DIRECT VERSUS PREDICTION-BASED PARTICLE FILTER ALGORITHMS

François Desbouvries and Boujemaa Ait-El-Fquih

TELECOM & Management Sudparis / Dépt. CITI and CNRS UMR 5157
9 rue Charles Fourier, 91011 Evry, France
Francois.Desbouvries@it-sudparis.eu

ABSTRACT

Particle Filtering (PF) algorithms propagate in time a Monte Carlo (MC) approximation of the a posteriori filtering measure in a Hidden Markov Chain (HMC) model. In this paper we first shed some new light on two classical PF algorithms, which can be considered as natural MC implementations of two two-step direct recursive formulas for computing the filtering distribution. We next address the Particle Prediction (PP) problem, which happens to be simpler than the PF problem because the optimal prediction conditional importance distribution (CID) is much easier to sample from. Motivated by this result we finally develop two PP-based PF algorithms, and we compare our algorithms via simulations.

Index Terms— Particle Filtering, Sequential Importance Sampling, Sampling Importance Resampling, Optimal importance function, Hidden Markov Chains.

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 probability density function (pdf) of $\mathbf{x}_n$ given $\mathbf{y}_{0:n} = \{\mathbf{y}_i\}_{i=0}^n$. Let $p(d\mathbf{x}) = p(\mathbf{x})d\mathbf{x}$ be the continuous measure with density $p(\mathbf{x})$. We assume that $\{\mathbf{x}_n, \mathbf{y}_n\}$ is an HMC :

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

We deal with the Bayesian filtering problem, which consists in computing $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ at each time instant n. From (1)

$$p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = \frac{p(\mathbf{x}_n, \mathbf{y}_n|\mathbf{x}_{n-1})}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1})} p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}), \quad (2)$$

in which $p(\mathbf{x}_n, \mathbf{y}_n|\mathbf{x}_{n-1})$ can be factorized as

$$\begin{aligned} p(\mathbf{x}_n, \mathbf{y}_n|\mathbf{x}_{n-1}) &= p(\mathbf{y}_n|\mathbf{x}_n) p(\mathbf{x}_n|\mathbf{x}_{n-1}) \quad (3) \\ &= p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{y}_n) p(\mathbf{y}_n|\mathbf{x}_{n-1}). \quad (4) \end{aligned}$$

Injecting (3) into (2), and taking the marginal, we see that $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ can be computed recursively by

$$\begin{aligned} p(\mathbf{x}_n|\mathbf{y}_{0:n-1}) &= \int p(\mathbf{x}_n|\mathbf{x}_{n-1}) p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1}), \quad (5) \\ p(\mathbf{x}_n|\mathbf{y}_{0:n}) &= \frac{p(\mathbf{y}_n|\mathbf{x}_n) p(\mathbf{x}_n|\mathbf{y}_{0:n-1})}{\int p(\mathbf{y}_n|\mathbf{x}_n) p(d\mathbf{x}_n|\mathbf{y}_{0:n-1})}. \quad (6) \end{aligned}$$

It remains to compute $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ in practice. In the linear Gaussian case, exact recursive solutions are provided by Kalman filtering (KF) techniques. In the general case however, computing either $p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n})$ or $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ is either difficult or impossible, so approximate techniques have been developed. Among them, PF (see e.g. [1] [2] [3] [4] and references therein) are sequential MC (SMC) methods which aim at computing a discrete approximation of $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$.

More precisely, many efforts have been devoted to the MC approximation of the global (resp. local) recursive formula (2) (resp. (5)-(6)). At this point, let us notice however that (5)-(6), say, is one recursive formula for $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ which explicitely computes this density as the recursion $p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1}) \rightarrow p(\mathbf{x}_n|\mathbf{y}_{0:n-1}) \rightarrow p(\mathbf{x}_n|\mathbf{y}_{0:n})$ (or, in short, $p_{n-1|n-1} \rightarrow p_{n|n-1} \rightarrow p_{n|n}$); so computing $p_{n|n}$ from $p_{0|0}$ via (5)-(6) amounts to walking along one particular path, made of the succession of a predictive step and an updating step. However following this path is not the only possible way to compute $p_{n|n}$. In other words, we should not necessarily try to get the best possible approximation of (5)-(6); what we really look for is a good approximation of $p_{n|n}$, no matter how it is obtained, even if $p_{n|n}$ is obtained indirectly, as the byproduct of some recursive algorithm. These ideas are well known in the context of KF, but not in PF, so the aim of the present paper is to investigate them in the context of SMC techniques.

