Comparison of simulation-based algorithms for parameter estimation and state reconstruction in nonlinear state-space models

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

DISCRETE AND CONTINUOUS
DYNAMICAL SYSTEMS SERIES S

doi:10.3934/dcdss.2022054

COMPARISON OF SIMULATION-BASED ALGORITHMS FOR PARAMETER ESTIMATION AND STATE RECONSTRUCTION IN NONLINEAR STATE-SPACE MODELS

Thi Tuyet Trang Chau*

Univ Rennes, IRMAR-UMR CNRS 6625, F-35000 Rennes, France

Pierre Ailliot

Univ Brest, CNRS UMR 6205, Laboratoire de Mathematiques de Bretagne Atlantique, France

Valérie Monbet

Univ Rennes, INRIA/SIMSMART, CNRS, IRMAR-UMR 6625, F-35000 Rennes, France

Pierre Tandeo

IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France

Abstract. This study aims at comparing simulation-based approaches for estimating both the state and unknown parameters in nonlinear state-space models. Numerical results on different toy models show that the combination of a Conditional Particle Filter (CPF) with Backward Simulation (BS) smoother and a Stochastic Expectation-Maximization (SEM) algorithm is a promising approach. The CPFBS smoother run with a small number of particles allows to explore efficiently the state-space and simulate relevant trajectories of the state conditionally to the observations. When combined with the SEM algorithm, this algorithm provides accurate estimates of the state and the parameters in nonlinear models, where the application of EM algorithms combined with a standard particle smoother or an ensemble Kalman smoother is limited.

1. **Introduction.** State space models (SSMs) are used in many fields such as geoscience, economics, statistics, computer science, neuroscience, and electrical engineering since they provide a flexible and interpretable framework for analyzing many signal and time series (see e.g., [21, 3, 2]). General SSMs are defined by the recursive equation,

$$\begin{cases} \mathbf{x}_t = \mathcal{M}_\theta\left(\mathbf{x}_{t-1},\ \eta_t\right) \\ \mathbf{y}_t = \mathcal{H}_\theta\left(\mathbf{x}_t,\ \epsilon_t\right) \end{cases} \tag{1}$$

where $\mathbf{x}_t$ denotes the latent (i.e., non observed) state and $\mathbf{y}_t$ the observations at time t. $\mathcal{M}_\theta$ and $\mathcal{H}_\theta$ describe respectively the dynamical evolution of the latent state $\{\mathbf{x}_t\}$ and the transformation between the latent state and the observations. $\{\eta_t\}$ and $\{\epsilon_t\}$ are independent white noise sequences with covariance matrices denoted respectively

2020 *Mathematics Subject Classification.* Primary: 62M05, 62F10, 62F15, 62F86.

Key words and phrases. EM algorithms, conditional particle filtering, backward simulation, nonlinear models, statistical inference.

* Corresponding author: Thi Tuyet Trang Chau,

Present address: Laboratoire des Sciences du Climat et de l'Environnement (LSCE/ IPSL UMR CEA-CNRS-UVSQ), F-91191 Gif-Sur-Yvette Cedex, France.

$\mathbf{Q}$ and $\mathbf{R}$. They describe respectively the various sources of uncertainties present in the dynamics of the state and observation errors. $\theta \in \Theta$ denotes the vector of unknown parameters. For instance, θ may contain parameters in the dynamical model $\mathcal{M}_\theta$, observation operator $\mathcal{H}_\theta$, and error covariance matrices $(\mathbf{Q}, \mathbf{R})$.

When working with state-space models, a usual problem consists in reconstructing the latent state $\mathbf{x}_t$ at time t given a sequence of observations $\mathbf{y}_{1:T} = \{\mathbf{y}_1, ..., \mathbf{y}_T\}$. Filtering corresponds to the case where observations are available until the time t (i.e., $T = t$) whereas smoothing to the case where observations are available after time t (i.e., $T > t$). Another key question is to identify a reasonable value of the unknown parameter θ. Actually, both questions are closely related. Indeed, incorrect values of θ may lead to bad reconstructions of the latent space. This is 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 state (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 SSMs (see also in [4]). Various methods have been proposed in the

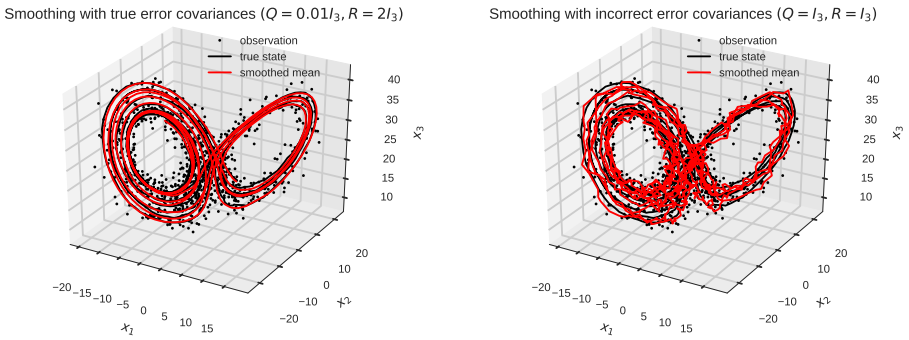

FIGURE 1. Impact of parameter values on smoothing distributions for the Lorenz-63 model (20). The true state (black curve) and observations (black points) have been simulated with $\theta^* = (\mathbf{Q}, \mathbf{R}) = (0.01\mathbf{I}_3, 2\mathbf{I}_3)$. The mean of the smoothing distributions (red curve) are computed using a standard particle smoother [16] with 100 particles. Results obtained with the true parameter value $\theta^* = (0.01\mathbf{I}_3, 2\mathbf{I}_3)$ (left panel) and a wrong parameter value $\tilde{\theta} = (\mathbf{I}_3, \mathbf{I}_3)$ (right panel) are plotted.

literature to estimate the parameters of SSMs and recent reviews can be found in [26, 45]. In this paper we focus on maximum likelihood estimation which is probably the most usual approach. There are two main 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 [26] *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 models with latent components. Since the seminal work of [14], various variants of the EM algorithm have been proposed in the literature (see e.g. [10, 26, 32, 39, 42, 45] and references therein). 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 a convergence criterion is reached.

Within the EM machinery, the more challenging issue is generally to compute the smoothing distribution. For linear Gaussian SSMs, the Kalman smoother (KS) [43] 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. The ensemble Kalman smoother (EnKS) [9, 22, 23] and its variants [6, 5, 7] are the most favorite choices for geophysical applications. They are based on a best linear unbiased estimate strategy and they allow to compute an approximation of the smoothing distribution using a relatively low number of simulations of the dynamical model. Unfortunately, for nonlinear state-space models, the approximations based on Kalman recursions generally do not converge to the smoothing distribution when the number of members increases [30]. Particle smoothers have been proposed as an alternative in [8, 16, 17, 24]. However, a large amount of particles and thus simulations of the dynamical model is typically required to get good approximations of the smoothing distribution. Since 2010, conditional particle filters (CPFs) [34, 36, 33, 44] pioneered by [1] have been developed as an alternative strategy to approximate smoothing distributions at a lower computational cost. Contrary to the more usual particle smoothing algorithms discussed above, CPFs simulate samples of the smoothing distribution using an iterative algorithm. At each iteration, one conditioning trajectory is plugged in a standard particle smoothing scheme. It helps the algorithm 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 [32], the author proposes to use a smoothing algorithm based on CPF, named Conditional particle filtering-Ancestor sampling (CPFAS), within a Stochastic Expectation-Maximization (SEM) algorithm (CPFAS-SEM algorithm). The authors showed using numerical simulations of univariate toy models that the algorithm can estimate the variances $\mathbf{Q}$ and $\mathbf{R}$ using only a few particles. However, CPFAS suffers from degeneracy (see [33]) and consequently the estimators obtained with CPFAS-SEM may be biased and/or have large variance.

In order to avoid the above-mentioned degeneracy issue, we propose in the present paper to combine CPF with the Backward Simulation (BS) algorithm originally proposed in [24] and the SEM algorithm (CPFBS-SEM algorithm). The main contribution of this paper is to show, using numerical simulations, that the proposed algorithm outperforms other EM algorithms which have been proposed in the literature in terms of parameter estimation, state reconstruction, and computational cost. We also provide an open-source Python library of all mentioned algorithms which is available on-line at `https://github.com/tchau218/parEMDA`.

The paper is organized as follows. In Section 2, we first remind sequential smoothing methods in an incremental way which permits to highlight the differences from one algorithm to the other. It starts from the usual particle filter/smoother and ends with the conditional particle smoother referred to as CPFBS. Then, the combination of CPFBS and the SEM algorithms is described. The performances of the different algorithms are compared in Section 3 using numerical experiments on toy models. Section 4 contains conclusions and perspectives.

2. Methods.

2.1. Smoothing using conditional particle-based methods.

2.1.1. *Particle Filtering (PF) and Conditional Particle Filtering (CPF).* In the state-space model (1), the latent state $\{\mathbf{x}_t\}$ is a Markov process with values in $\mathcal{X}$, defined by its initial distribution $p_\theta(\mathbf{x}_0)$ and transition kernel $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$. The observations $\{\mathbf{y}_t\}$ with values in $\mathcal{Y}$ are conditionally independent given the state process and we denote $p_\theta(\mathbf{y}_t|\mathbf{x}_t)$ the conditional distribution of $\mathbf{y}_t$ given $\mathbf{x}_t$. The transition kernel $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$ depends on both the dynamical model $\mathcal{M}_\theta$ and the distribution of the model error η_t whereas the conditional observation distribution $p_\theta(\mathbf{y}_t|\mathbf{x}_t)$ is a function of the observation model $\mathcal{H}_\theta$ and the distribution of the observation error ϵ_t.

