Nudging the Particle Filter

Ömer Deniz Akyıldız, Joaquín Míguez

Department of Signal Theory and Communications
Universidad Carlos III de Madrid
Avda. de la Universidad 30, 28911 Leganes, Madrid (Spain).

{omerdeniz.akyildiz, joaquin.miguez}@uc3m.es

September 15, 2017

Abstract

We investigate a new sampling scheme to improve the performance of particle filters in scenarios where either (a) there is a significant mismatch between the assumed model dynamics and the actual system producing the available observations, or (b) the system of interest is high dimensional and the posterior probability tends to concentrate in relatively small regions of the state space. The proposed scheme generates nudged particles, i.e., subsets of particles which are deterministically pushed towards specific areas of the state space where the likelihood is expected to be high, an operation known as *nudging* in the geophysics literature. This is a device that can be plugged into any particle filtering scheme, as it does not involve modifications in the classical algorithmic steps of sampling, computation of weights, and resampling. Since the particles are modified, but the importance weights do not account for this modification, the use of nudging leads to additional bias in the resulting estimators. However, we prove analytically that particle filters equipped with the proposed device still attain asymptotic convergence (with the same error rates as conventional particle methods) as long as the nudged particles are generated according to simple and easy-to-implement rules. Finally, we show numerical results that illustrate the improvement in performance and robustness that can be attained using the proposed scheme. In particular, we show the results of computer experiments involving misspecified Lorenz 63 model, object tracking with misspecified models, and a large dimensional Lorenz 96 chaotic model. For the examples we have investigated, the new particle filter outperforms conventional algorithms empirically, while it has only negligible computational overhead.

1 Introduction

1.1 Background

Inferring the hidden states of state-space models (SSMs) has received considerable attention since SSMs are ubiquitous in many fields including object tracking, mathematical

finance, weather dynamics, population dynamics, machine learning, and other topics where inferring the states of dynamical systems from data plays a big role.

An SSM comprises a pair of stochastic processes $(x_t)_{t\geq 0}$ and $(y_t)_{t\geq 1}$ called *signal process* and *observation process* respectively. The conditional relations between these processes are defined with a transition and an observation model (also called likelihood model) where observations are conditionally independent given the signal process, which is itself a Markov process. Given an observation sequence $y_{1:t}$, the filtering problem in SSMs is then estimation of expectations with respect to the posterior probability distribution of the hidden states conditioned on the sequence of observations $y_{1:t}$, which is also called as the filtering distribution.

Apart from few special cases, neither the filtering distribution nor the integrals (or expectations) with respect to can be computed exactly, hence one needs to resort to numerical approximations of these quantities. Bootstrap particle filters (BPFs) have been a classical choice for this task since their introduction [1–5]. The BPF constructs an empirical approximation of the posterior probability distribution via a set of Monte Carlo samples which are modified or killed sequentially as more data are taken into account. These samples are then used to estimate the relevant expectations. The BPF, in its original form, has received significant attention due to its efficiency in a variety of problems, its intuitive form, and its straightforward implementation. A large body of theoretical work concerning the BPF is also conducted, for example, it has been proved that the expectations with respect to the empirical measure that the BPF constructs converge to the expectations with respect to the true posterior distribution when the number of Monte Carlo particles tends to infinity [6–9]. Also, it has been shown that the BPF converges uniformly over time, i.e., the errors do not accumulate under regularity assumptions about the state-space and the model [10]. We refer the reader to the monographs [11–13] for an extensive compilation of theoretical results.

Despite the success of the BPF in relatively low dimensional settings, its use has been shown to be impractical in models where $(x_t)_{t\geq 0}$ and $(y_t)_{t\geq 1}$ are sequences of high-dimensional random variables. This problem is an instance of general class of problems named as *the curse of dimensionality*. It is observed that, in high dimensional systems, the estimates provided by the BPF significantly deviate from the ground truth, rendering the algorithm useless in practice. The problem is referred to as *the collapse of the particle filter* within the particle filtering context and it has been studied from various empirical and theoretical viewpoints in recent years. The authors of [14] study the weights of the BPF in a high-dimensional setting and they show that the maximum weight tends to 1 which, they argue, causes the filter to collapse effectively (also see [15, 16] for additional research on the collapse of the BPF). In [17], it is argued that the error bounds for the BPF depend on the dimension exponentially and, to overcome this problem, they propose a block particle filter which requires a model property called decaying correlations.

BPFs can be seen as one of the possible implementations of a more general class of algorithms called sequential Monte Carlo (SMC) methods [18]. In the BPF setup, one generates samples from the transition model only, using a proposal distribution which is the same as the assumed transition model. One can also use different proposal densities, in which more information about the system is encoded, so effectively generating better samples to prevent the collapse. Within this approach, numerous studies have been conducted, leading to, e.g. auxiliary particle filters (APFs) [19], Rao-Blackwellized particle

filters [20], or Gaussian sum particle filters [21].

The collapse of the particle filter has received significant attention, also, in the weather dynamics literature, where models are high-dimensional, obviously approximate, and do not have analytic solutions. In particular, in the data assimilation literature, high-dimensional systems are handled via a well-known operation called *nudging* [22–25]. The history of nudging can be traced back to [22] where it is introduced as pushing the state-variables towards available observations. Since then, nudging has been extensively explored in the data assimilation literature leading to a variety of algorithms (see [26] for a survey).

Over the years, in the filtering and data assimilation literature, the term *nudging* has been increasingly used for operations where state variables (particles in our setup) are pushed towards observations via some observation-dependent operation. If the dimensions of the observations and the hidden states are different, which is often the case, a gain matrix is computed in order to perform the nudging operation. Specifically, in the particle filtering context, [27], and others [28–30] developed different nudging schemes for particle filters. In these works, nudging is performed after the sampling step of the particle filter to push the state-variables towards the observations. In order to compensate for the movement of particles, the weighting step is adjusted so that weights are still unbiased and do not degenerate. It is shown that proposed schemes in [27–30] can track high-dimensional systems with a low number of particles. However, generating samples from proposal requires costly computations for each particle and computation of weights is not trivial from a computational perspective. It is also not trivial to apply the proposed nudging schemes when non-Gaussianity and nontrivial nonlinearities are present in the observation model.

1.2 Contribution

In this work, we propose a modification of the particle filter, termed *nudged particle filter* (NuPF) and assess its performance in high dimensional settings and with misspecified models. Although we use the same idea for nudging that is presented in the literature, our algorithm has subtle but crucial differences.

First, we define the nudging step not just as a relaxation step towards observations but as a step that increases the likelihood of particles strictly. We define the nudging step as an operation aiming at pushing the particles to the regions in the state-space where they have higher likelihoods. This definition paves the way for using different nudging schemes, such as using the gradients of likelihoods or employing random search schemes to move around the state-space. In particular, classical nudging (relaxation) operations arise as a special case of nudging using gradients when the observation model is linear and the likelihood is assumed to be Gaussian.

Second, unlike the other nudging based particle filters, we do not correct the bias induced by the nudging operation during the weighting step. Instead, we compute the weights in the same way they are computed in the standard BPF. However, we carry out the nudging step in a way that preserves the convergence rate of the BPF under mild standard assumptions. In particular, we push only a fraction of particles and leave others untouched, which is key to preserving the Monte Carlo convergence rate, in addition to a

significant computational gain[1]. Also, computing biased weights using only the likelihood function (i.e. regardless of the proposal) is usually much faster than computing full weights as one needs to evaluate only one function for each weight. Since we do not have to perform any heavy computation, the algorithm has almost negligible computational overhead compared to the BPF.

