The one step fixed-lag particle smoother as a strategy to improve the prediction step of particle filtering

Samuel Nyobe, Fabien F. Campillo, Serge Moto, Vivien Rossi

The one step fixed-lag particle smoother as a strategy to improve the prediction step of particle filtering

S. Nyobe*, F. Campillo†, S. Moto‡, V. Rossi§

December 3, 2021

Abstract

Sequential Monte Carlo methods have been a major breakthrough in the field of numerical signal processing for stochastic dynamical state-space systems with partial and noisy observations. However, these methods still present certain weaknesses. One of the most fundamental is the degeneracy of the filter due to the impoverishment of the particles: the prediction step allows the particles to explore the state-space and can lead to the impoverishment of the particles if this exploration is poorly conducted or when it conflicts with the following observation that will be used in the evaluation of the likelihood of each particle. In this article, in order to improve this last step within the framework of the classic bootstrap particle filter, we propose a simple approximation of the one step fixed-lag smoother. At each time iteration, we propose to perform additional simulations during the prediction step in order to improve the likelihood of the selected particles.

Keywords: particle filter, bootstrap particle filter, one step fixed-lag particle smoother, prediction step, extended Kalman filter, unscented Kalman filter.

1 Introduction

Since the 1980s, sequential Monte Carlo (SMC) methods, also called particle filter (PF) methods [11, 6, 2, 7, 4], have met with vast success and incredible

*MIBA research Unity, Faculty of Science, University of Yaoundé 1, P.O. Box 812 Yaoundé, Cameroon, UMI 209, UMMISCO-Cameroon, IRD, Sorbonne University, P.O. Box 93143 Bondy, France Cedex, samuel.nyobe@facsciences-uy1.cm

†Inria, MathNeuro Team, Montpellier, France, fabien.campillo@inria.fr

‡MIBA research Unity, Faculty of Science, University of Yaoundé 1, P.O. Box 812 Yaoundé, Cameroon, UMI 209, UMMISCO-Cameroon, IRD, Sorbonne University, P.O. Box 93143 Bondy, France Cedex, serge-constant.moto@facsciences-uy1.cm

§RU Forets and Societies, CIRAD, Yaoundé, Cameroon, Computer Engineering Dpt, National Advanced School of Engineering, University of Yaoundé 1, P.O. Box 8390, Yaoundé, Cameroon vivien.rossi@cirad.fr

expansion in the context of problems of filtering for hidden Markov models (HMM), also called nonlinear filtering for state-space models with partial and noisy observations. The success of these methods is due to their ability to take into account, in a numerically realistic and efficient way, the non-linearity of the dynamics and the non-Gaussianity of the underlying conditional distributions.

The exact (i.e., non approximated) dynamics of these HMMs takes the form of a sequential Bayes formula represented in Eqs. (3)-(4) also called Bayesian filter. In the case of linear models with Gaussian additive noises, the Kalman filter makes it possible to calculate the exact solution. Otherwise, except for few particular models, it is not possible to calculate the exact solution, it is necessary to calculate approximations. The first approximation methods, the extended Kalman filter (EKF) [1] and variants, consisted in linearizing the models then in applying the Kalman filter. But in the case of strongly nonlinear models, the EKF often diverges.

Since the 1980s, Monte Carlo methods have become very effective alternatives. Monte Carlo methods are now recognized as a powerful tool in estimation of Bayesian filter in nonlinear/non-Gaussian hidden Markov models.

Among them, the PF techniques rely on an online importance sampling approximation of the sequential Bayes formula (3)-(4). Indeed, at each iteration, the PF builds of set of particles i.e. an independently and identically distributed sample from an approximation of the theoretical solution of the Bayesian filter.

An iteration of the FP classically takes place in two steps: the prediction step, which explores the state space by moving the particles according to the equation of state, followed by the correction step consisting on the one hand in weighting the particles according to their correspondence with the new observation, i.e. according to their likelihood, and on the other hand in resampling the particles according to these weights. These two steps can be understood respectively as mutation and selection steps of genetic algorithms.

The first really efficient PF algorithm, namely the bootstrap particle filter (BPF) or sampling-importance-resampling filter proposed in 1993 [11], follows exactly these two steps.

The resampling step is essential, indeed without it we observe an impoverishment of the particles in a few iterations: a few particles, even only one, will concentrate all the likelihood; all the other particles thus becoming useless, the filter then loses the "track" of the current state.

However, despite this resampling step particle filters often suffer from another type of particle degeneration [2, 18, 7, 22, 16]. This occurs when the particles explore areas of the state space that are not in correspondence with the new observation. This is due to the fact that in its classical version, especially in the case of BPF, the prediction step propagates the particles in the state space without taking into account the next observation. At the time of the weighting via the likelihood, a very large part of the particles are associated with an almost zero weight, this phenomenon is aggravated by the machine epsilon. In the worst case, all the particles can be associated with a zero weight, the filter has then totally lost the track of the state evolution.

To overcome this problem it is relevant to take into account the observations

in the prediction step consisting in exploring the state space. One can consult the review articles [9, 10] concerning the possibilities of improvement of particle filters.

The present paper aims to tackle the problem of degeneracy. We propose a new PF algorithm, called predictive bootstrap particle smoother (PBPS), to further improve the prediction step but keeping the simplicity of the BF. The principle of the PBPS is to take into account the current and the next observations for the prediction and correction steps. At each iteration during the prediction step, we perform additional predictions for assessing particle likelihood with the next observation. It can be seen as a simple approximation of the one step fixed-lag smoother. The additional predictions contribute to the correction step in order to improve the likelihood of the selected particles with the current and next observation.

We assess the performance of the PBPS by comparisons on simulation studies with the extended Kalman filter (EKF) [1], the unscented Kalman filter (UKF) [21, 13], and the bootstrap particle filter (BPF) [11]. The performance criterium is the accuracy of the estimates versus the computation time. For simulation studies we consider two nonlinear models: a classical one-dimensional one and a four-dimensional bearings-only tracking one.

2 Problem statement

2.1 The state space model

We consider a Markovian state-space model with state process $(X_k)_{k\geq 0}$ taking values in $\mathbb{R}^n$ and observation process $(Y_k)_{k\geq 1}$ taking values in $\mathbb{R}^d$. We suppose that conditionally on $(X_k)_{k\geq 0}$, the observations Y_k are independent.

The ingredients of the state-space model are:

$$
\begin{aligned}
q_k(x|x') &\stackrel{\text{def}}{=} p_{X_k|X_{k-1}=x'}(x)\,, && \text{(state transition kernel)}\\
\psi_k(x|y) &\stackrel{\text{def}}{=} p_{Y_k|X_k=x}(y)\,, && \text{(local likelihood function)}\\
\mu_0(x) &\stackrel{\text{def}}{=} p_{X_0}(x)\,, && \text{(initial distribution)}
\end{aligned}
$$

for any $x, x' \in \mathbb{R}^n$, $y \in \mathbb{R}^d$.

These ingredients can be made explicit by considering for example the following state space model:

$$X_k = f_{k-1}(X_{k-1}) + g_{k-1}(X_{k-1})\, W_{k-1}\,, \tag{1}$$

$$Y_k = h_k(X_k) + V_k\,, \tag{2}$$

for $1 \leq k \leq K$, where X_k (resp. Y_k, W_k, V_k) takes values in $\mathbb{R}^n$ (resp. $\mathbb{R}^d$, $\mathbb{R}^m$, $\mathbb{R}^d$), f_k (resp. g_k, h_k) is a differentiable and at most linear growth, uniformly in k, from $\mathbb{R}^n$ to $\mathbb{R}^n$ (resp. $\mathbb{R}^d$, $\mathbb{R}^{n\times m}$, $\mathbb{R}^d$), g_k being also bounded. Random

sequences W_k and V_k are independently and identically distributed centered white Gaussian noises; W_k, V_k, X_0 being independent. In this case:

$$q_k(x|x') = \frac{\exp\Big(-\frac{1}{2}\big[x - f_{k-1}(x')\big]^* \big[g_k(x')\, Q_{k-1}\, g_k(x')^*\big]^{-1} \big[x - f_{k-1}(x')\big]\Big)}{\sqrt{(2\,\pi)^n\ \det Q_{k-1}}}$$

and

$$\psi_k(x|y) \propto \exp\Big(-\frac{1}{2}\big[y - h_k(x)\big]^* R_k^{-1} \big[y - h_k(x)\big]\Big)$$

(ψ_k has to be known up to a multiplicative constant).

