Signal Processing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Direct, prediction- and smoothing-based Kalman and particle filter algorithms

François Desbouvries [a,*], Yohan Petetin [a], Boujemaa Ait-El-Fquih [b]

[a] *Telecom Institute/Telecom SudParis/CITI Dpt. & CNRS UMR 5157, 91011 Evry, France*
[b] *IMS-Bordeaux/Équipe Signal et Image & CNRS UMR 5218, 33405 Talence, France*

ARTICLE INFO

Article history:
Received 2 June 2010
Received in revised form
17 December 2010
Accepted 21 March 2011

Keywords:
Kalman filters
Sequential Monte Carlo
Particle filtering
Sequential importance sampling
Sampling importance resampling

ABSTRACT

We address the recursive computation of the filtering probability density function (pdf) $p_{n|n}$ in a hidden Markov chain (HMC) model. We first observe that the classical path $p_{n-1|n-1} \rightarrow p_{n|n-1} \rightarrow p_{n|n}$ is not the only possible one that enables to compute $p_{n|n}$ recursively, and we explore the direct, prediction-based (P-based) and smoothing-based (S-based) recursive loops for computing $p_{n|n}$. We next propose a common methodology for computing these equations in practice. Since each path can be decomposed into an updating step and a propagation step, in the linear Gaussian case these two steps are implemented by Gaussian transforms, and in the general case by elementary simulation techniques. By proceeding this way we routinely obtain in parallel, for each filtering path, one set of Kalman filter (KF) equations and one generic sequential Monte Carlo (SMC) algorithm. Finally we classify in a common framework four KF (two of which are original), which themselves can be associated to four generic SMC algorithms (two of which are original). We finally compare our algorithms via simulations. S-based filters behave better than P-based ones, and within each class of filters better results are obtained when updating precedes propagation.

1. Introduction

Let $\mathbf{x}_n \in \mathbb{R}^m$ and $\mathbf{y}_n \in \mathbb{R}^p$ be respectively a hidden and observed process. Let $p(\mathbf{x}_n|\mathbf{y}_{0:n})$, say, denote the pdf (w.r.t. Lebesgue measure) of $\mathbf{x}_n$ given $\mathbf{y}_{0:n} = \{\mathbf{y}_i\}_{i=0}^{n}$, and $p(d\mathbf{x}) = p(\mathbf{x})\,d\mathbf{x}$ the continuous measure with density $p(\mathbf{x})$. We assume that $\{\mathbf{x}_n, \mathbf{y}_n\}$ is an HMC:

$$p(\mathbf{x}_{0:n},\mathbf{y}_{0:n}) = p(\mathbf{x}_0)\prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{x}_{i-1})\prod_{i=0}^{n} p(\mathbf{y}_i|\mathbf{x}_i). \tag{1}$$

Bayesian filtering consists in computing $p(\mathbf{x}_n|\mathbf{y}_{0:n})$, or at least some approximation of $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$.

Recursive solutions are of particular interest, and indeed $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ can be computed from $p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$ by the well known recursion (here $\mathcal{N}$ stands for numerator):

$$p(\mathbf{x}_n|\mathbf{y}_{0:n}) = \frac{p(\mathbf{y}_n|\mathbf{x}_n)\overbrace{\int p(\mathbf{x}_n|\mathbf{x}_{n-1})p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})\,d\mathbf{x}_{n-1}}^{p(\mathbf{x}_n|\mathbf{y}_{0:n-1})}}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1}) = \int \mathcal{N}\,d\mathbf{x}_n}. \tag{2}$$

Many efforts have thus been devoted to the actual computation of (2). In the linear Gaussian case, (2) can be computed exactly by KF techniques [1]. In the general case however, computing $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ is either difficult or impossible, so many approximate techniques have been developed, see e.g. [2–5]. Among them, particle filtering (PF) methods [5–7] are SMC methods which aim at propagating a discrete approximation of $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$.

In this paper we do not try to further improve the existing PF algorithms based on (2); by contrast, we focus on (2) itself, or indeed explore alternate paths for computing $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ (or, in short, $p_{n|n}$) recursively. Once these paths are

* Corresponding author. Tel.: +33 1 60 76 45 27; fax: +33 1 60 76 44 33.
E-mail address: francois.desbouvries@it-sudparis.eu (F. Desbouvries).

0165-1684/$ - see front matter © 2011 Published by Elsevier B.V.
doi:10.1016/j.sigpro.2011.03.013

derived we develop either KF or generic SMC implementations by mechanically applying appropriate standard tools.

More precisely, (2) describes recursion $p_{n-1|n-1} \to p_{n|n-1} \to p_{n|n}$, so computing $p_{n|n}$ from $p_{0|0}$ via (2) consists in walking along one particular path, made of the succession of a propagation step *P* and an updating step *U*. However, following this path is not the only possible way to compute $p_{n|n}$ recursively. Let us consider only those paths in which one time index is incremented at a time. A first alternative is path $p_{n-1|n-1} \overset{U}{\to} p_{n-1|n} \overset{P}{\to} p_{n|n}$. Both solutions compute $p_{n|n}$ recursively and differ only by the intermediate step which is either the one-step ahead predictive distribution $p_{n|n-1}$, or the one-step backward smoothing distribution $p_{n-1|n}$.

Now, in turn $p_{n|n-1}$ and $p_{n-1|n}$ can be propagated via the two different paths which are obtained when moving one index and next the other. This observation naturally yields six algorithms for computing $p_{n|n}$ recursively; the two paths $p_{n-1|n-1} \to p_{n-1|n} \to p_{n|n}$ and $p_{n-1|n-1} \to p_{n|n-1} \to p_{n|n}$ are "direct", i.e. $p_{n|n}$ is computed as the output of a loop with input $p_{n-1|n-1}$; two other paths are *P*-based, i.e. $p_{n|n}$ is computed (indirectly) from the predictive distribution, but the recursion itself now acts on $p_{n|n-1}$; and two paths are *S*-based, see Fig. 1. Out of these six paths only four are distinct, because the two paths at the boundary direct/*P*-based and direct/*S*-based coincide; for instance, the direct filter in which we first propagate and then update, coincides (up to a shift) with the *P*-based filter in which we first update and then propagate (see Fig. 2).

We next address the practical computation of these four integral equations. We classically consider two cases: the linear Gaussian case, for which propagating densities amounts to propagating their parameters; and the general case, in which we resort to Monte Carlo (MC) approximations. However, the novelty here relies in the systematic parallelization of the derivations. Since each path $p_{i|j} \to p_{i+1|j+1}$ consists of a *P* step and an *U* step (or vice versa, depending on which index is incremented first), we actually need a tool for implementing these two basic operations. In the linear Gaussian case, the two steps are implemented by two elementary transformations among

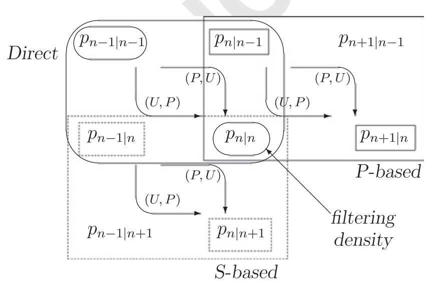

Fig. 1. The direct, *P*-based and *S*-based paths.

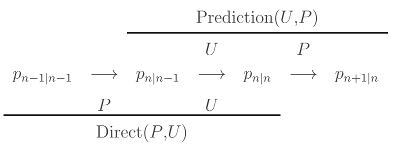

Fig. 2. The (*P*, *U*) direct filter vs. the (*U*, *P*) *P*-based filter.

Gaussian variables. In the general case, the mappings among densities are replaced by the propagation of sets of points (approximately) sampled from these densities; the *P* and *U* steps are then implemented by two elementary simulation techniques. Finally each path provides simultaneously one KF and one generic SMC algorithm. Two of the KF solutions are well known but two others are original; and similarly the SMC algorithms include two well known solutions (the bootstrap algorithm and the fully adapted auxiliary particle filter (APF)) and two original algorithms. This paper is organized as follows. PF is briefly recalled in Section 2. In Section 3 we set up a routine mechanism which will be used for implementing (in the linear Gaussian and general cases) the four distinct filtering paths in Fig. 1. In Section 4 we derive the direct filtering algorithms, address their implementation and discuss the solutions. *P*- and *S*-based algorithms are addressed respectively in Sections 5 and 6. We summarize our results in section Section 7, provide simulation results in Section 8, and finally conclude the paper.

2. The PF methodology

2.1. The generic PF algorithm

Let us briefly recall the principle of PF algorithms [4–7] which are based on importance sampling (IS) techniques. Assume that at time $n-1$ we have a random discrete measure which approximates $p(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) : p(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) \simeq \sum_{i=1}^{N} w_{n-1}^{i}\delta_{\mathbf{x}_{0:n-1}^{i}}(d\mathbf{x}_{0:n-1})$, where $\delta_x(\cdot)$ is the Dirac mass at point *x*, the samples $\mathbf{x}_{0:n-1}^{i}$ are generated from an importance distribution $q(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$, and the importance weight w_{n-1}^{i} associated to the *i*-th trajectory $\mathbf{x}_{0:n-1}^{i}$ is given by $w_{n-1}^{i} \propto p(\mathbf{x}_{0:n-1}^{i}|\mathbf{y}_{0:n-1})/q(\mathbf{x}_{0:n-1}^{i}|\mathbf{y}_{0:n-1})$, $\sum_{i=1}^{N} w_{n-1}^{i} = 1$. At time *n* we would like to get some empirical measure approximating $p(d\mathbf{x}_{0:n}|\mathbf{y}_{0:n})$. Let us start from

$$p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = \frac{p(\mathbf{x}_n,\mathbf{y}_n|\mathbf{x}_{n-1})}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1})}p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}). \tag{3}$$

We first see how to update the trajectories. If we assume that the importance pdf factorizes as

$$q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = q(\mathbf{x}_n|\mathbf{x}_{0:n-1},\mathbf{y}_{0:n})q(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}), \tag{4}$$

i.e. that $q(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$ is a marginal of $q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n})$, then for all $1 \le i \le N$, $\mathbf{x}_{0:n}^{i} = [\mathbf{x}_{0:n-1}^{i},\mathbf{x}_{n}^{i}]$, in which $\mathbf{x}_{n}^{i}$ is sampled from the conditional importance distribution (CID) $q(\mathbf{x}_n|\mathbf{x}_{0:n-1}^{i},\mathbf{y}_{0:n})$. In other words, when stepping from time $n-1$ to time *n*, due to (4) we can keep the old trajectories $\{\mathbf{x}_{0:n-1}^{i}\}_{i=1}^{N}$, and we just need to extend each of them by sampling a new particle $\mathbf{x}_{n}^{i}$.

As for the weights w_{n}^{i}, we see from (3) and (4) that they can be computed recursively as

$$w_{n}^{i} \propto \underbrace{\frac{p(\mathbf{x}_{n}^{i},\mathbf{y}_n|\mathbf{x}_{n-1}^{i})}{q(\mathbf{x}_{n}^{i}|\mathbf{x}_{0:n-1}^{i},\mathbf{y}_{0:n})}}_{\lambda_{n}^{i}} \times \underbrace{\frac{p(\mathbf{x}_{0:n-1}^{i}|\mathbf{y}_{0:n-1})}{q(\mathbf{x}_{0:n-1}^{i}|\mathbf{y}_{0:n-1})}}_{\propto w_{n-1}^{i}}. \tag{5}$$

Finally $\sum_{i=1}^{N} w_{n}^{i}\delta_{\mathbf{x}_{0:n}^{i}}(d\mathbf{x}_{0:n})$ approximates $p(d\mathbf{x}_{0:n}|\mathbf{y}_{0:n})$, and thus $\sum_{i=1}^{N} w_{n}^{i}\delta_{\mathbf{x}_{n}^{i}}(d\mathbf{x}_n)$ is an MC approximation of $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$.

2.2. Practical considerations

Now, PF algorithms are well known to suffer from weights degeneracy. Two main rescues are available. First, it has proved important in this generic algorithm to resample from $\sum_{i=1}^{N} w_n^i \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$. The second important point to take into account is to choose the CID carefully. Two particular choices are of interest:

- Sampling from the *a priori* transition kernel of Markov chain $\mathbf{x}$ (i.e. choosing $q(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i,\mathbf{y}_{0:n}) = p(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i) = p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$) is popular [8,9] because $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ is available, and sampling from $p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$ is often straightforward [4]. Moreover, computing λ_n^i in (5) reduces to evaluating $p(\mathbf{y}_n|\mathbf{x}_n^i)$, which is available from model (1).
- However, this choice of the prior density can lead to poor performances, and it is often preferable [10,11] to sample the particles from the optimal CID

$$q^{opt}(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i,\mathbf{y}_{0:n}) = p(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i,\mathbf{y}_{0:n}) = p(\mathbf{x}_n|\mathbf{x}_{n-1}^i,\mathbf{y}_n), \tag{6}$$

  i.e. the distribution which minimizes the variance of w_n^i, conditionally on the observations $\mathbf{y}_{0:n}$ and past samples $\mathbf{x}_{0:n-1}^i$. For this choice of the CID, $\lambda_n^i = p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)$.