We present computer simulation results on tracking a stochastic chaotic Lorenz 63 attractor with model misspecification, object tracking with a misspecified model involving a heavy-tailed nonlinear observation model, and tracking a high-dimensional Lorenz 96 model. We show that the algorithm can track a high dimensional system without requiring heavy computations and the algorithm can be robust to model misspecification in the transition model provided that the likelihood is well-specified. This is helpful in real-world applications because practitioners often have more control over measurement systems, which determine the likelihood, than they have over the state dynamics.

The paper is structured as follows. After a brief note about notation, we describe the state-space models of interest and the BPF in Section 2. Then in Section 3, we outline our algorithm and the specific nudging schemes we propose to use within the particle filter. We prove a convergence result in Section 4 which shows that our algorithm has the same asymptotic convergence rate as the BPF. We discuss the results of our computer experiments in Section 5, and then, finally, we conclude with Section 6.

1.3 Notation

We denote the real line with $\mathbb{R}$ and $\mathbb{R}^d$ is a d-dimensional product space where d is a positive integer. We denote the set of positive integers with $\mathbb{N}_+$ and the set of positive reals with $\mathbb{R}_+$. We represent the state space with $\mathsf{X} \subset \mathbb{R}^{d_x}$ and the observation space with $\mathsf{Y} \subset \mathbb{R}^{d_y}$. We denote the supremum norm of a function $f : \mathsf{X} \to \mathbb{R}$ by $\|f\|_\infty = \sup_{x \in \mathsf{X}} |f(x)|$. We indicate the space of bounded functions on X with $B(\mathsf{X})$.

In order to denote vectors, we use the shorthand notation $x_{i_1:i_2} = \{x_{i_1}, \ldots, x_{i_2}\}$. For sets of integers, we use $[n] = \{1, \ldots, n\}$. The p-norm of a vector $x \in \mathbb{R}^d$ is defined by $\|x\|_p = (x_1^p + \cdots + x_d^p)^{1/p}$. The L_p norm of a random variable z is denoted with $p \geq 1$ is $\|z\|_p = \left(\int |z|^p P(z) \mathrm{d}z\right)^{1/p}$. We use a shorthand notation for the likelihood, i.e., $g_t(x_t) := g(y_t|x_t)$.

When needed, we use the generic measure theoretic-notation $\mu(\mathrm{d}x)$ to denote the probability measure μ and we abuse the notation to denote its density with respect to Lebesgue measure with $\mu(x)$. We denote the integral with respect to this distribution as $\mu(f) = \int f(x)\mu(x)\mathrm{d}x$. We define the Dirac distribution $\delta_y(\mathrm{d}x)$ located at y as $f(y) = \int f(x)\delta_y(\mathrm{d}x)$. To denote the empirical measure that approximates $\mu(\mathrm{d}x)$, we use $\mu^N(\mathrm{d}x)$ where N denotes the number of Monte Carlo samples used for approximation.

[1]In principle, we can also nudge all particles. However, although it would enjoy the same theoretical properties as nudging only a fraction of particles, it has a much higher computational cost and it is harder to tune empirically. So we do not consider this possibility.

Algorithm 1 Bootstrap Particle Filter

1: Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ and set $W_0^{(i)} \propto 1$.
2: **for** $t \geq 1$ **do**
3: Sampling

$$\bar{x}_t^{(i)} \sim \tau_t(x_t|x_{t-1}^{(i)})$$

for all $i = 1, \ldots, N$.
4: Weighting:

$$w_t^{(i)} = g_t(\bar{x}_t^{(i)})/\bar{Z}_t^{(i)}$$

for all $i = 1, \ldots, N$ where $\bar{Z}_t^{(i)} = \sum_i g_t(\bar{x}_t^{(i)})$.
5: Resampling:

$$x_t^{(i)} \sim \sum_i w_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x)$$

6: **end for**

2 State-space models and bootstrap particle filters

In Section 1, we briefly described our model of interest. We now return to a more detailed description of the class of state-space models, which are our main interest in this paper. We consider the following system,

$$x_0 \sim \pi_0(x_0) \tag{1}$$
$$x_t|x_{t-1} \sim \tau_t(x_t|x_{t-1}) \tag{2}$$
$$y_t|x_t \sim g_t(y_t|x_t). \tag{3}$$

where $x_t \in \mathsf{X}$ and $y_t \in \mathsf{Y}$, and τ_t is a Markov transition kernel on X, and $g_t(y_t|x_t)$ denotes the likelihood. We assume the observation sequence is arbitrary but fixed, hence we use the notation $g_t(y_t|x_t) := g_t(x_t)$ for brevity. We are interested in inferring the conditional distributions of $(x_t)_{t\geq 0}$ given the observation sequence $y_{1:t}$, which are referred as the filtering distributions, denoted with $\pi_t(x_t) := \pi(x_t|y_{1:t})$ for each time t. We denote the filtering measure as $\pi_t(\mathrm{d}x_t)$ with a slight abuse of notation. Similarly, we denote the predictive distribution of the state variable x_t with $\xi(x_t|y_{1:t-1})$ and the predictive measure with $\xi(\mathrm{d}x_t)$. Accordingly, Monte Carlo approximations of the filtering and predictive measures will be denoted with $\pi_t^N(\mathrm{d}x_t)$ and $\xi_t^N(\mathrm{d}x_t)$ respectively.

Bootstrap particle filters are sequential importance sampling algorithms, aiming at obtaining a Monte Carlo based approximation of the filtering distribution for each time t with a fixed computational budget. BPFs propagate the samples according to the assumed dynamic model and modify them upon receiving the most recent observation in a way that the new samples are approximating the filtering distribution of the current time step. Algorithmically, it includes three main steps, called as sampling, weighting, and resampling. A depiction of the algorithm can be seen from Algorithm 1. More compactly, Algorithm 1 proceeds in a three step which can be depicted as,

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N.$$

where ξ_t^N, π_t^N are the approximations of the predictive and filtering distributions respectively. Given a set of samples $\{x_{t-1}^{(i)}\}_{i=1}^N$ approximating the filtering distribution π_{t-1} (denoted with π_{t-1}^N above), the BPF aims at obtaining a new set of samples $\{x_t^{(i)}\}_{i=1}^N$ to approximate π_t (denoted with π_t^N). As depicted in the Algorithm 1, given $\{x_{t-1}^{(i)}\}_{i=1}^N$, the BPF first obtains the particle approximation of the predictive distribution ξ_t via the sampling step. Hence its Monte Carlo approximation provided by the BPF is given as,

$$\xi_t^N(\mathrm{d}x_t) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x_t).$$

Next, the BPF obtains weights by weighting each sample according to the likelihood $g_t(x_t)$. After the weighting step in Algorithm 1, we obtain the weighted distribution, which is an approximation of the filtering distribution, as follows.

$$\tilde{\pi}_t^N(\mathrm{d}x_t) = \sum_{i=1}^{N} w_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x_t).$$

Finally, the BPF employs the resampling step to obtain the another approximation of the filtering distribution

$$\pi_t^N(\mathrm{d}x_t) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_t^{(i)}}(\mathrm{d}x_t).$$

As can be seen from the discussion, both $\tilde{\pi}_t^N$ and π_t^N can be used to approximate expectations with respect to the filtering distribution, however, the former is argued to be providing estimates with less variance. The resampling step commonly employed to prevent degeneracy and it is an optional step [4].

