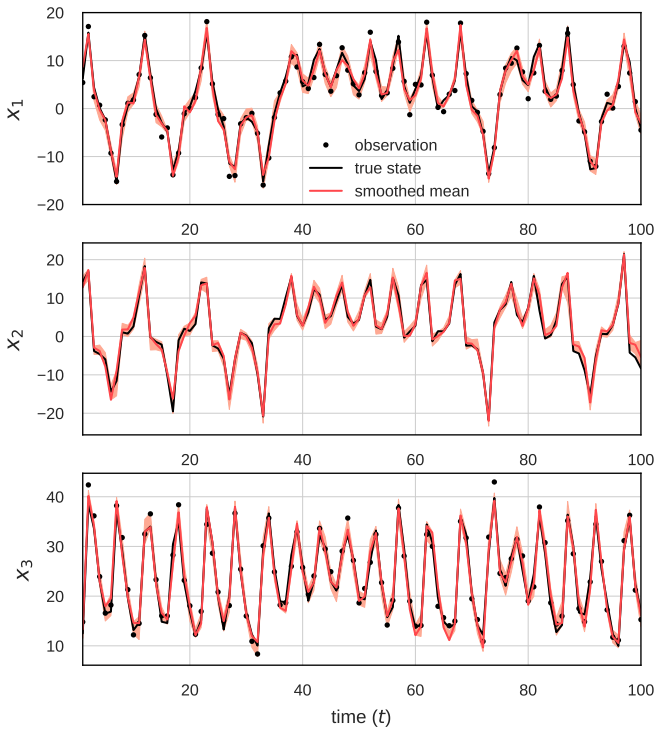

Figure 15. Reconstruction of the true state for Lorenz-63 model (21) with $\triangle = 0.15$, $T = 100$ by using the CPF-BS-SEM algorithm with 20 particles/realizations. Smoothed mean and 95% confidence interval of all realizations of last 10 iterations of the algorithm are computed.

the results obtained when reconstructing the latent space using the CPF-BS-SEM algorithm (using the same approach as for the linear model, based on storing the sequences simulated in the last 10 iterations of the algorithm). The smoothed means of three variables are close to the true state and RMSEs for each component are $(0.8875, 1.0842, 1.2199)$. 95% CIs covers the true state components with respect to CPs $(87\%, 84\%, 88\%)$. Although the second variable x_2 is unobserved the algorithm provides a reasonable reconstruction of this component.

Finally we preformed a cross-validation exercise to check the out-of-sample reconstruction ability of the proposed method. First we compute the mean values of the final estimates of the CPF-BS-SEM and the CPF-AS-SEM shown on Figure 14. This provides point estimates for the parameters based on a first sequence of $T = 100$ observations. Then the CPF-BS and CPF-AS algorithms are run on an another (test) sequence of observations of length $T' = 1000$ of the Lorenz-63 model with the same parameter values. This provide an estimate of the smoothing distribution for the test sequence. Table 1 gives RMSEs and CPs for the unobserved component of all smoothing samples with respect to number of iterations in $\{5, 10, 50, 100\}$. As expected the CPs of the two algorithms tend to 95% when the number of samples is large enough. The CPF-BS smoother clearly outperforms the CPF-AS as it gets smaller RMSEs and larger CPs with small number of iterations and thus less computational cost. Similar conclusions hold true when comparing the scores for the first and third components.

Table 1. Comparison of the reconstruction quality between the CPF-BS and CPF-AS smoothers on a test sequence in terms of root of mean square error-(RMSE) and coverage probability (CP). The parameters are estimated on a sequence of length $T = 100$ (mean values of the final estimates shown on Figure 13). The CPF-BS and CPF-AS algorithms are run on a test sequence simulated using the Lorenz-63 model (21) with $\triangle = 0.15, T' = 1000, \theta^* = (1, 2)$. The two scores are computed on the second component of the samples drawn from these smoothers with 20 particles/realizations.

number of iterations		5	10	50	100
CPF-BS	RMSE	1.5310	1.2507	1.0098	0.9891
	CP	83.8%	88.6%	94.3%	95.7%
CPF-AS	RMSE	2.1595	1.5711	1.0125	0.9769
	CP	58.9%	78.5%	92.0%	94.8%

4. Conclusion

CPF-BS and CPF-AS algorithms permits to simulate conditional trajectories of the latent state given observations with a low number of particles (eg. $5 - 20$). That allows to apply particle filtering (smoothing) on data assimilation context. Compare to EnKS, these algorithms permits to consider highly nonlinear and/or non-Gaussian state-space models. The CPF-BS sampler leads to a better description of the smoothing distribution at the same computational cost as the CPF-AS ones which only permits to generate one trajectory. Combined with EM methodology, it provides an efficient method to estimate the parameters such as error covariances. It also permits a better estimation of the uncertainty on the reconstructed space in data assimilation.