3. A practical toolbox

In the view of forthcoming Sections 4–6 we now provide a simple toolbox for routinely deriving Kalman like or SMC implementations of a given filtering algorithm. Whatever i and k, each path $p_{i|k} \to p_{i+1|k+1}$ in Fig. 1 is made of the succession of a P step $p_{i|k} \to p_{i+1|k}$ and of an U step $p_{i+1|k} \to p_{i+1|k+1}$ (or vice versa). The P step transforms $p(\mathbf{x}_i|\mathbf{y}_{0:k})$, say, into $p(\mathbf{x}_{i+1}|\mathbf{y}_{0:k}) = \int p(\mathbf{x}_{i+1}|\mathbf{x}_i,\mathbf{y}_{0:k}) p(\mathbf{x}_i|\mathbf{y}_{0:k})\, d\mathbf{x}_i$, i.e. if we drop dependence on $\mathbf{y}_{0:k}$, transforms some pdf $p(\mathbf{x}_1)$ into

$$p(\mathbf{x}_1) \overset{P}{\mapsto} p(\mathbf{x}_2) = \int p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_1)\, d\mathbf{x}_1, \tag{7}$$

and the U step transforms $p(\mathbf{x}_{i+1}|\mathbf{y}_{0:k})$, say, into $p(\mathbf{x}_{i+1}|\mathbf{y}_{0:k+1}) \propto p(\mathbf{y}_{k+1}|\mathbf{x}_{i+1},\mathbf{y}_{0:k})\, p(\mathbf{x}_{i+1}|\mathbf{y}_{0:k})$, i.e. if we drop time indices, transforms some pdf $p(\mathbf{x})$ into

$$p(\mathbf{x}) \overset{U}{\mapsto} p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\, d\mathbf{x}}. \tag{8}$$

The problem consists in implementing (7) and (8) in practice. In a very classical way, we separate the linear Gaussian case (see Section 3.1) and the general one (see Section 3.2).

3.1. The linear Gaussian case

In this case it is possible to compute (7) and (8) exactly:

1. *Propagating.* Let $p(\mathbf{x}_1) \sim \mathcal{N}(\hat{\mathbf{x}}_1,\mathbf{P}_1)$ and $p(\mathbf{x}_2|\mathbf{x}_1) \sim \mathcal{N}(\mathbf{A}\mathbf{x}_1+\mathbf{b},\mathbf{P}_{2|1})$. Then $p(\mathbf{x}_2) \sim \mathcal{N}\ (\mathbf{A}\hat{\mathbf{x}}_1+\mathbf{b},\ \mathbf{P}_{2|1}+\mathbf{A}\mathbf{P}_1\mathbf{A}^T)$.
2. *Updating.* Let $p(\mathbf{x}) \sim \mathcal{N}(\hat{\mathbf{x}},\mathbf{P}_\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x}) \sim \mathcal{N}(\mathbf{A}\mathbf{x}+\mathbf{b},\mathbf{P}_{\mathbf{y}|\mathbf{x}})$. Then
$$p(\mathbf{x}|\mathbf{y}) \sim \mathcal{N}(\hat{\mathbf{x}}+\mathbf{P}_\mathbf{x}\mathbf{A}^T[\mathbf{P}_{\mathbf{y}|\mathbf{x}}+\mathbf{A}\mathbf{P}_\mathbf{x}\mathbf{A}^T]^{-1}(\mathbf{y}-\mathbf{A}\hat{\mathbf{x}}-\mathbf{b}),\ \mathbf{P}_\mathbf{x}-\mathbf{P}_\mathbf{x}\mathbf{A}^T[\mathbf{P}_{\mathbf{y}|\mathbf{x}}+\mathbf{A}\mathbf{P}_\mathbf{x}\mathbf{A}^T]^{-1}\mathbf{A}\mathbf{P}_\mathbf{x}).$$

3.2. The general case

In the general case, (7) and (8) often cannot be computed exactly and one needs to resort to approximations. Let us focus here on MC techniques. At least two points of view are available: either we try to transform a set of points sampled from $p(\mathbf{x}_1)$ (resp. from $p(\mathbf{x})$) into points sampled (at least approximatively) from $p(\mathbf{x}_2)$ (resp. from $p(\mathbf{x}|\mathbf{y})$), see Section 3.2.2; or we plug a discrete weighted measure into the P and the U steps, see Section 3.2.3. As we shall see, both mechanisms involve the same three elementary operations: sampling (S) new particles; updating old weights (W); and resampling (R) from a weighted measure, which indeed are the three building blocks of any PF algorithm. Before we proceed let us first briefly recall three well known simulation techniques that will prove useful in Sections 3.2.2 and 3.2.3.

3.2.1. Three simulation techniques

1. *Hierarchical sampling.* Let $\tilde{\mathbf{x}}_1 \sim p(\mathbf{x}_1)$ and $\tilde{\mathbf{x}}_2 \sim p(\mathbf{x}_2|\tilde{\mathbf{x}}_1)$. Then $(\tilde{\mathbf{x}}_1,\tilde{\mathbf{x}}_2)$ is a sample from $p(\mathbf{x}_1,\mathbf{x}_2)$.
2. *Sampling from a discrete mixture by the composition method.* Let us consider the mixture density $p(\mathbf{x}_2) = \sum_{i=1}^{N} \alpha_i p_i(\mathbf{x}_2)$, where p_i is a pdf for all i, $\alpha_i > 0$ and $\sum_{i=1}^{N} \alpha_i = 1$. Let us sample an index j from the discrete distribution $\sum_{i=1}^{N} \alpha_i \delta_i(dx)$, and next let us sample $\tilde{\mathbf{x}}_2 \sim p_j(\mathbf{x}_2)$. Then $\tilde{\mathbf{x}}_2$ is a sample from $p(\mathbf{x}_2)$.

Remark 1. If $\mathbf{x}_2$ is seen as the marginal of variable $(x_1,\mathbf{x}_2)$, in which x_1 is a discrete latent variable with probability measure $\sum_{i=1}^{N} \alpha_i \delta_i(dx)$ and $p(\mathbf{x}_2|x_1 = i) = p_i(\mathbf{x}_2)$, then the composition method reduces to a particular instance of hierarchical sampling in which the first sample is marginalized out.

3. *Rubin's SIR mechanism* [12–14, 7, Section 9.2]: Let $\{\mathbf{x}^i\}_{i=1}^N$ be N i.i.d. samples from $p(\mathbf{x})$, and conditionally on $\{\mathbf{x}^i\}_{i=1}^N$, let $\{\tilde{\mathbf{x}}^i\}_{i=1}^I$ be I i.i.d. samples from $\sum_{i=1}^{N} (l(\mathbf{x}^i)/\sum_{i=1}^{N} l(\mathbf{x}^i))\delta_{\mathbf{x}^i}(d\mathbf{x})$. Then $\{\tilde{\mathbf{x}}^i\}_{i=1}^I$ become i.i.d. samples from $q(\mathbf{x}) \propto l(\mathbf{x})p(\mathbf{x})$ if $N \to \infty$.

3.2.2. The unweighted mechanism

1. *Propagating.* By hierarchical sampling, starting from N i.i.d. samples $\{\mathbf{x}_1^i\}_{i=1}^N$ from $p(\mathbf{x}_1)$, we get N i.i.d. samples $\{\mathbf{x}_2^i\}_{i=1}^N$ from $p(\mathbf{x}_2)$ by sampling, for each i, $\mathbf{x}_2^i$ from $p(\mathbf{x}_2|\mathbf{x}_1^i)$. This is nothing but the sampling step S of PF algorithms.
2. *Updating.* Getting N samples from $p(\mathbf{x}|\mathbf{y})$ and N samples from $p(\mathbf{x})$ can be achieved (asymptotically) by Rubin's SIR mechanism: Starting from $\{\mathbf{x}^i\}_{i=1}^N \sim p(\mathbf{x})$, we get N

points $\{\tilde{\mathbf{x}}^i\}_{i=1}^N$ (approximately) independently sampled from $p(\mathbf{x}|\mathbf{y})$ by associating to each sample $\mathbf{x}^i$ a weight proportional to $p(\mathbf{y}|\mathbf{x}^i)$, and then sampling $\{\tilde{\mathbf{x}}^i\}_{i=1}^N$ i.i.d. from $\sum_{i=1}^N (p(\mathbf{y}|\mathbf{x}^i)/\sum_{i=1}^N p(\mathbf{y}|\mathbf{x}^i))\delta_{\mathbf{x}^i}(d\mathbf{x})$. We just described nothing but the weighting step *W*, followed by the (re)sampling step *R* of PF algorithms.

3.2.3. The weighted mechanism

1. *Propagating.* Let $\sum_{i=1}^N w_i \delta_{\mathbf{x}_1^i}(d\mathbf{x}_1)$ be some discrete approximation of $p(d\mathbf{x}_1) = p(\mathbf{x}_1)\,d\mathbf{x}_1$. Injecting into (7), $p(\mathbf{x}_2)$ is approximated by $\hat{p}(\mathbf{x}_2) = \sum_{i=1}^N w_i p(\mathbf{x}_2|\mathbf{x}_1^i)$. To get i.i.d. samples $\{\mathbf{x}_2^i\}_{i=1}^N$ from this mixture pdf we can first sample *N* points $\tilde{\mathbf{x}}_1^i$ from $\sum_{i=1}^N w_i \delta_{\mathbf{x}_1^i}(d\mathbf{x}_1)$, and next sample $\mathbf{x}_2^i \sim p(\mathbf{x}_2|\tilde{\mathbf{x}}_1^i)$ (see Section 3.2.1, point 2). We just described the resampling step *R* (which from this point of view is indeed nothing but the first step of the composition method for sampling from a mixture), followed by the sampling step *S* of PF algorithms.
2. *Updating.* If $p(d\mathbf{x}) = p(\mathbf{x})\,d\mathbf{x}$ is replaced by $\sum_{i=1}^N w_i \delta_{\mathbf{x}^i}(d\mathbf{x})$, then $p(d\mathbf{x}|\mathbf{y}) = p(\mathbf{x}|\mathbf{y})\,d\mathbf{x}$ becomes $\sum_{i=1}^N (w_i p(\mathbf{y}|\mathbf{x}_i)/\sum_{i=1}^N w_i p(\mathbf{y}|\mathbf{x}_i))\delta_{\mathbf{x}^i}(d\mathbf{x})$. The points $\mathbf{x}^i$ are unchanged but each weight w_i is replaced by a weight proportional to $w_i p(\mathbf{y}|\mathbf{x}_i)$: this is the weighting step *W* of PF algorithms.

3.2.4. Unweighted vs. weighted mechanisms

As we have just seen, both mechanisms involve the same three elementary operations: *S*, *W* and *R*, in that order, and indeed coincide up to a cyclic permutation, which in fact simply means that the propagation of a weighted discrete measure reduces to that of a set of *N* particles, once these *N* points have first been simulated by the resampling step *R* (see Fig. 3, in which $\hat{p}^u_{i|j}$ (resp. $\hat{p}^w_{i|j}$) denotes an unweighted (resp. weighted) approximation of $p_{i|j}(d\mathbf{x})$). Finally the two points of view are equivalent, and as a consequence in Sections 4–6 we shall derive SMC implementations of exact filtering formulae by applying any of these mechanisms (the unweighted one, say).

Remark 2. The weighted and unweighted mechanisms we just described only serve as simple schematic toolboxes for deriving an SMC algorithm. In particular, as far as resampling is concerned Algorithms 1-*P*, 2-*P*, 1-*S* and 2-*S* below always involve a multinomial resampling step at each time instant. This does not mean that we recommend this resampling scheme, but simply that these algorithms were obtained either from Rubin's SIR mechanism (see Section 3.2.2) or the composition method (see Section 3.2.3). Now, as is well known, in Efron's bootstrap [15] some particles can be sampled several times while some will not be resampled at all; this drawback led to the development of more sophisticated alternatives such as smooth [16] or weighted [17] bootstrap resampling techniques. In the context of this paper, once a generic PF algorithm is obtained, other versions can be derived, which in particular call for resampling only if some criterion is satisfied; alternatives such as stratified of residual resampling schemes [18,19] are also available. Even if a thorough study of the improvements which can be brought to our algorithms by the existing resampling scenarios is out of the scope of this paper, we will nevertheless in Section 8.2 compare the naive and smooth bootstrap (SB) techniques.

4. Direct filtering algorithms

We now derive the two direct filtering algorithms for computing $p_{n|n}$, as well as their Kalman and SMC versions. We will see in particular that the bootstrap algorithm, and a reorganized version of the SIR algorithm with optimal CID are indeed the two generic MC implementations (in the point of view of Section 3) of these two direct filtering algorithms.

4.1. Exact direct filters

Let us start from (3) again. Due to the HMC assumption (1) the fundamental transition pdf $p(\mathbf{x}_n,\mathbf{y}_n|\,\mathbf{x}_{0:n-1},\mathbf{y}_{0:n-1}) = p(\mathbf{x}_n,\mathbf{y}_n|\mathbf{x}_{n-1})$ can be factorized as

$$p(\mathbf{x}_n,\mathbf{y}_n|\mathbf{x}_{n-1}) = p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1}), \tag{9}$$

$$p(\mathbf{x}_n,\mathbf{y}_n|\mathbf{x}_{n-1}) = \underbrace{p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)}_{q_F^{opt}}\,p(\mathbf{y}_n|\mathbf{x}_{n-1}), \tag{10}$$

note that q_F^{opt} in (10) is indeed (for the direct filtering problem, whence index *F*) the optimal CID (6) of Section 2.2.