Using the empirical distributions the BPF constructs, the expectation of a test function $f \in B(\mathsf{X})$ with respect to the filtering distribution can be estimated using the weighted distribution,

$$\pi_t(f) = \int f(x_t)\pi_t(\mathrm{d}x_t) \approx \sum_{i=1}^{N} w_t^{(i)} f(\bar{x}_t^{(i)}) = \tilde{\pi}_t^N(f)$$

where $w_t^{(i)}$ are normalized weights as given in the Algorithm 1. Alternatively, one can estimate the expectation using the resampled distribution,

$$\pi_t(f) \approx \pi_t^N(f) = \frac{1}{N}\sum_{i=1}^{N} f(x_t^{(i)}).$$

Convergence results mentioned in the introduction are usually proven for expectations with respect to bounded test functions f, see e.g. [6, 10, 11]

BPFs can be extended for more general, possibly observation-dependent proposals, known more generally as SMC algorithms. Instead of drawing samples from $\tau_t(x_t|x_{t-1})$, we can draw samples from a general proposal $q_t(x_t|x_{1:t-1}^{(i)}, y_t)$ which includes (possibly)

past samples and observations. Then, evaluation of weights become more elaborate, i.e., the unnormalized weights must be computed as [4],

$$W_t^{(i)} = \frac{g_t(\bar{x}_t^{(i)})\tau_t(\bar{x}_t^{(i)}|x_t^{(i)})}{q_t(\bar{x}_t^{(i)}|x_{1:t-1}^{(i)}, y_t)}$$

and the rest of the algorithm can be employed in the same way. This point is related to our algorithm as we also use the most recent observation to generate new samples. However, unlike in a classical SMC scheme where weights should be modified accordingly with respect to the proposal, we leave weights same as the BPF setting – an idea we will discuss in the next section.

3 Nudged Particle Filter

As we mentioned before, the nudged particle filter differs from the BPF with an extra step, *the nudging step*, employed after the sampling step. It is a mechanism to push particles to the regions in state-space where they have higher likelihoods. Since it is employed right after the sampling step, it can be seen as a modification of the proposal, employed in two-steps. However, unlike other approaches which use different proposals, we do not correct the effect of using a different proposal in the weighting step. We then carry out the weighting step as if we use the proposal as in the BPF, i.e., as if the proposal is the transition density itself. In effect, doing so introduces some bias without compensating for it. However, we only employ this step to a small subset of particles, leaving others untouched. We show that, provided that the nudging step is employed carefully, it is possible to preserve the usual convergence rate of the BPF, but with a worse constant than usual. This means that, asymptotically, our algorithm provides same performance with the BPF. In practice, however, there are significant differences and the step improves the performance significantly, as can be seen from Section 5.

From a general viewpoint, we can illustrate the nudged particle filter in four steps,

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{nudging}} \tilde{\xi}_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N$$

where we have an extra step after the sampling step. The general algorithm is given in Algorithm 2.

Next, we define the nudging operator in a precise way. In order to capture a general class of nudging operations, we define the nudging operator as follows.

Definition 1. *The nudging operator* $\alpha_t : \mathsf{X} \to \mathsf{X}$ *associated with the likelihood function* $g_t(x)$ *is a mapping such that*

$$\text{if} \quad x' = \alpha_t(x) \quad \text{then} \quad g_t(x') \geq g_t(x) \tag{4}$$

for every $x, x' \in \mathsf{X}$.

Intuitively, we define nudging as an operation that increases the likelihood deterministically. There are many possible ways to perform it in practice, we will describe some of them in the next section.

Algorithm 2 Nudged Particle Filter

1: Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ and set $W_0^{(i)} \propto 1$.
2: **for** $t \geq 1$ **do**
3: Sampling:

$$\bar{x}_t^{(i)} \sim \tau_t(x_t | x_{t-1}^{(i)})$$

for all $i = 1, \ldots, N$.
4: **Nudging**:

$$\tilde{x}_t^{(i)} = \alpha_t(\bar{x}_{t-1}^{(i)})$$

for $i \in \mathcal{I}$ where $\mathcal{I} \subset [N]$.
5: Weighting:

$$w_t^{(i)} = g_t(\tilde{x}_t^{(i)}) / \tilde{Z}_t^{(i)}$$

for all $i = 1, \ldots, N$ where $\tilde{Z}_t^{(i)} = \sum_i g(\tilde{x}_t^{(i)})$.
6: Resample:

$$x_t^{(i)} \sim \sum_i w_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x)$$

7: **end for**

The nudging step is implemented as follows. We choose $M \leq N$ number of particles to be pushed around the state-space. For theoretical reasons which we will make clear in the next section, this number is usually chosen so that $M \leq \sqrt{N}$. There are two important aspects of the step where different choices are possible. First, we have to decide how to choose the particles to be nudged. Second, we need to decide how to nudge the particles.

3.1 How to choose the particles to be nudged

In this paper, we identify two ways of choosing the particles to be modified. The ways we propose to choose particles are straightforward. More concretely, we propose two random mechanisms to choose particles, one of which suitable for parallel implementations of particle filters.

3.1.1 Batch nudging

In the first way, one can push M randomly chosen particles at once. So one can choose M unique indices between $[1, N]$ at once and apply the operator α_t to each of them. The index set of nudged particle is denoted with $\mathcal{I}$. Algorithmically, the step can be seen from Algorithm 3. We refer to this scheme as *batch nudging*, referring to selection of the indices at once.

Algorithm 3 Batch nudging

1: Choose $\mathcal{I} = \{i_1, \ldots, i_M\} \sim [N]$ without replacement.
2: Set

$$\tilde{x}_t^{(i_k)} = \alpha_t(\bar{x}_t^{(i_k)})$$

for $i_k \in \mathcal{I}$.

3.1.2 Independent nudging

Second way, which is suitable for parallel implementations, would be to nudge particles independently. One can choose each particle for some probability, which will make the nudging step trivial to implement under parallel implementations. The step can be seen from Algorithm 4.

Algorithm 4 Independent nudging

1: **for** $i \in [N]$ **do**

$$\tilde{x}_t^{(i)} = \alpha_t(\bar{x}_t^{(i)})$$

with probability M/N.

2: **end for**

This scheme, which we call as *independent nudging*, is based on the idea that we would like to nudge M particles in expectation. Therefore $\mathcal{I}$ would be a random index set with random cardinality. The price to pay for enabling parallelization is that we cannot be sure that M is below a certain number.

3.2 How to nudge

As it follows from our definition, the nudging step is aiming at increasing the likelihood of individual particles. This means that we need operations for maximizing the likelihood for individual particles we fixed to nudge. Here, one straightforward idea is to employ steps where the likelihood is maximized, in the same way that one would do when estimating the maximum likelihood solution. However, we do not want to go all the way down to the maximum likelihood solution (this would result in degeneracy). Also, to keep the algorithm lightweight, we focus on single steps where the likelihood is increased.