This paper is organized as follows. In §2 we recall PF. Next in §3 we revisit two classical PF algorithms (the Bootstrap filter [1], and a (reorganized) SIR algorithm with optimal CID [5]) from two different points of view. First, both algorithms can be seen as direct MC implementations of (5)-(6), which differ only by the instant when we sample from an approximate continuous distribution in order to get a discrete one; next, they can be seen as natural MC implementations of the recursive loops $p_{n-1|n-1} \rightarrow p_{n|n-1} \rightarrow p_{n|n}$

and $p_{n-1|n-1} \rightarrow p_{n-1|n} \rightarrow p_{n|n}$ which, in turn, directly stem from the two Bayes factorizations (3) and (4) of the transition pdf $p(\mathbf{x}_n, \mathbf{y}_n|\mathbf{x}_{n-1})$ in (2). Now a drawback of direct PF is that the optimal CID is often difficult to compute or to sample from. So in §4 we address the PP problem, which happens to be easier than the PF problem because in the prediction case the optimal CID coincides with the prior transition $p(\mathbf{x}_n|\mathbf{x}_{n-1})$ of the Markov chain $\mathbf{x}_n$, which is often straightforward to sample from. Motivated by this observation we address PP-based PF, in which the distribution of interest $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ is computed indirectly, *via* a PP algorithm (i.e., the recursive loop is now on $p(\mathbf{x}_n|\mathbf{y}_{0:n-1})$). Simulations are performed in §5, and we end the paper with some concluding remarks.

2. THE PF METHODOLOGY

2.1. The generic PF algorithm

Let us recall the principle of PF [2] [3] which is based on importance sampling (IS). Assume that at time $n-1$ we have a discrete measure which approximates $p(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$:

$$p(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) \simeq \sum_{i=1}^{N} w_{n-1}^i \delta_{\mathbf{x}_{0:n-1}^i}(d\mathbf{x}_{0:n-1}) \ , \quad (7)$$

where δ_x 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 $w_{n-1}^i \propto p(\mathbf{x}_{0:n-1}^i|\mathbf{y}_{0:n-1})/q(\mathbf{x}_{0:n-1}^i|\mathbf{y}_{0:n-1})$, $\sum_{i=1}^N w_{n-1}^i = 1$. Let us see how to compute (7) recursively. If we assume that the importance distribution factorizes as

$$q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = q(\mathbf{x}_n|\mathbf{x}_{0:n-1}, \mathbf{y}_{0:n}) q(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}), \quad (8)$$

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 $1 \leq i \leq N$, $[\mathbf{x}_{0:n}^i] = [\mathbf{x}_{0:n-1}^i, \mathbf{x}_n^i]$, in which $\mathbf{x}_n^i$ is sampled from the CID $q(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i, \mathbf{y}_{0:n})$. As for the weights w_n^i, we see from (2) that they can be computed recursively as

$$w_n^i \propto \underbrace{\frac{p(\mathbf{x}_n^i, \mathbf{y}_n|\mathbf{x}_{n-1}^i)}{q(\mathbf{x}_n^i|\mathbf{x}_{0:n-1}^i, \mathbf{y}_{0:n})}}_{\lambda_n^i} \times \underbrace{\frac{p(\mathbf{x}_{0:n-1}^i|\mathbf{y}_{0:n-1})}{q(\mathbf{x}_{0:n-1}^i|\mathbf{y}_{0:n-1})}}_{\propto w_{n-1}^i} . \quad (9)$$

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

2.2. Practical considerations

Now, sequential IS (SIS) algorithms are well known to suffer from weights degeneracy. It has thus proved important in the above generic algorithm to resample from $\sum_{i=1}^N w_n^i \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$ (either systematically or according to some strategy), and also to choose the CID $q(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i, \mathbf{y}_{0:n})$ carefully.

To that respect, sampling from the *a priori* transition kernel of the Markov chain $\mathbf{x}$ (i.e., choosing $q(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i, \mathbf{y}_{0:n}) = p(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i) = p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$, which was the original choice in the so-called Bootstrap filter [1]) is popular because sampling from $p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$ is often straightforward. Moreover, computing the incremental weight λ_n^i in (9) reduces to evaluating the conditional likelihood of the new observation given the updated particle position.

However, this choice can lead to poor performances, and it is often preferable to sample from the optimal CID $q^{opt}(\mathbf{x}_n|\mathbf{x}_{0:n-1}^i, \mathbf{y}_{0:n}) = p(\mathbf{x}_n|\mathbf{x}_{n-1}^i, \mathbf{y}_n)$ [5], i.e. the distribution which minimizes the variance of the importance weights w_n^i, conditionally on the observations $\mathbf{y}_{0:n}$ and past samples $\mathbf{x}_{0:n-1}^i$. In this case $\lambda_n^i = p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)$.