2.2 Nonlinear filtering

Nonlinear filtering aims at determining the conditional distribution η_k of the current state X_k given the past observations $Y_1, \ldots, Y_k$, namely:

$$\eta_k(x) \stackrel{\text{def}}{=} p_{X_k|Y_{1:k}=y_{1:k}}(x)\,,\ x \in \mathbb{R}^n$$

for any $k \geq 1$ and $y_{1:k} \in (\mathbb{R}^d)^k$, here we use the notation:

$$\text{“}Y_{1:k}\text{” for } (Y_1, \ldots, Y_k)$$

(e.g. $Y_{1:k} = y_{1:k}$ means $Y_\ell = y_\ell$ for all $\ell = 1, \ldots, k$).

The nonlinear filter allows to determine η_k from η_{k-1} using the classical two-step recursive Bayes formula:

Prediction step. The predicted distribution $\eta_{k^-}(x) \stackrel{\text{def}}{=} p_{X_k|Y_{1:k-1}=y_{1:k-1}}(x)$ of X_k given $Y_{1:k-1} = y_{1:k-1}$ is given by:

$$\eta_{k^-}(x) = \int_{\mathbb{R}^n} q_k(x|x')\ \eta_{k-1}(x')\ \mathrm{d}x'\,, \quad x \in \mathbb{R}^n\,. \tag{3}$$

Correction step. The new observation $Y_k = y_k$ allows to update the predicted distribution in order to obtain η_k according to the Bayes formula:

$$\eta_k(x) = \frac{\psi_k(x|y_k)\,\eta_{k^-}(x)}{\int_{\mathbb{R}^n} \psi_k(x'|y_k)\,\eta_{k^-}(x')\,\mathrm{d}x'}\,, \quad x \in \mathbb{R}^n\,. \tag{4}$$

Note that in the correction step (4) , η_k is proportional to the product of η_{k-1} and the local likelihood function, that is:

$$\eta_k(x) \propto \psi_k(x|y_k)\,\eta_{k^-}(x)\,.$$

2.3 Nonlinear smoothing

To get $\bar{\eta}_{k-1}(x)$ the conditional distribution of X_{k-1} given $Y_{0:k} = y_{0:k}$, we consider the extended state vector $\mathbf{X}_k = (X_k, X_{k-1})$, it's a Markov with transition:

$$\begin{aligned} Q_k(x', x''|x_{k-1}, x_{k-2}) &= p_{X_k, X_{k-1}|X_{k-1}=x_{k-1}, X_{k-2}=x_{k-2}}(x', x'') \\ &= p_{X_k|X_{k-1}=x'', X_{k-1}=x_{k-1}, X_{k-2}=x_{k-2}}(x')\; p_{X_{k-1}|X_{k-1}=x_{k-1}, X_{k-2}=x_{k-2}}(x'') \\ &= p_{X_k|X_{k-1}=x_{k-1}, X_{k-2}=x_{k-2}}(x')\; \delta_{x_{k-1}}(x'') \\ &= p_{X_k|X_{k-1}=x_{k-1}}(x')\; \delta_{x_{k-1}}(x'') \\ &= q_k(x'|x_{k-1})\; \delta_{x_{k-1}}(x'')\,. \end{aligned} \tag{5}$$

The conditional distribution $\boldsymbol{\eta}_k(x', x'')$ of $\mathbf{X}_k = (X_k, X_{k-1})$ given $Y_{0:k} = y_{0:k}$, we apply the previous filter formula:

$$\begin{aligned} \boldsymbol{\eta}_{k^-}(x', x'') &= \iint Q_k(x', x''|x_{k-1}, x_{k-2})\, \boldsymbol{\eta}_{k-1}(x_{k-1}, x_{k-2})\, \mathrm{d}x_{k-1}\, \mathrm{d}x_{k-2} \\ &= \iint q_k(x'|x_{k-1})\, \delta_{x_{k-1}}(x'')\, \boldsymbol{\eta}_{k-1}(x_{k-1}, x_{k-2})\, \mathrm{d}x_{k-1}\, \mathrm{d}x_{k-2} \\ &= \iint q_k(x'|x_{k-1})\, \delta_{x_{k-1}}(x'')\, \boldsymbol{\eta}_{k-1}(x'', x_{k-2})\, \mathrm{d}x_{k-1}\, \mathrm{d}x_{k-2}\,, \end{aligned} \tag{6}$$

and the distribution of Y_k given $(X_k = x_k, X_{k-1} = x_{k-1})$ is the distribution of Y_k given $X_k = x_k$, hence:

$$\boldsymbol{\eta}_k(x', x'') \propto \psi_k(x'|y_k)\, \boldsymbol{\eta}_{k^-}(x', x'')\,, \tag{7}$$

which the x''-marginal distribution gives the conditional distribution of X_{k-1} given $Y_{0:k} = y_{0:k}$.

2.4 Approximations

The main difficulty encountered by the nonlinear filter (3)-(4) lies in the two integrations. These integrations can be solved explicitly in only in the linear/Gaussian case, leading to the Kalman filter, and in a very few other specific nonlinear/non-Gaussian cases. In the latter cases, the optimal filter can be solved *explicitly* in the form of a finite dimensional filter; hence in the vast majority of cases, it is necessary to use approximation techniques [4]. Among the approximation techniques, we will consider the extended Kalman filter (EKF) [1], the unscented Kalman filter (UKF) [21, 13] and particle filter techniques, also called sequential Monte Carlo techniques [11, 6], see [8] for a recent overview.

Concerning the particle filters, as we will see, it is important to notice that on the one hand we do not need to know the analytical expressions of the state transition kernel and of the initial distribution, we just need to be able to sample (efficiently) from them; on the other hand, we do need the analytical expression of the local likelihood function (up to a multiplicative constant), indeed, for a given y, we need to compute $\psi_k(x|y)$ for a very large number of x values.