References

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

Berry T, Sauer T. 2013. Adaptive ensemble kalman filtering of non-linear systems. *Tellus A: Dynamic Meteorology and Oceanography* **65**(1): 20 331.

Cappé O, Godsill SJ, Moulines E. 2007. An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceedings of the IEEE* **95**(5): 899–924, doi:10.1109/jproc.2007.893250.

Carrassi A, Bocquet M, Bertino L, Evensen G. 2017. Data assimilation in the geosciences-an overview on methods, issues and perspectives. *arXiv preprint arXiv:1709.02798* .

Celeux G, Chauveau D, Diebolt J. 1995. On Stochastic Versions of the EM Algorithm. Research Report RR-2514, INRIA.

Chan KS, Ledolter J. 1995. Monte carlo em estimation for time series models involving counts. *Journal of the American Statistical Association* **90**(429): 242–252, doi:10.1080/01621459.1995.10476508.

Chopin N, Singh SS, *et al.* 2015. On particle gibbs sampling. *Bernoulli* **21**(3): 1855–1883.

Delyon B, Lavielle M, Moulines E. 1999. Convergence of a stochastic approximation version of the em algorithm. *Annals of statistics* : 94–128.

Dempster AP, Laird NM, Rubin DB. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **39**(1): 1–38.

Desroziers G, Berre L, Chapnik B, Poli P. 2005. Diagnosis of observation, background and analysis-error statistics in observation space. *Quarterly Journal of the Royal Meteorological Society* **131**(613): 3385–3396.

Douc R, Cappé O. 2005. Comparison of resampling schemes for particle filtering. In: *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*. IEEE, pp. 64–69.

Douc R, Garivier A, Moulines E, Olsson J. 2009. On the forward filtering backward smoothing particle approximations of the smoothing distribution in general state spaces models. *arXiv preprint arXiv:0904.0316* .

Doucet A, de Freitas N, Gordon N (eds). 2001. *Sequential monte carlo methods in practice*. Statistics for Engineering and Information Science, Springer-Verlag, New York, ISBN 0-387-95146-6, doi:10.1007/978-1-4757-3437-9.

Doucet A, Godsill S, Andrieu C. 1998. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering.

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

Dreano D, Tandeo P, Pulido M, Ait-El-Fquih B, Chonavel T, Hoteit I. 2017. Estimating model-error covariances in nonlinear state-space models using kalman smoothing and the expectation–maximization algorithm. *Quarterly Journal of the Royal Meteorological Society* **143**(705): 1877–1885.

Evensen G, Van Leeuwen PJ. 2000. An ensemble kalman smoother for nonlinear dynamics. *Monthly Weather Review* **128**(6): 1852–1867.

Ghil M, Malanotte-Rizzoli P. 1991. Data assimilation in meteorology and oceanography. *Advances in geophysics* **33**: 141–266.

Godsill SJ, Doucet A, West M. 2004. Monte carlo smoothing for nonlinear time series. *Journal of the american statistical association* **99**(465): 156–168.

Hol JD, Schon TB, Gustafsson F. 2006. On resampling algorithms for particle filters. In: *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*. IEEE, pp. 79–82.

Kantas N, Doucet A, Singh SS, Maciejowski J, Chopin N, *et al.* 2015. On particle methods for parameter estimation in state-space models. *Statistical science* **30**(3): 328–351.

Kitagawa G. 1998. A self-organizing state-space model. *Journal of the American Statistical Association* : 1203–1215.

Kokkala J, Solin A, Särkkä S. 2014. Expectation maximization based parameter estimation by sigma-point and particle smoothing. In: *FUSION*. IEEE, pp. 1–8.

Kuhn E, Lavielle M. 2004. Coupling a stochastic approximation version of em with an mcmc procedure. *ESAIM: Probability and Statistics* **8**: 115–131.

Le Gland F, Monbet V, Tran VD. 2009. Large sample asymptotics for the ensemble kalman filter. PhD thesis, INRIA.

Lguensat R, Tandeo P, Ailliot P, Pulido M, Fablet R. 2017. The analog data assimilation. *Monthly Weather Review* **145**(10): 4093–4107.

Lindsten F. 2013. An efficient stochastic approximation EM algorithm using conditional particle filters. In: *ICASSP*. IEEE, pp. 6274–6278.

Lindsten F, Jordan MI, Schön TB. 2014. Particle gibbs with ancestor sampling. *Journal of Machine Learning Research* **15**: 2145–2184.

Lindsten F, Schön T, Jordan MI. 2012. Ancestor sampling for particle gibbs. In: *Advances in Neural Information Processing Systems*. pp. 2591–2599.