Given a fixed vector θ and a sequence of length T of observations $\mathbf{y}_{1:T} = (\mathbf{y}_1, ..., \mathbf{y}_T)$, a classical inference problem consists in computing the filtering distributions $p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t})$ and smoothing distributions $p_\theta(\mathbf{x}_t|\mathbf{y}_{1:T})$. For linear Gaussian models, the filtering distributions are Gaussian distributions whose means and covariances can be computed using the Kalman recursions. When state-space models are nonlinear or non-gaussian, the filtering distributions do not admit a closed form and particle filtering (PF) methods have been proposed to compute approximations of these quantities [8, 17, 18]. The general PF algorithm is based on the following relation between the filtering distributions at time $t-1$ and t,

$$p_\theta(x_{0:t}|\mathbf{y}_{1:t}) = \frac{p_\theta(\mathbf{y}_t|\mathbf{x}_t)\; p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{y}_t|\mathbf{y}_{1:t-1})}\; p_\theta(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}). \tag{2}$$

Note that if we are able to compute the joint filtering distribution $p_\theta(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, then it is possible to deduce the marginal filtering distribution $p_\theta(\mathbf{x}_t|\mathbf{y}_{1:t})$ by integrating over the variables $\mathbf{x}_{0:t-1}$.

PF is based on importance sampling and it leads to an approximation of $p_\theta(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ as follows,

$$\widehat{p}_\theta\left(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}\right) = \sum_{i=1}^{N_f} \delta_{\mathbf{x}_{0:t}^{(i)}}\left(\mathbf{x}_t\right) w_t^{(i)} \tag{3}$$

where N_f denotes the number of particles, δ_x the Dirac function, and $\{w_t^{(i)}\}$ are normalized positive weights. Starting from an approximation (3) at time $t-1$, the iteration at time t of PF algorithms usually consists of the three main steps described below (see left panel of Figure 2 for an illustration).

- **Resampling.** A resampling method is used to duplicate particles with large weights and remove particles with very small weights (see in [15, 25] for a discussion on different resampling methods).
- **Forecasting.** It consists in propagating the particles from time $t-1$ to time t with a proposal kernel $\pi_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})$.
- **Weighting.** Importance weights $\{w_t^{(i)}\}^{i=1:N_f}$ of the particles $\{\mathbf{x}_{0:t}^{(i)}\}^{i=1:N_f}$ are computed according to the formula

$$\begin{aligned} W\left(\mathbf{x}_{0:t}\right) &= \frac{p_\theta\left(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}\right)}{\pi_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})} \\ &\overset{(2)}{\propto} \frac{p_\theta\left(\mathbf{y}_t|\mathbf{x}_t\right) p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})}{\pi_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})}\; p_\theta\left(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}\right). \end{aligned} \tag{4}$$

The PF algorithm is presented in **Algorithm 1**. Notation $\{I_t^i\}_{t=1:T}^{i=1:N_f}$ in **Algorithm 1** is used to store the indices of the particles across time steps and permits to reconstruct the past trajectory of a particle. This is further discussed below since it is a key ingredient in some smoothing algorithms discussed in this study. Also note that, in a general PF algorithm, particles can be propagated according to any proposal distribution π_θ (see [8, 18]). In this paper, the boostrap filter is used where $\pi_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) = p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})\ p_\theta(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})$. Accordingly, the forecasting step consists in sampling with respect to the dynamical model $\mathcal{M}_\theta$ and the importance weight function (4) can be simplified as $W(\mathbf{x}_{0:t}) \propto p_\theta(\mathbf{y}_t|\mathbf{x}_t)$.

One of the drawback of PF algorithms is that a large number of particles is generally required to get a good approximation of the filtering distributions and this may lead to prohibitive computational costs in practical applications. Conditional particle filtering (CPF) was introduced in [1] as an alternative to approximate the smoothing distributions using a lower number of particles. CPF algorithms differ from PF algorithms by adding a replacing step between the forecasting and weighting steps. In this step, one of the particle path is replaced by a conditioning trajectory $\mathbf{X}^* = (\mathbf{x}_1^*, \cdots, \mathbf{x}_T^*) \in \mathcal{X}^T$. For instance, if the N_f-th particle is replaced it leads to the following scheme at time t,

$$\mathbf{x}_t^{(i)} = \begin{cases} \mathbf{x}_t^{(i)} \sim \pi_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(I_t^i)}, \mathbf{y}_{1:t}), & \forall i = 1:N_f - 1 \\ \mathbf{x}_t^*, \quad i = N_f. \end{cases} \tag{5}$$

Similarly to PF, the resulting sample $\{\mathbf{x}_t^{(i)}\}^{i=1:N_f}$ is weighted according to (4).

In **Algorithm 1**, the differences between PF and CPF algorithms are presented. The additional ingredients of CPF are highlighted in grey. The general principle of

---

Algorithm 1: Particle Filtering (PF)/ ***Conditional Particle Filtering (CPF)*** **given** ***the conditioning sequence*** $\mathbf{X}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \cdots, \mathbf{x}_T^*)$ ***(only for CPF)*****, observations** $\mathbf{y}_{1:T}$**, and parameter** θ**.**

- Initialization:
  - Sample $\{\mathbf{x}_0^{(i)}\}^{i=1:N_f} \sim\ p_\theta(\mathbf{x}_0)$.
  - Set initial weights $w_0^{(i)} = 1/N_f, \forall i = 1:N_f$.
- For $t = 1:T$,
  - **Resampling**: draw indices $\{I_t^i\}^{i=1:N_f}$ with respect to particle weights $\{w_{t-1}^{(i)}\}^{i=1:N_f}$.
  - **Forecasting**:

$$\mathbf{x}_t^{(i)} \sim \pi_\theta\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}^{(I_t^i)}, \mathbf{y}_{1:t}\right), \forall i = 1:N_f.$$

  - ***Replacing*** *(only for CPF): set* $\mathbf{x}_t^{(N_f)} = \mathbf{x}_t^*$ *and* $I_t^{N_f} = N_f$.
  - **Weighting**: compute $\tilde{w}_t^{(i)} = W\left(\mathbf{x}_{0:t-1}^{(I_t^i)}, \mathbf{x}_t^{(i)}\right)$ by using (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 CPF algorithm is also illustrated on Figure 2. Let us discuss informally the role of the conditioning trajectory. When selecting from a sample composed of particles simulated from the proposal kernel π_θ and the conditioning particle, two opposite

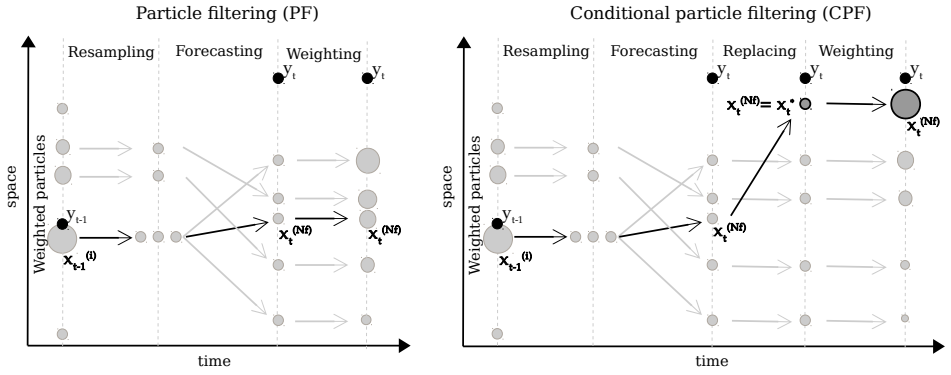

FIGURE 2. Comparison of one iteration of PF and CPF algorithms using $N_f = 5$ particles (light grey points). The differences are highlighted in black : CPF replaces the particle $\mathbf{x}_t^{(N_f)}$ of the PF with the conditioning particle $\mathbf{x}_t^*$ (dark grey point).

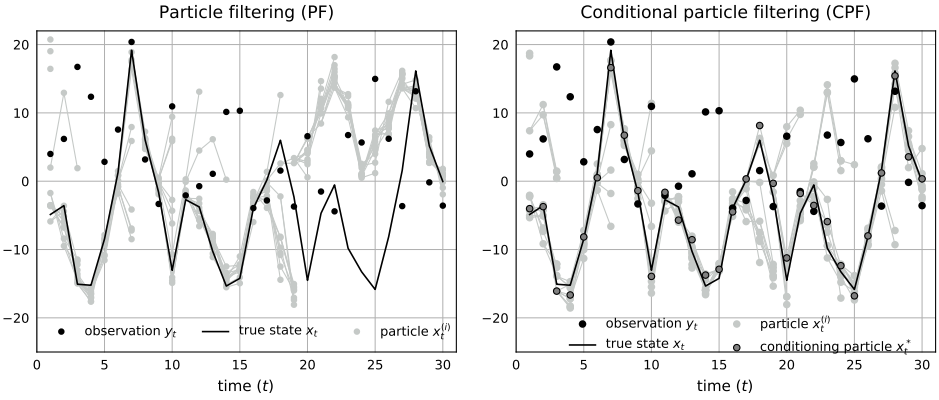

FIGURE 3. Comparisons of PF and CPF algorithms with 10 particles on the Kitagawa model defined in Section 3.2. Grey lines show the ancestors of the particles.

situations may occur. If the conditioning particle is far from the true state then it will have a low probability to be duplicated after weighting and resampling in the filtering procedure. But if the conditioning particle is close to the true state, then it will have a high probability to be duplicated and propagated at the next time step. Consequently, a good sequence set for the conditioning trajectory ensures that the CPF algorithm will explore the state space in its neighborhood, 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 (see 19). The forecasting distribution can be bimodal given this model due to the cos-term and the observation operator which is quadratic. In addition, a large value of the observation error variance R leads to observations which may not bring useful information about the state and this complicates the identification of the filtering distribution. On the left panel of Figure 3, PF starts to simulate trajectories (grey lines) which are far away from the

true state around time $t = 17$ (black line). At the same time, the observation y_t does not bring enough information on the state and the correction step is unable to correct the forecast. It leads to a bad approximation of the filtering distribution at time $t = 18$ and this effect persists during several time steps. CPF gives better results thanks to a good conditioning trajectory which helps generating relevant forecasts (see right panel of Figure 3).