- Injecting (9) into (3) and marginalizing provides (2). Since this direct path $p_{n-1|n-1} \overset{P}{\rightarrow} p_{n|n-1} \overset{U}{\rightarrow} p_{n|n}$ involves

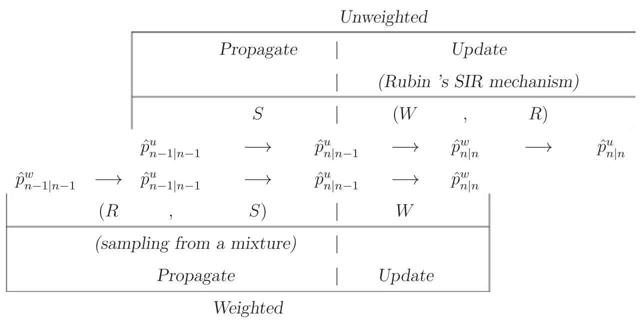
Unweighted
Propagate
Update
(Rubin 's SIR mechanism)
S
(W , R)
$\hat{p}^u_{n-1|n-1} \longrightarrow \hat{p}^u_{n|n-1} \longrightarrow \hat{p}^w_{n|n} \longrightarrow \hat{p}^u_{n|n}$
$\hat{p}^w_{n-1|n-1} \longrightarrow \hat{p}^u_{n-1|n-1} \longrightarrow \hat{p}^u_{n|n-1} \longrightarrow \hat{p}^w_{n|n}$
(R , S)
W
(sampling from a mixture)
Propagate
Update
Weighted

Fig. 3. The weighted vs. unweighted mechanisms.

the one-step ahead prediction pdf $p_{n|n-1}$ we will denote it by 1-*P*.

- Injecting (10) into (3) provides

$$p(\mathbf{x}_n|\mathbf{y}_{0:n}) \overset{(10)}{=} \int p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n) \underbrace{\left[\frac{p(\mathbf{y}_n|\mathbf{x}_{n-1})p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1})=\int \mathcal{N} d\mathbf{x}_{n-1}}\right]}_{p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n})} d\mathbf{x}_{n-1}, \tag{11}$$

which is the path $p_{n-1|n-1} \overset{U}{\to} p_{n-1|n} \overset{P}{\to} p_{n|n}$. Since this path involves the one-step backward smoothing density $p_{n-1|n}$ we will denote it by 1-*S*.

We now use the toolbox of Section 3 to implement in practice formulas (2) and (11).

4.2. Implementing (2): 1-*P* algorithm

Let us first consider (2). The solution is well known in the context of linear Gaussian state-space systems. More precisely, let

$$\begin{cases} \mathbf{x}_{n+1} = \mathbf{F}_n\mathbf{x}_n + \mathbf{G}_n\mathbf{u}_n, \\ \mathbf{y}_n = \mathbf{H}_n\mathbf{x}_n + \mathbf{v}_n, \end{cases} \tag{12}$$

in which $\mathbf{x}_0 \sim \mathcal{N}(\overline{\mathbf{x}}_0, \mathbf{P}_0)$, $\{\mathbf{u}_n\}$ and $\{\mathbf{v}_n\}$ are independent, mutually independent and independent of $\mathbf{x}_0$, $\mathbf{u}_n \sim \mathcal{N}(\mathbf{0},\mathbf{Q}_n)$ and $\mathbf{v}_n \sim \mathcal{N}(\mathbf{0},\mathbf{R}_n)$. Let also $\tilde{\mathbf{Q}}_n = \mathbf{G}_n\mathbf{Q}_n\mathbf{G}_n^T$. Then $p(\mathbf{x}_i|\mathbf{y}_{0:n}) \sim \mathcal{N}(\hat{\mathbf{x}}_{i|n}, \mathbf{P}_{i|n})$ for some $\hat{\mathbf{x}}_{i|n}$ and $\mathbf{P}_{i|n}$. Transforming $p_{n-1|n-1}$ into $p_{n|n-1}$ and $p_{n|n}$ via (2), reduces to transforming $(\hat{\mathbf{x}}_{n-1|n-1}, \mathbf{P}_{n-1|n-1})$ into $(\hat{\mathbf{x}}_{n|n-1}, \mathbf{P}_{n|n-1})$ and $(\hat{\mathbf{x}}_{n|n}, \mathbf{P}_{n|n})$. This is done by the well known KF equations ((13), (14) implement the *P* step and (15)–(18) the *U* step) [2,21]:

$$\hat{\mathbf{x}}_{n|n-1} = \mathbf{F}_{n-1}\hat{\mathbf{x}}_{n-1|n-1}, \tag{13}$$

$$\mathbf{P}_{n|n-1} = \mathbf{F}_{n-1}\mathbf{P}_{n-1|n-1}\mathbf{F}_{n-1}^T + \tilde{\mathbf{Q}}_{n-1}, \tag{14}$$

$$\tilde{\mathbf{y}}_n = \mathbf{y}_n - \mathbf{H}_n\hat{\mathbf{x}}_{n|n-1}, \tag{15}$$

$$\mathbf{L}_n = \mathbf{R}_n + \mathbf{H}_n\mathbf{P}_{n|n-1}\mathbf{H}_n^T, \tag{16}$$

$$\hat{\mathbf{x}}_{n|n} = \hat{\mathbf{x}}_{n|n-1} + \mathbf{P}_{n|n-1}\mathbf{H}_n^T\mathbf{L}_n^{-1}\tilde{\mathbf{y}}_n, \tag{17}$$

$$\mathbf{P}_{n|n} = \mathbf{P}_{n|n-1} - \mathbf{P}_{n|n-1}\mathbf{H}_n^T\mathbf{L}_n^{-1}\mathbf{H}_n\mathbf{P}_{n|n-1}. \tag{18}$$

Let us turn to MC approximations of (2). We assume that at $n-1$ we have *N* samples from $p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$. Using Section 3.2.2, point 1, we get *N* samples $\tilde{\mathbf{x}}_n^i$ from $p(\mathbf{x}_n|\mathbf{y}_{0:n-1})$ by drawing $\tilde{\mathbf{x}}_n^i$ from $p(\mathbf{x}_n|\mathbf{x}_{n-1}^i) = p(\mathbf{x}_n|\mathbf{x}_{n-1}^i, \mathbf{y}_{0:n-1})$. Next using Section 3.2.2, point 2, we get *N* (approximate) samples from $p(\mathbf{x}_n|\mathbf{y}_{0:n}) \propto p(\mathbf{y}_n|\mathbf{x}_n)$ $p(\mathbf{x}_n|\mathbf{y}_{0:n-1})$ by sampling from $\sum_{i=1}^N (p(\mathbf{y}_n|\tilde{\mathbf{x}}_n^i)/\sum_{i=1}^N p(\mathbf{y}_n|\tilde{\mathbf{x}}_n^i))\delta_{\tilde{\mathbf{x}}_n^i}(d\mathbf{x}_n)$. We just described the sampling step *S*, followed by the updating step (*W*, *R*) of the Bootstrap algorithm:

1-*P* algorithm (= Bootstrap algorithm, [8]). Let $\hat{p}(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N (1/N)\delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$ approximate $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$.

```
S. For $1 \le i \le N$, sample $\tilde{\mathbf{x}}_n^i$ from $p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$;
W. For $1 \le i \le N$, compute $w_n^i \propto p(\mathbf{y}_n|\tilde{\mathbf{x}}_n^i)$, $\sum_i w_n^i = 1$
R. For $1 \le i \le N$, sample $\mathbf{x}_n^i$ from $\sum_{i=1}^N w_n^i \delta_{\tilde{\mathbf{x}}_n^i}(d\mathbf{x}_n)$;
```

Then $\hat{p}(d\mathbf{x}_n|\mathbf{y}_{0:n}) = \sum_{i=1}^N (1/N)\delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$ approximates $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$.

4.3. Implementing (11): 1-*S* algorithm

We assume (12) again. Let us first define $\mathbf{H}_n^1 = \mathbf{H}_n\mathbf{F}_{n-1}$, $\mathbf{R}_n^1 = \mathbf{R}_n + \mathbf{H}_n\tilde{\mathbf{Q}}_{n-1}\mathbf{H}_n^T$, $\tilde{\mathbf{K}}_{n|n}^1 = \tilde{\mathbf{Q}}_{n-1}\mathbf{H}_n^T(\mathbf{R}_n^1)^{-1}$, $\mathbf{F}_{n-1}^1 = (\mathbf{I} - \tilde{\mathbf{K}}_{n|n}^1\mathbf{H}_n)\mathbf{F}_{n-1}$, $\tilde{\mathbf{Q}}_{n-1}^1 = \tilde{\mathbf{Q}}_{n-1} - \tilde{\mathbf{K}}_{n|n}^1\mathbf{R}_n^1(\tilde{\mathbf{K}}_{n|n}^1)^T$. Implementing (11) leads to the following KF equations which, up to our best knowledge, are original[1] ((19)–(22) implement the *U* step and (23), (24) the *P* step):

$$\tilde{\mathbf{y}}_n = \mathbf{y}_n - \mathbf{H}_n^1\hat{\mathbf{x}}_{n-1|n-1}, \tag{19}$$

$$\mathbf{L}_n = \mathbf{R}_n^1 + \mathbf{H}_n^1\mathbf{P}_{n-1|n-1}(\mathbf{H}_n^1)^T, \tag{20}$$

$$\hat{\mathbf{x}}_{n-1|n} = \hat{\mathbf{x}}_{n-1|n-1} + \mathbf{P}_{n-1|n-1}(\mathbf{H}_n^1)^T\mathbf{L}_n^{-1}\tilde{\mathbf{y}}_n, \tag{21}$$

$$\mathbf{P}_{n-1|n} = \mathbf{P}_{n-1|n-1} - \mathbf{P}_{n-1|n-1}(\mathbf{H}_n^1)^T\mathbf{L}_n^{-1}\mathbf{H}_n^1\mathbf{P}_{n-1|n-1}, \tag{22}$$

$$\hat{\mathbf{x}}_{n|n} = \mathbf{F}_{n-1}^1\hat{\mathbf{x}}_{n-1|n} + \tilde{\mathbf{K}}_{n|n}^1\mathbf{y}_n = \mathbf{F}_{n-1}\hat{\mathbf{x}}_{n-1|n} + \tilde{\mathbf{Q}}_{n-1}\mathbf{H}_n^T\mathbf{L}_n^{-1}\tilde{\mathbf{y}}_n, \tag{23}$$

$$\mathbf{P}_{n|n} = \mathbf{F}_{n-1}^1\mathbf{P}_{n-1|n}(\mathbf{F}_{n-1}^1)^T + \tilde{\mathbf{Q}}_{n-1}^1. \tag{24}$$

Remark 3. Algorithms (13)–(18) and (19)–(24) do not share the same internal variables. However, they both have a common structure, and both filters are driven by the so-called innovations process $\tilde{\mathbf{y}}_n$ (see e.g. [2,21]), the covariance of which is matrix $\mathbf{L}_n$. As we shall see in the sequel, these features also hold for the KF algorithms of Sections 5 and 6.

Let us now address the general case. Assume again that at $n-1$ we have *N* samples $\mathbf{x}_{n-1}^i$ from $p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$. From the SIR mechanism described in Section 3.2.2, point 2, we compute weights w_n^i proportional to $p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)$; $\sum_{i=1}^N w_n^i\delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$ approximates $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n})$, and (re)sampling from this distribution provides (approximate) samples $\{\tilde{\mathbf{x}}_{n-1}^i\}_{i=1}^N$ from $p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n})$. Using Section 3.2.2, point 1, we finally sample $\mathbf{x}_n^i$ from $p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i, \mathbf{y}_n) = p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i, \mathbf{y}_{0:n})$. We obtained the:

1-*S* algorithm (= Algorithm 8.1.1., [7, p. 253]). Let $\hat{p}(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N (1/N)\delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$ approximate $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$.

[1] More precisely, the set of Eqs. (19)–(24) is original as a Kalman *filter* algorithm, but some of these equations are known otherwise: (21)–(22) are one-step fixed-lag smoothing recursions (see [20, Eqs. (19)–(20)], where however the parameters are computed differently); and (23)–(24) are final time instant fixed-interval smoothing recursions, see [21, Chapter 10] or [22, Section III-C].

W. For $1 \le i \le N$, compute $w_n^i \propto p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)$, $\sum_{i=1}^N w_n^i = 1$;
R. For $1 \le i \le N$, sample $\tilde{\mathbf{x}}_{n-1}^i \sim \sum_{i=1}^N w_n^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$;
S. For $1 \le i \le N$, sample $\mathbf{x}_n^i$ from $p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i,\mathbf{y}_n)$.

Then $\hat{p}(d\mathbf{x}_n|\mathbf{y}_{0:n}) = \sum_{i=1}^N (1/N)\delta_{\mathbf{x}_n^i}(d\mathbf{x}_n) \simeq p(d\mathbf{x}_n|\mathbf{y}_{0:n})$.

4.4. Comments and remarks on the 1-P and 1-S SMC algorithms

We just saw that 1-*P* and 1-*S* can be understood as natural MC implementations (using the tools of Section 3.2.2) of paths $p_{n-1|n-1} \to p_{n|n-1} \to p_{n|n}$ and $p_{n-1|n-1} \to p_{n-1|n} \to p_{n|n}$, respectively. We now see that they can also be interpreted as MC approximations of (2), which differ only by the point where sampling is introduced (see Section 4.4.1), or as two particular instances of APF (see Section 4.4.2). In Section 4.4.3 we finally compare 1-*S* to the SIR algorithm with optimal CID.

4.4.1. Sampling before or after the updating step

Eq. (2) transforms $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$ into $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$. If exact computing is not possible, MC approximations of $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$ can still be obtained by plugging a discrete approximation of $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$ into (2). However, the resulting measure is continuous, so a sampling mechanism is needed somewhere in (2) if we want to further proceed at time $n+1$ and finally get an SMC algorithm.