3 Particle approximations

The particle approximation η_k^N of η_k is a Monte Carlo empirical approximation of the form:

$$\eta_k^N(x) = \sum_{i=1}^N \omega_k^i \, \delta_{\xi_k^i}(x) \,, \; x \in \mathbb{R}^n \,.$$

composed of N particles ξ_k^i in $\mathbb{R}^n$ and weights ω_k^i, the weights are positive and sum to one [6]. Ideally the particles are sampled from η_k and the importance weight are all equal to $1/N$. *For the sake of notation simplicity, we now omit the superscript N in η_k^N.*

3.1 The bootstrap particle filter

The bootstrap particle filter (BPF) filter is a classical sequential importance resampling method: suppose we have a good approximation $\eta_{k-1} = \frac{1}{N} \sum_{i=1}^N \delta_{\xi_{k-1}^i}$ of η_{k-1}. We can apply the prediction step (3) to η_{k-1} and get:

$$\int_{\mathbb{R}^n} q_k(x|x') \; \eta_{k-1}(x') \, \mathrm{d}x' = \sum_{i=1}^N \omega_{k-1}^i \; q_k(x|\xi_{k-1}^i)$$

and then apply the correction step (4) and get:

$$\sum_{i=1}^N \omega_{k-1}^i \, \psi_k(x|y_k) \, q_k(x|\xi_{k-1}^i) \; \Big/ \int_{\mathbb{R}^n} \sum_{i=1}^N \omega_{k-1}^i \, \psi_k(x'|y_k) \, q_k(x'|\xi_{k-1}^i) \, \mathrm{d}x' \,.$$

Both the two last expressions are mixture of the densities $q_k(\,\cdot\,|\xi_{k-1}^i)$ and therefor not of the particle type. The bootstrap particle filter (BPF) proposed by [11] is the simplest method to propose a particle approximation. For the prediction step we use the sampling technique:

$$\xi_{k^-}^i \sim q_k(\,\cdot\,|\xi_{k-1}^i) \,, \; i = 1 : N \quad \text{(independently)} \tag{8}$$

which is:

$$\xi_{k^-}^i = f_{k-1}(\xi_{k-1}^i) + g_{k-1}(\xi_{k-1}^i) \, w^i$$

where w^i are i.i.d. $N(0, R_{k-1})$ samples. Then we let

$$\eta_{k^-}(x) \stackrel{\text{def}}{=} \sum_{i=1}^N \omega_{k-1}^i \, \delta_{\xi_{k^-}^i}(x) \,. \tag{9}$$

Through the correction step (4), this approximation η_{k^-} gives $\sum_{i=1}^N \omega_k^i \, \delta_{\xi_{k^-}^i}$ where:

$$\omega_k^i \stackrel{\text{def}}{=} \frac{\psi_k(\xi_{k^-}^i|y_k) \, \omega_{k^-}^i}{\sum_{j=1}^N \psi_k(\xi_{k^-}^j|y_k) \, \omega_{k^-}^j} \tag{10}$$

are the updated weight taking account of the new observation y_k through the likelihood function ψ_{k,y_k}. This correction step *must* be completed by a resampling of the particles $\xi_{k^-}^i$ according to the importance weights ω_k^i:

$$\xi_k^i \sim \sum_{j=1}^{N} \omega_k^j \, \delta_{\xi_{k^-}^j} \,, \; i = 1:N \quad \text{(independently)} \tag{11}$$

leading the to the particle approximation:

$$\eta_k(x) \stackrel{\text{def}}{=} \sum_{i=1}^{N} \omega_k^i \, \delta_{\xi_k^i}(x) \,, \text{ with } \omega_k^i = \frac{1}{N} \,. \tag{12}$$

Algorithm 1 gives a summary of the BPF, in this version a resampling is performed at each time step. The multinomial resampling step (11), the basic idea proposed in [11], could be deeply improved, there are several resampling techniques, see [5, 15] for more details.

1: $\xi_0^i \overset{\text{iid}}{\sim} \mu_0(\,\cdot\,) \,, \; i = 1:N$ `# initialization`
2: **return** $\xi_0^{1:N}$
3: **for** $k = 1:K$ **do**
4: $\quad \xi_{k^-}^i \sim q_k(\,\cdot\,|\xi_{k-1}^i) \,, \; i = 1:N$ `# evolution of particles`
5: $\quad \omega_k^i \leftarrow \psi_k(\xi_{k^-}^i|y_k) \,, \; i = 1:N$ `# likelihood`
6: $\quad \omega_k^i \leftarrow \omega_k^i / \sum_{j=1}^N \omega_k^j \,, \; i = 1:N$ `# reweighting`
7: $\quad \xi_k^i \overset{\text{iid}}{\sim} \sum_{j=1}^N \omega_k^j \, \delta_{\xi_{k^-}^j} \,, \; i = 1:N$ `# resampling`
8: $\quad$ **return** $\xi_k^{1:N}$
9: **end for**

Algorithm 1: Bootstrap particle filter (BPF).

It is not necessary to resample the particles at each time iteration like in Algorithm 1. However, this resampling step should be done regularly in terms of time iterations to avoid degeneracy of the weights [6].

We will look at another degeneracy problem. Suppose that in step (10), the local likelihood values $\psi_k(\xi_{k^-}^i|y_k)$ associated with the predicted particles $\xi_{k^-}^i$ are all very small, or even equal to zero due to rounding in floating point arithmetic. This normalization step (10) is then impossible. This problem occurs when the filter "loses track" of the true state X_k, i.e. the likelihood of the predicted particules $\xi_{k^-}^i$ w.r.t. the observation y_k are all negligible, in this case the observation y_k appears as an outlier. This occurs especially when the period of time between two successive instants of observation is very large compared to the dynamics of the state process. Strategies to the overcome that weakness encompass the iterated extended Kalman particle filter [17] or the iterated unscented Kalman particle filter [12]; see [10] and [16] for reviews of the subject.

3.2 The predictive bootstrap particle smoother

The purpose of the predictive bootstrap filter (PBPS) is to improve the correction step (10) of the BPF by using both $Y_k = y_k$ and $Y_{k+1} = y_{k+1}$ at each iteration k. In a way, the PBPS can be seen as an approximation of the distribution of X_k given $Y_{1:k+1} = y_{1:k+1}$, the one step fixed-lag smoother presented in Section 2.3.

In the proposed algorithm, the prediction step consists first in propagating the N particles of η_{k-1} to time k to get η_{k^-}; second, for each one of these N particles propagating, one-step-ahead at time $k+1$, an offspring particle. Then the correction step consists in updating the weights of the N particles of η_{k^-} according to their likelihood with y_k and the likelihood of their one-step ahead offspring particles with y_{k+1}.

The iteration $k-1 \to k$ of the filter is more precisely:

Prediction step. Like for the BPF we sample N particles and compute N normalized likelihood weights:

$$\tilde{\xi}_k^i \sim q_k(\,\cdot\,|\xi_{k-1}^i)\,, \qquad \tilde{\omega}_k^i \propto \psi_k(\xi_{k^-}^i|y_k)\,\omega_{k^-}^i\,, \qquad i = 1:N\,.$$

Again we will resample the particles $\tilde{\xi}_k^{1:N}$, but instead of doing so according to the weights $\tilde{\omega}_k^{1:N}$, we will first modify the latter ones. For each index i, we generate a one-step-ahead offspring particles and compute its weight:

$$\tilde{\xi}_{k+1}^i \sim \tilde{q}_{k+1}(\,\cdot\,|\tilde{\xi}_k^i)\,, \qquad \tilde{\omega}_{k+1}^i \stackrel{\text{def}}{=} \psi_{k+1}(\tilde{\xi}_{k+1}^i|y_{k+1}).$$

Note that the likelihood weights $\tilde{\omega}_{k+1}^i$ depend on the next observation y_{k+1}. See later for the choice of one-step-ahead sampler $\tilde{q}_{k+1}$.

Correction step. We compute the weights at time k according to Eq. (7):

$$\omega_k^i \stackrel{\text{def}}{=} \tilde{\omega}_k^i\;\tilde{\omega}_{k+1}^i \qquad \text{for } i = 1:N. \tag{13}$$

Then, N particles $\tilde{\xi}_k^{1:N}$ are resampled according to the weights $\omega_k^{1:N}$.

The PBPS is depicted in Algorithm 2.