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

$$\widehat{p}_\theta\left(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}\right) = \sum_{i=1}^{N_f} \delta_{\mathbf{x}_{0:T}^{(i)}}\left(\mathbf{x}_{0:T}\right) w_T^{(i)} \tag{6}$$

where $\mathbf{x}_{0:T}^{(i)}$ is a particle path, $w_T^{(i)}$ is its corresponding weight, and i is the index of the particle at the final time step. In practice, given the final particle, e.g., $\mathbf{x}_T^s = \mathbf{x}_T^{(i)}$, the rest of the smoothing path $\mathbf{x}_{0:T}^s$ can be retrieved by tracking the ancestors (parent, grandparent, etc) of the particle $\mathbf{x}_T^{(i)}$. 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 $\mathbf{x}_t^{(i)}$. The technique, which is named ancestor tracking in the literature, is illustrated on Figure 4 with $N_f = 3$ and $T = 4$. Given $i = 1$, the parent of particle $\mathbf{x}_4^{(1)}$ is the particle $\mathbf{x}_3^{(I_4^1)} = \mathbf{x}_3^{(3)}$, its grandparent is the particle $\mathbf{x}_2^{(I_3^3)} = \mathbf{x}_2^{(3)}$ and its highest ancestor is $\mathbf{x}_1^{(I_2^3)} = \mathbf{x}_1^{(2)}$. At the end, we obtain one realization $\mathbf{x}_{1:4}^s = \mathbf{x}_{1:4}^{(1)} = (\mathbf{x}_1^{(2)}, \mathbf{x}_2^{(3)}, \mathbf{x}_3^{(3)}, \mathbf{x}_4^{(1)})$.

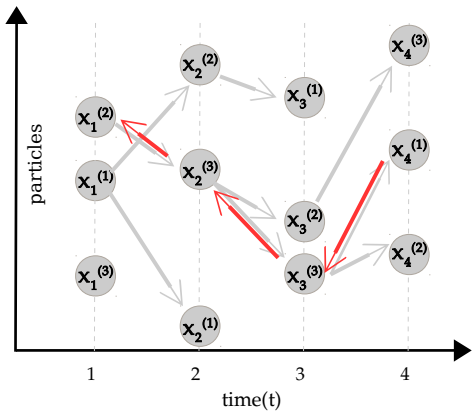

FIGURE 4. Example of ancestor tracking based on ancestral links of filtering particles. Particles (grey balls) are obtained using a filtering algorithm with $N_f = 3$ particles.

In the CPF smoother (**Algorithm 2**), the conditioning particle is updated iteratively, using the particles obtained at the final time step in the previous iteration. According to Theorem A in [1], starting from any initial conditioning trajectory, the CPF smoother (**Algorithm 2**) will generate trajectories which are approximately distributed according to the smoothing distribution after a certain number of iterations, even if the numbers of particles N_f and N_s are low. However, in practice,

Algorithm 2: Smoothing with Conditional Particle Filtering (CPF) given the conditioning $\mathbf{X}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \cdots, \mathbf{x}_T^*)$, observations $\mathbf{y}_{1:T}$, and parameter θ.

- Run CPF (**Algorithm 1**) given $\mathbf{X}^*$, observations $\mathbf{y}_{1:T}$, parameter θ, and N_f particles.
- Repeat N_s times to simulate N_s trajectories :
  - For $t = T$, draw index J_T with $p(J_T = i) \propto w_T^{(i)}$ and set $\mathbf{x}_T^s = \mathbf{x}_T^{(J_T)}$.
  - For $t < T$, set index $J_t = I_{t+1}^{J_{t+1}}$ and $\mathbf{x}_t^s = \mathbf{x}_t^{(J_t)}$.
- Update the conditioning particle $\mathbf{X}^*$ with one of these trajectories.

this algorithm generally has a poor mixing and a low rate of convergence. The main reason for this is the so-called degeneracy issue [33]: all the particles present at the final time step T share the same ancestors after a few generations. This is illustrated on Figure 4 where 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 many particles disappear whereas other particles have many children. As a consequence, all 10 particles at the final time step $T = 30$ share 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 one, except maybe for the last time steps.

To improve the mixing properties of the algorithm, [34, 37] proposed to modify the replacing step in the CPF (**Algorithm 1**) as follows. After setting the final particle $\mathbf{x}_t^{(N_f)} = \mathbf{x}_t^* \in \mathbf{X}^*$ to the conditioning particle, the index of its parent $I_t^{N_f}$ is drawn following Bayes' rule,

$$p_\theta(I_t^{N_f} = i | \mathbf{x}_t^*, \mathbf{y}_{1:t}) \propto p_\theta(\mathbf{x}_t^* | \mathbf{x}_{t-1}^{(i)})\ w_{t-1}^{(i)}. \tag{7}$$

Resampling $I_t^{N_f}$ helps to break the conditioning trajectory $\mathbf{X}^*$ into pieces so that the algorithm is less likely to simulate trajectories which are similar to $\mathbf{X}^*$. The resulting algorithm is referred to as Conditional Particle Filtering-Ancestor Sampling (CPFAS) in the sequel. In [33, 34], 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 theoretical properties of the original CPF algorithm hold true for the CPFAS (see Theorem A in Appendix).

The comparison of the left and middle panels of Figure 5 shows that resampling indices permits to obtain ancestor tracks which are different from the conditioning particles. However, the CPFAS smoother also suffers from the degeneracy problem mentioned above: all the trajectories simulated with the CPFAS coincide for $t < 20$ and thus cannot describe the spread of the smoothing distribution except maybe for the last time steps. In the next sections, we propose to replace ancestor tracking by backward simulation in order to better use the information brought by the particles.

2.1.3. *Smoothing with Conditional particle filtering- Backward simulation (CPFBS).* Backward simulation (BS) was first proposed in the statistical literature [16, 19, 24] to sample smoothing distribution in association with the regular particle filter (PFBS algorithm), and then combined with CPF in studies of [47, 33]. For BS,

the smoothing distribution is decomposed as

$$p_\theta\left(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}\right) = p_\theta\left(\mathbf{x}_T|\mathbf{y}_{1:T}\right) \prod_{t=0}^{T-1} p_\theta\left(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}\right), \tag{8}$$

where

$$p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}) \propto p_\theta\left(\mathbf{x}_{t+1}|\mathbf{x}_t\right)\ p_\theta\left(\mathbf{x}_t|\mathbf{y}_{1:t}\right) \tag{9}$$

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

$$\widehat{p}_\theta\left(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{y}_{1:t}\right) \propto \sum_{i=1}^{N_f} p_\theta(\mathbf{x}_{t+1}|\mathbf{x}_t^{(i)}) w_t^{(i)} \delta_{\mathbf{x}_t^{(i)}}\left(\mathbf{x}_t\right) \tag{10}$$

Combining the relation (8) and estimator (10), one smoothing trajectory $\mathbf{x}_{0:T}^s = \mathbf{x}_{0:T}^{\mathbf{J}_{0:T}} = (\mathbf{x}_0^{(J_0)}, \mathbf{x}_1^{(J_1)}, \cdots, \mathbf{x}_{T-1}^{(J_{T-1})}, \mathbf{x}_T^{(J_T)})$ can be simulated recursively backward in time. The algorithm is described more precisely below.

Algorithm 3: Smoothing with Conditional Particle Filtering - Backward Simulation (CPFBS) given the conditioning sequence $\mathbf{X}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*, \cdots, \mathbf{x}_T^*)$, observations $\mathbf{y}_{1:T}$, and parameter θ.

- Run CPF (**Algorithm 1**) given $\mathbf{X}^*$, observations $\mathbf{y}_{1:T}$, parameter θ, and N_f particles.
- Repeat the Backward Simulation for N_s times:
  - For $t = T$, draw J_T with $p(J_T = i) \propto w_T^{(i)}$.
  - For $t < T$,
    - Compute weights $w_t^{s,(i)} = p_\theta(\mathbf{x}_{t+1}^{(J_{t+1})}|\mathbf{x}_t^{(i)})\ w_t^{(i)}$ using (10), for all $i = 1 : N_f$.
    - Sample J_t with $p(J_t = i) \propto w_t^{s,(i)}$.

    end for
- Update the conditioning trajectory $\mathbf{X}^*$ with one of these trajectories.

Results displayed on Figure 5 suggest that the CPFBS algorithm (right panel) is better in simulating different smoothing trajectories than the ones with ancestor tracking (left and middle panels) which are prone to the degeneracy issue. Figure 6 illustrates this algorithm with a small sample simulating the state in the Kitagawa model. CPFBS is initialized with the constant sequence equal to 0 ($\mathbf{x}_t^* = 0$ for $t \in \{1, ..., T\}$). This impacts the quality of the simulated trajectories which are far from the true state at the first iteration. The conditioning trajectory is then updated at each iteration and it helps driving 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.

Generating new trajectories for each iteration conditionally on a trajectory in the previous one of the CPF, CPFAS, and CPFBS algorithms defines a Markov kernel on $\mathcal{X}^T$. Theorem A states that these Markov kernels have interesting theoretical properties (see also [12] for more results). In particular, the second property of this theorem implies that running the algorithm with any initial conditioning trajectory

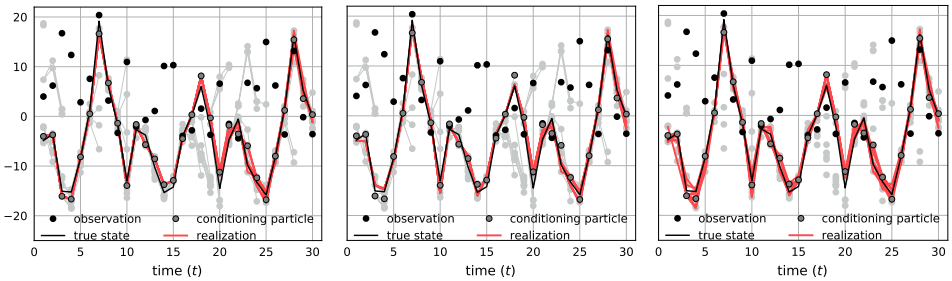

FIGURE 5. Comparison of CPF (left), CPFAS (middle), and CPFBS (right). The state (black line) and the observations (black points) have been simulated using the Kitagawa model (19) with $Q = 1$ and $R = 10$. $N_f = 10$ particles (grey points with grey lines showing the genealogy) are used in the three algorithms. The red curves show $N_s = 10$ realizations simulated with the algorithms.