- *Sampling before updating.* Let us position the sampling step between the *P* and *U* steps. Let us start from $\{\mathbf{x}_{n-1}^i, w_{n-1}^i\}_{i=1}^N$, i.e. let $\hat{p}(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N w_{n-1}^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$. Injecting into (2) we get an approximation $\hat{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n-1})$ (c stands for continuous) of $\hat{p}(\mathbf{x}_n|\mathbf{y}_{0:n-1})$:
$$\hat{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N w_{n-1}^i p(\mathbf{x}_n|\mathbf{x}_{n-1}^i). \tag{25}$$
If we sample at this point we need to get N samples $\{\mathbf{x}_n^i\}_{i=1}^N$ from the mixture pdf (25). Using Section 3.2.1, point 2, for all i, $1 \le i \le N$, we first sample $\tilde{\mathbf{x}}_{n-1}^i$ according to $\sum_{i=1}^N w_{n-1}^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$, and next sample $\mathbf{x}_n^i$ from $p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i)$. Then $\hat{p}(d\mathbf{x}_n|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N (1/N)\delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$ is a discrete approximation of $p(d\mathbf{x}_n|\mathbf{y}_{0:n-1})$, and from (2) we get $\hat{p}(d\mathbf{x}_n|\mathbf{y}_{0:n}) \overset{def}{=} \sum_{i=1}^N w_n^i \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$ with $w_n^i \propto p(\mathbf{y}_n|\mathbf{x}_n^i)$. Up to a cyclic permutation, this (*R*, *S*, *W*) algorithm is the Bootstrap algorithm which we just described. In particular, from this point of view the famous *resampling* step *R* of the Bootstrap algorithm is indeed nothing but the first step of the sampling procedure according to the mixture $\sum_{i=1}^N w_{n-1}^i p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$ (see Section 3.2.1, point 2).
- *Sampling after updating.* Let us now sample after the end of the *P* and *U* steps. Injecting (25) into (2) we get the following continuous approximation of $p_{n|n}$: $\tilde{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n}) = \sum_{i=1}^N w_{n-1}^i p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1}^i) / \int \mathcal{N} d\mathbf{x}_n$, which, due to (10), can be rewritten as the (continuous) mixture pdf
$$\tilde{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n}) = \sum_{i=1}^N \frac{w_{n-1}^i p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)}{\sum_{i=1}^N w_{n-1}^i p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)} p(\mathbf{x}_n|\mathbf{x}_{n-1}^i,\mathbf{y}_n). \tag{26}$$
One should finally sample from (26); using again Section 3.2.1, point 2, we see that one should build $\sum_{i=1}^N w_n^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$ with $w_n^i \propto w_{n-1}^i p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)$, sample $\tilde{\mathbf{x}}_{n-1}^i$ from it, and finally sample $\mathbf{x}_n^i$ from $p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i,\mathbf{y}_n)$, which again gives algorithm 1-*S*.

4.4.2. Connection with APF

Let us briefly recall the principle of APF ([23]; see also [24–26] for recent developments[2]). Let us start from (26) again. Pdf $\tilde{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n})$ is a finite mixture approximating $p(\mathbf{x}_n|\mathbf{y}_{0:n})$. If sampling from $\tilde{p}^c$ is difficult or impossible, one can approximate $\tilde{p}^c$ by the mixture pdf
$$q(\mathbf{x}_n) = \sum_{i=1}^N \frac{w_{n-1}^i \tau_{n-1}^i}{\sum_{i=1}^N w_{n-1}^i \tau_{n-1}^i} \tilde{q}(\mathbf{x}_n|\mathbf{x}_{n-1}^i), \tag{27}$$
sample from q (the so-called first-stage weights $\tilde{w}_{n-1}^i \propto w_{n-1}^i \tau_{n-1}^i$ and pdf $\tilde{q}$ are degrees of freedom used for designing this importance density), and use IS. Since the target density $\tilde{p}^c$ in (26) and the importance density q in (27) are both mixture densities, the (so-called second-stage) weights computed by this IS method would be the ratio of a sum of N terms over another sum of N terms. This drawback is avoided by using data augmentation (i.e. by sampling both from q and from the index of the mixture in q, and taking the marginal, according to the mechanism described in Remark 1, see [23] for details.

Now, both 1-*P* and 1-*S* are particular APF: 1-*P* is obtained if we take $\tau_{n-1}^i = 1$ for all i, and $\tilde{q}(\mathbf{x}_n|\mathbf{x}_{n-1}^i) = p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$; and 1-*S* corresponds of course to the case where $q = \tilde{p}^c$, i.e. in the so-called *fully-adapted* case, in the terminology of [23].

4.4.3. SIR with optimal CID: original or reorganized?

Comparing with Section 2.2, we see that 1-*S* is a *reordering* of the SIR algorithm with optimal CID (and with systematic resampling), see [11]. However, these two algorithms are not simply related by a shift in time. More precisely, in [11] the successive steps are $S \to W \to R$ (or, equivalently, $W \to S \to R$: steps *S* and *W* commute since λ_n^i defined in (5) does not depend on the new particle $\mathbf{x}_n^i$), while 1-*S* consists of the successive steps $W \to R \to S$. So $\mathbf{x}_n^i$ is sampled from $p(\mathbf{x}_n|\mathbf{x}_{n-1}^i,\mathbf{y}_n)$ in [11], while in 1-*S* $\mathbf{x}_n^i \sim p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i,\mathbf{y}_n)$ in which $\tilde{\mathbf{x}}_{n-1}^i \sim \sum_{i=1}^N w_n^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$. In other words, each old particle $\mathbf{x}_{n-1}^i$ is taken uniformly into account in [11], while only those with high weights do contribute to the updated trajectory in 1-*S*. We now

[2] In particular, in [25] the optimal (in terms of asymptotical variance) APF filter for a given moment is derived, but this filter by nature is function dependent.

explain more deeply the differences between both algorithms.

- The original *(S, W, R)* algorithm of Section 2.1 is, by nature, a batch IS algorithm which has been rendered adaptive, but in which the trajectories, in the absence of resampling, tend to depart (as time increases) from the target pdf $p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n})$. More precisely, at fixed time $n-1$, we would like to compensate the missing target pdf $p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$ by a set of samples from $p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$. Since it is easier to sample from q, we indeed dispose of $\hat{p}(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^{N} w_{n-1}^{i}\delta_{\mathbf{x}_{0:n-1}^{i}}(d\mathbf{x}_{0:n-1})$, where the N trajectories $\mathbf{x}_{0:n-1}^{i}$ are sampled from q and $w_{n-1}^{i} \propto p(\mathbf{x}_{0:n-1}^{i}\ |\mathbf{y}_{0:n-1})/\ q(\mathbf{x}_{0:n-1}^{i}|\mathbf{y}_{0:n-1})$. Let us now address adaptivity. Due to (3) and condition (4), this IS algorithm can be sequentialized easily: each trajectory $\mathbf{x}_{0:n-1}^{i}$ is kept unaltered and is simply extended by a new particle $\mathbf{x}_{n}^{i}$; and the associated weight can be updated recursively. However, a drawback of this simple computational scheme is that in the absence of resampling, the trajectories do not fully take into account the data. To see this, assume that resampling has occurred at time k, but not between k and n with $n>k$. From Section 3.2.2, point 2, $\mathbf{x}_{0:k}^{i}$ are (approximately[3]) sampled from $p(\mathbf{x}_{0:k}|\mathbf{y}_{0:k})$, and trajectories $\mathbf{x}_{0:n}^{i}$, which should be sampled from
$$p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) \propto p(\mathbf{x}_{0:k}|\mathbf{y}_{0:k}) \prod_{j=k+1}^{n} p(\mathbf{x}_j|\mathbf{x}_{j-1})p(\mathbf{y}_j|\mathbf{x}_j), \tag{28}$$
$$p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) \propto p(\mathbf{x}_{0:k}|\mathbf{y}_{0:k}) \prod_{j=k+1}^{n} p(\mathbf{x}_j|\mathbf{x}_{j-1},\mathbf{y}_j)p(\mathbf{y}_j|\mathbf{x}_{j-1}), \tag{29}$$
are indeed sampled either from
$$q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = p(\mathbf{x}_{0:k}|\mathbf{y}_{0:k}) \prod_{j=k+1}^{n} p(\mathbf{x}_j|\mathbf{x}_{j-1}) = p(\mathbf{x}_{0:n}|\mathbf{y}_{0:k}) \tag{30}$$
(if the bootstrap algorithm is used), or from
$$q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = p(\mathbf{x}_{0:k}|\mathbf{y}_{0:k}) \prod_{j=k+1}^{n} p(\mathbf{x}_j|\mathbf{x}_{j-1},\mathbf{y}_j) \tag{31}$$
(if the SIR algorithm with optimal CID is used). So for $0 \le j \le n$, each particle $\mathbf{x}_{j}^{i}$ does not depend on the whole data $\mathbf{y}_{0:n}$, but only on $\mathbf{y}_{0:k}$ (if (30) is used) or on $\mathbf{y}_{0:\max(k,j)}$ (if (31) is used). So trajectories $\mathbf{x}_{0:n}^{i}$ sampled from (30) or (31) are all the more likely to diverge from trajectories sampled from (28)–(29) as n increases and departs from k. This discrepancy is (somehow) corrected by the weights, which take into account the new data $\mathbf{y}_{k+1:n}$.
- By contrast, 1-*S* is not a SIS (or SIR) algorithm, in the sense that it is not a sequential version of an algorithm based on IS. Considered as a batch algorithm, at time $n-1$ $\hat{p}(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^{N}(1/N)\delta_{\mathbf{x}_{0:n-1}^{i}}(d\mathbf{x}_{0:n-1})$, in which the N trajectories are (exactly) sampled from $q(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) = p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$. Let us now address the sequential version. At time n we need to sample from
$$q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = \underbrace{p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)}_{\substack{p(\mathbf{x}_n|\mathbf{x}_{0:n-1},\mathbf{y}_{0:n}) \\ (\mathit{Section}\ \ 3.2.2,\ \ \mathit{point}\ \ 1)}} \times \underbrace{\left[\frac{p(\mathbf{y}_n|\mathbf{x}_{n-1})}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1})} p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})\right]}_{\substack{p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n}) \\ (\mathit{Section}3.2.2,\ \ \mathit{point}\ \ 2)}}. \tag{32}$$
For this pdf the sufficient condition (4) is *not* satisfied (because of factor $p(\mathbf{y}_n|\mathbf{x}_{n-1})/p(\mathbf{y}_n|\mathbf{y}_{0:n-1})$ in the bracket), so the trajectories cannot be updated so easily. The presence of this factor means that we should transform trajectories sampled from $p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$ into trajectories sampled from $p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n})$. To that end we can use IS: from Rubin's sampling scheme (see Section 3.2.1, point 3), by updating the weights first we step from $p(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) \simeq \sum_{i=1}^{N}(1/N)\delta_{\mathbf{x}_{0:n-1}^{i}}(d\mathbf{x}_{0:n-1})$ to $p(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n}) \simeq \sum_{i=1}^{N}(p(\mathbf{y}_n|\mathbf{x}_{n-1}^{i})/\sum_{i=1}^{N} p(\mathbf{y}_n|\mathbf{x}_{n-1}^{i}))\delta_{\mathbf{x}_{0:n-1}^{i}}(d\mathbf{x}_{0:n-1})$, and next by resampling we get (approximate) trajectories from $p(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n})$. Comparing with the SIR algorithm, instead of keeping all trajectories unaltered and extending each of them by one new particle, we weight all trajectories first and then reselect them according to their weights. Once reselected, the ith trajectory is extended by a new particle $\mathbf{x}_{n}^{i}$ (using Section 3.2.2, point 1). Finally in the bootstrap and SIR algorithms the batch estimates are fundamentally based on IS, but due to condition (4) no Bayes step is needed when updating the trajectories, and so no secondary IS mechanism is introduced. In 1-*S* the batch estimates are not based on IS, but updating trajectories requires an *U* step followed by a *P* step (see (32)), so IS is introduced locally, as a means to implement the Bayes mechanism.