3. REVISITING SOME DIRECT PF ALGORITHMS

In this section, we shall revisit some classical PF algorithms (the Bootstrap filter and the PF with optimal CID evoked above) from two different points of view (see §3.2 and 3.3). But it will prove useful to first recall the 3 simulation techniques that are used whenever we deal with PF algorithms.

3.1. Three generic simulation techniques

In all PF algorithms we sample (S) new particles, update some weights (W) and (possibly) resample (R) from a weighted discrete measure. These three SMC elementary operations S, W and R are indeed related to the following three simulation techniques.

1. (S *step*). Let $x_i \sim p(x)$ and let us sample y_i from $p(y|x_i)$. Then (x_i, y_i) is a sample from $p(x, y)$ and thus y_i is a sample from $p(y)$.

2. ((R, S) *steps*). Let $p(y)$ be the mixture $p(y) = \sum_{i=1}^m \alpha_i \, p_i(y)$ with $\alpha_i > 0$ and $\sum \alpha_i = 1$. Let j be sampled from $p(x) = \sum_{i=1}^m \alpha_i \delta_i(dx)$ and next y be sampled from $p_j(y)$. Then y is a sample from $p(y)$.

3. ((W, R) *steps*). Let $\tilde{p}(dx) = \sum_{i=1}^N \frac{p(x^i)/q(x^i)}{\sum_{i=1}^N p(x^i)/q(x^i)} \delta_{x^i}(dx)$ in which x_i are i.i.d. samples from q. Conditionnally on $\{x_i\}$, let $\{\tilde{x}^i\}_{i=1}^M$ be i.i.d. samples from $\tilde{p}$; then $\{\tilde{x}^i\}_{i=1}^M$ become i.i.d. samples from p if $N \rightarrow \infty$.

3.2. Sampling before or after updating ?

Eqs. (5)-(6) transform $p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$ into $p(\mathbf{x}_n|\mathbf{y}_{0:n})$ or, equivalently, $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$ into $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$. If exact computing is impossible, 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 (5)-(6). However, the resulting approximate measure is continuous, so sampling is needed at

some point of the process if we want to further proceed at time $n+1$ and finally get an SMC algorithm. Now, a natural question is *when* should one sample the new particles; we shall now revisit from this point of view the Bootstrap algorithm [1], and the SIR algorithm with optimal CID [5].

3.2.1. Sampling before updating.

In this option we implement (5)-(6), with a sampling step positioned after (5). Let us thus assume that at time $n-1$ we have a discrete random approximation of $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$:

$$\widehat{p}(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^{N} w_{n-1}^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1}) \ . \qquad (10)$$

Injecting into (5) we get an approximation $\hat{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n-1})$ (superscript c stands for continuous) of $p(\mathbf{x}_n|\mathbf{y}_{0:n-1})$:

$$\hat{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n-1}) \quad = \quad \sum_{i=1}^{N} w_{n-1}^i p(\mathbf{x}_n|\mathbf{x}_{n-1}^i). \qquad (11)$$

Since we sample at the end of the prediction step we need to get N samples $\{\mathbf{x}_n^i\}_{i=1}^N$ from the mixture pdf (11). To that end we can use §3.1 (point 2), i.e. first sample N points $\tilde{\mathbf{x}}_{n-1}^i$ 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 $p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i)$. At the end of this two-step sampling mechanism, $\widehat{p}(d\mathbf{x}_n|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N \frac{1}{N} \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$ is a discrete approximation of $p(d\mathbf{x}_n|\ \mathbf{y}_{0:n-1})$, and from (6)

$$\widehat{p}(d\mathbf{x}_n|\mathbf{y}_{0:n}) \stackrel{def}{=} \sum_{i=1}^{N} \underbrace{\frac{p(\mathbf{y}_n|\mathbf{x}_n^i)}{\sum_{i=1}^N p(\mathbf{y}_n|\mathbf{x}_n^i)}}_{w_n^i} \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n). \qquad (12)$$

Let us summarize the

Algorithm 1.

Let $\widehat{p}(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N w_{n-1}^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$ approximate $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$.

R. For $1 \le i \le N$, sample $\tilde{\mathbf{x}}_{n-1}^i$ from $\sum_{i=1}^N w_{n-1}^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$;

S. For $1 \le i \le N$, sample $\mathbf{x}_n^i$ from $p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i)$;

W. For $1 \le i \le N$, compute $\tilde{w}_n^i = p(\mathbf{y}_n|\mathbf{x}_n^i)$.

Compute $w_n^i \propto \tilde{w}_n^i$, $\sum_{i=1}^N w_n^i = 1$. Then $\widehat{p}(d\mathbf{x}_n|\mathbf{y}_{0:n}) = \sum_{i=1}^N w_n^i \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$ approximates $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$.