Choice of one-step-ahead sampler $\tilde{q}_{k+1}$. For the special case of the system (1)-(2) we can choose a simpler "deterministic sampler":

$$\tilde{\xi}_{k+1}^i = f_k(\tilde{\xi}_k^i)$$

which corresponds to the one-step-ahead mean:

$$\begin{aligned}\tilde{\xi}_{k+1}^i &= \mathbb{E}\big(X_{k+1}\big|X_k = \tilde{\xi}_k^i\big)\\ &= \mathbb{E}\big(f_k(X_k) + g_k(X_k)\,W_k\big|X_k = \tilde{\xi}_k^i\big)\\ &= f_k(\tilde{\xi}_k^i) + g_k(\tilde{\xi}_k^i)\,\underbrace{\mathbb{E}(W_k|X_k = \tilde{\xi}_k^i)}_{=0} = f_k(\tilde{\xi}_k^i)\end{aligned}$$

This choice greatly reduces the computation burden and it is enough, as we will see, in linear state equation. For highly nonlinear state equation, we can choose $\tilde{q}_{k+1} = q_{k+1}$, which corresponds to simulate the state dynamic.

$\xi_0^i \overset{\text{iid}}{\sim} \mu(\,\cdot\,),\ i = 1:N$ `# initialization`

for $k = 1:K$ **do**

  for $i = 1:N$ **do**

    $\tilde{\xi}_k^i \sim q_k(\,\cdot\,|\xi_{k-1}^i)$ `# particles propagation`

    $\tilde{\omega}_k^i \leftarrow \psi_k(\tilde{\xi}_k^i|y_k)$

    $\tilde{\xi}_{k+1}^i = f_k(\tilde{\xi}_k^i)$ `# offspring particles generating`

    $\tilde{\omega}_{k+1}^i \leftarrow \psi_{k+1}(\tilde{\xi}_{k+1}^i|y_{k+1})$ `# offspring particles weighting`

    $\omega_k^i \leftarrow \tilde{\omega}_k^i\,\tilde{\omega}_{k+1}^i$ `# particles weighting`

  end for

  $\omega_k^i \leftarrow \omega_k^i / \sum_{i'=1}^N \omega_k^{i'}\ ,\ i = 1:N$ `# weights normalization`

  $\xi_k^i \overset{\text{iid}}{\approx} \sum_{i'=1}^N \omega_k^{i'}\,\delta_{\tilde{\xi}_k^{i'}}\ ,\ i = 1:N$ `# particles resampling`

  return $\xi_k^{1:N}$

end for

Algorithm 2: predictive bootstrap particle smoother (PBPS).

4 Simulation studies

We compare numerically the PBPS to other filters on two space-state models described below. The filters performance is assessed trough Monte Carlo scheme by a criteria based on mean squared error between the state trajectory and the filter estimates. We assess the performance versus computational time by testing several numbers of particles.

4.1 Space-state models simulated

First case study: one-dimensional model

We consider the following one-dimensional nonlinear model [11, 14, 6]:

$$\begin{aligned} X_k &= \frac{1}{2}\,X_{k-1} + \frac{25\,X_{k-1}}{1+X_{k-1}^2} + 8\cos\big(1.2\,(k-1)\big) + W_{k-1}\,, \\ Y_k &= \frac{X_k^2}{20} + V_k\,, \end{aligned} \tag{14}$$

with $1 \leq k \leq K$ $(K = 50)$, $X_0 \sim \mathcal{N}(0,1)$; $W_k \overset{\text{iid}}{\sim} \mathcal{N}(0,3^2)$, $V_k \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$; X_0, $(W_k)_{k\geq 1}$ and $(V_k)_{k\geq 1}$ mutually independent. Note that the state process X_k is observed only through X_k^2 so the filters have difficulties to determine whether X_k is positive or negative, especially since the state process X_k regularly changes of sign. Thus this model is regularly used as benchmark for testing filters, as filters may easily lose track of X_k.

Second case study: a four-dimensional bearings-only tracking model

We consider the following four-dimensional model [11, 19, 3]:

$$\begin{aligned} X_k &= \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} X_{k-1} + \sigma_W \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} W_{k-1}\,, \\ Y_k &\sim \text{wrapped Cauchy}\Big(\arctan\Big(\frac{X_k[1]}{X_k[2]}\Big), \rho\Big), \end{aligned} \tag{15}$$

with $1 \leq k \leq K = 20$, $X_0 \sim \mathcal{N}(\bar{X}_0, P_0)$ with:

$$\bar{X}_0 = \big(-0.05, 0.2, 0.001, -0.055\big)^*\,, \quad P_0 = 0.01\,\mathrm{diag}\big(0.5^2, 0.3^2, 0.005^2, 0.01^2\big)\,,$$

$\sigma_W = 0.001$, $W_k \overset{\text{iid}}{\sim} \mathcal{N}(0, I_{2\times 2})$, $\rho = 1 - 0.005^2$, X_0 and $(W_k)_{k\geq 1}$ mutually independent; conditionally on $(X_k)_{k\geq 0}$, $(Y_k)_{k\geq 1}$ and $(W_k)_{k\geq 1}$ are independent.

The state equation corresponds to a target moving on a plane. The state vector is:

$$X_k = (X_k[1], X_k[2], X_k[3], X_k[4])^* = (x_1, x_2, \dot{x}_1, \dot{x}_2)^*\,,$$

where (x_1, x_2) are the Cartesian coordinates of the target in the plane and $(\dot{x}_1, \dot{x}_2)$ are the corresponding velocities. The observer is located at the origin of the plane and accessed only to the azimuth angle $\beta = \arctan(x_1/x_2) \in [-\pi, \pi]$ corrupted by noise.

The conditional probability density function of the measured angle Y_k given the state X_k is assumed to be a wrapped Cauchy distribution with concentration parameter ρ [19]:

$$p_{Y_k|X_k=x}(y) = \frac{1}{2\,\pi}\,\frac{1-\rho^2}{1+\rho^2-2\,\rho\,\cos\Big(y - \arctan(\frac{x[1]}{x[2]}))\Big)}\,, \quad -\pi \leq y < \pi\,, \tag{16}$$

where $\rho \in [0, 1]$ is the mean resultant length. Note that the state dynamics is linear and Gaussian, but the observation dynamics is nonlinear and non-Gaussian.

4.2 Performance criteria

For both case studies, we compare the filters performances through a Monte Carlo scheme. We simulate S independent trajectories $(X^{(s)}_{0:K}, Y^{(s)}_{1:K})_{s=1:S}$ of the state-space models. For each simulation s, we ran R times each particle filter $\mathcal{F} \in \{\text{BPF, PBPS}\}$ and we compute the root mean squared error at time k:

$$\mathrm{RMSE}_k(\mathcal{F}) \overset{\text{def}}{=} \frac{1}{S}\sum_{s=1}^{S}\sqrt{\frac{1}{R}\sum_{r=1}^{R}\big|\hat{X}_k^{\mathcal{F}(s,r)} - X_k^{(s)}\big|^2}\,, \tag{17}$$

where:

$$\hat{X}_k^{\mathcal{F}(s,r)} \overset{\text{def}}{=} \int_{\mathbb{R}^n} x\,\eta_k^{\mathcal{F}(s,r)}(x)\,\mathrm{d}x = \frac{1}{N}\sum_{i=1}^{N}\xi_k^{\mathcal{F}(s,r),i}$$

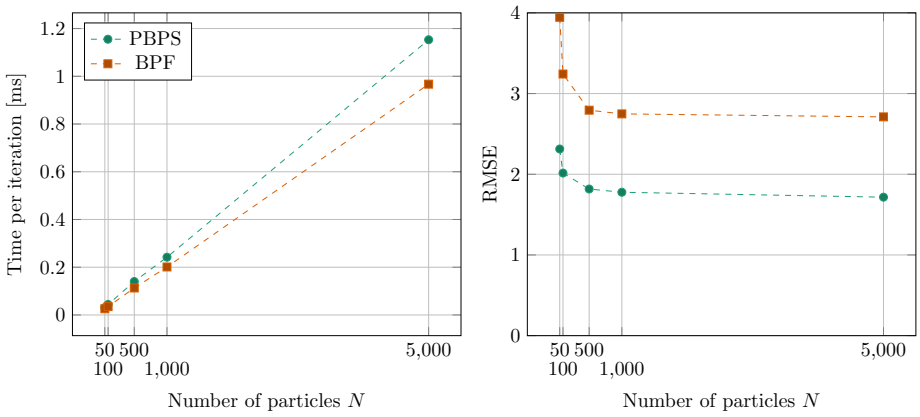

Figure 1: One-dimensional case study (14) — We compare the performances of the PBPS and the BPF with 5 different values of N: 50, 100, 500, 1,000, and 5,000. We plot the average computation time per iteration (left) and RMSE (right) with $S = 100$ and $R = 40$. We can see that PBPS is clearly superior to BPF. For example, the PBPS with 50 particles is significantly faster than the BPF with 5000 particles while giving better results.