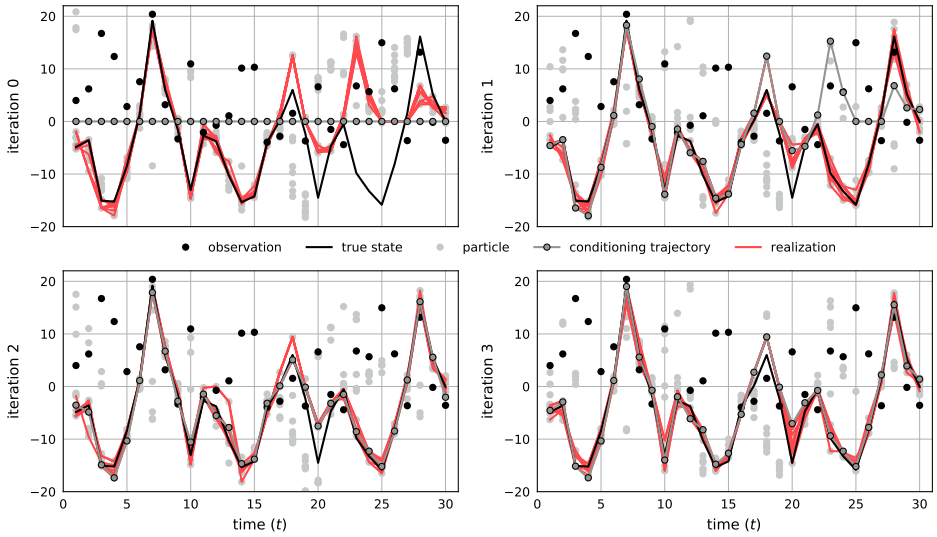

FIGURE 6. Four iterations of the CPFBS smoother (**Algorithm 3**). The state (black line) and the observations (black points) have been simulated using the Kitagawa model (19) with $Q = 1$ and $R = 10$. At the first iteration, the conditioning trajectory (grey dotted line) is initialized with the constant sequence equal to 0. CPFBS is run with $N_f = 10$ particles (grey points) and $N_s = 10$ trajectories (red curves).

permits to simulate samples distributed approximately according to the smoothing distribution after a sufficient number of iterations, whatever the values of N_f and N_s. This theorem was first proven for the CPF smoother in [1]. The results were then extended to CPFBS in [35] and to CPFAS in [34] on target to solving inverse problems in non-Markovian models.

Though the CPFBS smoother shares the same good theoretical properties as the CPF and CPFAS, we will show further in Section 3 that it gives better results in practice. This is due to its ability to reduce the degeneracy problem and hence provide better descriptions of the smoothing distribution. At first glance, running BS seems to be more costly than tracking ancestors. Nevertheless, the transition probability in the backward kernel (10) is computed reusing the information of the particles simulated within CPF and thus does not require extra simulations of the state equation. In practice, the computational complexity of the CPFBS algorithm is thus similar to that of the CPF or CPFAS algorithm and grows linearly with N_f.

Recently, the CPFBS with few particles $(5-20)$ has been applied to sample θ and the latent state in a Bayesian framework [33, 35]. In the next section, we propose to combine CPFBS and Stochastic EM algorithm to perform maximum likelihood estimation.

2.2. **Stochastic EM algorithm and parameter estimation.** This section presents the estimation of the unknown parameter θ given a sequence $\mathbf{y}_{1:T}$ of observations of the SSM (1). For a SSM, the likelihood function is given by

$$L(\theta) = p_\theta(\mathbf{y}_{1:T}) = \int p_\theta\left(\mathbf{x}_{0:T}, \mathbf{y}_{1:T}\right) d\mathbf{x}_{0:T}. \tag{11}$$

The EM algorithm originally proposed in [14] is the most classical numerical method in the statistical literature to maximize the likelihood function in models with latent variables. This iterative algorithm maximizes at each step an auxiliary function G which is generally easier to optimize than the likelihood function (11). G is defined as

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

$$\triangleq \int \ln p_\theta\left(\mathbf{x}_{0:T}, \mathbf{y}_{1:T}\right)\ p_{\theta'}\left(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}\right) d\mathbf{x}_{0:T} \tag{13}$$

where θ and θ' denote two possible values for the parameters. Starting from an initial parameter θ_0, each iteration r $(r \geq 1)$ of the EM algorithm comprises 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 this algorithm increases the likelihood function at each iteration and gives a sequence (θ_r) which converges to a maximum likelihood estimate (MLE).

Remark that the intermediate function (12) of the EM algorithm is defined as the expectation of the full likelihood function

$$p_\theta(\mathbf{x}_{0:T}, \mathbf{y}_{1:T}) = p_\theta\left(\mathbf{x}_0\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right) \prod_{t=1}^{T} p_\theta\left(\mathbf{y}_t|\mathbf{x}_t\right) \tag{14}$$

with respect to the smoothing distribution $p_{\theta'}\left(\mathbf{x}_{0:T}|\mathbf{y}_{1:T}\right)$. The EM algorithm combined with Kalman smoothing (KS-EM, [43]) has been the dominant approach to estimate parameters in linear Gaussian models. In nonlinear and/or non-Gaussian models, the smoothing distribution and thus the intermediate function of the EM algorithm are generally intractable. The EM algorithm needs to be adapted in such situation. In [10, 11, 46], the authors proposed to use as alternative a Monte Carlo

approximation of (12),

$$\widehat{G}(\theta,\theta') \triangleq \frac{1}{N_s}\sum_{j=1}^{N_s} \ln p_\theta\left(\mathbf{x}_{0:T}^j, \mathbf{y}_{1:T}\right), \tag{15}$$

where $\{\mathbf{x}_{0:T}^j\}^{j=1:N_s}$ are N_s trajectories simulated according to the smoothing distribution $p_{\theta'}(\mathbf{x}_{0:T}|\mathbf{y}_{1:T})$. This algorithm is referred to as Stochastic EM (SEM) algorithm in the literature.

SEM requires generating samples of the smoothing distribution at each iteration. In the literature, standard particle smoothing methods are generally used (see [26]). However, the computational cost of these algorithms can be prohibitive for practical applications. A possible alternative is to use an approximate smoother such as EnKS leading to the EnKS-EM algorithm originally proposed in [20]. It was found using numerical simulations that this algorithm performs well as long as the non-linearities are not too important. Nevertheless, the EnKS-EM may lead to bad estimations or divergence problems in case of strong non-linearities (see numerical experiments in Section 3). Hereafter, we explore alternatives based on the smoothers introduced in Section 2.1.

In [32], it is proposed to combine SEM and CPFAS leading to the CPFAS-SEM algorithm described below. Starting from an initial parameter value $\widehat{\theta}_0$ and a conditioning trajectory $\mathbf{X}_0^*$, each iteration r in the CPFAS-SEM algorithm consists of

- **E-step**:
  - i. Draw N_s realizations using the CPFAS smoother with parameter $\widehat{\theta}_{r-1}$, conditioning sequence $\mathbf{X}_{r-1}^*$, and observations $\mathbf{y}_{1:T}$. $\mathbf{X}_r^*$ denotes the new conditioning trajectory obtained after updating.
  - ii. Compute the quantity $\widehat{G}(\theta,\widehat{\theta}_{r-1})$ via (14) and (15).
- **M-step**: Compute $\widehat{\theta}_r = \arg\max_\theta \widehat{G}(\theta,\widehat{\theta}_{r-1})$.

Note that the CPFAS-SEM algorithm is slightly different from a regular SEM algorithm because at iteration r, N_s smoothing trajectories are sampled given the previous conditioning trajectory $\mathbf{X}_{r-1}^*$. This creates some (stochastic) dependence between successive iterations in the algorithm.

Numerical illustrations shown in Figure 5 and results given in Section 3 indicate that the degeneracy issue in the CPFAS algorithm may lead to estimates with some bias and large variances. As discussed in the preceding section, the CPFBS algorithm is expected to be less prone to degeneracy and to provide a better description of the smoothing distribution at the same computational cost than the CPFAS algorithm. We thus propose to replace CPFAS by CPFBS in the CPFAS-SEM algorithm leading to the CPFBS-SEM algorithm.

Depending on the complexity of the SSM (1), analytical or numerical procedures may be used to maximize $\widehat{G}$ in the **M-step**. Usual practical applications are based on Gaussian SSM defined as

$$\begin{cases} \mathbf{x}_t = m(\mathbf{x}_{t-1}) + \eta_t, \\ \mathbf{y}_t = h(\mathbf{x}_t) + \epsilon_t. \end{cases} \tag{16}$$

where m and h can be linear or nonlinear functions, $\eta_t \sim \mathcal{N}(\mathbf{0},\mathbf{Q})$, and $\epsilon_t \sim \mathcal{N}(\mathbf{0},\mathbf{R})$. In this particular case, the following analytical expressions can be derived

for updating $\mathbf{Q}$ and $\mathbf{R}$ in the **M-step**.

$$\widehat{\mathbf{Q}}_r = \frac{1}{TN_s} \sum_{t=1}^{T} \sum_{j=1}^{N_s} \left[\mathbf{x}_t^j - m\left(\mathbf{x}_{t-1}^j\right)\right] \left[\mathbf{x}_t^j - m\left(\mathbf{x}_{t-1}^j\right)\right]',$$

$$\widehat{\mathbf{R}}_r = \frac{1}{TN_s} \sum_{t=1}^{T} \sum_{j=1}^{N_s} \left[\mathbf{y}_t - h\left(\mathbf{x}_t^j\right)\right] \left[\mathbf{y}_t - h\left(\mathbf{x}_t^j\right)\right]'. \tag{17}$$

The number N_s of simulated trajectories is a key parameter in the CPFBS-SEM algorithm. If N_s is large, the law of large numbers implies that $\widehat{G}$ is a good approximation of G and the SEM algorithm is close to the EM algorithm. However, a large N_s means a high computational cost. Different strategies have been proposed in the SEM literature to get a good trade-off between the quality of the estimation and the computational time. For instance, it is possible to increase the value of N_s at each iteration of the EM (Monte Carlo EM algorithm, MCEM, see [10]) or to re-use the smoothing trajectories simulated in the previous iterations (stochastic approximation EM algorithm, SAEM, see [13, 29]) to decrease the variance of the estimates. In this article, we focus on the usual SEM to simplify the presentation.

