Adv. Appl. Prob. **49,** 24–48 (2017)
*doi:*10.1017.apr.2016.77

A STABLE PARTICLE FILTER FOR A CLASS OF HIGH-DIMENSIONAL STATE-SPACE MODELS

ALEXANDROS BESKOS,* *University College London*
DAN CRISAN,** *Imperial College London*
AJAY JASRA,*** **** *National University of Singapore*
KENGO KAMATANI,***** *Osaka University*
YAN ZHOU,*** *National University of Singapore*

Abstract

We consider the numerical approximation of the filtering problem in high dimensions, that is, when the hidden state lies in $\mathbb{R}^d$ with large d. For low-dimensional problems, one of the most popular numerical procedures for consistent inference is the class of approximations termed particle filters or sequential Monte Carlo methods. However, in high dimensions, standard particle filters (e.g. the bootstrap particle filter) can have a cost that is exponential in d for the algorithm to be stable in an appropriate sense. We develop a new particle filter, called the *space–time particle filter*, for a specific family of state-space models in discrete time. This new class of particle filters provides consistent Monte Carlo estimates for any fixed d, as do standard particle filters. Moreover, when there is a spatial mixing element in the dimension of the state vector, the space–time particle filter will scale much better with d than the standard filter for a class of filtering problems. We illustrate this analytically for a model of a simple independent and identically distributed structure and a model of an L-Markovian structure ($L \geq 1$, L independent of d) in the d-dimensional space direction, when we show that the algorithm exhibits certain stability properties as d increases at a cost $\mathcal{O}(nNd^2)$, where n is the time parameter and N is the number of Monte Carlo samples, which are fixed and independent of d. Our theoretical results are also supported by numerical simulations on practical models of complex structures. The results suggest that it is indeed possible to tackle some high-dimensional filtering problems using the space–time particle filter that standard particle filters cannot handle.

Keywords: State-space model; high dimensions; particle filter

2010 Mathematics Subject Classification: Primary 62M20
Secondary 60G35

1. Introduction

We consider the numerical resolution of filtering problems and the estimation of the associated normalising constants for state-space models. In particular, the data is modelled by a discrete-time process $\{Y_n\}_{n\geq 1}$, $Y_n \in \mathbb{R}^{d_y}$, associated to a hidden signal modelled by a Markov chain $\{X_n\}_{n\geq 0}$, $X_n \in \mathbb{R}^d$; we are concerned with high dimensions, that is, large d.

Received 23 June 2015; revision received 1 July 2016.
* Postal address: Department of Statistical Science, University College London, London WC1E 6BT, UK.
** Postal address: Department of Mathematics, Imperial College London, London SW7 2AZ, UK.
*** Postal address: Department of Statistics and Applied Probability, National University of Singapore, 117546, Singapore.
**** Email address: staja@nus.edu.sg
***** Postal address: Graduate School of Engineering Science, Osaka University, Osaka 565-0871, Japan.

For simplicity, we assume that the location of the signal at time 0 is fixed and known, but the algorithm can easily be extended to the general case. We write the joint density (with respect to an appropriate dominating measure) of $(x_{1:n}, y_{1:n})$ as

$$p(x_{1:n}, y_{1:n}) = \prod_{k=1}^{n} g(x_k, y_k) f(x_{k-1}, x_k) \tag{1.1}$$

for kernel functions f, g, and $X_0 = x_0$ so that, given the hidden states $X_{1:n} = \{X_1, \ldots, X_n\}$, the data $Y_{1:n} = \{Y_1, \ldots, Y_n\}$ consist of independent entries with Y_k depending only on X_k. The objective is to approximate the filtering distribution $X_n \mid Y_{1:n} = y_{1:n}$. This filtering problem for large d is notoriously difficult in many scenarios.

In general, the filter cannot be computed exactly and one often has to resort to numerical methods, for example by using particle filters (see, e.g. [10]). Particle filters make use of a sequence of proposal densities and sequentially simulate from these a collection of $N > 1$ samples, termed particles. In most scenarios it is not possible to use the distribution of interest as a proposal. Therefore, one must correct for the discrepancy between proposal and target via importance weights. In the majority of cases of practical interest, the variance of these importance weights increases with algorithmic time. This can, to some extent, be dealt with via a resampling procedure consisting of sampling with replacement from the current weighted samples and resetting them to $1/N$. The variability of the weights is often measured by the effective sample size (ESS). If d is small to moderate then particle filters can, in many cases, be effective for increasing the time parameter n, for instance, by possessing a time-uniform error under conditions; see [6].