is the numerical approximation of $\hat{X}_k^{(s)} = \mathbb{E}(X_k^{(s)}|Y_{1:k}^{(s)})$ by the filter $\mathcal{F}$. We also compute the global root mean squared error:

$$\mathrm{RMSE}(\mathcal{F}) \stackrel{\text{def}}{=} \frac{1}{K+1} \sum_{k=0}^{K} \mathrm{RMSE}_k(\mathcal{F})\,. \tag{18}$$

We proceed to the systematic resampling technique [5, 15] in all particle filters. To fairly compare the different particle filters, we compute the RMSE and the mean computational times of an iteration on the same simulation of states and observations trajectories.

All implementations are done in R language [20] using a 2.10 GHz core i3 intel running Linux Mint. 19.1 with a 8 Go RAM.

4.3 Simulation results

One-dimensional case study (14)

First we compare the BPF and the PBPS. In Fig. 1, we plot the average computation time per iteration (left) and the RMSE (right) with $S = 100$, $R = 40$ and for different values of N: 50, 100, 500, 1000 and 5000.

Compared to the BPF, and with the same number of particules, the PBPS requires a little more computation time (Fig. 1 left) but is significantly more accurate (Fig. 1 right).

For instance, the PBPS with $N = 50$ particles is more accurate than the BPF with $N = 5000$ particles while being significantly faster. Similarly, the PBPS with $N = 1000$ particles is 5 times faster than the BPF with $N = 5000$ particles, while being 30% better. The strategy proposed by the PBPS is very advantageous here, in particular due to the very nonlinear nature of this model for both the state equation and the observation equation.

Next, in Fig. 2, we simulate one trajectory of the state/observation process and we compare the BPF and the PBPS (with $N = 1000$) with the extended Kalman filter (EKF) and une unscented Kalman filter (UKF).

For each filter $\mathcal{F} \in \{\text{PBPS, BPF, EKF, UKF}\}$, we plot the true state trajectory $k \to X_k$, the approximation $k \to \hat{X}_k^{\mathcal{F}}$, and associated 95% confidence region. The PBPS approximation appears to have a smaller error $k \to |\hat{X}_k^{\mathcal{F}} - X_k|$ and a smaller confidence region than the other filters.

Due to the nature of the system, it is not surprising that the EKF has a very poor quality behavior. In particular, the EKF has a lot of difficulty to take into account the passages of the state variable by 0: once it has "chosen" to go to one side or the other of 0, it no longer has the possibility of changing sides even in the case where the sign of the state X_k is different. Let us note that the UKF, even if it is a little less precise compared to particle filters, allows to take into account the non-linearities of the system and is much faster than all the other filters.

Finally, in Fig. 3, we compare $k \to \text{RMSE}_k(\mathcal{F})$ for $\mathcal{F} \in \{\text{PBPS, BPF, EKF, UKF}\}$ (with $N = 1,000$). Note that for EKF and UKF, the summation on replicas r is useless. Once again, it can be seen that the PBPS filter behaves significantly better than the others. As said before the EKF filter behaves very poorly on this example, unlike the UKF which is closer to the behavior of particle filters. We note again the very bad behavior of the EKF and the relatively good behavior of the UKF, the latter remains less accurate than the BPF and the PBPS but requires much less computing time.

Four-dimensional case study (15)

Classically, in this case of study the trajectory of the target is not simulated according to the equation (15) but according to a uniform rectilinear motion or according to a uniform motion with some course changes. On the other hand, the filters follow the model (15): there is thus a mismatch between the simulation and the filter. This will allow us to test the robustness of the filters which is an essential property of the domain.

We run four simulations: one where the target stays in a straight line and one where the target changes course; and in each case the filter uses a value of σ_W small ($\sigma_W = 0.001$) and a value of σ_W large ($\sigma_W = 0.01$). The first value of 0.001 corresponds better to the target motion while the second value 0.01 is too large compared to the target trajectory.

In the case of Fig. 4 the target follows a uniform rectilinear motion and the

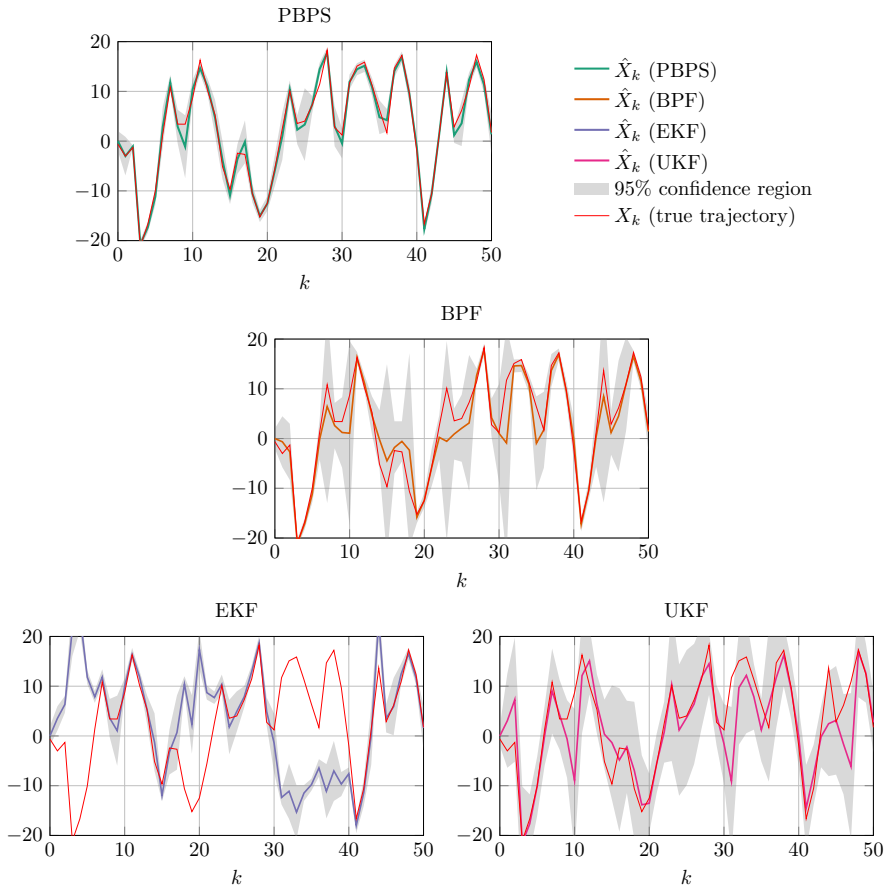

Figure 2: One-dimensional case study (14) — For each filter, with $N = 1000$, we plot the true state trajectory $k \to X_k$ (red), the approximation $k \to \hat{X}_k$, and the associated 95% confidence region (grey area). The PBPS approximation appears to have a smaller error $k \to \hat{X}_k - X_k$ and a smaller confidence region than the other filters.