5. *P*-based filtering algorithms

We now address the filtering algorithms which compute $p_{n|n}$ via a recursive loop involving $p_{n|n-1}$. Let us begin with
$$p(\mathbf{x}_{0:n+1}|\mathbf{y}_{0:n}) = \frac{p(\mathbf{x}_{n+1},\mathbf{y}_n|\mathbf{x}_n)}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1})} p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n-1}). \tag{33}$$
The role played by the elementary transition $p(\mathbf{x}_n,\mathbf{y}_n|\mathbf{x}_{n-1})$ in direct filtering algorithms is now played by $p(\mathbf{x}_{n+1},\mathbf{y}_n|\mathbf{x}_n)$ which, due to Bayes's rule in model (1), can be factorized as
$$p(\mathbf{x}_{n+1},\mathbf{y}_n|\mathbf{x}_n) = \underbrace{p(\mathbf{x}_{n+1}|\mathbf{x}_n)}_{q_P^{opt}} p(\mathbf{y}_n|\mathbf{x}_n), \tag{34}$$
$$p(\mathbf{x}_{n+1},\mathbf{y}_n|\mathbf{x}_n) = p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_{n+1}|\mathbf{x}_n). \tag{35}$$
Note that $q_P^{opt} = p(\mathbf{x}_{n+1}|\mathbf{x}_n)$ is the optimal CID (in the sense of Section 2.2) for the one-step ahead prediction problem (whence index *P*). Injecting (34) into (33) and integrating w.r.t. $\mathbf{x}_{0:n}$ yields the 1-*P* path (2) (up to a shift); injecting (35) into (33) and integrating w.r.t. $\mathbf{x}_{0:n}$ yields
$$p(\mathbf{x}_{n+1}|\mathbf{y}_{0:n}) \overset{(35)}{=} \int \frac{p(\mathbf{y}_n|\mathbf{x}_n)\overbrace{[p(\mathbf{x}_{n+1}|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{y}_{0:n-1})]}^{p(\mathbf{x}_n,\mathbf{x}_{n+1}|\mathbf{y}_{0:n-1})}}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1}) = \int \mathcal{N} d\mathbf{x}_n d\mathbf{x}_{n+1}}\ d\mathbf{x}_n, \tag{36}$$

[3] Though $\mathbf{x}_{0:k}^{i}$ are sampled from $p(\mathbf{x}_{0:k}|\ \mathbf{y}_{0:k})$ only if $N \to \infty$, we shall assume it is true, in particular in (30)–(31).

finally we also have

$$p(\mathbf{x}_{n+1}|\mathbf{y}_{0:n}) = \frac{p(\mathbf{y}_n|\mathbf{x}_{n+1},\mathbf{y}_{0:n-1})\overbrace{\int p(\mathbf{x}_{n+1}|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{y}_{0:n-1})d\mathbf{x}_n}^{p(\mathbf{x}_{n+1}|\mathbf{y}_{0:n-1})}}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1}) = \int \mathcal{N} d\mathbf{x}_{n+1}}. \tag{37}$$

Eqs. (2) and (37) describe the two different ways of moving from $p_{n|n-1}$ to $p_{n+1|n}$ which are obtained when incrementing one index and next the other: (2) is the path $p_{n|n-1} \to p_{n|n} \to p_{n+1|n}$ (already studied in Section 4.2) and (37) the path $p_{n|n-1} \to p_{n+1|n-1} \to p_{n+1|n}$. Finally (36) is the alternative path $p_{n|n-1} \to p_{n,n+1|n-1} \to p_{n,n+1|n} \to p_{n+1|n}$, which indeed will prove useful in SMC implementations (see below). Finally $p_{n|n}$ is computed from $p_{n|n-1}$ via (2).

Let us now consider practical implementations of formulas (37) or (36). We first assume the linear Gaussian case. Assuming (12) again, (36) and (37) yield a common solution which is well known in the context of linear Gaussian state-space systems [2, Eqs. (4.9), (4.10), (4.12), (5.6) and (5.11)], [21, Theorem 9.5.1] ((38)–(39) implement the *P* step and (40)–(43) the *U* step of the prediction loop, and the filtering path is implemented by (16)–(18)):

$$\hat{\mathbf{x}}_{n+1|n-1} = \mathbf{F}_n \hat{\mathbf{x}}_{n|n-1}, \tag{38}$$

$$\mathbf{P}_{n+1|n-1} = \mathbf{F}_n \mathbf{P}_{n|n-1} \mathbf{F}_n^T + \tilde{\mathbf{Q}}_n, \tag{39}$$

$$\tilde{\mathbf{y}}_n = \mathbf{y}_n - \mathbf{H}_n \hat{\mathbf{x}}_{n|n-1}, \tag{40}$$

$$\mathbf{L}_n = \mathbf{H}_n \mathbf{P}_{n|n-1} \mathbf{H}_n^T + \mathbf{R}_n, \tag{41}$$

$$\hat{\mathbf{x}}_{n+1|n} = \hat{\mathbf{x}}_{n+1|n-1} + \mathbf{F}_n \mathbf{P}_{n|n-1} \mathbf{H}_n^T \mathbf{L}_n^{-1} \tilde{\mathbf{y}}_n, \tag{42}$$

$$\mathbf{P}_{n+1|n} = \mathbf{P}_{n+1|n-1} - \mathbf{F}_n \mathbf{P}_{n|n-1} \mathbf{H}_n^T \mathbf{L}_n^{-1} \mathbf{H}_n \mathbf{P}_{n|n-1} \mathbf{F}_n^T. \tag{43}$$

We now consider the general case. Implementing (37) would require the knowledge of $p(\mathbf{y}_n|\mathbf{x}_{n+1},\mathbf{y}_{0:n-1})$, which is not directly available. Implementing (36) is indeed simpler; using Section 3.2.2 again, we get the following algorithm[4] (denoted by 2-*P* since (36) involves $p_{n+1|n-1}$):

2-*P* Algorithm. Let $p(d\mathbf{x}_n|\mathbf{y}_{0:n-1}) \simeq \sum_{i=1}^N (1/N)\delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$.

```
Prediction.   S. For $1 \le i \le N$, sample $\tilde{\mathbf{x}}_{n+1}^i \sim p(\mathbf{x}_{n+1}|\mathbf{x}_n^i)$;
              W. For $1 \le i \le N$, compute $w_n^i \propto p(\mathbf{y}_n|\mathbf{x}_n^i)$, $\sum_{i=1}^N w_n^i = 1$;
              R. For $1 \le i \le N$, sample $\mathbf{x}_{n+1}^i \sim \sum_{i=1}^N w_n^i \delta_{\tilde{\mathbf{x}}_{n+1}^i}(d\mathbf{x}_{n+1})$;
              Then $p(d\mathbf{x}_{n+1}|\mathbf{y}_{0:n}) \simeq \sum_{i=1}^N (1/N)\delta_{\mathbf{x}_{n+1}^i}(d\mathbf{x}_{n+1})$.
Filtering.    $p(d\mathbf{x}_n|\mathbf{y}_{0:n}) \simeq \sum_{i=1}^N w_n^i \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$.
```

6. *S*-based filtering algorithms

Let us finally see how $p_{n|n}$ can be computed recursively via the propagation of a one-step backward smoothing distribution. We start from

$$p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n+1}) = \frac{p(\mathbf{x}_n,\mathbf{y}_{n+1}|\mathbf{x}_{0:n-1},\mathbf{y}_{0:n})}{p(\mathbf{y}_{n+1}|\mathbf{y}_{0:n})} p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n}). \tag{44}$$

From the HMC model (1) one can check easily that $(\mathbf{x}_n,\mathbf{y}_{n+1})$ is a Markov chain. So factor $p(\mathbf{x}_n,\mathbf{y}_{n+1}|\mathbf{x}_{0:n-1},\mathbf{y}_{0:n})$ in (44) reduces to $p(\mathbf{x}_n,\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{y}_n)$, which itself can be factorized as

$$p(\mathbf{x}_n,\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{y}_n) = \underbrace{p(\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{x}_n,\mathbf{y}_n)}_{p(\mathbf{y}_{n+1}|\mathbf{x}_n)} p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n), \tag{45}$$

$$p(\mathbf{x}_n,\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{y}_n) = \underbrace{p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n,\mathbf{y}_{n+1})}_{q_S^{opt}} p(\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{y}_n), \tag{46}$$

in which q_S^{opt} is now the optimal CID (in the sense of Section 2.2) for the one-step backward smoothing problem (whence index *S*). Injecting (45) into (44), and taking the marginal, we get (11) again (up to a shift); on the other hand, injecting (46) into (44), and taking the marginal, we get

$$p(\mathbf{x}_n|\mathbf{y}_{0:n+1}) \overset{(46)}{=} \int p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n,\mathbf{y}_{n+1}) \times \underbrace{\left[\frac{p(\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{y}_n)p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n})}{p(\mathbf{y}_{n+1}|\mathbf{y}_{0:n}) = \int \mathcal{N} d\mathbf{x}_{n-1}}\right]}_{p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n+1})} d\mathbf{x}_{n-1}. \tag{47}$$

Eq. (11) describes the path $p_{n-1|n} \to p_{n|n} \to p_{n|n+1}$, already studied in Section 4.3, while (47) describes the path $p_{n-1|n} \to p_{n-1|n+1} \to p_{n|n+1}$; (47) is the smoothing counterpart of the prediction equation (37) (the path $p_{n-1|n} \to p_{n-1|n+1} \to p_{n|n+1}$ can be considered as the mirror path of $p_{n|n-1} \to p_{n+1|n-1} \to p_{n+1|n}$, in which the time indices of the observed and hidden processes are inverted).

Let us now implement (47). We first address the linear Gaussian case. Let $\mathbf{H}_{n+1}^2 = \mathbf{H}_{n+1}^1 \mathbf{F}_{n-1}^1$, $\mathbf{R}_{n+1}^2 = \mathbf{R}_{n+1}^1 + \mathbf{H}_{n+1}^1 \tilde{\mathbf{Q}}_{n-1}^1 (\mathbf{H}_{n+1}^1)^T$, $\tilde{\mathbf{K}}_{n|n+1}^2 = \tilde{\mathbf{Q}}_{n-1}^1 (\mathbf{H}_{n+1}^1)^T (\mathbf{R}_{n+1}^2)^{-1}$, $\mathbf{F}_{n-1}^2 = (\mathbf{I} - \tilde{\mathbf{K}}_{n|n+1}^2 \mathbf{H}_{n+1}^1)\mathbf{F}_{n-1}^1$ and $\tilde{\mathbf{Q}}_{n-1}^2 = \tilde{\mathbf{Q}}_{n-1}^1 - \tilde{\mathbf{K}}_{n|n+1}^2 \mathbf{R}_{n+1}^2 (\tilde{\mathbf{K}}_{n|n+1}^2)^T$, in which $\mathbf{H}_{n+1}^1$, $\mathbf{F}_{n-1}^1$ and $\tilde{\mathbf{Q}}_{n-1}^1$ are defined in Section 4.3. We get the following *S*-based KF, which up to our best knowledge is original ((48)–(51) implement the *U* step and (52)–(53) the *P* step of the smoothing loop, and the filtering path is implemented by (23)–(24)):

$$\tilde{\mathbf{y}}_{n+1} = \mathbf{y}_{n+1} - \mathbf{H}_{n+1}^2 \hat{\mathbf{x}}_{n-1|n} - \mathbf{H}_{n+1}^1 \tilde{\mathbf{K}}_{n|n}^1 \mathbf{y}_n, \tag{48}$$

$$\mathbf{L}_{n+1} = \mathbf{R}_{n+1}^2 + \mathbf{H}_{n+1}^2 \mathbf{P}_{n-1|n} (\mathbf{H}_{n+1}^2)^T, \tag{49}$$

$$\hat{\mathbf{x}}_{n-1|n+1} = \hat{\mathbf{x}}_{n-1|n} + \mathbf{P}_{n-1|n} (\mathbf{H}_{n+1}^2)^T \mathbf{L}_{n+1}^{-1} \tilde{\mathbf{y}}_{n+1}, \tag{50}$$

$$\mathbf{P}_{n-1|n+1} = \mathbf{P}_{n-1|n} - \mathbf{P}_{n-1|n} (\mathbf{H}_{n+1}^2)^T \mathbf{L}_{n+1}^{-1} \mathbf{H}_{n+1}^2 \mathbf{P}_{n-1|n}, \tag{51}$$

$$\hat{\mathbf{x}}_{n|n+1} = \mathbf{F}_{n-1}^2 \hat{\mathbf{x}}_{n-1|n+1} + \tilde{\mathbf{K}}_{n|n+1}^2 \mathbf{y}_{n+1} + (\mathbf{I} - \tilde{\mathbf{K}}_{n|n+1}^2 \mathbf{H}_{n+1}^1) \tilde{\mathbf{K}}_{n|n}^1 \mathbf{y}_n, \tag{52}$$

$$\mathbf{P}_{n|n+1} = \mathbf{F}_{n-1}^2 \mathbf{P}_{n-1|n+1} (\mathbf{F}_{n-1}^2)^T + \tilde{\mathbf{Q}}_{n-1}^2. \tag{53}$$

On the other hand, by using Section 3.2.2 we get the following generic SMC implementation of (47), which will be denoted by 2-*S*, since (47) involves the two-step backward smoothing pdf $p_{n-1|n+1}$:

2-*S* **algorithm**. Let $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n}) \simeq \sum_{i=1}^N (1/N) \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$.

[4] We do not resample in the filtering step in order not to introduce unnecessary MC variation.

Smoothing. *W*. For $1 \le i \le N$, compute $w^i_{n+1} \propto p(\mathbf{y}_{n+1}|\mathbf{x}^i_{n-1},\mathbf{y}_n)$, $\sum_{i=1}^N w^i_{n+1} = 1$;
R. For $1 \le i \le N$, sample $\tilde{\mathbf{x}}^i_{n-1}$ from $\sum_{i=1}^N w^i_{n+1}\delta_{\mathbf{x}^i_{n-1}}(d\mathbf{x}_{n-1})$;
S. For $1 \le i \le N$, sample $\mathbf{x}^i_n \sim p(\mathbf{x}_n|\tilde{\mathbf{x}}^i_{n-1},\mathbf{y}_n,\mathbf{y}_{n+1})$; then $p(d\mathbf{x}_n|\mathbf{y}_{0:n+1}) \simeq \sum_{i=1}^N (1/N)\delta_{\mathbf{x}^i_n}(d\mathbf{x}_n)$.

Filtering. For $1 \le i \le N$, sample $\overline{\mathbf{x}}^i_n \sim p(\mathbf{x}_n|\mathbf{x}^i_{n-1},\mathbf{y}_n)$; then $p(d\mathbf{x}_n|\mathbf{y}_{0:n}) \simeq \sum_{i=1}^N (1/N)\delta_{\overline{\mathbf{x}}^i_n}(d\mathbf{x}_n)$.

7. Classification of the KF and SMC algorithms

Before we proceed to simulations let us first summarize the results of Sections 4–6. We derived six integral formulas for computing $p_{n|n}$: two of them directly compute $p_{n|n}$ from $p_{n-1|n-1}$, two compute $p_{n|n}$ *via* the prediction loop ($p_{n|n-1} \to p_{n+1|n}$), and two *via* the smoothing loop ($p_{n-1|n} \to p_{n|n+1}$). Each of these six formulas transforms $p_{j|k}$ into $p_{j+1|k+1}$ by first updating one index and then the other, and the filtering pdf of interest $p_{n|n}$ is computed either directly (i.e. within the recursive loop) or indirectly.