3. Numerical results.

The performance of the CPFBS-EM algorithm is assessed using simulations and compared with other algorithms including CPFAS-SEM, PFBS-SEM, and EnKS-EM. Simulations are performed using three different toys models. We first focus on a simple univariate linear Gaussian model. For this model, the KS-EM algorithm, which can provide an exact numerical approximation to the MLE, is run to check the accuracy of the estimates derived from the SEM algorithms. Then, we consider nonlinear models, starting with the univariate Kitagawa model, before discussing results obtained with the three-dimensional Lorenz-63 model.

3.1. Linear model.

The linear Gaussian SSM is one classical toy model in the literature since the Kalman recursions give exact expressions for the filtering and smoothing expressions and the **M-step** of the EM algorithm can also be solved exactly. For this model, the KS-EM algorithm [43] can thus be implemented to compute an accurate numerical approximation of the MLE at a low computational cost and be used as a reference for other estimation algorithms. Implementations of stochastic versions of the EM algorithm for linear Gaussian SSMs are discussed for example in [26, 32, 41].

Consider a linear SSM defined as

$$\begin{cases} x_t = Ax_{t-1} + \eta_t, \\ y_t = x_t + \epsilon_t, \end{cases} \tag{18}$$

where $\{x_t\}$ and $\{y_t\}$ have values in $\mathbb{R}$, η_t and ϵ_t are independent Gaussian white noise sequences with variances Q and R, and A is the autoregressive coefficient. $\theta = (A, Q, R)$ denotes the vector of unknown parameters. The true parameter value is fixed to $\theta^* = (0.9, 1, 1)$ and the length of simulated sequences to $T = 100$. An example of simulated trajectory is shown in Figure 7.

The initial parameter $\widehat{\theta}_0$ for all the algorithms is sampled using a uniform distribution on the interval $[0.5,\ 1.5]^3$. The KS-EM is run with 1000 iterations and is expected to obtain an exact numerical approximation to the MLE. In all the experiments below, the initial conditioning trajectories $\mathbf{X}_0^*$ of the CPFBS-SEM and

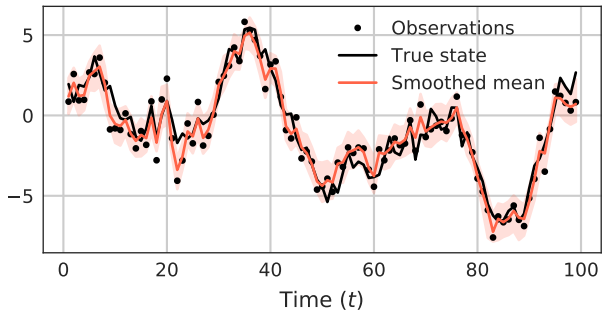

FIGURE 7. Sequence simulated with the linear Gaussian SSM model (18) with $\theta^* = (0.9, 1, 1)$. The mean of the smoothing distribution (red curve) and 95% prediction interval (light red area) are computed based on the smoothing trajectories simulated in the last 10 iterations of CPFBS-SEM algorithm with $N_f = N_s = 10$ particles.

CPFAS-SEM algorithms are simulated by running standard PF-based smoothing algorithms given $\widehat{\theta}_0$ and a sequence of observations. In order to estimate the distribution of the estimators, each experiment is run 100 times. Besides the quality of the estimate of θ, we also compare the algorithms through their abilities to reconstruct the latent state $x_{1:T}$ from the observations $y_{1:T}$. In practice, all the smoothing trajectories obtained in the the last 10 iterations of the SEM algorithms are used to compute an empirical mean for the smoothing distribution and a 95% confidence interval (CI) for the latent state as illustrated on Figure 7. Finally, the reconstruction ability of the algorithms is measured using the root mean square error (RMSE) between the estimated mean of the smoothing distribution and the true latent state.

Figure 8 shows the evolution of the distributions of the estimators of θ and the RMSEs between the true state and the mean of the smoothing distributions as a function of the number of EM iterations. The estimates given by the KS-EM algorithm are shown in Figure 8 (dotted line). Although the CPFBS-EM is run with a low number $N_f = N_s = 10$ of particles, it provides estimates and reconstruction errors with similar distributions than KS-EM. It is also noticeable that the algorithms begin to stabilize after only 10 iterations. As expected from the discussion in Section 2.1.3, CPFBS-SEM estimates and reconstruction errors have a smaller variance than those of CPFAS-SEM.

In the next experiment, the performances of the CPFBS-SEM, CPFAS-SEM, PFBS-SEM, and EnKS-EM algorithms are compared for different numbers of particles $N_f = N_s \in \{10, 50, 100\}$. The empirical distribution of the estimators at iteration 100 are shown on Figure 9. When only $N_f = N_s = 10$ particles are used, the CPFBS-SEM algorithm clearly outperforms all the other algorithms based on Monte-Carlo simulations and gives similar results than KS-EM. The PFBS-SEM algorithm with $N_f = N_s = 10$ or even $N_f = N_s = 50$ particles leads to estimates with a bias which is significantly larger than the ones of other algorithms. It illustrates that the PFBS-SEM algorithm, based on the usual particle filter, needs much more particles than the ones based on CPF (see also in [32]). With $N_f = 100$ particles,

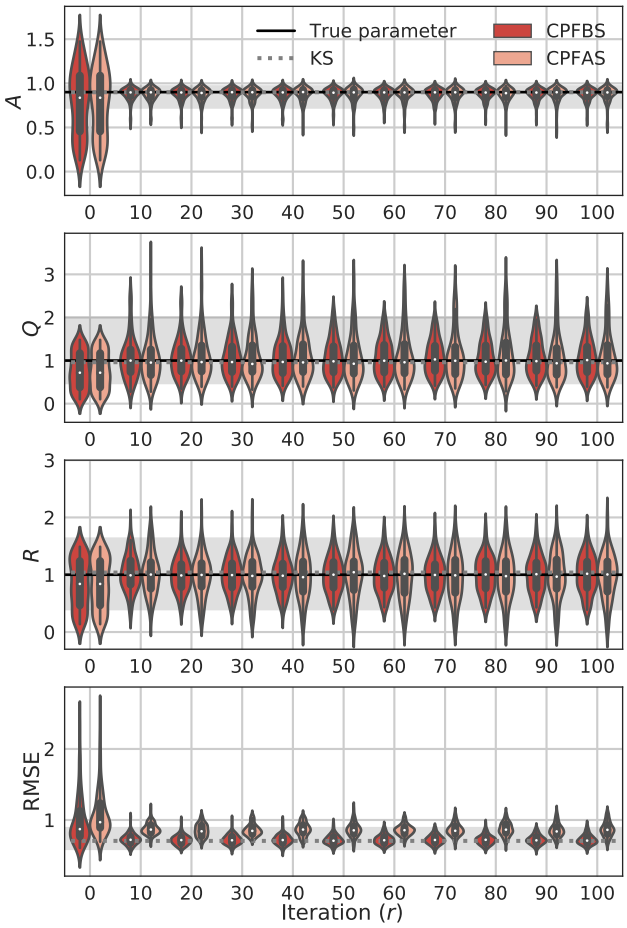

FIGURE 8. Distribution of the estimates obtained with CPFBS-SEM and CPFAS-SEM algorithms as a function of the number of EM iterations for the linear Gaussian SSM model (18) with $\theta^* = (0.9, 1, 1)$, $T = 100$, $N_f = N_s = 10$. The empirical distributions are computed using 100 simulated samples. The median (grey dotted line) and 95% confidence interval (grey shaded area) are computed using 10^3 iterations of the KS-EM algorithm.

the effect of conditioning becomes less important and the PFBS-SEM and CPFBS-SEM give similar results. With $N_f = 10$ members, the EnKS-EM algorithm leads to biased estimates for Q and R but with $N_f = 50$ or $N_f = 100$ members it exhibits similar good performances than CPFBS-SEM and KS-EM algorithms. Finally, the CPFAS-SEM algorithm leads to estimates with a larger variance compared to the other algorithms, and even with $N_f = N_s = 100$ its performance is not as good than the one of KS-EM or CPFBS-SEM with $N_f = N_s = 10$.

3.2. **Kitagawa model.** The SEM algorithms are now applied on the univariate Kitagawa SSM defined as

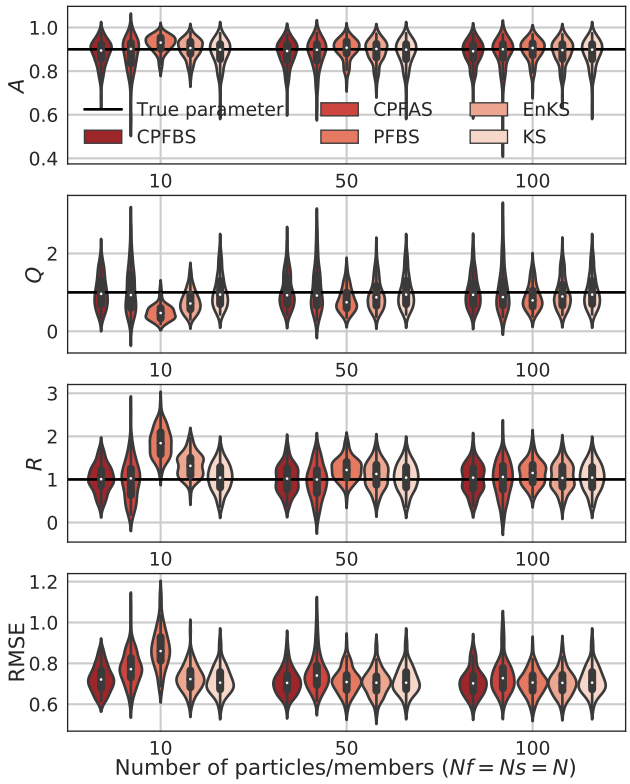

FIGURE 9. Distribution of the estimates obtained with CPFBS-SEM, CPFAS-SEM, PFBS-EM, and EnKS-EM algorithms as a function of the number of particles for the linear Gaussian SSM model (18) with $\theta^* = (0.9, 1, 1)$ and $T = 100$. Results obtained by running 100 iterations of the algorithms. The empirical distributions are computed using 100 simulated samples.

$$\begin{cases} x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + \ \eta_t, \\ y_t = 0.05x_t^2 + \ \epsilon_t, \end{cases} \tag{19}$$

where $\{x_t\}$ and $\{y_t\}$ have values in $\mathbb{R}$, η_t and ϵ_t are independent Gaussian white noise sequences with variances Q and R. $\theta = (Q, R)$ denotes the unknown parameter with true parameter value fixed to $\theta^* = (1, 10)$ in the experiments below, unless stated otherwise. Figure 10 shows a sequence simulated with this model. It was first proposed in [40] and then widely considered in the literature as a toy model to perform numerical experiments (see e.g., [18, 24, 27, 28, 42, 32]). It is a challenging univariate SSM since both the dynamical and observation models are non-linear.