3.2.1 Gradient nudging

In principle, one can apply single step of any operator α_t whose fixed points are solutions to the problem,

$$\max_{x_t \in \mathsf{X}} g_t(x_t)$$

i.e. any operator that move particles in the direction of the maximum likelihood estimate. If $g_t(x_t)$ is a differentiable function of x_t, one straightforward way to do nudging would be to take gradient steps. The generic *gradient nudging* step can be seen from Algorithm 5.

Algorithm 5 Gradient nudging

1: **for** $i \in \mathcal{I}$ **do**

$$\tilde{x}_t^{(i)} = \bar{x}_t^{(i)} + \gamma \nabla_{x_t} g_t(\bar{x}_t^{(i)}).$$

2: **end for**

Alternatively, one can take more gradient steps as well, but in this work, we limit it to a single step in order to keep the step lightweight. Note that, it is also possible to

use different operators that target the maximum of $g_t(x)$, such as single steps of similar operators that target the maximum of the likelihood. But we leave the investigation of those options to future work.

3.2.2 Random search nudging

Another option is to use gradient-free techniques to push particles towards regions where they have higher likelihoods. For example, a stochastic search based technique can be employed [31]. This nudging scheme can be seen from Algorithm 6. We call this scheme as *random search nudging*.

Algorithm 6 Random search nudging

1: **repeat**
2: Perturb $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)} + \eta_t$ where $\eta_t \sim \mathcal{N}(0, C)$ for some covariance matrix C.
3: Keep $\tilde{x}_t^{(i)}$ if $g_t(\tilde{x}_t^{(i)}) > g_t(\bar{x}_t^{(i)})$, otherwise $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)}$.
4: **until** the particle is nudged.

3.2.3 Model specific nudging

Finally, one can nudge particles using the specific model information. If the model variables have a tractable relationship to each other, one can modify the position of unobserved state variables with respect to the observed ones. For example, in an object tracking scenario, positions and velocities have a special and simple physical relationship but usually only position variables are observed through a linear or nonlinear transformation. In this case, nudging would only effect position variables. However, using position variables, one can also nudge velocity variables with simple rules. We discuss this idea in the object tracking experiment and show its utility in the Section 5.

4 Analysis

In this section, we prove a convergence result for our algorithm by using well-known tools from the literature. In particular, we derive an L_p bound for the error made at each time step. The proofs of L_p convergence for the BPFs are well known, see e.g. [32]. Since our algorithm differs from the bootstrap particle filter with a single step, it suffices to bound the nudging step in order to prove an L_p convergence result. For the proof of Theorem 1, we only sketch the bounds for other steps, except the nudging step, for which we detail the proof.

4.1 An L_p convergence result

Assumption 1. *The likelihood function is positive and bounded, i.e.,*

$$g_t(x_t) > 0 \quad \text{and} \quad \|g_t\|_\infty = \sup_{x_t \in \mathsf{X}} |g_t(x_t)| < \infty$$

for $t = 1, \ldots, T$.

Theorem 1. *Let $y_{1:T}$ be arbitrary but fixed and choose any $0 < t \leq T$. Let $f \in B(\mathsf{X})$ and fix the number of nudged particles $M \in \mathbb{N}_+$ such that $M \leq \sqrt{N}$. If Assumption 1 holds, then*

$$\|\pi_t^N(f) - \pi_t(f)\|_p \leq \frac{c_t \|f\|_\infty}{\sqrt{N}}$$

where $c_t < \infty$ is a constant independent of N.

Proof. We prove the result by induction. Note that, for the case of $t = 0$, the bound is satisfied trivially as $\pi_0^N(f)$ will be an i.i.d. approximation of the expectation $\pi_0(f)$, we have [33],

$$\left\|\pi_0(f) - \pi_0^N(f)\right\|_p \leq \frac{c_0 \|f\|_\infty}{\sqrt{N}}.$$

where $c_0 < \infty$ is a constant independent of N. Now, as an induction hypothesis, let us assume that,

$$\left\|\pi_{t-1}^N(f) - \pi_{t-1}(f)\right\|_p \leq \frac{c_{t-1} \|f\|_\infty}{\sqrt{N}}$$

for some constant $c_{t-1} < \infty$ independent of N. Given the particles from π_{t-1}^N, particle filters first employ a sampling step, that is proposing new particles to obtain the predictive measure ξ_t^N. Through a combinatorial argument, it is possible to show that (see e.g. [32, Lemma 1]),

$$\left\|\xi_t^N(f) - \xi_t(f)\right\|_p \leq \frac{c_{1,t} \|f\|_\infty}{\sqrt{N}}. \tag{5}$$

where $c_{1,t} < \infty$ is a constant independent of N. Next, we analyze the error introduced in the nudging step. Let $\{\bar{x}_t^{(i)}\}_{i=1}^N$ be the particles of the empirical predictive distribution ξ_t^N and $\{\tilde{x}_t^{(i)}\}_{i=1}^N$ be the particles of the nudged distribution $\tilde{\xi}_t^N$. There are M nudged particles. Therefore,

$$\left\|\xi_t^N(f) - \tilde{\xi}_t^N(f)\right\|_p = \left\|\frac{1}{N}\sum_{i \in \mathcal{I}} \left(f(\bar{x}_t^{(i)}) - f(\tilde{x}_t^{(i)})\right)\right\|_p \leq \frac{2\|f\|_\infty M}{N} \leq \frac{2\|f\|_\infty}{\sqrt{N}} \tag{6}$$

where the first inequality holds because $|f(x) - f(x')| \leq 2\|f\|_\infty$ for every $(x, x') \in \mathsf{X} \times \mathsf{X}$ and the second inequality follows from the construction of the algorithm where we chose $M \leq \sqrt{N}$. Combining (5) and (6), we arrive at,

$$\left\|\xi_t(f) - \tilde{\xi}_t^N(f)\right\| \leq \frac{\tilde{c}_{1,t} \|f\|_\infty}{\sqrt{N}} \tag{7}$$

where $\tilde{c}_{1,t} = 2 + c_{1,t} < \infty$ is a constant independent of N. The rest of the proof is standard. Following [32, Lemma 1], we readily obtain that,

$$\left\|\pi_t(f) - \tilde{\pi}_t^N(f)\right\|_p \leq \frac{c_{2,t} \|f\|_\infty}{\sqrt{N}}, \tag{8}$$

where

$$c_{2,t} = \frac{2\|g_t\|_\infty \tilde{c}_{1,t}}{\xi_t(g_t)} < \infty$$

is a constant independent of N but depends on $\tilde{c}_{1,t}$ and the upper bound of the likelihood which we assumed to be finite. Note that our assumptions imply $\xi_t(g_t) > 0$. The analysis of the resampling step is also standard. Given $\tilde{\pi}_t^N(f)$, we obtain the particle approximation of the expectation with respect to the filter measure $\pi_t^N(f)$ by employing the resampling step. If we perform multinomial resampling, the same argument in the base case yields

$$\left\|\tilde{\pi}_t^N(f) - \pi_t^N(f)\right\|_p \leq \frac{c_{3,t}\|f\|_\infty}{\sqrt{N}} \tag{9}$$

where $c_{3,t} < \infty$ is a constant independent of N. Now we can combine bounds (8) and (9) and conclude,

$$\left\|\pi_t^N(f) - \pi_t(f)\right\|_p \leq \frac{c_t\|f\|_\infty}{\sqrt{N}}$$

which concludes the proof. ■

We can also analyze specific choices of nudging schemes. The easiest setup to consider is gradient nudging. Within this scheme, we push the particles to the regions where they have higher likelihood using the gradient information. In what follows, we derive the bound specific to the batch gradient nudging.