Up to a shift, the above algorithm (R, S, W) is thus nothing but the well-known Bootstrap algorithm [1], which consists of the successive steps $S \to W \to R$. In particular, from this point of view the famous *resampling* (or *selection*) step R of the Bootstrap algorithm is indeed nothing but the first step of the composition method used when sampling from the mixture $\sum_{i=1}^N w_{n-1}^i p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$ (see §3.1, point 2).

3.2.2. Sampling after updating.

Let us now choose the option of sampling after the end of the prediction and updating steps. Starting from (11) again, and injecting (11) into (6) we get

$$\hat{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n}) \quad = \quad \frac{\sum_{i=1}^N w_{n-1}^i p(\mathbf{y}_n|\mathbf{x}_n) p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)}{\sum_{i=1}^N w_{n-1}^i \int p(\mathbf{y}_n|\mathbf{x}_n) p(\mathbf{x}_n|\mathbf{x}_{n-1}^i) d\mathbf{x}_n},$$

which, due to (4), can be rewritten as the mixture

$$\hat{p}^c(\mathbf{x}_n|\mathbf{y}_{0:n}) = \sum_{i=1}^{N} \frac{w_{n-1}^i p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)}{\sum_{i=1}^N w_{n-1}^i p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)} p(\mathbf{x}_n|\mathbf{x}_{n-1}^i, \mathbf{y}_n),$$

which one should finally sample from. Using again (§3.1, point 2) leads to the following algorithm, which coincides with [4, Algorithm 8.1.1 p. 253] :

Algorithm 2.

Let $\widehat{p}(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N \frac{1}{N} \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$ approximate $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$.

W. For $1 \le i \le N$, compute $w_n^i \propto p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)$.

R. For $1 \le i \le N$, sample $\tilde{\mathbf{x}}_{n-1}^i \sim \sum_{i=1}^N w_{n-1}^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$;

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

Comparing with §2.2, we see that Algorithm 2 is a reordering of the SIR PF algorithm with optimal CID (and with systematic resampling) [5]. However, these two algorithms are not simply related by a shift in time, as were Algorithm 1 and the Bootstrap algorithm above. More precisely, in [5] the successive steps are $S \to W \to R$ (or $W \to S \to R$, because the S and W steps commute), while the recursive loop of Algorithm 2 is made of the successive steps $W \to R \to S$. So if there is resampling, $\mathbf{x}_n^i$ is sampled from $p(\mathbf{x}_n|\mathbf{x}_{n-1}^i, \mathbf{y}_n)$ in [5], while in Algorithm 2 $\mathbf{x}_n^i \sim p(\mathbf{x}_n|\tilde{\mathbf{x}}_{n-1}^i, \mathbf{y}_n)$ in which $\tilde{\mathbf{x}}_{n-1}^i \sim \sum_{i=1}^N w_{n-1}^i \delta_{\mathbf{x}_{n-1}^i}(d\mathbf{x}_{n-1})$; in others words, each old particle $\mathbf{x}_{n-1}^i$ is taken equally into account in [5], while only those with high weights do contribute to the updated trajectory in Algorithm 2.

3.3. The role of the transition pdf $p(\mathbf{x}_n, \mathbf{y}_n|\mathbf{x}_{n-1})$

As we have already seen, the two Bayes factorizations (3), (4) of $p(\mathbf{x}_n, \mathbf{y}_n|\mathbf{x}_{n-1})$ play an important role. Injecting them into (2) respectively gives (here N stands for numerator) :

$$p(\mathbf{x}_{n-1}, \mathbf{x}_n|\mathbf{y}_{0:n}) \stackrel{(3)}{=} \frac{p(\mathbf{y}_n|\mathbf{x}_n) \overbrace{[p(\mathbf{x}_n|\mathbf{x}_{n-1}) p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})]}^{p(\mathbf{x}_{n-1}, \mathbf{x}_n|\mathbf{y}_{0:n-1})}}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1}) = \int N d\mathbf{x}_{n-1} d\mathbf{x}_n} \qquad (13)$$

$$\stackrel{(4)}{=} p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)\,\underbrace{[\frac{p(\mathbf{y}_n|\mathbf{x}_{n-1})p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1})=\int N d\mathbf{x}_{n-1}}]}_{p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n})}. \quad (14)$$

Both formulas enable to compute $p_{n|n}$ from $p_{n-1|n-1}$; (13) uses the path $p_{n-1|n-1} \rightarrow p_{n|n-1} \rightarrow p_{n|n}$, and (14) the path $p_{n-1|n-1} \rightarrow p_{n-1|n} \rightarrow p_{n|n}$. So in the first solution we predict $\mathbf{x}_n$, based on the same data, and then update $p_{n|n-1}$ thanks to $\mathbf{y}_n$; while in the second path we update first, and next predict the state $\mathbf{x}_n$.