The SEM algorithms were initialized using values of θ simulated according to the uniform distribution in $[1, 10]^2$. In this section, we only compare results obtained with the CPFBS-SEM and CPFAS-SEM algorithms run with $N_f = N_s = 10$ particles and 100 iterations. As shown in the linear case, these algorithms outperform the

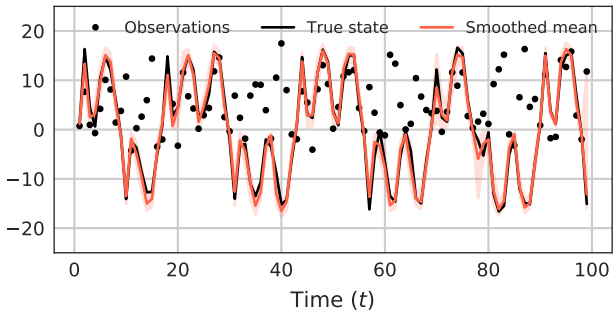

FIGURE 10. Sequence simulated with the Kitagawa model (19) with $\theta^* = (1, 10)$. The mean of the smoothing distribution (red curve) and 95% prediction interval (light red area) are computed based on the smoothing trajectories simulated in the last 10 iterations of CPFBS-SEM algorithm with $N_f = N_s = 10$ particles.

PFBS-SEM with such a small number of particles. We also found that the EnKS-EM algorithm provides estimates with very large bias and thus their corresponding numerical results are not reported.

Figure 11 shows the distribution of the estimates obtained with CPFBS-SEM and CPFAS-SEM algorithms as a function of the number of SEM iterations. Both methods stabilize rapidly, after about 30 iterations, and provide estimates with a low bias. Again the estimates obtained with CPFBS-SEM have a lower variance compared to CPFAS-SEM.

Figure 12 shows the distribution of the estimates obtained with CPFBS-SEM and CPFAS-SEM algorithms as a function of the observation error variance R. As expected, the reconstruction error increases with the observation error. Both methods give similar results when the observation error is small, but the CPFBS-SEM algorithm provides estimates with a lower variance when R increases.

3.3. **Lorenz-63 model.** In this section, we consider the SSM defined as

$$\begin{cases} \mathbf{x}_t = m(\mathbf{x}_{t-1}) + \eta_t, \\ \mathbf{y}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_t + \epsilon_t, \end{cases} \tag{20}$$

where $\mathbf{x}_t$ and $\mathbf{y}_t$ have values respectively in $\mathbb{R}^3$ and $\mathbb{R}^2$ (only the first and third components of the state are observed), $\eta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$, and $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$.

The dynamical model m is defined as the solution on the time interval $[0, \triangle]$ of the Lorenz-63 model [38],

$$\begin{cases} \mathbf{z}(0) = \mathbf{x} \\ \frac{d\mathbf{z}(\tau)}{d\tau} = g(\mathbf{z}(\tau)), \quad \tau \in [0, \triangle], \\ m(\mathbf{x}) = \mathbf{z}(\triangle) \end{cases} \tag{21}$$

for $\mathbf{x} \in \mathbb{R}^3$ and $g(\mathbf{z}) = (10(z_2 - z_1),\ z_1(28 - z_3) - z_2,\ z_1 z_2 - 8/3 z_3), \quad \forall \mathbf{z} = (z_1, z_2, z_3)^\top \in \mathbb{R}^3$.

In order to compute $m(\mathbf{x}_{t-1})$, a Runge-Kutta scheme (order 5) is used to integrate (21) on the time interval $[0, \triangle]$ with initial condition $\mathbf{x}_{t-1}$. The value of $\triangle$ affects the

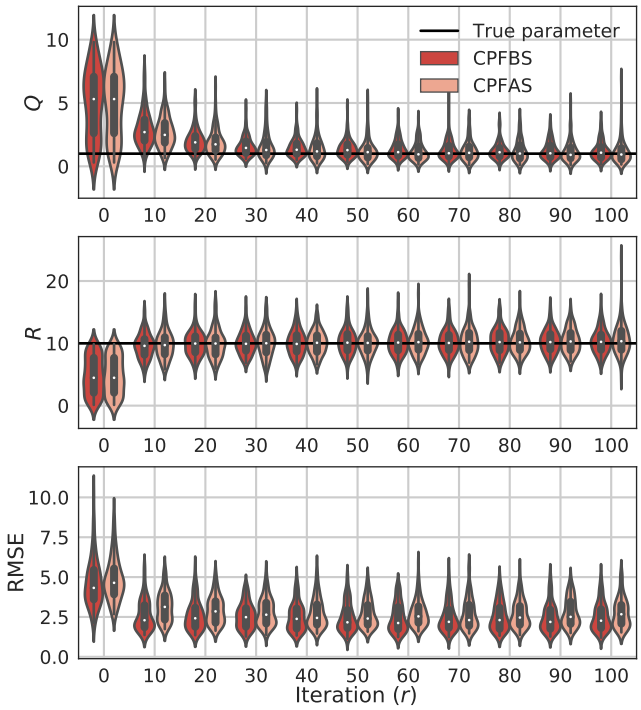

FIGURE 11. Distribution of the estimates obtained with CPFBS-SEM and CPFAS-SEM algorithms as a function of the number of SEM iterations for the Kitagawa model (19) with $\theta^* = (1, 10)$, $T = 100$, $N_f = N_s = 10$. The empirical distributions are computed using 100 simulated samples.

non-linearity of the dynamical model m. According to Figure 15 (see top panels), when $\triangle = 0.01$ the relation between $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ is well approximated by a linear model, but when $\triangle = 0.15$ the non-linearity is more pronounced. The intermediate value $\triangle = 0.08$ corresponds to 6-hour recorded data in atmospheric applications [20, 31].

For the sake of simplifying illustrations, error covariance matrices are assumed to be diagonal. More precisely, we denote $\mathbf{Q} = \sigma^2_{\mathbf{Q}}\mathbf{I}_3$ and $\mathbf{R} = \sigma^2_{\mathbf{R}}\mathbf{I}_2$ and the unknown parameter to be estimated is $\theta = (\sigma^2_{\mathbf{Q}}, \sigma^2_{\mathbf{R}}) \in \mathbb{R}^2$. Analytical expression can be derived for the **M-step** of the SEM algorithm:

$$\widehat{\theta}_r = (\widehat{\sigma}^2_{\mathbf{Q},r}, \widehat{\sigma}^2_{\mathbf{R},r}) = \left(\frac{\mathrm{Tr}[\widehat{\mathbf{Q}}_r]}{3}, \frac{\mathrm{Tr}[\widehat{\mathbf{R}}_r]}{2} \right) \tag{22}$$

where $\widehat{\mathbf{Q}}_r$ and $\widehat{\mathbf{R}}_r$ are defined in (17).

Unless stated otherwise, the model time step is $\triangle = 0.15$ and the true parameter value is $\theta^* = (0.01, 2)$. Figure 13 shows a sequence simulated with this model together with the reconstruction of the state obtained after running 100 iterations of the CPFBS-EM algorithm. Remark that the algorithm seems to be able to reconstruct the second component of the state which is not observed but which is

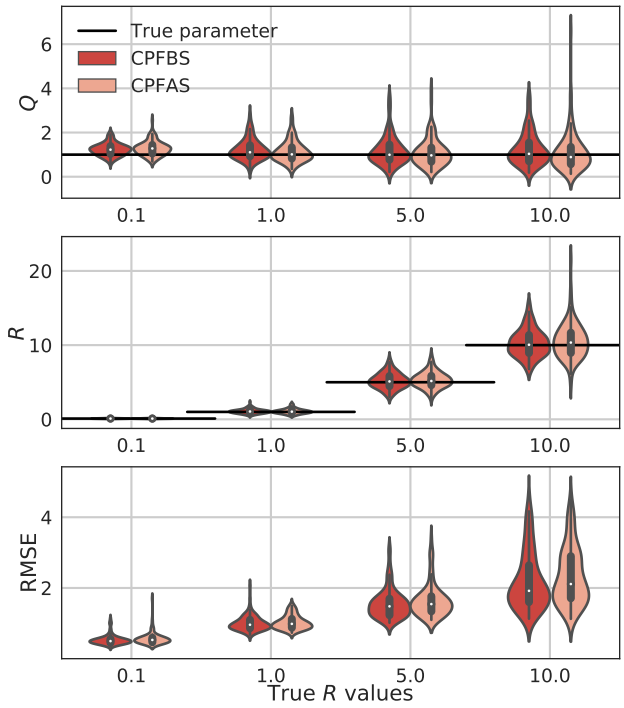

FIGURE 12. Distribution of the estimates obtained with CPFBS-SEM and CPFAS-SEM algorithms for the Kitagawa model (19) with $\theta^* = (1, R^*)$, $R^* \in \{0.1, 1, 5, 10\}$, $T = 100$, $N_f = N_s = 10$. Results obtained by running 100 iterations of the SEM algorithms. The empirical distributions are computed using 100 simulated samples.

strongly related to the other components. In the experiments below, the reconstruction error (RMSE) is computed over all the components of the state, including the non-observed one.

When running the SEM algorithms, the initial parameter values of θ are drawn uniformly in $[0.001, 1] \times [0.1, 3]$. Figure 14 shows the distribution of the estimates obtained with CPFBS-SEM and CPFAS-SEM algorithms as a function of the number of SEM iterations. Compared to the results obtained with the univariate models, more iterations are needed before the algorithms stabilize. After 100 iterations, both algorithms provide estimates with a low bias and again the CPFBS-SEM is better in terms of variance and reconstruction error.