Assumption 2. *The gradient of the likelihood is bounded, in particular,* $\|\nabla_x g_t(x)\|_2 \leq G < \infty$ *for every* $x \in \mathsf{X}$.

Theorem 2. *Fix a step-size* $\gamma \in \mathbb{R}_+$ *and the number of nudged particles* $M \in \mathbb{N}_+$ *such that* $\gamma M \leq \sqrt{N}$. *If Assumption 2 holds and* f *is a Lipschitz test function, then the error introduced by the batch gradient nudging step can be bounded as,*

$$\left\|\xi_t^N(f) - \tilde{\xi}_t^N(f)\right\|_p \leq \frac{LG}{\sqrt{N}}$$

where L *is the Lipschitz constant of* f.

Proof. Notice that since $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)} + \gamma \nabla_{x_t} g_t(\bar{x}_t^{(i)})$, we arrive at,

$$\begin{aligned}\left|f(\tilde{x}_t^{(i)}) - f(\bar{x}_t^{(i)})\right| &\leq L\left\|\tilde{x}_t^{(i)} - \bar{x}_t^{(i)}\right\|_2 \\ &= L\gamma\left\|\nabla_{x_t} g(x_t^{(i)})\right\|_2 \\ &\leq \gamma LG \end{aligned} \tag{10}$$

where the first line follows from the Lipschitz assumption and the equality in the second line follows from the implementation of the nudging step as a gradient step and finally

the last inequality follows from the Assumption 2. Then we have,

$$\begin{aligned}\left\|\xi_t^N(f) - \tilde{\xi}_t^N(f)\right\|_p &= \left\|\frac{1}{N}\sum_{i\in\mathcal{I}}\left(f(\bar{x}_t^{(i)}) - f(\tilde{x}_t^{(i)})\right)\right\|_p \\ &\leq \frac{1}{N}\sum_{i\in\mathcal{I}}\left\|f(\bar{x}_t^{(i)}) - f(\tilde{x}_t^{(i)})\right\|_p \\ &\leq \frac{M}{N}\gamma LG\end{aligned}$$

where the first inequality follows from Minkowski's inequality and the second inequality follows from (10). Here, by assumption, we have γ and M so that $\gamma M \leq \sqrt{N}$. So we have,

$$\left\|\xi_t^N(f) - \tilde{\xi}_t^N(f)\right\|_p \leq \frac{LG}{\sqrt{N}}.$$

which concludes the proof. ■

This theorem provides important intuition on our method. First, given Theorem 2, by the same argument used in Theorem 1, we can obtain the convergence of the filter π_t^N by replacing the bound in the nudging step with the one we have in the latter theorem. This would be of same rate of convergence as in Theorem 1. However, it suggests a way to choose M and γ. In particular, one can choose $M = N^\beta$ where $0 < \beta < 1$. This would suggest choosing $\gamma \leq N^{\frac{1}{2}-\beta}$ in order to have $\gamma M \leq \sqrt{N}$. One can also choose $\gamma \leq CN^{\frac{1}{2}-\beta}$ so that one can allow for some flexibility and still guarantee $\gamma M \leq C\sqrt{N}$ for some constant $C > 1$. More concretely, this theorem provides us with a heuristic to balance the step size with the number of nudged particles. We can increase the number of nudged particles but in that case we need to shrink the step size accordingly so as to keep $\gamma M \leq \sqrt{N}$. Second, instead of the gradient of the likelihood, one can use the gradient of the log-likelihood if the likelihood comes from the exponential family densities. Third, by using standard tools, one can prove a uniform convergence result in the same vein of [11] without extra trouble in addition to what is available in the literature.

5 Numerical simulations

In this section, we conduct three experiments. In the first experiment, we use a misspecified chaotic Lorenz 63 model. In the second experiment, we consider an object tracking scenario where the model is again misspecified, the observation model is nonlinear and the observation noise is heavy-tailed. Finally, we describe an experiment where we demonstrate the results on a high-dimensional Lorenz 96 model in order to show that our algorithm is capable of estimating high-dimensional systems[2].

For all experiments, we have chosen $M = \sqrt{N}$. This ensures the validity of the theoretical results we have presented in Section 4. For the object tracking experiment, we have used a large-step size, but note that this does not change the convergence rate as well, see the discussion following the Theorem 2.

[2]For the experiments involving Lorenz 96 model, simulation from the model is implemented in C++ and integrated to Matlab. The rest of the algorithms are implemented in Matlab.

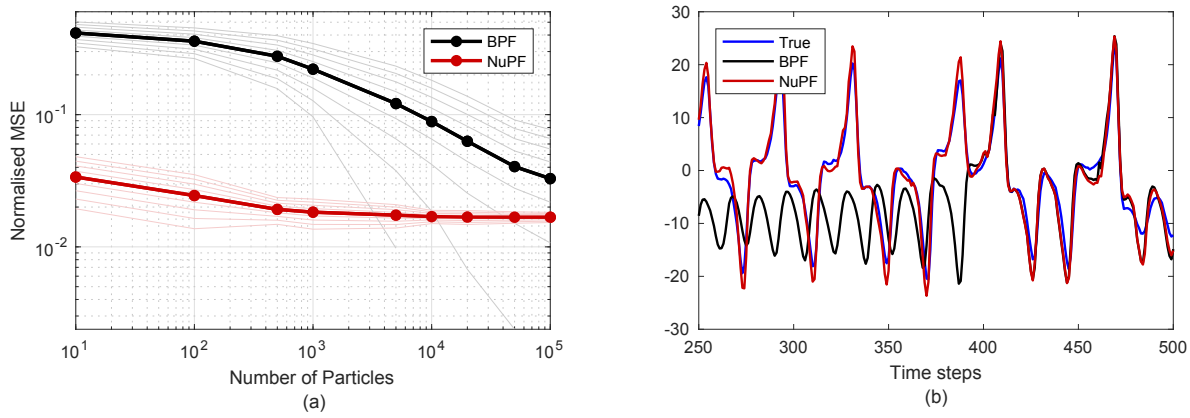

Figure 1: (a) Normalised mean squared error (NMSE) results of Lorenz 63 system. The results obtained with 1000 Monte Carlo runs for each $N \in \{10, 100, 500, 1K, 5K, 10K, 20K, 50K, 100K\}$. Dashes mark up to 1 standard deviation (σ). For the misspecified parameter $\mathsf{b}_\epsilon = \mathsf{b} + \epsilon$, we set $\mathsf{b} = 8/3$ and $\epsilon = 0.75$. (b) A sample path from estimation of an *unobserved* dimension (second dimension) in a run where $N = 500$.