In the linear Gaussian case these equations can be computed exactly, which yields six KF algorithms, four of which are distinct. Two of these algorithms are well-known and two are original. Any of these KF is a two-in-one algorithm, in the sense that it computes the parameters $\mathbf{x}_{n|n}$ and $\mathbf{P}_{n|n}$ of the filtering pdf $p_{n|n}$, but also those of another Gaussian pdf ($p_{n|n-1}$ or $p_{n+1|n-1}$ for the *P*-based filters, $p_{n-1|n}$ or $p_{n-1|n+1}$ for the *S*-based ones). In the general case, the formulas can be implemented by SMC approximations, which yields two direct, two *P*-based and two *S*-based PF algorithms. Among each pair of algorithms, the optimal solution (optimal in terms of the conditional variance of the weights) is obtained when updating precedes propagation. Among the six solutions some coincide, and two are well known: the Bootstrap algorithm is both a direct and *P*-based PF algorithm, and is optimal for the prediction loop, but not for the filtering one; the reorganized SIR algorithm with optimal CID (or fully adapted APF) is both a direct and *S*-based PF algorithm, and is optimal for the filtering loop, but not for the smoothing one. Finally the two remaining *P*- and *S*-based algorithms mirror each other. A summary of these algorithms is given in Table 1.

8. Simulations

We are now going to compare the performances of the four SMC algorithms 1-*P*, 2-*P*, 1-*S*, 2-*S*, and of the SIR algorithm with optimal CID $p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)$ (see Section 2.2 and also the discussion in Section 4.4.3), from now on simply denoted by SIR. In practice using a given SMC algorithm requires that one can sample from the associated optimal CID, and compute the associated likelihood at any point; the optimal CID and associated likelihood are the first and second factors of (10), (34) or (46) (in particular the optimal CID for the direct, *P*- and *S*-based algorithms is recalled in Table 1). There is no difficulty with *P*-based algorithms, because $p(\mathbf{x}_{n+1}|\mathbf{x}_n)$ and $p(\mathbf{y}_n|\mathbf{x}_n)$ are available. But as is well known, in many cases $p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)$ and $p(\mathbf{y}_n|\mathbf{x}_{n-1})$ are neither directly available nor can be computed exactly; and we would expect (at first sight) that computing $p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n,\mathbf{y}_{n+1})$ and $p(\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{y}_n)$ is even more difficult. Let us observe however that computing $(p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n,\mathbf{y}_{n+1}),\ p(\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{y}_n))$ from $(p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n),\ p(\mathbf{y}_n|\mathbf{x}_{n-1}))$ is of the same difficulty as computing $(p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n),\ p(\mathbf{y}_n|\mathbf{x}_{n-1}))$ from $(p(\mathbf{x}_n|\mathbf{x}_{n-1}),\ p(\mathbf{y}_n|\mathbf{x}_n))$, because

$$p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n) \propto p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1}), \tag{54}$$

$$p(\mathbf{y}_n|\mathbf{x}_{n-1}) = \int p(\mathbf{y}_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1})d\mathbf{x}_n, \tag{55}$$

$$p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n,\mathbf{y}_{n+1}) \propto p(\mathbf{y}_{n+1}|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n), \tag{56}$$

$$p(\mathbf{y}_{n+1}|\mathbf{x}_{n-1},\mathbf{y}_n) = \int p(\mathbf{y}_{n+1}|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)d\mathbf{x}_n. \tag{57}$$

So (54) and (55) (resp. (56) and (57)) are particular instances of (8) and (7), which consist in computing $(p(\mathbf{x}_1|\mathbf{x}_2),p(\mathbf{x}_2))$ from $(p(\mathbf{x}_1),p(\mathbf{x}_2|\mathbf{x}_1))$, *i.e. in computing one of the two Bayes factorization of a joint pdf, given the other one*. As a consequence, up to some necessary adaptations (see Section 8.3 below) the 2-*S* SMC algorithm can be used in situations where $(p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n),\ p(\mathbf{y}_n|\mathbf{x}_{n-1}))$ can be computed exactly (as in non-linear state-space models with linear measurement equation and additive Gaussian noise), provided we use for computing (56)–(57) one of the approximation techniques [11,29,27] which have already been developed for computing $p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)$ and $p(\mathbf{y}_n|\mathbf{x}_{n-1})$ from $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ and $p(\mathbf{y}_n|\mathbf{x}_n)$.

8.1. Simulations, linear model

We first consider the following state-space model:

$$\begin{cases} x_{n+1} = 0.2x_n + u_n, \\ y_n = 5x_n + v_n, \end{cases} \tag{58}$$

Table 1
Direct, *P*- and *S*-based KF and PF algorithms.

Nature	Eq.	Loop	q^{opt}	KF	SMC
P-based	(36)–(37)	$p_{n\|n-1} \to p_{n+1\|n-1} \to p_{n+1\|n}\overset{\downarrow}{p_{n\|n}}$		(38)–(47), (16)–(18).	2-*P* (original)
	(2)	$p_{n\|n-1} \to p_{n\|n} \to p_{n+1\|n}$	$q_P^{opt} = p(\mathbf{x}_{n+1}\|\mathbf{x}_n)$	(13)–(18)	1-*P* (= Bootstrap)
Direct	(2)	$p_{n-1\|n-1} \to p_{n\|n-1} \to p_{n\|n}$			
	(11)	$p_{n-1\|n-1} \to p_{n-1\|n} \to p_{n\|n}$	$q_F^{opt} = p(\mathbf{x}_n\|\mathbf{x}_{n-1}\mathbf{y}_n)$	(19)–(24) (original)	1-*S* (=reorg. SIR + q_F^{opt})
S-based	(11)	$p_{n-1\|n} \to p_{n\|n} \to p_{n\|n+1}$			
	(47)	$p_{n-1\|n} \to p_{n-1\|n+1} \to p_{n\|n+1}\overset{\downarrow}{p_{n\|n}}$	$q_S^{opt} = p(\mathbf{x}_n\|\mathbf{x}_{n-1},\mathbf{y}_n\mathbf{y}_{n+1})$	(48)–(53), (23)–(24) (original)	2-*S* (original)

in which u_n and v_n are i.i.d., mutually independent and independent of x_0, with $x_0 \sim \mathcal{N}(0.5,0.5)$, $u_n \sim \mathcal{N}(0,Q)$ and $v_n \sim \mathcal{N}(0,R)$. Of course, in this model exact KF is available and serves as a benchmark solution. For all five SMC algorithms the optimal CID and associated incremental weight can be computed exactly. More precisely, by using Section 3.1 and (54)–(57) we get $p(x_{n+1}|x_n) \sim \mathcal{N}(0.2x_n,Q)$, $p(y_n|x_n) \sim \mathcal{N}(5x_n,R)$; $p(x_n|x_{n-1},y_n) \sim \mathcal{N}(m,P)$ and $p(y_n|x_{n-1}) \sim \mathcal{N}(x_{n-1},\Sigma)$ with $m=(0.2R/(R+25Q))x_{n-1}+(5Q/(R+25Q))y_n$, $P=RQ/(R+25Q)$, and $\Sigma=R+25Q$; and finally $p(x_n|x_{n-1},y_n,y_{n+1}) \sim \mathcal{N}((\Sigma/(\Sigma+P))m+(P/(\Sigma+P))y_{n+1}, \Sigma P/(\Sigma+P))$ and $p(y_{n+1}|x_{n-1},y_n) \sim \mathcal{N}(m,\Sigma+P^2)$.

Let $\mathcal{J}$ be defined as $\mathcal{J}=\frac{1}{50}\sum_{n=1}^{50}[\frac{1}{1000}\sum_{j=1}^{1000}(\hat{x}^j_{n|n}-x^j_n)^2]^{1/2}$ ($\mathcal{J}$ is averaged over 1000 realizations, and 50 time indices). Let $N=100$ and $R=2$. Table 2 displays $\mathcal{J}$ for different values of Q. As we can see S-based algorithms always outperform P-based ones, and for a class of algorithms (P- or S-based) better results occur when updating precedes propagation. 1-S slightly outperforms the SIR algorithm, but $\mathcal{J}$ differs only by the third digit. For $Q=0.1$ all algorithms behave almost identically, but the P-based ones degrade quickly as Q increases: in that case the strong variations of x_n are better tracked when y_n (for SIR or 1-S) or even y_n and y_{n+1} (for 2-S) are taken into account. In Fig. 4 we set $Q=0.1$ and we see how $\mathcal{J}$ evolves with N. The ordering of the algorithms is maintained, but when N becomes higher the performances of the SIR algorithm and of the two S-based ones become very close to that of the KF.

8.2. Simulations, non-linear model

Let us now consider the model [9]:

$$\begin{cases} x_{n+1}=f_n(x_n)+u_n, \\ y_n=g(x_n)+v_n, \end{cases} \tag{59}$$

with $f_n(x_n)=0.5x_n+25x_n/(1+x_n^2)+8\cos(1.2(n+1))$, $g(x_n)=x_n^2/20$, and in which u_n and v_n are i.i.d., mutually independent and independent of x_0, with $x_0 \sim \mathcal{N}(0.5,0.5)$. Let also $u_n \sim \mathcal{N}(0,Q)$ and $v_n \sim \mathcal{N}(0,R)$. In this model $p(x_n|x_{n-1})$ and $p(y_n|x_n)$ are available, so implementing the P-based algorithms is straightforward. However, $p(x_n|x_{n-1},y_n)$ and $p(y_n|x_{n-1})$ cannot be computed exactly. Let us first briefly recall three well known techniques for approximating $p(x_n|x_{n-1},y_n)$ and $p(y_n|x_{n-1})$.

8.2.1. Approximating $p(x_n|x_{n-1},y_n)$ and $p(y_n|x_{n-1})$

- *Linearizing the observation equation* [28,11,29]. In (59) $p(x_n|x_{n-1}) \sim \mathcal{N}(f_{n-1}(x_{n-1}),Q)$, $p(y_n|x_n) \sim \mathcal{N}(g(x_n),R)$. Since $p(x_n|x_{n-1})$ and $p(y_n|x_n)$ are Gaussian, from Section 3.1, point 2, pdf $p(x_n|x^i_{n-1},y_n) \propto p(y_n|x_n)p(x_n|x^i_{n-1})$ could be computed easily if moreover $E(y_n|x_n)=ax_n+b$ for some a and b. This second condition does not hold here, so the method consists in linearizing the observation equation in a neighbourhood of $f_{n-1}(x^i_{n-1})$. We get

$$y_n \simeq g(f_{n-1}(x^i_{n-1}))+g'(f_{n-1}(x^i_{n-1}))(x_n-f_{n-1}(x^i_{n-1}))+v_n, \tag{60}$$

$$y_n \simeq \underbrace{-\frac{f_{n-1}(x^i_{n-1})^2}{20}}_{b}+\underbrace{\frac{f_{n-1}(x^i_{n-1})}{10}}_{a}x_n+v_n. \tag{61}$$

- *Exact matching moment* (*EMM*) [29]. The method consists in approximating the true pdf $p(x_n,y_n|x_{n-1})$ by a Gaussian pdf $\hat{p}_{EMM}(x_n,y_n|x_{n-1})$. The parameters of $\hat{p}_{EMM}$ can be computed exactly: E$[x_n|x^i_{n-1}]$ and var$[x_n|x^i_{n-1}]$ are available, E$[y_n|x^i_{n-1}]$ and var $[y_n|x^i_{n-1}]$ can be computed from $y_n=(f_{n-1}(x_{n-1})+u_{n-1})^2/20+v_n$ and Covar$[x_n,y_n|$

Table 2
Empirical standard deviation $\mathcal{J}$, linear model.

Q	2-P	1-P	SIR	1-S	2-S	KF
0.1	0.2183713	0.2155558	0.2147512	0.2134734	0.2129922	0.2126259
1	0.3489346	0.2844732	0.2754586	0.2739999	0.2731135	0.2726688
3	0.6020060	0.3092687	0.2820246	0.2809878	0.2809739	0.2794307
5	0.8511697	0.3287174	0.2829190	0.2819416	0.2815135	0.2801607
10	1.3505633	0.3723547	0.2843347	0.2833163	0.2830501	0.2817664

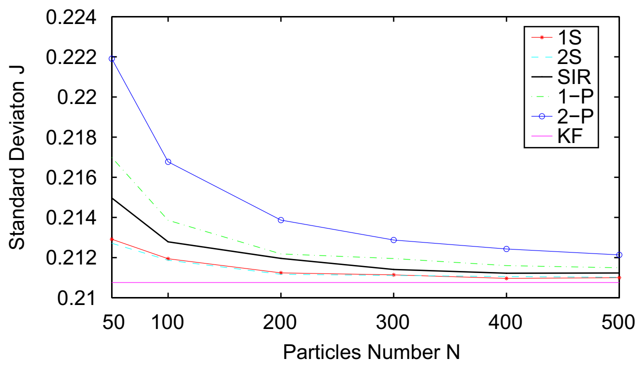

Fig. 4. Empirical standard deviation, linear model.

x^i_{n-1}] from E[$x_n y_n|x^i_{n-1}$]. Finally

$$\hat p_{EMM}(x_n,y_n|x^i_{n-1}) \sim \mathcal{N}\left(\begin{bmatrix} f_{n-1}(x^i_{n-1}) \\ \frac{f_{n-1}(x^i_{n-1})^2}{20}+\frac{Q}{20}\end{bmatrix}, \begin{bmatrix} Q & \frac{f_{n-1}(x^i_{n-1})Q}{10} \\ \frac{f_{n-1}(x^i_{n-1})Q}{10} & \frac{f_{n-1}(x^i_{n-1})^2Q}{100}+\frac{Q^2}{200}+R\end{bmatrix}\right), \qquad (62)$$

from which one can compute an approximation $\hat p_{EMM}(x_n|x^i_{n-1},y_n)$ of $p(x_n|x^i_{n-1},y_n)$ and $\hat p_{EMM}(y_n|x^i_{n-1})$ of $p(y_n|x^i_{n-1})$.