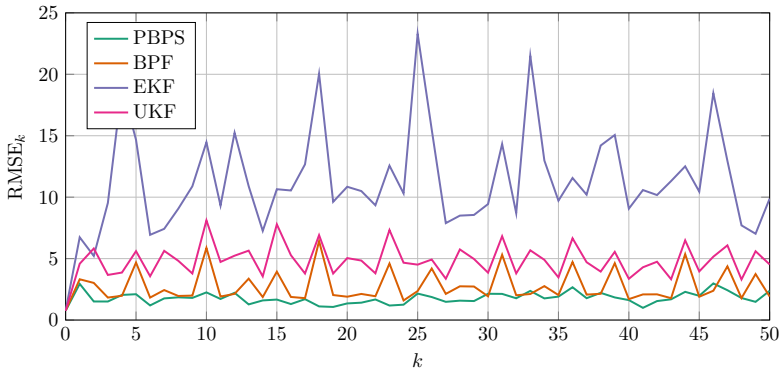

Figure 3: One-dimensional case study (14) — We compare $k \to \mathrm{RMSE}_k$ ($S = 100$ and $R = 40$), for the PBPS ($N = 1000$), BPF ($N = 1000$), EKF, and the UKF (for the two latter filters, the summation on replicas r is useless).

filter assumes a small value of σ_W (0.001). The computation time per iteration according to the number of particles is presented in Fig. 4 (bottom left); the RMSE also according to the number of particles is presented in Fig. 4 (bottom right). The PBPS filter is slightly more computationally time consuming than the BPF filter; the accuracies are equivalent.

In Fig. 4 (top), we present an example of simulation of the trajectory and of the two filters: in addition to the real trajectory, we plot the trajectories estimated by each of the two filters, to which we associate the 95% uncertainty ellipses corresponding to the empirical covariance matrices of the position (x_1, x_2). These ellipses are very stretched in the direction of the line of sight represented by lines coming from the position $(0, 0)$ of the observer.

In the case of Fig. 5 the target follows a uniform rectilinear motion and the filter assumes a large value of σ_W (0.01). The computation time per iteration is equivalent to the previous case. On the other hand, in terms of RMSE, the PBPS performs much better than the BPF.

In Fig. 5 (top), we see that the BPF loses the target track while the PBPS is much more robust. Indeed, here the value $\sigma_W = 0.01$ used in the filter is too large compared to the real target track and the PBPS reacts much better to this model mismatch.

In the case of Fig. 6 the target follows a uniform rectilinear motion except for a single change of course and the filter assumes a small value of $sigma_W$ (0.001). The computation time per iteration is equivalent to the previous cases. In terms of RMSE, the two filters are equivalent.

In Fig. 6 (top), we see that both filters lose the target track. Indeed, the value $\sigma_W = 0.001$ is too small to account for the change in heading. A classical

idea is to increase this value in order to allow the filter to maintain the track. This is what we do in the next example.

In the case of Fig. 7 the target follows a uniform rectilinear motion except for a single change of course and the filter assumes a large value of σ_W (0.01). The computation time per iteration is equivalent to the previous cases. On the other hand, in terms of RMSE the PBPS performs much better than the BPF.

In Fig. 7 (top), we see that the BPF loses the target track while the PBPS is clearly more robust.

In conclusion, for a slightly longer computation time, the PBPS filter is clearly more robust than the BPF in terms of assumptions on the target trajectory, which is particularly sought after in this type of application.

5 Discussion and conclusion

For the first example presented, at an equivalent level of accuracy and computation time, the PBPS requires significantly fewer particles. With the same number of particles, there is a strong reduction of the empirical variance of the error, this is also due to the fact that the PBPS is a smoother which uses one more observation than the BPF which is a filter; if this is done with a slight increase of the computation time per iteration for the same number of particles, the PBPS is still much more efficient than the BPF since it requires much less particles for an equivalent level of accuracy. For the second example, in the case where the BPF works well, the PBPS has an equivalent level of accuracy for a slightly higher computation time. On the other hand, in cases where the BPF loses the target trajectory, the PBPS does not lose it and has a much higher accuracy for a slightly longer computation time.

When the equation of state is strongly nonlinear, as in Example 1, PBPS is significantly more accurate than BPF with half as many particles. For example 2, it is easy to adjust the variance of the noise on the dynamics of the PBPS so that it supports a change in the course of the target. Whereas for the BPF we were not able to find a variance value that would allow it to withstand a change in course. So the example 2 illustrates the greater robustness of the SPBF compared to the BPF when a nonlinearity occurs on the state dynamics.

We introduced an improved version of the BPF, the *predictive bootstrap particle smoother* (PBPS), for nonlinear state-space models, which presents a markedly improved behavior. Although the PBPS relied on an observation horizon of one-step forward, for systems with nonlinear state dynamics it produced more accurate estimates than other filters for a significantly lower computation time. Beyond the specific interest of PBPS, our results demonstrated that it is possible to build particle filtering algorithms based on step forward horizons with competitive computation time.

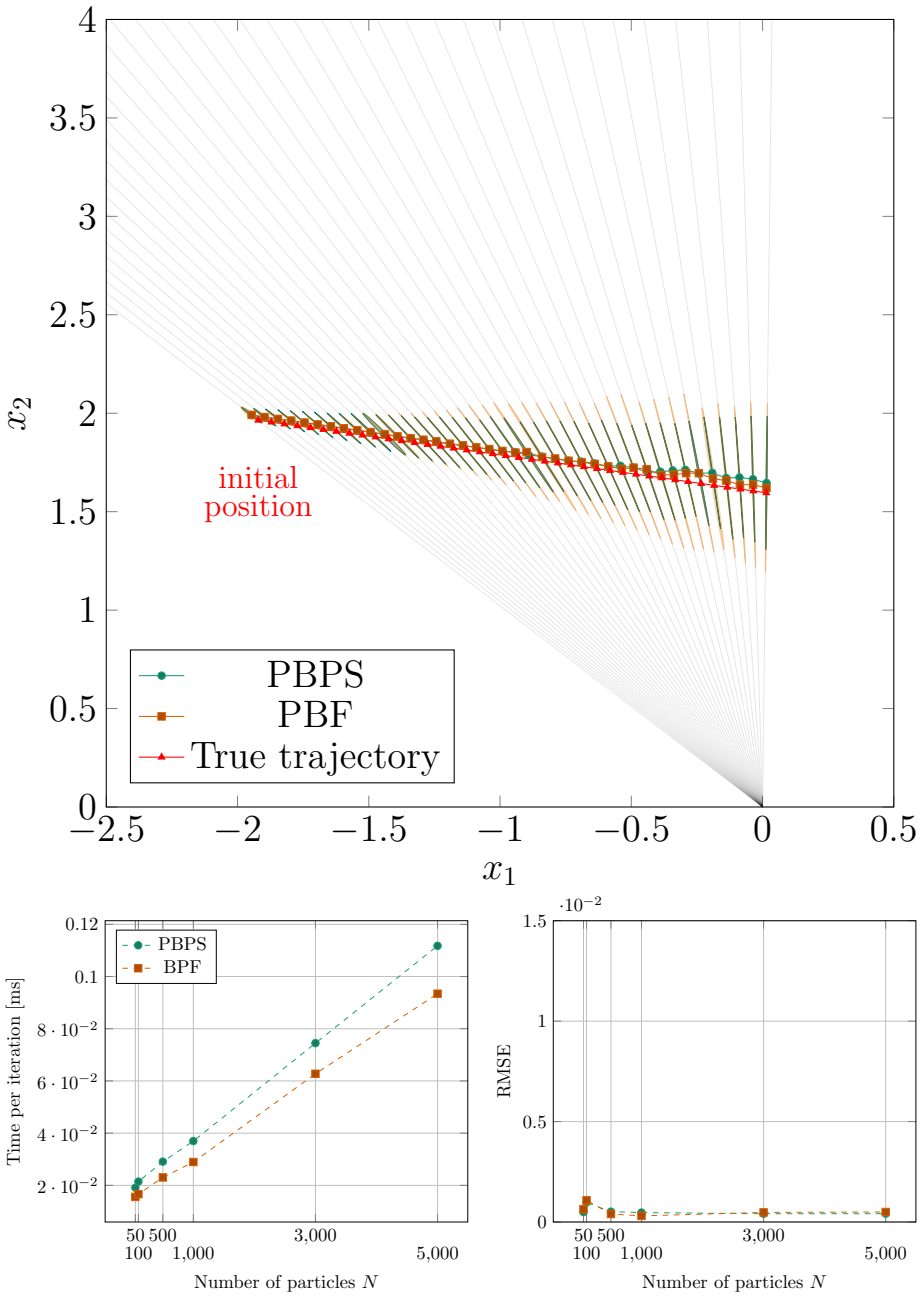

Figure 4: Four-dimensional case study (15) — The target follows a uniform rectilinear motion and the filter assumes a small value of $\sigma_W = 0.001$.