5.1 Misspecified stochastic Lorenz 63 model

In this experiment, we demonstrate the algorithm on a misspecified stochastic Lorenz 63 model. In continuous time, the model equations can be written as a stochastic differential equation in three dimensions,

$$\begin{aligned}
\mathrm{d}x_1 &= -\mathsf{s}(x_1 - x_2) + \mathrm{d}w_1, \\
\mathrm{d}x_2 &= \mathsf{r}x_1 - x_2 - x_1 x_3 + \mathrm{d}w_2, \\
\mathrm{d}x_3 &= x_1 x_2 - \mathsf{b}x_3 + \mathrm{d}w_3,
\end{aligned}$$

where $\{w_i(s)\}_{s\in(0,\infty)}$ for $i = 1, 2, 3$ are unidimensional Wiener processes and $(\mathsf{s}, \mathsf{r}, \mathsf{b}) \in \mathbb{R}$ are model parameters. We discretise the model using the Euler-Maruyama scheme with integration step $\mathsf{T}_e > 0$ and obtain the following discrete-time system,

$$\begin{aligned}
x_{1,t} &= x_{1,t-1} - \mathsf{T}_e\mathsf{s}(x_{1,t-1} - x_{2,t-1}) + \sqrt{\mathsf{T}_e}u_{1,t} \\
x_{2,t} &= x_{2,t-1} + \mathsf{T}_e(\mathsf{r}x_{1,t-1} - x_{2,t-1} - x_{1,t-1}x_{3,t-1}) + \sqrt{\mathsf{T}_e}u_{2,t} \\
x_{3,t} &= x_{3,t-1} + \mathsf{T}_e(x_{1,t-1}x_{2,t-1} - \mathsf{b}x_{3,t-1}) + \sqrt{\mathsf{T}_e}u_{3,t}
\end{aligned}$$

where $(u_{i,t})_{t\in\mathbb{N}}$, $i = 1, 2, 3$ are i.i.d. Gaussian random variables with zero mean and unit variance. We assume that the system is partially observed *both through time and space*. We assume that we can partially observe only one dimension from the system which is the first one. So essentially we have an observation model,

$$y_n = k_o x_{1,t_s n} + v_n$$

where $(v_n)_{n\in\mathbb{N}}$ is a sequence of i.i.d. Gaussian random variables with zero mean and unit variance and k_o assumed known.

We set parameters $(\mathsf{s}, \mathsf{r}, \mathsf{b})$ in a certain way which makes the system chaotic. In particular, while generating the true data, we set

$$(\mathsf{s}, \mathsf{r}, \mathsf{b}) = \left(10, 28, \frac{8}{3}\right).$$

Moreover, we choose a specific initial condition which is known to be generating chaotic behaviour $x_0 = [-5.91652, -5.52332, 24.5723]^\top$ [32]. We assume that the system is observed every t_s time steps which is set to $t_s = 40$. We have simulated the system for $T = 20000$. Since $t_s = 40$, we have T/t_s number observations from the system where in this example it is 500.

We misspecify the model while running the filtering algorithms. We can choose to misspecify any parameter and in this work we choose to misspecify b. In particular, we run the filters with $\mathsf{b}_\epsilon = \mathsf{b} + \epsilon$ where we set $\epsilon = 0.75$ which amounts to a serious misspecification considering that the system is chaotic and any small perturbation would effect estimation performance significantly.

We have implemented the NuPF with independent gradient nudging with $\gamma = 0.75$. Normalised mean square errors (NMSE) and a sample path from estimated trajectories can be seen in Fig. 1. The experiment shows that the NuPF is able to track a misspecified model whereas the BPF catches the performance of the NuPF with low number of particles only with a large number of particles.

5.2 Object tracking with a misspecified model

In this experiment, we consider an object tracking scenario where objects are observed through nonlinear measurement models (sensors) with heavy-tailed noise. The object of interest is following a model,

$$x_t = Ax_{t-1} + BL(x_{t-1} - x_{\text{target}}) + v_t$$

where $x_t \in \mathbb{R}^4$ denotes the state of the object, consisted of two coordinates and two velocity states. The variable x_{target} is the deterministic, pre-chosen target state. The sequence $(v_t)_{t\in\mathbb{N}}$ here is a sequence of zero-mean Gaussian random variables with covariance matrix Q. The parameters A, B, Q are set as,

$$A = \begin{bmatrix} I_2 & \kappa I_2 \\ 0 & 0.99 I_2 \end{bmatrix} \quad B = \begin{bmatrix} 0 & I_2 \end{bmatrix}^\top \quad Q = \begin{bmatrix} \frac{\kappa^3}{3} I_2 & \frac{\kappa^2}{2} I_2 \\ \frac{\kappa^2}{2} I_2 & \kappa I_2 \end{bmatrix}$$

where I_2 is 2×2 identity matrix and $\kappa = 0.04$. The object is simulated so that it follows a policy and the policy matrix $L \in \mathbb{R}^{2\times 4}$ is found by solving the Riccati equation, see [34] for details. In particular, the object tries to reach a certain state which results in a highly maneuvering trajectory. The nudged particle filter, on the other hand, is not informed about the true model and the policy it follows, it only assumes the following model,

$$x_t = Ax_{t-1} + v_t$$

as the simplest model to assume. Observations are assumed to be nonlinear and coming from 10 sensors placed around the object[3]. For each sensor, the observations are modelled

[3]With a well-specified model, fewer sensors would suffice. But we found that with misspecification, more observations are needed.

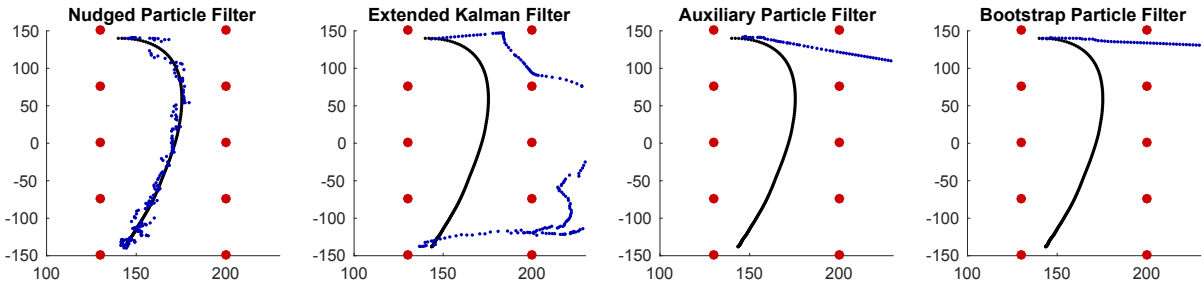

Figure 2: Visualisation of a sample run of the experiment for different algorithms using 500 particles. Black dots denote the real trajectory of the object, red dots are sensors where observations were collected, and blue points are estimates provided by the different algorithms. As can be seen from the left, NuPF manages to closely track the target, while other algorithms collapse. EKF is the second best performing algorithm thanks to the gradient information, however, Gaussianity assumptions cause the collapse. We have run this experiment for 10000 Monte Carlo runs. Average errors are tabulated in Table 1.