For some state-space models with specific structures, particle algorithms can be effective in high dimensions, or at least can be appropriately modified to be so. We note, for instance, that one can setup an effective particle filter even when $d = \infty$, provided one assumes a finite (and small, relative to d) amount of information in the likelihood (see, e.g. [12] for details). This is *not* the class of problems for which we are interested in here. In general, it is mainly the variability of the likelihood $g(x_k, y_k)$ that determines the algorithmic challenge rather than the dimension d of the hidden space per se (this is related to what is called the 'effective dimension' in [4]). The function $x_k \mapsto g(x_k, y_k)$ can convey a lot of information about the hidden state, especially so in high dimensions. If this is the case, using the prior transition kernel $f(x_{k-1}, x_k)$ as the proposal will be ineffective. We concentrate here on the challenging class of problems with large state-space dimension d and an amount of information in the likelihood that increases with d. The standard particle filter will typically perform poorly in this context, often requiring that $N = \mathcal{O}(\kappa^d)$ for some $\kappa > 1$; see, e,g. [4]. The results of [4], amongst others, have motivated substantial research in the literature on particle filters in high dimensions, such as the recent work in [15], which attempted an approximate split of the d-dimensional state vector to confront the curse of dimensionality for importance sampling, at the cost of introducing a difficult to quantify bias with magnitude that depends on the position along the d coordinates. See [15] and the references therein for some algorithms designed for high-dimensional filtering. To date, there are few particle filtering algorithms that are simultaneously

(i) asymptotically consistent (as N grows),

(ii) of fixed computational cost per time-step ('online'),

(iii) supported by theoretical analysis demonstrating a subexponential cost in d.

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

There is substantial interest in developing an algorithm which can possess these attributes. In this paper we attempt to provide an algorithm which has the above properties. However, in the context of (iii) we can verify this only for a subclass of filtering problems for which there is a spatial mixing element in the dimension. It is stressed that we have found that the algorithm we develop can be applied in other contexts, with empirical evidence suggesting effective performance in high dimensions, but there is no mathematical proof that there is a subexponential cost in d.

Our method develops as follows. In a general setting, we assume that there exists an increasing sequence of sets $\{\mathcal{A}_{k,j}\}_{j=1}^{\tau_{k,d}}$, with $\mathcal{A}_{k,1} \subset \mathcal{A}_{k,2} \subset \cdots \subset \mathcal{A}_{k,\tau_{k,d}} = \{1\colon d\}$ for integer $0 < \tau_{k,d} \leq d$, such that we can factorise

$$g(x_k, y_k) f(x_{k-1}, x_k) = \prod_{j=1}^{\tau_{k,d}} \alpha_{k,j}(y_k, x_{k-1}, x_k(\mathcal{A}_{k,j})) \tag{1.2}$$

for appropriate functions $\alpha_{k,j}(\cdot)$, where we denote $x_k(\mathcal{A}) = \{x_k(j)\colon j \in \mathcal{A}\} \in \mathbb{R}^{|\mathcal{A}|}$. As we remark later on, this structure is not an absolutely necessary requirement for the subsequent algorithm, but will clarify the ideas in the development of the method. Within a sequential Monte Carlo context, one can think of augmenting the sequence of distributions of increasing dimension $X_{1\colon k} \mid Y_{1\colon k}$, $1 \leq k \leq n$, moving from $\mathbb{R}^{d(k-1)}$ to $\mathbb{R}^{dk}$, with intermediate laws on $\mathbb{R}^{d(k-1)+|\mathcal{A}_{k,j}|}$ for $j = 1, \ldots, \tau_{k,d}$. The structure in (1.2) is not uncommon. For instance, one should typically be able to obtain such a factorisation for the prior term $f(x_{k-1}, x_k)$ by marginalising over subsets of coordinates. Then, for the likelihood component $g(x_k, y_k)$, this could, for instance, be implied when the model assumes a local dependence structure for the observations. Critically, for this approach to be effective it is necessary that the factorisation is such that it allows for a gradual introduction of the 'full' likelihood term $g(x_k, y_k)$ along the $\tau_{k,d}$ steps. For instance, trivial choices such as

$$\alpha_{k,j} = p(x_k(j) \mid x_{k-1}, x_k(1\colon j-1)), \qquad 1 \leq j \leq d-1,$$
$$\alpha_{k,d} = p(x_k(d) \mid x_{k-1}, x_k(1\colon d-1)) g(x_k, y_k)$$

are ineffective, as they introduce the complete likelihood term only in the last step.