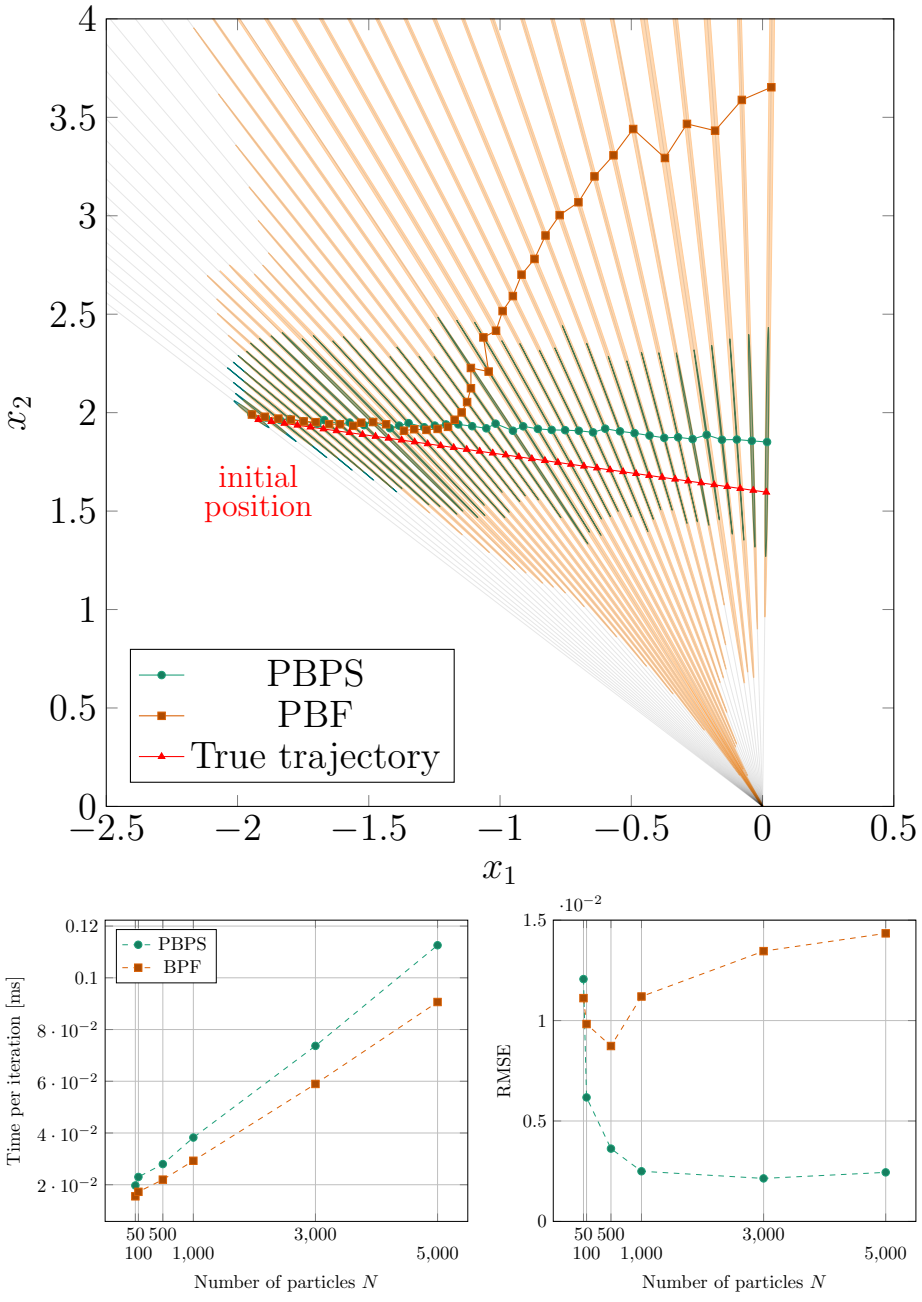

Figure 5: Four-dimensional case study (15) — The target follows a uniform rectilinear motion and the filter assumes a large value of $\sigma_W = 0.01$.