Figure 15 shows the performances of the CPFBS-SEM, CPFAS-SEM, and EnKS-EM algorithms as a function of the time step $\triangle$ and thus of the non-linearities presented in the dynamical model. For the model with small non-linearity ($\triangle = 0.01$), the EnKS-EM algorithm performs better. This may be due to the ability of the EnKS to describe the smoothing distribution with a low number of members in such setting. When the non-linearities increase, the EnKS does not provide relevant approximations of the smoothing distribution and the EnKS-EM provides estimates with a large bias and variance compared to the SEM algorithms based on

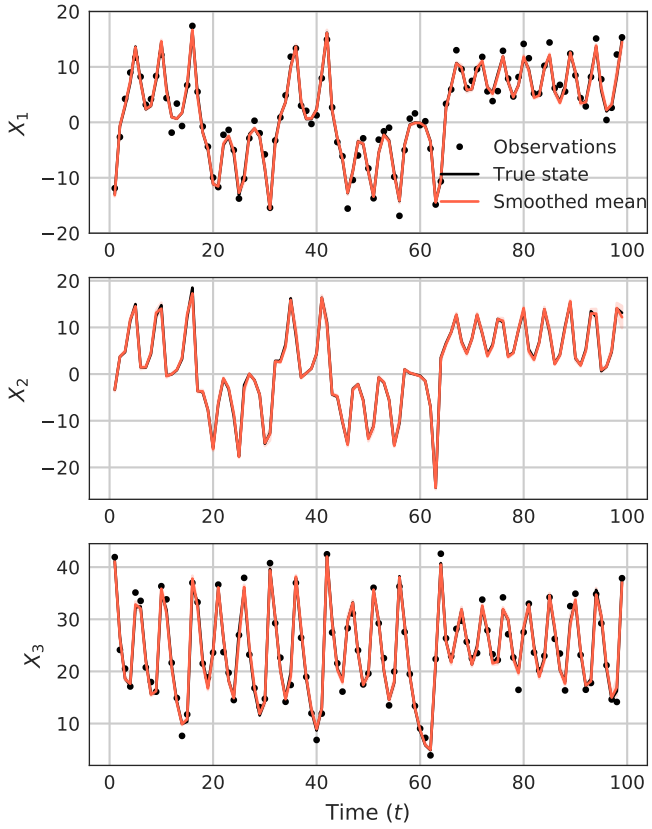

FIGURE 13. Sequence simulated with the Lorenz-63 model (20) with $\theta^* = (0.01, 2)$ and time step $\triangle = 0.15$. The mean of the smoothing distribution (red curve) and 95% prediction interval (light red area) are computed based on the smoothing trajectories simulated in the last 10 iterations of CPFBS-SEM algorithm with $N_f = N_s = 10$ particles.

CPF. Again, the CPFBS-SEM algorithm leads to estimates with a lower variance compared to CPFAS-SEM for all $\triangle$. It has been found that with $\triangle = 0.25$, the CPF-based algorithms still give reasonable estimates whereas the EnKS-EM algorithm completely diverges (not shown; a Python library is provided for repeating the experiments).

Finally, a cross-validation exercise is conducted to check the out-of-sample reconstruction ability of the algorithms. The CPFBS-SEM and the CPFAS-SEM algorithms are first run to estimate the parameter θ on a learning sequence. The estimate obtained in the last iteration, which distribution is shown in Figure 14, is then used as input of the CPFBS and CPFAS smoothing algorithms on a validation sequence to estimate the smoothing distribution. RMSE and coverage probability (CP) are reported in Table 1 as a function of the number of iterations of these smoothing algorithms. As expected RMSEs decrease and CPs tend to 95% when the number of iterations and thus the number of trajectories simulated with the

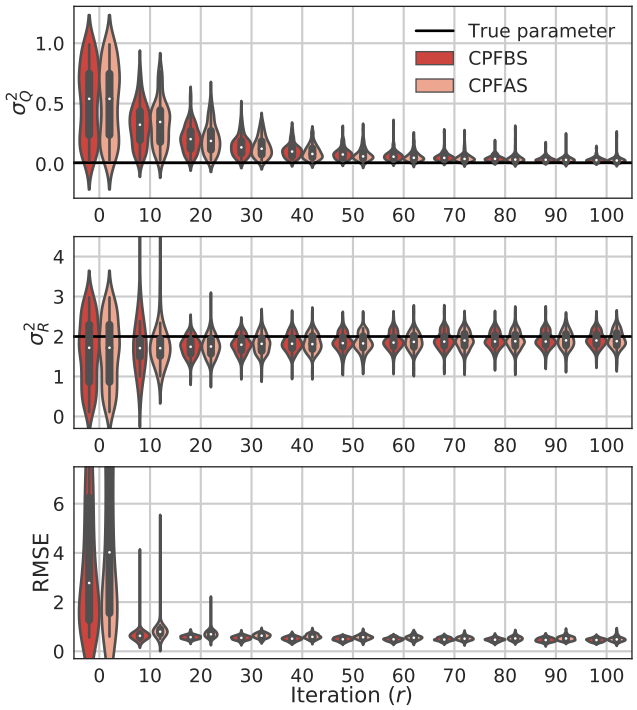

FIGURE 14. Distribution of the estimates obtained with CPFBS-SEM and CPFAS-SEM algorithms as a function of the number of EM iterations for the Lorenz-63 models (20) with $\theta^* = (0.01, 2)$, $\triangle = 0.15$ $T = 100$, $N_f = N_s = 20$. The empirical distributions are computed using 100 simulated samples.

smoothing algorithms increase. The scores computed over the second (unobserved) variable of the Lorenz model are generally close to those computed over the other components. This confirms the ability of the proposed approach to reconstruct non-observed component. The CPFBS smoother clearly outperforms the CPFAS as it leads to smaller RMSEs and CPs closer to 95%. For example, the CPFBS smoother run with 50 iterations provides similar results than the CPFAS run with 100 iterations. This is another benefit of using the BS approach to reduce the degeneracy issue.

4. **Conclusion.** The numerical results obtained in this study promote the combination of CPF, BS, and SEM algorithms to estimate both the unknown parameters and the state in non-linear state-space models. The use of CPF permits to better handle non-linearities than the EnKS and the use of BS permits to avoid degeneracy issues and provide a better description of the smoothing distribution compared to AS. It is shown that running 100 iterations of the CPFBS-SEM algorithm with a low number of particles $(10 - 20)$ is generally sufficient to obtain good estimates in low-dimensional $(d \leq 3)$ state-space models.

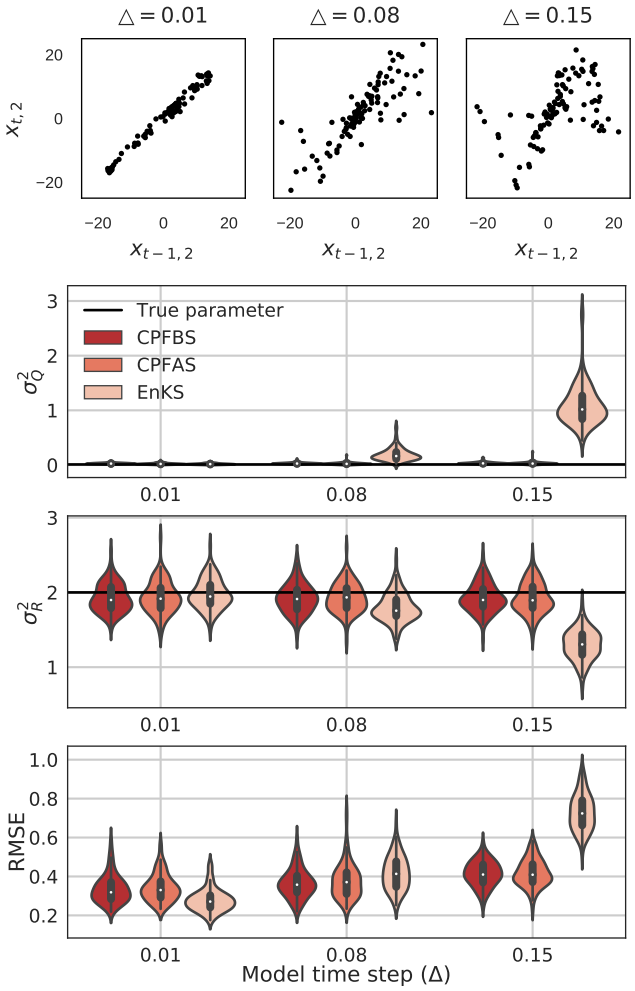

FIGURE 15. Distribution of the estimates obtained with CPFBS-SEM, CPFAS-SEM, and EnKS-EM algorithms as a function of the time step $\triangle$ for the Lorenz-63 models (20) with $\theta^* = (0.01, 2)$, $T = 100$, $N_f = N_s = 20$ and 20 members for the EnKS algorithm. The empirical distributions are computed using 100 simulated samples.

In the future, we plan to test the methodology on mid- to large-dimensional nonlinear models, starting with models with a low number of parameters. Future works also include the study of the convergence properties of the proposed algorithm.

Appendix

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

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

TABLE 1. Comparison of the reconstruction ability of the CPFBS and CPFAS smoothers using cross-validation on the Lorenz-63 model (20) with $\triangle = 0.15, \theta^* = (0.01, 2)$. The parameter θ is estimated on learning sequences of length $T = 100$. Given these estimates, the CPFBS and CPFAS algorithms are run on validation sequences of length $T' = 100$. The two scores are computed on only the second component (top) and over all the three components (bottom). Algorithms run with $N_f = N_s = 20$ particles/realizations. The median and 95% CI of each score are evaluated based on 100 simulated sequences.