- *Unscented particle filter* (*UPF*) [27]. The method still consists in approximating the true pdf $p(x_n,y_n|x_{n-1}) = p(x_n|x_{n-1})p(y_n|x_n)$ by a Gaussian pdf $\hat p_{UPF}(x_n,y_n|x_{n-1})$, the parameters of which are now approximated via an unscented transform (UT) (see e.g. [30]). In (59) E $(x_n|x^i_{n-1})$ and var $(x_n|x^i_{n-1})$ are available, so we only need to evaluate the first two moments of $p(y_n|x_{n-1})$. We eventually get a Gaussian approximation $\hat p_{UPF}(x_n,y_n|x^i_{n-1})$ (which differs from $\hat p_{EMM}(x_n,y_n|x^i_{n-1})$ in (62)) of $p(x_n,y_n|x^i_{n-1})$, from which approximations $\hat p_{UPF}(x_n|x^i_{n-1},y_n)$ of $p(x_n|x_{n-1},y_n)$ and $\hat p_{UPF}(y_n|x^i_{n-1})$ of $p(y_n|x^i_{n-1})$ are eventually computed.

Remark 4. Note that the EMM or UPF methods which we just recalled yield an approximation $\hat p_{EMM}(y_n|x^i_{n-1})$ or $\hat p_{UPF}(y_n|x^i_{n-1})$ of $p(y_n|x^i_{n-1})$, needed to calculate the weights in the 1-*S* algorithm. However, these approximations sometimes lead to poor performances, especially when *Q* is high. In such cases, we alternatively use IS for estimating $p(y_n|x^i_{n-1})$. More precisely, we compute $\hat p_{IS}(y_n|x^i_{n-1})$ as $\hat p(y_n|x^i_{n-1})(\overline{x}^i_n) = p(y_n|\overline{x}^i_n)p(\overline{x}^i_n|x^i_{n-1})/\ \hat p(\overline{x}^i_n|x^i_{n-1},y_n)$ where $\overline{x}^i_n \sim \hat p(x_n|x^i_{n-1},y_n)$.

8.2.2. Simulations

We now turn to simulations. Let $Q=1$, $N=50$, and in UPF $\alpha=1$ and $\beta=0$. For 1-*S* we use either $\hat p_{EMM}(y_n|x^i_{n-1})$ or $\hat p_{UPF}(y_n|x^i_{n-1})$. Table 3 displays $\mathcal{J}$ for different values of *R*. The ordering of the algorithms is maintained, and the difference between SIR and 1-*S* becomes significant (simulations using the linearization method have already been performed in [28], and are efficient only if $N \geq 500$). Note that in (59) f_n has strong variations, so the influence of y_n is essential. This explains the difference between *P*-based algorithms and the SIR and 1-*S* ones, at least when *R* is small. On the other hand if the observations get very noisy ($R=10$) the performance of 2-*P* remains unaltered, while by contrast the SIR and 1-*S* algorithms degrade. Also note that $\mathcal{J}_{2P}$ and $\mathcal{J}_{1P}$ (and also $\mathcal{J}_{SIR}$ to a lesser extent) decrease when *R* increases while remaining small (up to $R=3$ for 2-*P* and 1-*P*, and $R=2$ for SIR). This is not surprising because in 2-*P* and 1-*P* particles x^i_n are guided blindly by the prior and then weighted by the past likelihood $p(y_{n-1}|x^i_{n-1})$ (for 2-*P*) or the present one $p(y_n|x^i_n)$ (for 1-*P*); this likelihood is tight if *R* is small, so few different particles are kept after the resampling step, all the more than the number of particles is small ($N=50$ in this simulation). Now in SIR we do take into account the observation y_n when we sample from $p(x_n|x^i_{n-1},y_n)$, but we nevertheless get poor results if the likelihood between x^i_{n-1} and y^i_n (which depends on *R*) is weak. Interestingly enough, we do not observe this phenomenon with 1-*S*, because particles x^i_{n-1} are first resampled according to a weight proportional to $p(y_n|x^i_{n-1})$, which guarantees that x_n will be sampled from $p(x_n|x^i_{n-1},y_n)$ where x^i_{n-1} and y_n already have a high likelihood; this simulation illustrates the importance of resampling before sampling, especially when the model is very informative.

Next in Table 4 we set $Q=10$, $R=1$, in UPF $\alpha=0.94$ and $\beta=0$, and we use $\hat p_{IS}(y_n|x^i_{n-1})$ (see Section 8.2.1, Remark 4). We see how $\mathcal{J}$ evolves with the number of particles *N*. As above, 1-*S* outperforms the *P*-based and SIR algorithms. This simulation also includes columns 2-*P*-SB and 1-*P*-SB, where an SB resampling technique is used (see Section 3.2, Remark 2). Following [31], SB resampling was done from the Gaussian mixture $\sum_{i=1}^N w^i_n \mathcal{N}(x^i_n,B_n)$ with $B_n=Vb_n$, $b_n=4/N(n_x+2)^{1/(n_x+4)}$, $V=\sum_{i=1}^N w^i_n(x^i_n-\sum_{j=1}^N w^j_n x^j_n)\,(x^i_n-\sum_{j=1}^N w^j_n x^j_n)^T$ and n_x is the dimension of the state vector x_k (here $n_x=1$). For these model and parameters smooth and naive bootstrap resampling provide similar results for 1-*P*, but SB improves the results of the 2-*P* algorithm. We actually observed in other simulations that the ordering of the algorithms is maintained if using SB resampling. Even though SB ensures that all resampled particles will be different, it does not necessarily outperform multinomial bootstrap resampling, see [32,31]; and indeed it seems that the effect of the chosen resampling

Table 3
Empirical standard deviation $\mathcal{J}$, non-linear model.

R	2-*P*	1-*P*	SIR(EMM)	SIR(UPF)	1-*S*(EMM)	1-*S*(UPF)
0.5	4.6937	3.7397	3.3746	3.4745	2.7655	2.7460
1	4.3601	3.6139	3.3611	3.4082	2.8616	2.8823
2	4.1670	3.5613	3.4065	3.4571	2.9705	2.9683
3	3.9561	3.5611	3.4232	3.4599	3.0079	3.0028
5	4.1270	3.6724	3.6820	3.6766	3.3171	3.2730
10	4.3013	4.1054	4.0950	4.0524	3.8455	3.8002
20	4.8327	4.7053	4.7030	4.7249	4.4992	4.5309

Table 4
Empirical standard deviation $\mathcal{J}$, non-linear model.

N	2-*P*	2-*P*-SB	1-*P*	1-*P*-SB	SIR(EMM)	SIR(UPF)	1-*S*(EMM)	1-*S*(UPF)
50	7.0848	6.4659	5.5501	5.1970	5.1884	5.0528	4.9003	4.9302
100	5.8079	5.3658	4.9031	4.8522	4.7207	4.7397	4.5679	4.5796
200	5.2276	4.8377	4.5898	4.5987	4.4926	4.5260	4.4780	4.4698
300	4.9139	4.6783	4.4642	4.4951	4.4522	4.4477	4.4310	4.4396
500	4.6379	4.5467	4.4397	4.4267	4.4218	4.4122	4.4138	4.4077

scheme (classical or SB) depends on the model, on its parameters (Q and R), and also on the number of particles. It would certainly be of interest to analyse precisely when SB is likely to outperform the classical bootstrap, however such an extensive study is out of the scope of this paper.

8.3. Simulations, semi-linear models

In the previous section we compared the P-based, SIR and 1-S algorithms, but not the 2-S one, because in model (59) $p(x_n|x_{n-1},y_n,y_{n+1})$ and $p(y_{n+1}|x_{n-1},y_n)$ are rather difficult to implement. The 2-S algorithm can nevertheless be used in some situations, as we now see. Let us first consider the non-linear model with linear measurements equation

$$\begin{cases} x_{n+1} = f_n(x_n)+u_n, \\ y_n = 0.5x_n+v_n, \end{cases} \tag{63}$$

in which u_n and v_n are i.i.d., mutually independent and independent of x_0, with $x_0 \sim \mathcal{N}(0,1)$, $u_n \sim \mathcal{N}(0,Q)$ and $v_n \sim \mathcal{N}(0,R)$ (the first equation of (63) coincides with that of (59)). In this model $p(x_{n+1}|x_n)$ and $p(y_n|x_n)$ are available as above. Moreover E $(y_n|x_n)$ is linear in x_n, so $p(x_n|x_{n-1},y_n)$ and $p(y_n|x_{n-1})$ can also be computed easily: $p(x_n|x_{n-1},y_n) \sim \mathcal{N}(\tilde{f}_{n-1}(x_{n-1},y_n),\tilde{Q})$, $p(y_{n+1}|x_n) \sim \mathcal{N}(\tilde{g}(x_n),\tilde{R})$, with $\tilde{f}_{n-1}(x_{n-1},y_n) = (4R/(4R+Q))f_{n-1}(x_{n-1})+(2Q/(4R+Q))y_n$, $\tilde{Q} = 4RQ/(4R+Q)$, $\tilde{g}(x_n) = 0.5f_n(x_n)$ and $\tilde{R} = R+0.25Q$.

Next, even though $p(x_n|x_{n-1},y_n)$ and $p(y_{n+1}|x_n)$ are Gaussian, E $(y_{n+1}|x_n) = \mathrm{E}(y_{n+1}|x_n,x_{n-1},x_n) = \tilde{g}(x_n)$ is no longer linear in x_n, so $p(x_n|x_{n-1},y_n,y_{n+1}) \propto p(x_n|x_{n-1},y_n)\, p(y_{n+1}|x_n)$ is not Gaussian. However, the problem of computing $p(x_n|x_{n-1},y_n,y_{n+1})$ from $(p(x_n|x_{n-1},y_n), p(y_{n+1}|x_n))$ is the same as that of computing $p(x_n|x_{n-1},y_n)$ from $(p(x_n|x_{n-1}),p(y_n|x_n))$, so the techniques recalled in Section 8.2.1 can be adapted to (63) (in particular $p(y_{n+1}|x^i_{n-1},y_n)$ can be computed by adapting the EMM or UPF techniques, or the IS technique of Remark 4; Tables 5 and 6 are obtained by the IS approximation). From a numerical point of view, note however that there are differences between using 1-S in the non-linear model (59), or 2-S in the semi-linear one (63).

First, in (59) we apply linearization, UT or EMM to $g(x_n)=x_n^2$. Since this function is smooth, the UT or EMM approximations give good results, and consequently 1-S does not suffer from the approximations required by its practical implementation. On the other hand, in order to implement 2-S in (63) we need to apply UT or EMM to $\tilde{g}(x_n) = 0.5f_n(x_n)$, but now function f_n in (59) is not smooth, so 2-S will provide good results only if the approximations are valid. Also observe that the exact moments of $p(x_n|x_{n-1},y_n,y_{n+1})$ cannot be computed in model (63) (because $\tilde{g}(x_n)$ is not a polynomial in x_n any longer); the linearization and EMM methods will then be replaced by a one- or second-order Taylor series expansion of $\tilde{g}(x_n)$.

Let us turn to simulations. Let $R=2$ and $N=50$. For 2-S we used either a second order Taylor series expansion or UPF with parameters $\alpha = 0.73$ and $\beta = \alpha^2-1$, which actually gave better results. Table 5 displays $\mathcal{J}$ as a function of Q. As we can see, the ordering 2-P < 1-P < SIR < 1-S is

Table 5
Empirical standard deviation $\mathcal{J}$, first semi-linear model.

Q	2-P	1-P	SIR	1-S	2-S(UPF)
0.1	1.8806834	1.5315351	1.5157711	1.2715162	1.1597958
0.3	2.2708272	1.8352925	1.7089960	1.4593566	1.3207988
0.5	2.3162285	1.8401509	1.7103603	1.4729264	1.4229020
0.75	2.3338881	1.8495259	1.7552383	1.4987803	1.5487517
1	2.4440703	1.8517968	1.8399069	1.5774676	1.6391233
10	2.8704646	2.4588790	2.3428298	2.2977296	2.4482171
50	3.8532685	2.8746632	2.7053773	2.6711875	2.6882771

maintained. As expected, 2-S outperforms 1-S if Q is small, but 1-S actually performs better if Q increases. Again the reason why is that in the non-linear model f_n has strong variations; all orders actually matter, so all approximations of f_n at some point becomes very poor outside of a small neighbourhood of that point, and such situations do happen if Q gets larger.

In Fig. 5 we set $Q=10$ and we compare the algorithms as a function of N. As we see 1-S behaves better than SIR, which behaves better than 1-P, which behaves better than 2-P.