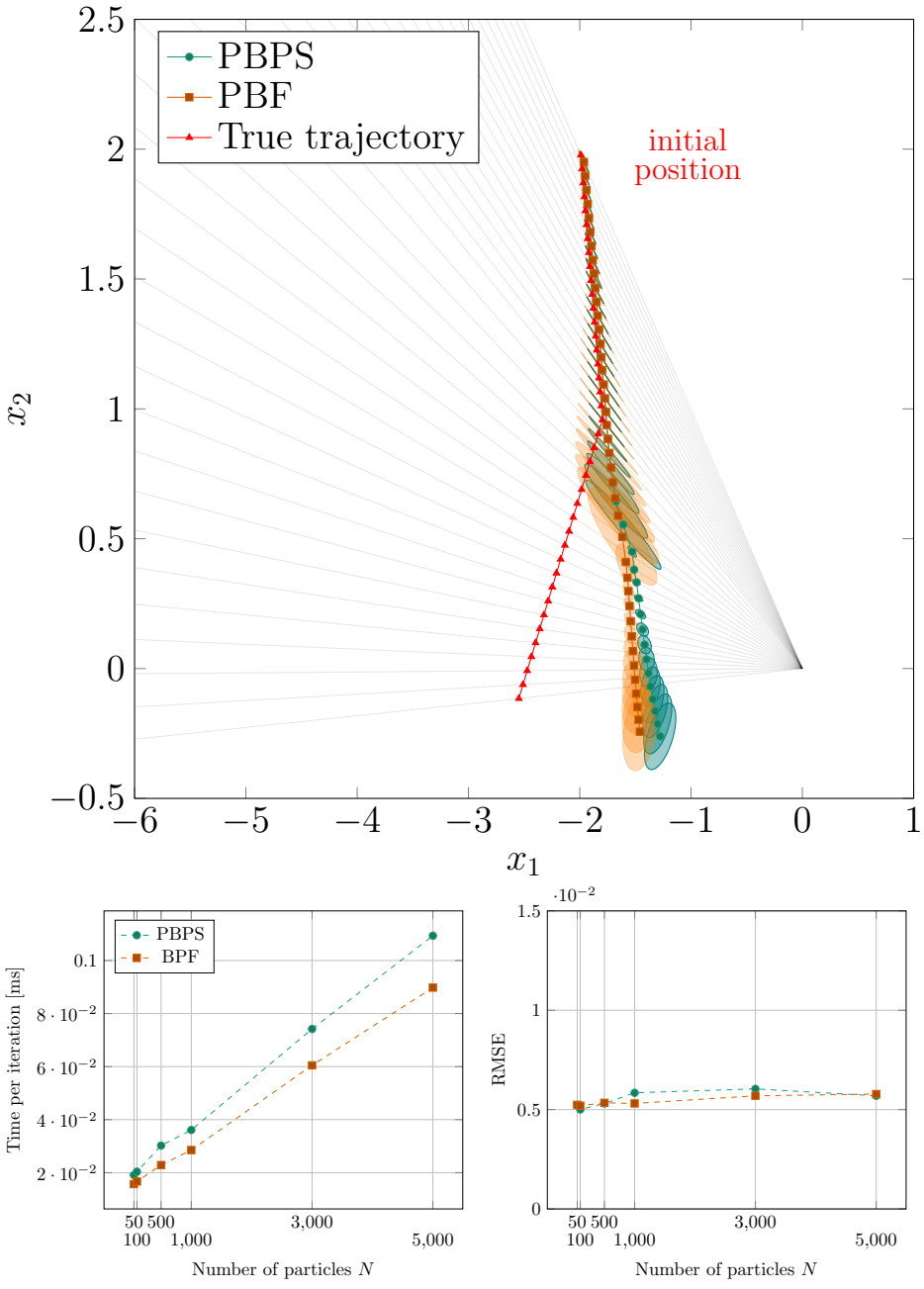

Figure 6: Four-dimensional case study (15) — The target follows a uniform rectilinear motion with a single change in the course and the filter assumes a small value of $\sigma_W = 0.001$.

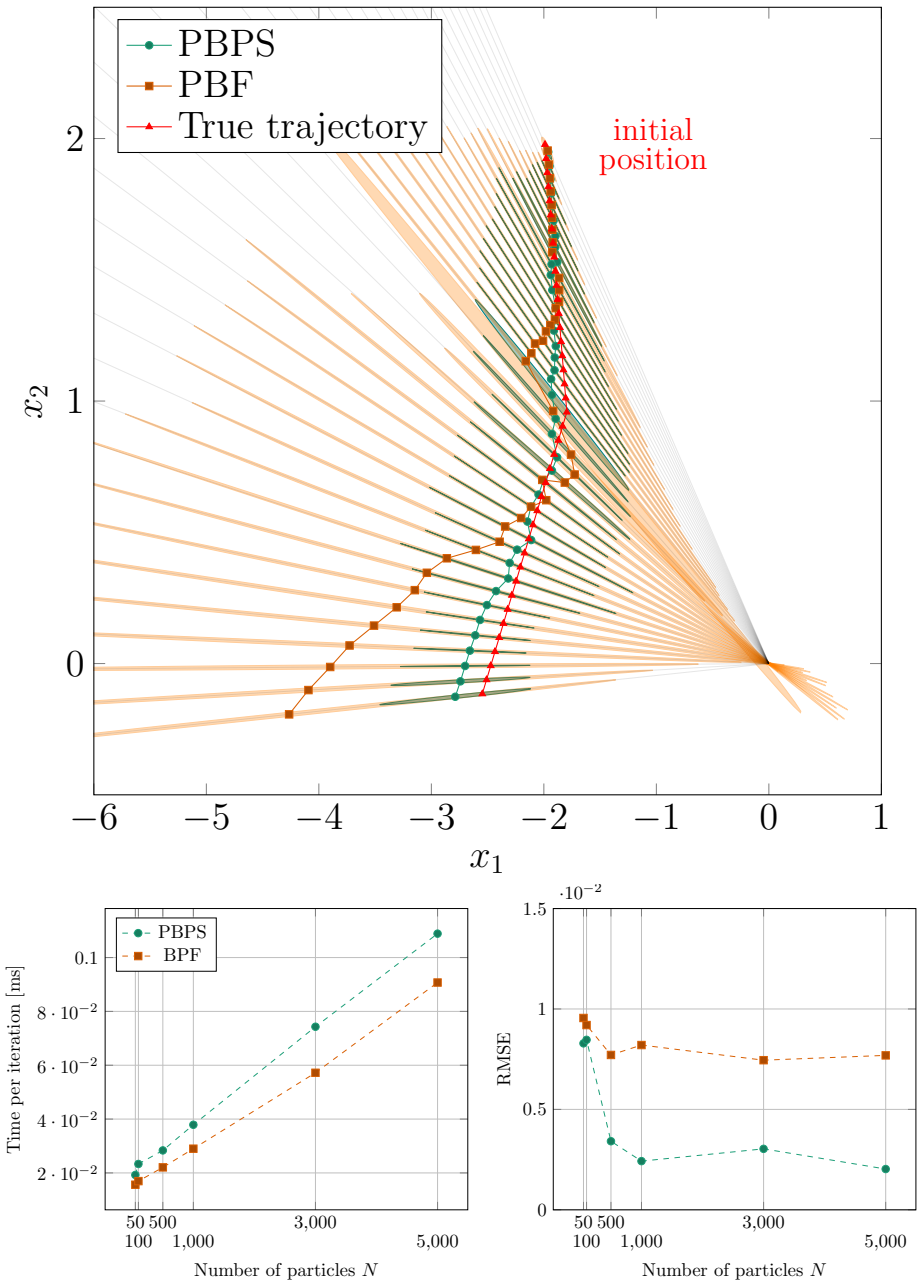

Figure 7: Four-dimensional case study (15) — The target follows a uniform rectilinear motion with a single change in the course and the filter assumes a large value of $\sigma_W = 0.01$.

References

[1] B. D. Anderson and J. B. Moore. *Optimal Filtering*. Prentice-Hall, 1979.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[3] F. Campillo and V. Rossi. Convolution particle filter for parameter estimation in general state-space models. *IEEE Transactions on Aerospace and Electronic Systems*, 45(3):1063–1072, 2009.

[4] D. Crisan and B. Rozovskii, editors. *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

[5] R. Douc and O. Cappé. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, ISPA*, 2005.

[6] A. Doucet, N. de Freitas, and N. J. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer New York, 2001.

[7] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In *[4]*, 2011.

[8] A. Doucet and A. Lee. Sequential Monte Carlo methods. In M. Maathuis, M. Drton, S. Lauritzen, and M. Wainwright, editors, *Handbook of Graphical Models*. Chapman & Hall/CRC, 2018.

[9] S. Godsill. Particle filtering: the first 25 years and beyond. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7760–7764. IEEE, 2019.

[10] S. J. Godsill and T. Clapp. Improvement strategies for Monte Carlo particle filters. In *[6]*, pages 139–158, 2001.

[11] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non–Gaussian Bayesian state estimation. *IEEE Proceedings F Radar and Signal Processing*, 140(2):107–113, 1993.

[12] W. Guo, C. Han, and M. Lei. Improved unscented particle filter for nonlinear Bayesian estimation. In *10th International Conference on Information Fusion*, 2007.

[13] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *IEEE Review*, 92(3):401–422, 2004.

[14] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[15] T. Li, M. Bolic, and P. M. Djuric. Resampling methods for particle filtering: Classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3):70–86, 2015.

[16] T. Li, S. Sun, T. P. Sattar, and J. M. Corchado. Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches. *Expert Systems with Applications*, 41(8):3944–3954, 2014.

[17] L. Liang-Qun, J. Hong-Bing, and L. Jun-Hui. The iterated extended Kalman particle filter. In *IEEE International Symposium on Communications and Information Technology*, volume 2, pages 1213–1216, 2005.

[18] S. Park, J. P. Hwang, E. Kim, and H.-J. Kang. A new evolutionary particle filter for the prevention of sample impoverishment. *IEEE Transactions on Evolutionary Computation*, 13(4):801–809, 2009.

[19] M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[20] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[21] E. A. Wan and R. van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.

[22] J. Zuo. Dynamic resampling for alleviating sample impoverishment of particle filter. *IET Radar, Sonar Navigation*, 7(9):968–977, 2013.