Algorithm	BPF	APF	EKF	NuPF
NMSE	0.5418	0.5418	0.2087	**0.0402**

Table 1: Average NMSE values for the algorithms on object tracking problem with 10000 Monte Carlo runs.

in the following way,

$$y_{t,i} = 10 \log_{10} \left(\frac{P_0}{\|r_t - s_i\|^2} + \eta \right) + w_{t,i}$$

where $r_t = [x_{1,t}, x_{2,t}]^\top$ and s_i is the position of the ith sensor and $w_{t,i} \sim \mathcal{MT}(\mu, \Sigma, \nu)$ assumed to be a Multivariate-t distributed random variable for each $i = \{1, \ldots, 10\}$, see [35] for a definition of the Multivariate-t distribution. We choose $\mu = 0$ and $\Sigma = I$, meaning that sensor observations are zero-mean and are not correlated. Intuitively, the closer the parameter ν to 1, the more explosive observations become. In particular, we set $\nu = 1.01$ to make the observations explosive and heavy-tailed to make the problem more challenging. For the sensor parameters, we set $\eta = 10^{-9}$ and $P_0 = 1$.

We have implemented the nudged particle filter with batch gradient nudging with a large-step size $\gamma = 5.5$, with an additional model specific nudging step. In other words, in addition to the batch gradient step, we also used model specific information to further nudge unobserved state variables. In particular, after nudging the observed variables $x^{(i)}_{1:2,t}$, we nudged the rest of the state variables as,

$$x^{(i)}_{3:4,t} = \frac{1}{\kappa}(x^{(i)}_{1:2,t} - x^{(i)}_{1:2,t-1}).$$

The motivation behind this nudging comes from the physical relationship between positions and velocities. We note, however, without this special nudging, the algorithm also outperforms other algorithms we mention in the next paragraph.

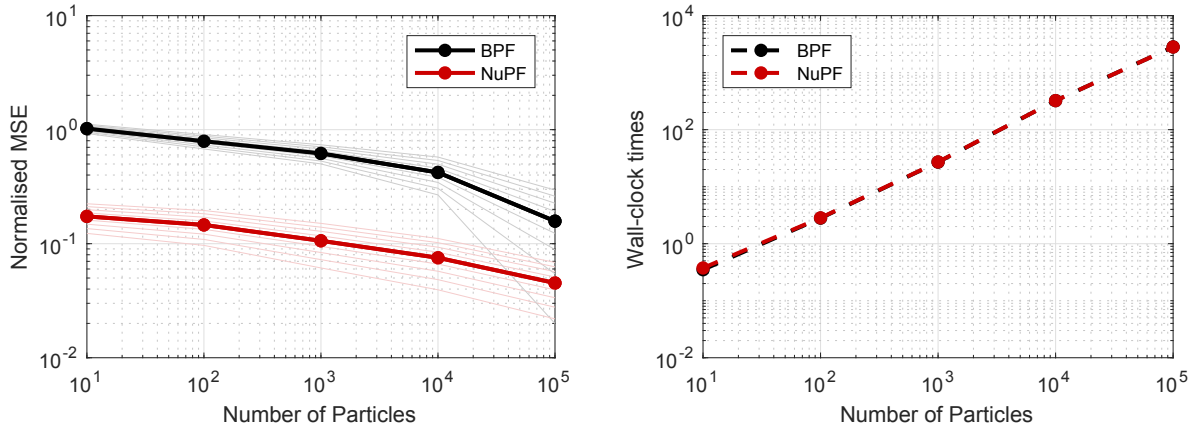

Figure 3: Comparison of NuPF and BPF for the Lorenz 96 model where model dimension $d = 40$ for 1024 Monte Carlo runs. On the left, NMSEs are plotted with respect to changing number of particles for $N \in \{10, 100, 1K, 10K\}$. Dashes mark up to 1 standard deviation (σ). It can be seen that NuPF provides much better performance for the same number of particles. On the right, we compared runtimes of BPF and NuPF for the same experiment. NuPF is only marginally slower than BPF so that it basically improves the performance with virtually no extra computational cost.

We have run 10000 Monte Carlo runs while setting the number of particles $N = 500$ for the auxiliary particle filter (APF), the BPF, and the NuPF. For comparison, we have also implemented the extended Kalman filter (EKF) which also uses the gradient of the observation model. An illustrative run of a single experiment can be seen from Fig. 2 and average NMSEs can be seen from Table 1. In particular, from Fig. 2, we can draw some insights. Although, the EKF also uses the gradient information of the observation model, there is no way to consider the nature of the noise distribution as the EKF approximates it as a Gaussian. BPF and APF collapse, simply, due to model mismatch in the transition model and we have not been able to see any significant difference between them, despite APF also uses the most recent observation to propose new samples.

5.3 High-dimensional stochastic Lorenz 96 model

In this experiment, we compare the nudged particle filter with the ensemble Kalman filter (EnKF) to show that the NuPF scales to high dimensions with a computational budget which is linear in dimension.

We consider Lorenz 96 model which is defined as follows,

$$\mathrm{d}x_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F + \mathrm{d}w_i$$

where $(w_i(s))_{s\in(0,\infty)}$ are Wiener processes for $i = 1, \ldots, d$ where d is the dimension of the system. We set $F = 8$ which is known to be generating a chaotic regime. The model is assumed to have a circular structure, so that $x_{-1} = x_{d-1}$, $x_0 = x_d$, and $x_{d+1} = x_1$. In order to simulate data from this model, we apply the Euler-Maruyama discretization scheme and obtain a discrete-time model,

$$x_{i,t} = x_{i,t-1} + \mathsf{T}_e((x_{i+1,t-1} - x_{i-2,t-1})x_{i-1,t-1} - x_{i,t-1} + F) + \sqrt{\mathsf{T}_e}u_{i,t}$$

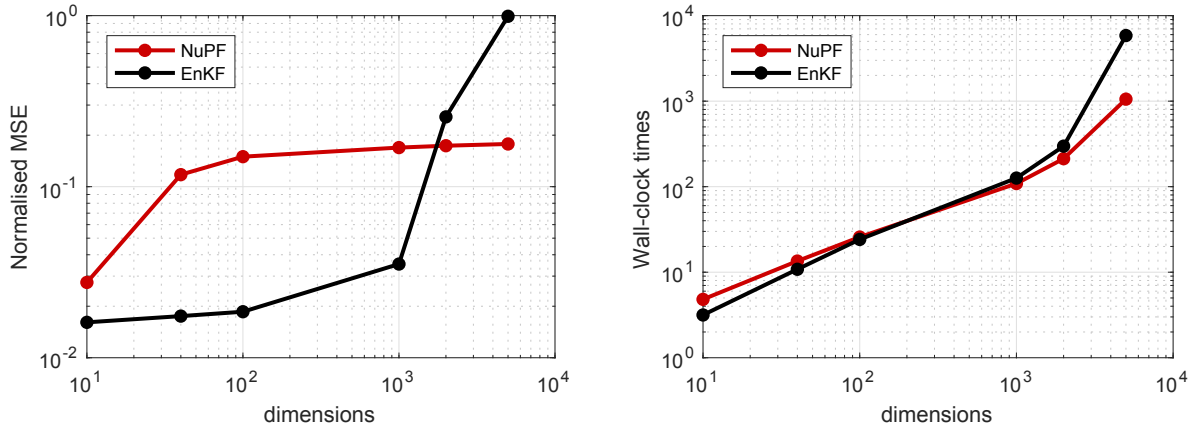

Figure 4: Comparison of NuPF with EnKF for the Lorenz 96 model with varying dimensions. More specifically, we changed the dimension $d = [10, 40, 100, 1000, 2000, 5000]$ while $N = 500$ kept fixed, we have run the experiment for 1000 Monte Carlo runs. On the left, the performance of the algorithms with increasing dimensions can be seen. EnKF with 500 ensemble members is superior to NuPF in terms of error up to 1000 dimensions. However, then it collapses while NuPF attains the same performance with increasing dimensions. On the right, the running times for the same experiments can be seen. The computational cost of EnKF increases drastically with dimension while NuPF has a linear increase, basically because the system dimension is bigger so the dimensions of particles are bigger.

where $u_{i,t}$ are zero-mean, unit-variance Gaussian random variables. We assume that this model is also observed partially through both time and space. In particular, we assume that we observe half of the state variables, with a certain period,

$$y_{k,n} = x_{(2k-1),t_s n} + v_n$$

where $t_s = 10$ and $(v_n)_{n\in\mathbb{N}}$ are zero-mean Gaussian random variables with variance $r = 1$ (observation noise variance).