Let us turn to MC approximations of these exact formulas. We first consider (13), and begin with the formula in brackets. We assume that at $n-1$ we have N samples from $p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$. Using (§3.1, point 1) we get N samples $\mathbf{x}_n^i$ from $p(\mathbf{x}_n|\mathbf{y}_{0:n-1})$ by drawing $\mathbf{x}_n^i$ from $p(\mathbf{x}_n|\mathbf{x}_{n-1}^i)$. Next using (§3.1, point 3), 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 (12). We indeed just described the sampling step S, followed by the updating step (W,R) of the Bootstrap algorithm.

Let us now address (14), beginning by the equation in brackets. Assume again that at $n-1$ we have N samples from $p(\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$. From the SIR mechanism described in (§3.1, point 3), 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 (§3.1, 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 indeed just described the updating step (W,R) of Algorithm 2, followed by its sampling step S, i.e. the reorganized SIR algorithm with optimal CID.

3.4. SIR with optimal CID : original or reorganized ?

We now come back on the differences between Algorithm 2 and the original SIR algorithm with optimal CID. Although both algorithms coincide in the absence of resampling, and both rely on IS techniques, they do not stem exactly from the same rationale, as we now explain.

Let us begin with the SIR algorithm. At fixed time $n-1$, we would like to compensate the missing pdf $p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$ by multiple imputations from $p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})$, and next replace the corresponding expectations accordingly. Since it is easier to sample from q, we indeed dispose of the weighted measure (7). Now, due to condition (8) this IS algorithm can be sequentialized easily : each old trajectory $\mathbf{x}_{0:n-1}^i$ is kept unaltered and is simply extended by one new particle $\mathbf{x}_n^i$. But a drawback of this simple computational scheme is that in the absence of resampling, the trajectories do not fully take into account the data : the new datum $\mathbf{y}_n$ can only modify (at least if the optimum CID is used) the present and future values of the trajectories, but not the past ones. So the original algorithm is, by nature, a batch IS algorithm which has been rendered adaptive, but in which the trajectories, in the absence of resampling, tend to diverge (as time increases) from trajectories sampled from the target density $p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n})$.

By contrast, Algorithm 2 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 $n-1$, $\hat{p}(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N \frac{1}{N}\delta_{\mathbf{x}_{0:n-1}^i}(d\mathbf{x}_{0:n-1})$, 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 n we need to sample from $q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) = p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n}) =$

$$\underbrace{p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)}_{\substack{p(\mathbf{x}_n|\mathbf{x}_{n-1},\mathbf{y}_n)\\ (\S 3.1,\ point\ 1)}} \times \underbrace{[\frac{p(\mathbf{y}_n|\mathbf{x}_{n-1})}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1})} p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1})]}_{\substack{p(\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n})\\ (\S 3.1,\ point\ 3)}}. \quad (15)$$

For this density the sufficient condition (8) is *not* satisfied (because of the multiplicative factor within 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 §3.1, point 3), by updating the weights first we step from $\hat{p}(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n-1}) = \sum_{i=1}^N \frac{1}{N}\ \delta_{\mathbf{x}_{0:n-1}^i}(d\mathbf{x}_{0:n-1})$ to $\hat{p}(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n}) = \sum_{i=1}^N \frac{p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)}{\sum_{i=1}^N p(\mathbf{y}_n|\mathbf{x}_{n-1}^i)}\ \delta_{\mathbf{x}_{0:n-1}^i}(d\mathbf{x}_{0:n-1})$, and next by resampling we get (approximate) trajectories from $p(d\mathbf{x}_{0:n-1}|\mathbf{y}_{0:n})$. Comparing with SIR algorithms, in this scheme we reweight all the trajectories first and then reselect them according to their weights. We finally sample a new particle $\mathbf{x}_n^i$ by using (§3.1, point 1).

Finally in SIR algorithms the batch estimates are fundamentally based on IS, but due to condition (8) no Bayes step is needed when updating the trajectories, and so no further IS mechanism is introduced. In Algorithm 2 the batch estimate is not based on IS, but updating trajectories requires a Bayes step and a Markovian step (see (15)), so IS is introduced locally as a means to implement the Bayes mechanism.

4. PREDICTION-BASED PF ALGORITHM

In §2 and 3 we focused on direct PF algorithms, i.e. on approximate ways of computing $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$ directly from $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$. Even though the process is very natural, it leads to the difficulty that the optimal CID is difficult to compute in practice. So many efforts have been expended in order to approximate this CID (see e.g. [5] [4]). In this section we will try to bypass this difficulty by computing $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$ indirectly, *via* the prediction distribution $p(d\mathbf{x}_n|\mathbf{y}_{0:n-1})$. So we shall first deal with the generic PP problem (see §4.1) and we shall next address the connection with the PF problem (see §4.2).