Finally let us consider the semi-linear model (63), but in which the evolution equation is replaced by $x_{n+1} = \arctan x_n + u_n$. We set $\alpha = 0.88$, $\beta = \alpha^2-1$, $N=100$ and $R=2$. Table 6 displays $\mathcal{J}$ in terms of Q. By contrast with the previous semi-linear model, the arctan function is now very smooth. As a result all algorithms give satisfactory results, especially if Q is low. Also observe that the ordering of the algorithms is maintained, and in particular that for this model 2-S always outperforms 1-S, even for large values of Q. The reason why is that for the arctan function limited order approximations are valid in a large domain, so the necessity of approximating $p(x_n|x_{n-1},y_n,y_{n+1})$ is no longer a handicap of 2-S w.r.t. 1-S.

8.4. Simulations, range bearing tracking model

We finally compare our algorithms in the range bearing tracking problem (see e.g. [33]). We track the Cartesian coordinates and velocity of a target with linear dynamics from range bearing measurements. So let $\mathbf{x}_n = [p_x(n),\dot{p}_x(n),p_y(n),\dot{p}_y(n)]^T$. The model is described as

$$\begin{cases} \mathbf{x}_{n+1} = \mathbf{F}\mathbf{x}_n+\mathbf{u}_n, \\ \mathbf{y}_n = \left[tan^{-1}\left(\dfrac{p_y(n)}{p_x(n)}\right), \sqrt{p_x(n)^2+p_y(n)^2}\right]^T + \mathbf{v}_n, \end{cases} \tag{64}$$

where $\mathbf{u}_k \sim \mathcal{N}(0,\mathbf{Q})$ and $\mathbf{v}_k \sim \mathcal{N}(0,\mathbf{R})$ are i.i.d., mutually independent and independent of $\mathbf{x}_0$, and

$$\mathbf{F} = \begin{pmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{Q} = \sigma_Q^2 \begin{pmatrix} \frac{T_s^3}{3} & \frac{T_s^2}{2} & 0 & 0 \\ \frac{T_s^2}{2} & T_s & 0 & 0 \\ 0 & 0 & \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ 0 & 0 & \frac{T_s^2}{2} & T_s \end{pmatrix},$$

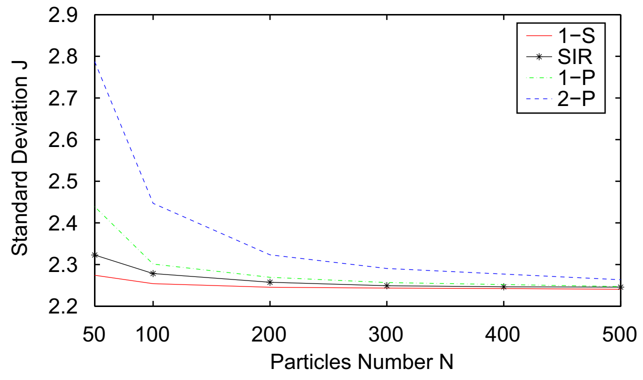

Fig. 5. Empirical standard deviation, non-linear model.

Table 6
Empirical standard deviation $\mathcal{J}$, alternate semi-linear model ($N=100$, $R=2$).

Q	2-*P*	1-*P*	SIR	1-*S*	2-*S*(UPF)
1	1.1264610	1.1222430	1.1232098	1.1185242	1.1170665
5	1.8533435	1.8397771	1.8385670	1.8262934	1.8252950
20	2.1870510	2.1645330	2.1558708	2.1451642	2.1449666
50	2.4919791	2.4292195	2.4116993	2.3984171	2.3981047

Table 7
Empirical standard deviation $\mathcal{J}(p_x)$, range-bearing tracking model.

N	2-*P*	1-*P*	SIR	1-*S*(UPF)
100	5.1673	4.4171	4.4218	4.2916
200	4.4404	3.9461	3.9160	3.7829
500	3.9941	3.5751	3.5137	3.4983

$$\mathbf{R}=\begin{pmatrix}\sigma_b^2 & 0\\ 0 & \sigma_r^2\end{pmatrix},$$

where $T_s=1$ s is the sampling period, $\sigma_Q=0.1$ is the process noise, $\sigma_b=5\pi/180$ (5°) (resp. $\sigma_r=5$ m) is the bearing (resp. range) measurements standard deviation.

For SIR and 1-*S* the exact CID $p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)$ cannot be computed exactly; as in Section 8.2 we use UPF with $\alpha=0.94$ and $\beta=0$ and $\hat{p}_{UPF}(\mathbf{y}_n|\mathbf{x}^i_{n-1})$. True state $\mathbf{x}_0$ is located at $\mathbf{x}_{init}=[5,5,-3,-3]^T$ and the first particles $\mathbf{x}^i_0$ are sampled from $p(\mathbf{x}_0)\sim\mathcal{N}(\mathbf{x}_{init},\mathbf{P}_{init})$ with $\mathbf{P}_{init}=diag([10,1,10,1])$. We use the same criterion $\mathcal{J}$ as above but for each component of the state vector. Table 7 presents results for the p_x component (similar results are obtained for the three other ones p_y, $\dot{p}_x$ and $\dot{p}_y$). The ordering of the algorithms is maintained, even if for this model $\mathcal{J}_{1-P}$ and $\mathcal{J}_{SIR}$ are close. This is because the model is almost deterministic ($\sigma_Q=0.1$). Velocity is a critical parameter here, because one single measurement does not bear information on the velocity of the target. In 1-*P* particles are sampled blindly, so after the *S* step it may happen that only very few particles bear a good estimate of the velocity parameters. This in turn will induce a bad estimation of the position parameters after the next prediction step. This phenomenon is observed in Fig. 6 where there is a gap between the true state and the state estimated by 1-*P*. This does not happen with 1-*S* because particles $\mathbf{x}^i_{n-1}$ are first guided toward the new observation, and then $\mathbf{x}^i_n$ is sampled by taking into account the new particle $\tilde{\mathbf{x}}^i_{n-1}$ and observation $\mathbf{x}_n$.

9. Conclusion

We addressed the recursive computation of the posterior pdf $p_{n|n}$ of a hidden variable $\mathbf{x}_n$ given all observations $\mathbf{y}_{0:n}$. We observed that the classical recursion ($p_{n-1|n-1}\rightarrow p_{n|n-1}\rightarrow p_{n|n}$) is not the only one that can be used for computing $p_{n|n}$ online, and we explored alternate direct, *P*- and *S*-based filters, which yields four distinct solutions. Since any of these paths can be decomposed into a propagation step and an updating one (or vice versa), we routinely derived practical algorithms by applying, for each step, either appropriate Gaussian formulas or ad hoc simulation techniques. This enabled us to enhance the fundamental role played in SMC algorithms by the sampling, weighting and resampling operations (always in that order, up to a cyclic permutation). Even though the tools that we use are not new, we do believe that the systematic and parallel derivation, for each filtering path, of both a KF and an associated generic SMC algorithm, is both original and economical. Up to our best knowledge, among the four KF (resp. SMC) algorithms that we get two are well known and two are original. Finally simulations showed that the reorganized SIR algorithm with optimal CID, i.e. the direct filter for which updating precedes propagation, behaves better than the original one; *S*-based algorithms outperform *P*-based ones, and in each class of algorithms better results are obtained (under fair conditions, i.e. when the necessary approximations are indeed valid) when updating precedes propagation.

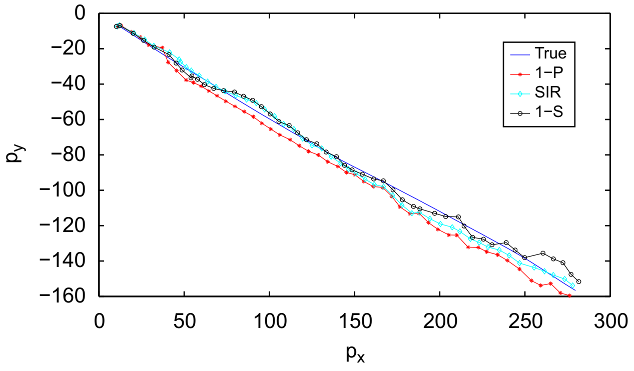

Fig. 6. Range-bearing tracking scenario—estimation of $p_y(n)$ vs. $p_x(n)$ for $N=100$.

Acknowledgement

The authors would like to thank the French MOD DGA/MRIS for financial support of the PhD of Y. Petetin.

References

[1] Y.C. Ho, R.C.K. Lee, A Bayesian approach to problems in stochastic estimation and control, IEEE Transactions on Automatic Control 9 (October) (1964) 333–339.
[2] B.D.O. Anderson, J.B. Moore, Optimal Filtering, Prentice-Hall, Englewood Cliffs, NJ, 1979.
[3] H. Tanizaki, Nonlinear Filters, Estimation and Applications, second ed., Springer, Berlin, 1996.
[4] J.S. Liu, R. Chen, Sequential Monte Carlo methods for dynamic systems, Journal of the American Statistical Association 93 (443) (1998) 1032–1044.
[5] M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Transactions on Signal Processing 50 (2) (2002) 174–188.
[6] P.M. Djuric, J.H. Kotecha, A. Jianqui Zhang, A. Yufei Huang, T. Ghirmai, M.F. Bugallo, J. Miguez, Particle filtering, IEEE Signal Processing Magazine 20 (5) (2003) 19–38.
[7] O. Cappé, É. Moulines, T. Rydén, Inference in Hidden Markov Models, Springer, 2005.
[8] N.J. Gordon, D.J. Salmond, A.F.M. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, IEE Proceedings-F 140 (2) (1993) 107–113.
[9] G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, Journal of Computational and Graphical Statistics 5 (1) (1996) 1–25.
[10] V. Zaritskii, V. Svetnik, L. Shimelevich, Monte Carlo technique in problems of optimal data processing, Automation and remote control (1975) 95–103.
[11] A. Doucet, S.J. Godsill, C. Andrieu, On sequential Monte Carlo sampling methods for Bayesian filtering, Statistics and Computing 10 (2000) 197–208.
[12] D.B. Rubin, Using the SIR algorithm to simulate posterior distributions, in: M.H. Bernardo, K.M. Degroot, D.V. Lindley, A.F.M. Smith (Eds.), Bayesian Statistics III, Oxford University Press, Oxford, 1988.
[13] A.E. Gelfand, A.F.M. Smith, Sampling based approaches to calculating marginal densities, Journal of the American Statistical Association 85 (410) (1990) 398–409.
[14] A.F.M. Smith, A.E. Gelfand, Bayesian statistics without tears: a sampling-resampling perspective, The American Statistician 46 (2) (1992) 84–87.
[15] B. Efron, Bootstrap methods: another look at the jacknife, Annals of Statistics 7 (1) (1979) 1–26.
[16] B. Efron, R.J. Tibshirani, An introduction to the bootstrap, Monographs on Statistics and Applied Probability, vol. 57, Chapman & Hall, New York, 1993.
[17] P. Barbe, P. Bertail, The weighted bootstrap, Lecture Notes in Statistics, vol. 98, Springer Verlag, New York, 1995.
[18] R. Douc, O. Cappé, É. Moulines, Comparison of resampling schemes for particle filtering, in: Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA), 2005.
[19] J.D. Hol, T.B. Schön, F. Gustafsson, On resampling algorithms for particle filtering, in: Proceedings of the IEEE Nonlinear Statistical Signal Processing Workshop (NSSPW'06), Cambridge, UK, September 13–15, 2006.
[20] R.E. Helmick, W.D. Blair, S.A. Hoffman, One-step fixed-lag smoothers for Markovian switching systems, IEEE Transactions on Automatic Control 41 (7) (1996) 1051–1056.
[21] T. Kailath, A.H. Sayed, B. Hassibi, Linear Estimation, Prentice-Hall, 2000.
[22] B. Ait-el-Fquih, F. Desbouvries, On Bayesian fixed-interval smoothing algorithms, IEEE Transactions on Automatic Control 53 (10) (2008) 2437–2442.
[23] M.K. Pitt, N. Shephard, Filtering via simulation: auxiliary particle filter, Journal of the American Statistical Association 94 (446) (1999) 550–599.
[24] P. Fearnhead, Computational methods for complex stochastic systems: a review of some alternatives to MCMC, Statistics and Computing 18 (2) (2008) 151–171.
[25] R. Douc, É. Moulines, J. Olsson, Optimality of the auxiliary particle filter, Probability and Mathematical Statistics 29 (1) (2009) 1–28.
[26] N. Whiteley, A.M. Johansen, Recent developments in auxiliary particle filtering, in: Barber, Cemgil, Chiappa (Eds.), Inference and Learning in Dynamic Models, Cambridge University Press, 2010 to appearto appear. Q2 Q1
[27] R. van der Merwe, A. Doucet, N. de Freitas, E. Wan, The unscented particle filter, in: Advances in Neural Information Processing Systems, 2000.
[28] A. Doucet, N.J. Gordon, V. Krishnamurthy, Particle Filters for state estimation of jump Markov linear systems, Technical Report, Cambridge University Engineering Department, 1999.
[29] S. Saha, P.K. Manda, Y. Boers, H. Driessen, A. Bagchi, Gaussian proposal density using moment matching in SMC methods, Statistics and Computing 19 (2) (2009).
[30] S. Julier, J. Uhlmann, Unscented filtering and nonlinear estimation, Proceedings of the IEEE 92 (3) (2004) 401–422.
[31] P. Stavropoulos, D.M. Titterington, Improved particle filters and smoothing. In: A. Doucet, N. de Freitas, N. Gordon (Eds.), Sequential Monte-Carlo Methods in Practice, Springer Verlag, 2001 (Chapter 14).
[32] B.W. Silverman, G.A. Young, The bootstrap: to smooth or not to smooth?, Biometrika 74 (3) (1987) 469–479
[33] B. Ristic, S. Arulampalam, N. Gordon, Beyond the Kalman Filter: Particle Filters for Tracking Applications, Artech House, Boston London, 2004.