We consider two experiments for Lorenz 96 model. For both experiments, we have run the NuPF with batch gradient nudging with a step size $\gamma = 0.075$. In the first experiment, we fix the dimension $d = 40$ and run the BPF and the NuPF with increasing number of particles. This experiment is mainly intended to show how computational load of the NuPF and the BPF compares to each other. The results can be seen from Fig. 3. We show that the NuPF has a superior performance to the BPF in terms of NMSEs while the running times of both algorithms are almost the same. In fact, the difference between running times is negligible.

For the second experiment, we compare the NuPF with the EnKF, which is the standard choice for data assimilation problems. The results can be seen from Fig. 4. In particular, we show that, as we increase the dimensions with fixed number of particles and ensemble members ($N = 500$), EnKF collapses as its performance depends on the number of ensemble members and as with increasing dimension, it needs more computational power whereas NuPF has a consistent performance among varying dimensions. We found that NuPF is also able to track higher dimensional systems with low number of particles,

similar to what has been reported in [29] and [30].

6 Conclusions

We proposed a modification of the particle filter which is capable of operating under model misspecification and estimating high-dimensional systems without needing heavy computations. We showed that the algorithm enjoys the same convergence rate with the BPF, it has almost negligible computational overhead compared to the BPF, and it enjoys much better empirical performance than the BPF. Our algorithmic step is general and generic and it can be used with any differentiable or non-differentiable likelihood. Since this scheme does not require any modification of the well-defined steps of the particle filter, it can be plugged into a variety of particle filters. We think that elaboration of the nudging step can provide a number of possibilities also for SMC algorithms. Just as the resampling step, which is employed solely for numerical reasons, the nudging step can become a standard step for improving the performance of the particle filters. We hope to pursue different directions implied by our algorithm in near future.

Acknowledgements

This work was partially supported by *Ministerio de Economía y Competitividad* of Spain (TEC2015-69868-C2-1-R ADVENTURE), the Office of Naval Research Global (N62909-15-1-2011), and the regional government of Madrid (program CASICAM-CM S2013/ICE-2845).

References

[1] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

[2] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[3] Jun S Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998.

[4] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[5] Nando de Freitas, Arnaud Doucet, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo in Practice*. Springer, 2001.

[6] Pierre Del Moral and Alice Guionnet. Central limit theorem for nonlinear filtering and interacting particle systems. *Annals of Applied Probability*, pages 275–297, 1999.

[7] Nicolas Chopin. Central limit theorem for sequential monte carlo methods and its application to bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.

[8] Hans R Künsch. Recursive monte carlo filters: algorithms and theoretical analysis. *Annals of Statistics*, pages 1983–2021, 2005.

[9] Randal Douc and Eric Moulines. Limit theorems for weighted samples with applications to sequential monte carlo methods. *Annals of Statistics*, pages 2344–2376, 2008.

[10] Pierre Del Moral and Alice Guionnet. On the stability of interacting processes with applications to filtering and genetic algorithms. In *Annales de l'Institut Henri Poincare (B) Probability and Statistics*, volume 37, pages 155–194. Elsevier, 2001.

[11] Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, 2004.

[12] Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[13] Pierre Del Moral. *Mean field simulation for Monte Carlo integration*. CRC Press, 2013.

[14] Thomas Bengtsson, Peter Bickel, and Bo Li. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In *Probability and statistics: Essays in honor of David A. Freedman*, pages 316–334. Institute of Mathematical Statistics, 2008.

[15] Chris Snyder, Thomas Bengtsson, Peter Bickel, and Jeff Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640, 2008.

[16] Peter Bickel, Bo Li, Thomas Bengtsson, et al. Sharp failure rates for the bootstrap particle filter in high dimensions. In *Pushing the limits of contemporary statistics: Contributions in honor of Jayanta K. Ghosh*, pages 318–329. Institute of Mathematical Statistics, 2008.

[17] Patrick Rebeschini, Ramon Van Handel, et al. Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866, 2015.

[18] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. Springer, 2001.

[19] Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

[20] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000.

[21] Jayesh H Kotecha and Petar M Djuric. Gaussian sum particle filtering. *IEEE Transactions on signal processing*, 51(10):2602–2612, 2003.

[22] James E Hoke and Richard A Anthes. The initialization of numerical models by a dynamic-initialization technique. *Monthly Weather Review*, 104(12):1551–1556, 1976.

[23] Paola Malanotte-Rizzoli and William R Holland. Data constraints applied to models of the ocean general circulation. part i: The steady case. *Journal of physical oceanography*, 16(10):1665–1682, 1986.

[24] Paola Malanotte-Rizzoli and William R Holland. Data constraints applied to models of the ocean general circulation. part ii: the transient, eddy-resolving case. *Journal of physical oceanography*, 18(8):1093–1107, 1988.

[25] X Zou, IM Navon, and FX LeDimet. An optimal nudging data assimilation scheme using parameter estimation. *Quarterly Journal of the Royal Meteorological Society*, 118(508):1163–1186, 1992.

[26] S Lakshmivarahan and John M Lewis. Nudging methods: a critical overview. In *Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications (Vol. II)*, pages 27–57. Springer, 2013.

[27] Peter Jan Van Leeuwen. Particle filtering in geophysical systems. *Monthly Weather Review*, 137(12):4089–4114, 2009.

[28] Peter Jan van Leeuwen. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, 136(653):1991–1999, 2010.

[29] Melanie Ades and Peter Jan Van Leeuwen. An exploration of the equivalent weights particle filter. *Quarterly Journal of the Royal Meteorological Society*, 139(672):820–840, 2013.

[30] Melanie Ades and Peter J Van Leeuwen. The equivalent-weights particle filter in a high-dimensional system. *Quarterly Journal of the Royal Meteorological Society*, 141(687):484–503, 2015.

[31] James C Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

[32] Dan Crisan and Joaquin Miguez. Nested particle filters for online parameter estimation in discrete-time state-space markov models. *Bernoulli (to appear)*.

[33] Alan Bain and Dan Crisan. *Fundamentals of stochastic filtering*. Springer, 2009.

[34] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control, Vol. I*. Athena Scientific, Belmont, MA, 2001.

[35] Samuel Kotz and Saralees Nadarajah. *Multivariate t-distributions and their applications*. Cambridge University Press, 2004.