4.1. The generic PP problem

Similarly to §2.1, particle *prediction* can be set as follows. From (1) we get

$$p(\mathbf{x}_{0:n+1}|\mathbf{y}_{0:n}) = \frac{p(\mathbf{x}_{n+1}, \mathbf{y}_n|\mathbf{x}_n)}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1})} p(\mathbf{x}_{0:n}|\mathbf{y}_{0:n-1}). \quad (16)$$

Let us thus start from $p(d\mathbf{x}_{0:n}|\mathbf{y}_{0:n-1}) \simeq \sum_{i=1}^N w_{n-1}^i \delta_{\mathbf{x}_{0:n}^i}(d\mathbf{x}_{0:n})$, in which samples $\mathbf{x}_{0:n}^i$ are generated from $q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n-1})$, and the prediction importance weights w_{n-1}^i now satisfy $w_{n-1}^i \propto p(\mathbf{x}_{0:n}^i|\mathbf{y}_{0:n-1}) / q(\mathbf{x}_{0:n}^i|\mathbf{y}_{0:n-1})$. Let us assume that

$$q(\mathbf{x}_{0:n+1}|\mathbf{y}_{0:n}) = q(\mathbf{x}_{n+1}|\mathbf{x}_{0:n}, \mathbf{y}_{0:n}) q(\mathbf{x}_{0:n}|\mathbf{y}_{0:n-1}).$$

As for the updating of the importance weights, (9) becomes

$$w_n^i \propto \underbrace{\frac{p(\mathbf{x}_{n+1}^i, \mathbf{y}_n|\mathbf{x}_n^i)}{q(\mathbf{x}_{n+1}^i|\mathbf{x}_{0:n}^i, \mathbf{y}_{0:n})}}_{\lambda_n^i} \times \underbrace{\frac{p(\mathbf{x}_{0:n}^i|\mathbf{y}_{0:n-1})}{q(\mathbf{x}_{0:n}^i|\mathbf{y}_{0:n-1})}}_{\propto w_{n-1}^i}. \quad (17)$$

Finally $\sum_{i=1}^N w_n^i \delta_{\mathbf{x}_{0:n+1}^i}(d\mathbf{x}_{0:n+1})$ approximates $p(d\mathbf{x}_{0:n+1}|\mathbf{y}_{0:n})$, and thus $p(d\mathbf{x}_{n+1}|\mathbf{y}_{0:n}) \simeq \sum_{i=1}^N w_n^i \delta_{\mathbf{x}_{n+1}^i}(d\mathbf{x}_{n+1})$.

Similarly to §2.2, one should also choose $q(\mathbf{x}_{n+1}|\mathbf{x}_{0:n}, \mathbf{y}_{0:n})$ properly. It is easy to show [6] that the CID $q(\mathbf{x}_{n+1}|\mathbf{x}_{0:n}^i, \mathbf{y}_{0:n}) = p(\mathbf{x}_{n+1}|\mathbf{x}_n^i)$ minimizes the variance of weight w_n^i conditionally on $\mathbf{x}_{0:n}^i$ and $\mathbf{y}_{0:n}$. So in the PP problem, the optimal CID coincides with the prior transition pdf of the Markov chain $\mathbf{x}$, from which it is often easy to sample from. Computing the incremental weight is also much easier that in PF, since λ_n^i now reduces to $p(\mathbf{y}_n|\mathbf{x}_n)$, which is nothing but the HMC elementary observational transition in (1). Stated otherwise, we cumulate the advantages of choosing both the prior (for practical issues) and the posterior (for optimality results) CIDs, since in the PP problem both transitions coincide.

4.2. Prediction based PF algorithms

In §3.3, we have seen that the two factorizations of $p(\mathbf{x}_n, \mathbf{y}_n|\mathbf{x}_{n-1})$ obtained by Bayes 's rule led to two different formulas (see (13) and (14)) which indeed expressed the two two-step paths of the elementary recursion $p_{n-1|n-1} \rightarrow p_{n|n}$ which are obtained when moving one index first and then the other (i.e., one can choose to first update pdf $p_{n-1|n-1}$, and then propagate the state variable, or vice versa).