Lindsten F, Schön TB. 2011. On the use of backward simulation in particle markov chain monte carlo methods. *arXiv preprint arXiv:1110.2873* .

Lindsten F, Schön TB. 2012. On the use of backward simulation in the particle gibbs sampler. In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, pp. 3845–3848.

Lindsten F, Schön TB, *et al.* 2013. Backward simulation methods for monte carlo statistical inference. *Foundations and Trends® in Machine Learning* **6**(1): 1–143.

Lorenz EN. 1963. Deterministic nonperiodic flow. *Journal of the atmospheric sciences* **20**(2): 130–141.

McLachlan G, Krishnan T. 2007. *The em algorithm and extensions*, vol. 382. John Wiley & Sons.

Miyoshi T. 2011. The gaussian approach to adaptive covariance inflation and its implementation with the local ensemble transform kalman filter. *Monthly Weather Review* **139**(5): 1519–1535.

Olsson J, Cappé O, Douc R, Moulines E, *et al.* 2008. Sequential monte carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli* **14**(1): 155–179.

Picchini U, Samson A. 2018. Coupling stochastic em and approximate bayesian computation for parameter inference in state-space models. *Computational Statistics* **33**(1): 179–212.

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

Poterjoy J, Anderson JL. 2016. Efficient assimilation of simulated observations in a high-dimensional geophysical system using a localized particle filter. *Monthly Weather Review* **144**(5): 2007–2020.

Pulido M, Tandeo P, Bocquet M, Carrassi A, Lucini M. 2017. Stochastic parameterization identification using ensemble kalman filtering combined with expectation-maximization and newton-raphson maximum likelihood methods. *arXiv preprint arXiv:1709.07328* .

Schön TB, Wills A, Ninness B. 2011. System identification of nonlinear state-space models. *Automatica* **47**(1): 39 – 49, doi:http://dx.doi.org/10.1016/j.automatica.2010.10.013.

Shumway RH, Stoffer DS. 1982. An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis* **3**(4): 253–264.

Snyder C. 2011. Particle filters, the optimal proposal and high-dimensional systems. In: *Proceedings of the ECMWF Seminar on Data Assimilation for atmosphere and ocean*. pp. 1–10.

Stroud JR, Bengtsson T. 2007. Sequential state and variance estimation within the ensemble kalman filter. *Monthly Weather Review* **135**(9): 3194–3208.

Stroud JR, Katzfuss M, Wikle CK. 2017. A bayesian adaptive ensemble kalman filter for sequential state and parameter estimation. *Monthly Weather Review* (2017).

Svensson A, Schön TB, Kok M. 2015. Nonlinear state space smoothing using the conditional particle filter. *arXiv preprint arXiv:1502.03697* .

Tandeo P, Ailliot P, Ruiz J, Hannart A, Chapron B, Cuzol A, Monbet V, Easton R, Fablet R. 2015a. Combining analog method and ensemble data assimilation: application to the lorenz-63 chaotic system. In: *Machine Learning and Data Mining Approaches to Climate Science*, Springer, pp. 3–12.

Tandeo P, Pulido M, Lott F. 2015b. Offline parameter estimation using enkf and maximum likelihood error covariance estimates: Application to a subgrid-scale orography parametrization. *Quarterly Journal of the Royal Meteorological Society* **141**(687): 383–395.

Ueno G, Higuchi T, Kagimoto T, Hirose N. 2010. Maximum likelihood estimation of error covariances in ensemble-based filters and its application to a coupled atmosphere–ocean model. *Quarterly Journal of the Royal Meteorological Society* **136**(650): 1316–1343.

Ueno G, Nakamura N. 2014. Iterative algorithm for maximum-likelihood estimation of the observation-error covariance matrix for ensemble-based filters. *Quarterly Journal of the Royal Meteorological Society* **140**(678): 295–315.

Ueno G, Nakamura N. 2016. Bayesian estimation of the observation-error covariance matrix in ensemble-based filters. *Quarterly Journal of the Royal Meteorological Society* **142**(698): 2055–2080.

Van Leeuwen PJ. 2015. Nonlinear data assimilation for high-dimensional systems. In: *Nonlinear Data Assimilation*, Springer, pp. 1–73.

Work DB, Blandin S, Tossavainen O, Piccoli B, Bayen AM. 2010. A traffic model for velocity data assimilation. *Applied Mathematics Research eXpress* **2010**(1): 1–35.

Zhen Y, Harlim J. 2015. Adaptive error covariances estimation methods for ensemble kalman filters. *Journal of computational physics* **294**: 619–638.

Zhu M, van Leeuwen PJ, Zhang W. 2017. Estimating model error covariances using particle filters. *Quarterly Journal of the Royal Meteorological Society* doi:10.1002/qj.3132.