2^{nd} component		Number of iterations			
		10	20	50	100
CPFBS	RMSE	0.4328 [0.3011, 0.7473]	0.3928 [0.2771, 0.6258]	0.3772 [0.2609, 0.5752]	0.3704 [0.2438, 0.5737]
	CP	89% [72%, 97%]	93% [78%, 99%]	96% [83%, 100%]	97% [87%, 100%]
CPFAS	RMSE	0.4351 [0.2927, 2.2515]	0.4146 [0.2532, 1.216]	0.3993 [0.2433, 0.7047]	0.3798 [0.2315, 0.7068]
	CP	73% [53%, 85%]	85% [69%, 95%]	92% [82%, 99%]	95% [86%, 100%]

Three components		Number of iterations			
		10	20	50	100
CPFBS	RMSE	0.4351 [0.2983, 0.7969]	0.3990 [0.2771, 0.6277]	0.3803 [0.2761, 0.5251]	0.3722 [0.2758, 0.5053]
	CP	89.33% [72.42%, 96.96%]	92.67% [77.71%, 98.88%]	95.67% [83.38%, 99.33%]	96.83% [88.71%, 99.67%]
CPFAS	RMSE	0.4354 [0.3199, 2.0301]	0.4172 [0.3022, 1.1063]	0.3912 [0.2611, 0.5682]	0.3813 [0.2448, 0.5665]
	CP	71.67% [54.17%, 84.5%]	85.17% [71.17%, 94.63%]	92.5% [82.08%, 98.29%]	95.0% [86.42%, 99.29%]

$$p_\theta(\mathbf{A}|\mathbf{y}_{1:T}) = \int \mathcal{K}_\theta(\mathbf{X}^*, \mathbf{A})\; p_\theta(\mathbf{X}^*|\mathbf{y}_{1:T})\; d\mathbf{X}^* \tag{23}$$

where $\mathcal{K}_\theta(\mathbf{X}^*, \mathbf{A}) = \mathbb{E}_{\theta, \mathbf{X}^*}\left[\mathbb{1}_{\mathbf{A}}(\mathbf{x}_{0:T}^{\mathbf{J}_{0:T}})\right]$, *and* $\mathbf{x}_{0:T}^{\mathbf{J}_{0:T}} = \{\mathbf{x}_0^{(J_0)}, \cdots, \mathbf{x}_T^{(J_T)}\}$.

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

$$\|\mathcal{K}_\theta^r(\mathbf{X}^*, \cdot) - p_\theta(\cdot|\mathbf{y}_{1:T})\|_{TV} \overset{r\to\infty}{\longrightarrow}{}^{\text{as}}\; 0. \tag{24}$$

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

Proof. Theorem A in this paper was proved corresponding to Theorem 5 in [1] for CPF (**Algorithm 2**), Propositions 1 and 2 in [35] for CPFBS (**Algorithm 3**), and Theorems 1 and 2 in [34] for CPFAS. □

REFERENCES

[1] C. Andrieu, A. Doucet and R. Holenstein, Particle markov chain monte Carlo methods, *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **72** (2010), 269–342.

[2] M. Aoki, *State Space Modeling of Time Series*, Springer-Verlag, Berlin, 1987.

[3] D. Barber, A. T. Cemgil and S. Chiappa, *Bayesian Time Series Models*, Cambridge University Press, 2011.

[4] T. Berry and T. Sauer, Adaptive ensemble Kalman filtering of non-linear systems, *Tellus A: Dynamic Meteorology and Oceanography*, **65** (2013), 20331.

[5] M. Bocquet and P. Sakov, Joint state and parameter estimation with an iterative ensemble Kalman smoother, *Nonlin. Processes Geophys.*, **20** (2013), 803–818.

[6] M. Bocquet and P. Sakov, Combining inflation-free and iterative ensemble Kalman filters for strongly nonlinear systems, *Nonlinear Processes in Geophysics*, **19** (2012), 383–399.
[7] M. Bocquet and P. Sakov, An iterative ensemble Kalman smoother, *Quarterly Journal of the Royal Meteorological Society*, **140** (2014), 1521–1535.
[8] O. Cappé, S. J. Godsill and E. Moulines, An overview of existing methods and recent advances in sequential monte carlo, *Proceedings of the IEEE*, **95** (2007), 899–924.
[9] A. Carrassi, M. Bocquet, L. Bertino and G. Evensen, Data assimilation in the geosciences: An overview of methods, issues, and perspectives, *Wiley Interdisciplinary Reviews: Climate Change*, **9** (2018), e535.
[10] G. Celeux, D. Chauveau and J. Diebolt, *On Stochastic Versions of the EM Algorithm*, Research Report RR-2514, INRIA, 1995.
[11] K. S. Chan and J. Ledolter, Monte Carlo EM estimation for time series models involving counts, *J. Amer. Statist. Assoc.*, **90** (1995), 242–252.
[12] N. Chopin, S. S. Singh, On particle gibbs sampling, *Bernoulli*, **21** (2015), 1855–1883.
[13] B. Delyon, M. Lavielle and E. Moulines, Convergence of a stochastic approximation version of the em algorithm, *Ann. Statist.*, **27** (1999), 94–128.
[14] A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *J. Roy. Statist. Soc. Ser. B*, **39** (1977), 1–38.
[15] R. Douc and O. Cappé, Comparison of resampling schemes for particle filtering, in *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, IEEE, 2005, 64–69.
[16] R. Douc, A. Garivier, E. Moulines and J. Olsson, On the forward filtering backward smoothing particle approximations of the smoothing distribution in general state spaces models, arXiv preprint, arXiv:0904.0316.
[17] A. Doucet, N. de Freitas and N. Gordon (eds.), *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, Springer-Verlag, New York, 2001.
[18] A. Doucet, S. Godsill and C. Andrieu, *On Sequential Monte Carlo Sampling Methods for Bayesian Filtering*, 1998.
[19] A. Doucet and A. M. Johansen, A tutorial on particle filtering and smoothing: Fifteen years later, *The Oxford Handbook of Nonlinear filtering*, 656–704, Oxford Univ. Press, Oxford, 2011.
[20] D. Dreano, P. Tandeo, M. Pulido, B. Ait-El-Fquih, T. Chonavel and I. Hoteit, 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** (2017), 1877–1885.
[21] J. Durbin and S. J. Koopman, *Time Series Analysis by State Space Methods*, Oxford university press, 2012.
[22] G. Evensen, The ensemble Kalman filter: Theoretical formulation and practical implementation, *Ocean Dynamics*, **53** (2003), 343–367.
[23] G. Evensen and P. J. van Leeuwen, An ensemble Kalman smoother for nonlinear dynamics, *Monthly Weather Review*, **128** (2000), 1852–1867.
[24] S. J. Godsill, A. Doucet and M. West, Monte Carlo smoothing for nonlinear time series, *J. Amer. Statist. Assoc.*, **99** (2004), 156–168.
[25] J. D. Hol, T. B. Schon and F. Gustafsson, On resampling algorithms for particle filters, in *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, IEEE, 2006, 79–82.
[26] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, N. Chopin, On particle methods for parameter estimation in state-space models, *Statist. Sci.*, **30** (2015), 328–351.
[27] G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, *J. Comput. Graph. Statist.*, **5** (1996), 1–25.
[28] J. Kokkala, A. Solin and S. Särkkä, Expectation maximization based parameter estimation by sigma-point and particle smoothing, in *FUSION*, IEEE, 2014, 1–8.
[29] E. Kuhn and M. Lavielle, Coupling a stochastic approximation version of EM with an MCMC procedure, *ESAIM Probab. Stat.*, **8** (2004), 115–131.
[30] F. Le Gland, V. Monbet and V.-D. Tran, Large sample asymptotics for the ensemble kalman filter, in *Handbook on Nonlinear Filtering* (eds. D. Crisan and B. Rozovskii), Oxford University Press, Oxford, 2011, chapter 22, 598–631.
[31] R. Lguensat, P. Tandeo, P. Ailliot, M. Pulido and R. Fablet, The analog data assimilation, *Monthly Weather Review*, **145** (2017), 4093–4107.

[32] F. Lindsten, An efficient stochastic approximation EM algorithm using conditional particle filters, in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, 6274–6278.
[33] F. Lindsten, *Particle Filters and Markov Chains for Learning of Dynamical Systems*, PhD thesis, Linköping University Electronic Press, 2013.
[34] F. Lindsten, M. I. Jordan and T. B. Schön, Particle Gibbs with ancestor sampling, *J. Mach. Learn. Res.*, **15** (2014), 2145–2184.
[35] F. Lindsten and T. B. Schön, On the use of backward simulation in the particle Gibbs sampler, in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2012, 3845–3848.
[36] F. Lindsten, T. B. Schön, Backward simulation methods for monte Carlo statistical inference, *Foundations and Trends® in Machine Learning*, **6** (2013), 1–143.
[37] F. Lindsten, T. Schön and M. I. Jordan, Ancestor sampling for particle Gibbs, in *Advances in Neural Information Processing Systems*, 2012, 2591–2599.
[38] E. N. Lorenz, Deterministic nonperiodic flow, *J. Atmospheric Sci.*, **20** (1963), 130–141.
[39] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, vol. 382, John Wiley & Sons, 2008.
[40] M. Netto, L. Gimeno and M. Mendes, On the optimal and suboptimal nonlinear filtering problem for discrete-time systems, *IEEE Transactions on Automatic Control*, **23** (1978), 1062–1067.
[41] J. Olsson, O. Cappé, R. Douc, E. Moulines, Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models, *Bernoulli*, **14** (2008), 155–179.
[42] T. B. Schön, A. Wills and B. Ninness, System identification of nonlinear state-space models, *Automatica J. IFAC*, **47** (2011), 39–49.
[43] R. H. Shumway and D. S. Stoffer, An approach to time series smoothing and forecasting using the em algorithm, *Journal of Time Series Analysis*, **3** (1982), 253–264.
[44] A. Svensson, T. B. Schön and M. Kok, Nonlinear state space smoothing using the conditional particle filter, *IFAC-PapersOnLine*, **48** (2015), 975–980.
[45] P. Tandeo, P. Ailliot, M. Bocquet, A. Carrassi, T. Miyoshi, M. Pulido and Y. Zhen, A review of innovation-based methods to jointly estimate model and observation error covariance matrices in ensemble data assimilation, *Monthly Weather Review*, **148** (2020), 3973–3994.
[46] G. C. G. Wei and M. A. Tanner, A Monte Carlo implementation of the em algorithm and the poor man's data augmentation algorithms, *Journal of the American statistical Association*, **85** (1990), 699–704.
[47] N. Whiteley, Discussion on particle markov chain monte carlo methods, *Journal of the Royal Statistical Society: Series B*, **72** (2010), 306–307.

Received for publication July 2021; early access March 2022.

E-mail address: trang.chau@lsce.ipsl.fr
E-mail address: pierre.ailliot@univ-brest.fr
E-mail address: valerie.monbet@univ-rennes1.fr
E-mail address: pierre.tandeo@imt-atlantique.fr