For PP the role played by the elementary transition $p(\mathbf{x}_n, \mathbf{y}_n|\mathbf{x}_{n-1})$ in PF is now played by $p(\mathbf{x}_{n+1}, \mathbf{y}_n|\mathbf{x}_n)$ which, due to Bayes's rule, can be factorized as :

$$\begin{aligned} p(\mathbf{x}_{n+1}, \mathbf{y}_n|\mathbf{x}_n) &= p(\mathbf{y}_n|\mathbf{x}_n) p(\mathbf{x}_{n+1}|\mathbf{x}_n) \quad (18) \\ &= p(\mathbf{x}_{n+1}|\mathbf{x}_n) p(\mathbf{y}_n|\mathbf{x}_n). \quad (19) \end{aligned}$$

So the two factorizations coincide if we commute the factors. However we should not, because injecting (18) and (19) in (16) leads to the following equations :

$$p(\mathbf{x}_n, \mathbf{x}_{n+1}|\mathbf{y}_{0:n}) \overset{(19)}{=} p(\mathbf{x}_{n+1}|\mathbf{x}_n) \frac{\overbrace{p(\mathbf{y}_n|\mathbf{x}_n) p(\mathbf{x}_n|\mathbf{y}_{0:n-1})}^{p(\mathbf{x}_n|\mathbf{y}_{0:n})}}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1}) = \int N d\mathbf{x}_n} \quad (20)$$

$$\overset{(18)}{=} \frac{p(\mathbf{y}_n|\mathbf{x}_n) \overbrace{[p(\mathbf{x}_{n+1}|\mathbf{x}_n) p(\mathbf{x}_n|\mathbf{y}_{0:n-1})]}^{p(\mathbf{x}_n, \mathbf{x}_{n+1}|\mathbf{y}_{0:n-1})}}{p(\mathbf{y}_n|\mathbf{y}_{0:n-1}) = \int N d\mathbf{x}_n d\mathbf{x}_{n+1}} \quad (21)$$

So, as above, (20) and (21) describe the two different ways of going from $p_{n|n-1}$ to $p_{n+1|n}$ which are obtained when incrementing one time index at a time. Let us now consider MC implementations of these formulas.

The first one is the path $p_{n|n-1} \rightarrow p_{n|n} \rightarrow p_{n+1|n}$: in (20) $p_{n|n-1}$ is first updated to $p_{n|n}$, and then we step to the next prediction pdf $p_{n+1|n}$. A direct MC implementation of (20) results in the steps W, R and S of Algorithm 1 (see §3.2.1). So this algorithm happens to be a (time shifted) version of the Bootstrap algorithm. At this point, it is interesting to observe that the Bootstrap algorithm, with its prior CID, is "optimal" (in the sense of the CID) for the PP problem.

Taking the marginals w.r.t. $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$, we see that (21) is a compact way of describing the paths $p_{n|n-1} \rightarrow p_{n+1|n-1} \rightarrow p_{n+1|n}$ (which is the recursive loop of the algorithm) and $p_{n|n-1} \rightarrow p_{n|n}$ (already obtained in (6)). This solution is well known in KF (see e.g. [7, Thm 9.5.1]). Using §3.1 again, a direct MC implementation of this algorithm is given by the following algorithm (first obtained in [6]) :

Algorithm 3. Let $p(d\mathbf{x}_n|\mathbf{y}_{0:n-1}) \simeq \sum_{i=1}^N w_n^i \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$.

Prediction.

- For $i = 1, \cdots, N$, sample $\mathbf{x}_{n+1}^i \sim p(\mathbf{x}_{n+1}|\mathbf{x}_n^i)$;
- For $i = 1, \cdots, N$, compute $w_{n+1}^i \propto p(\mathbf{y}_n|\mathbf{x}_n^i) \times w_n^i$;
- Resample (if necessary) from $\sum_{i=1}^N w_{n+1}^i \delta_{\mathbf{x}_{n+1}^i}(d\mathbf{x}_{n+1})$;
- Then $p(d\mathbf{x}_{n+1}|\mathbf{y}_{0:n}) \simeq \sum_{i=1}^N w_{n+1}^i \delta_{\mathbf{x}_{n+1}^i}(d\mathbf{x}_{n+1})$.

Filtering.

- For $i = 1, \cdots, N$, set $\kappa_n^i = w_{n+1}^i$;
- Then $p(d\mathbf{x}_n|\mathbf{y}_{0:n}) \simeq \sum_{i=1}^N \kappa_n^i \delta_{\mathbf{x}_n^i}(d\mathbf{x}_n)$.

5. SIMULATIONS

In the previous sections we considered direct MC implementations of some recursive formulas for computing $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$, and finally we described four algorithms (two direct PF algorithms, and two PP-based PF algorithms) :

1. The Bootstrap filter (see Algorithm 1), which corresponds to the path $p_{n-1|n-1} \rightarrow p_{n|n-1} \rightarrow p_{n|n}$;

2. The reorganized SIR algorithm with optimal CID (see Algorithm 2), which corresponds to $p_{n-1|n-1} \rightarrow p_{n-1|n} \rightarrow p_{n|n}$;

3. The algorithm which implements $p_{n|n-1} \rightarrow p_{n|n} \rightarrow p_{n+1|n}$; since it is nothing but the time-shifted Bootstrap, we shall not differentiate it from Algorithm 1;

4. And finally Algorithm 3, which corresponds to the path
$$\begin{array}{ccccc} p_{n|n-1} & \rightarrow & p_{n+1|n-1} & \rightarrow & p_{n+1|n} \\ \downarrow & & & & \\ p_{n|n} & & & & \end{array}.$$

In this section we perform some simulations. Let

$$\begin{aligned} x_{n+1} &= 0.5x_n + \frac{25x_n}{(1+x_n^2)} + 8\cos(1.2(n+1)) + u_n, \\ y_n &= x_n + v_n. \end{aligned}$$

We set $x_0 \sim \mathcal{N}(0,1)$ and $u_n \sim \mathcal{N}(0,10)$. Let j be the realization and n the running time. We set $1 \leq j \leq P$ with $P = 50$ and $0 \leq n \leq M$ with $M = 40$. We first set $N = 300$ particles, and we plot the empirical standard deviation $\mathcal{J}$, defined as

$$\mathcal{J} = \frac{1}{M+1} \sum_{n=0}^{M} \left(\frac{1}{P} \sum_{j=1}^{P} (\hat{x}_{n|n}^j - x_n^j)^2\right)^{1/2} \qquad (22)$$

as a function of $var(v_n)$. The results are displayed in Table 1. We next set $var(v_n) = 5$, and we display $\mathcal{J}$, as a function of the number of particles N (see Table 2).

As far as PP-PF algorithms are concerned, the Bootstrap algorithm always outperforms Algorithm 3, probably because data y_n is taken into account before we proceed to the sampling step. On the other hand, the relative performance of the Bootstrap w.r.t. Algorithm 2 depends on $var(v_n)$, i.e. on the shape of the likelihood; as is well known, the Bootstrap is expected not to give good results if the likelihood is very sharp, but on the other hand we can see that there are situations in which the Bootstrap gives better results than Algorithm 2. Finally the results of Table 2 confirm that for all algorithms $\mathcal{J}$ decreases with the number of particles.

$var(v_n)$	Alg. 3	Alg. 1	Alg. 2
.3	1.0686	0.5680	0.5473
3	1.5767	1.5512	2.0090
10	2.3692	2.3611	4.3705

Table 1. Emp. standard deviation as a function of $var(v_n)$

6. CONCLUSION

In this paper we first revisited two classical PF algorithms, the Bootstrap algorithm and the reorganized SIR algorithm with optimal CID. Both algorithms are two natural MC approximations of the same exact recursive formula, which only differ by the time instant in which sampling is introduced. Alternately they are natural MC implementations of the two two-step formulas for computing $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$ from $p(d\mathbf{x}_{n-1}|\mathbf{y}_{0:n-1})$, in which one first increments one time index and then the other.

N	Alg. 3	Alg. 1	Alg. 2
100	2.0496	1.9145	2.7945
200	1.9303	1.8757	2.8808
400	1.9010	1.8723	2.7980
600	1.8076	1.8047	2.7660

Table 2. Empirical standard deviation as a function of N

We next considered PP. This problem is simpler than PF, because the optimal CID happens to be the prior transition of $\mathbf{x}$, from which it is often simple to sample from. We observed in particular that the Bootstrap is indeed optimal for PP. Motivated by this result we considered PP-based PF algorithms, in which the distribution of interest $p(d\mathbf{x}_n|\mathbf{y}_{0:n})$ is obtained indirectly, as a byproduct of a recursive loop involving prediction distributions. Simulations showed that the Bootstrap can indeed behave better than the SIR algorithm with optimal CID, depending on the shape of the likelihood function.

7. REFERENCES

[1] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/ non-Gaussian Bayesian state estimation," *IEE Proceedings-F*, vol. 140, no. 2, 1993.

[2] 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.

[3] P.M. Djuric, J.H. Kotecha, Jianqui Zhang, Yufei Huang, T. Ghirmai, M.F. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Processing Magazine*, 2003.

[4] O. Cappé, E. Moulines, and T. Rydén, *Inference in Hidden Markov Models*, Springer-Verlag, 2005.

[5] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

[6] B. Ait el Fquih and F. Desbouvries, "A new Particle Filtering algorithm with structurally optimal importance function," in *Proc. Icassp*, Las Vegas, 2008.

[7] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear estimation*, Prentice Hall Information and System Sciences Series. Prentice Hall, Upper Saddle River, NJ, 2000.