Our contribution is based upon the idea that particle filters are, in general, robust with regard to the time parameter (e.g. the error in approximation can be shown to be time uniform). Thus, we exploit the structure in (1.2) to build up a particle filter in space–time moving vertically along the space index; for this reason, we call the new algorithm the space–time particle filter (STPF). We break the kth time-step of the particle filter into $\tau_{k,d}$ space-substeps and run a system of N independent particle filters for these substeps. This is similar to a tempering approach employed in [2], [3], in the context of sequential Monte Carlo algorithms [8] for a single target probability of dimension d. There, the idea is to use annealing steps, interpolating between an easy to sample distribution and the target with an $\mathcal{O}(d)$ number of steps. In the context of filtering, for the filter, say, at time 1 we break the problem of trying to perform importance sampling in one step for a d-dimensional object (which typically does not perform well, as noted by [4]) into $\tau_{1,d}$ easier steps via the particle filter along space; as the particle filter on low to moderate dimensions is typically well behaved, one expects the proposed procedure to work well even if d is large. A similar idea is used at subsequent time-steps of the filter.

In the main part of the paper and in all theoretical derivations, we work under the easier to present scenario $\tau_{k,d} = d$ and $\mathcal{A}_{k,j} = \{1\colon j\}$. We establish that our algorithm is consistent

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

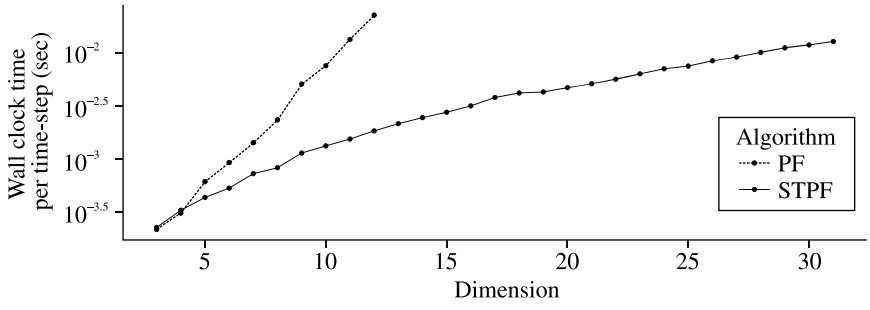

FIGURE 1: Computational cost per time-step to achieve a predetermined RMSE versus model dimension, for the standard PF and STPF. The two algorithms are applied on Model 1 in Section 4; see Section 4.1 for more details.

as N grows (for fixed d), that is, that one can estimate the filter with enough computational power in a manner that is online. Then we look at two simple models:

- an independent and identically distributed (i.i.d.) scenario both in space and time,
- an L-Markovian model along space.

In both cases, we present results indicating that the algorithm is stable at a cost of $\mathcal{O}(nNd^2)$. As we remark in Section 3.2, this cost is optimistic in general. We stress here that there is a lot more to be investigated in terms of the analytical properties of the proposed algorithm to fully explore its potential, certainly in more complex model structures than the above. In this paper we aim to make an important first contribution to a very significant and challenging problem and open up several directions for future investigation. In particular, it *is not claimed* that there is a subexponential cost in d for every filtering problem where the algorithm could be applied.

Numerical results shown later in the paper strongly suggest that the STPF can be very effective in high dimensions. Indicatively, in Figure 1, we show results from applying the standard particle filter (PF) and the STPF on Model 1 defined later in the paper (Section 4). In the figure we show the computational cost per time-step required to achieve a predetermined RMSE (root-mean squared error) versus model dimension for estimates of $\mathbb{E}[X_n(1) \mid Y_{1:n}]$, with $n = 1000$. In this case, the numerics suggest that the STPF is much more robust than the PF, which suffers from the curse of dimensionality.

This paper is structured as follows. In Section 2 the STPF algorithm is given. In Section 3 our mathematical results are given; some proofs are housed in the appendices. In Section 4 our algorithm is implemented and compared to existing methodology. In Section 5 we conclude the paper with several remarks for future work.

2. The STPF

We develop an algorithm that combines a local filter running d space-steps using M_d particles, with a global filter making time-steps and using N particles. In Section 3 we establish that, for any fixed $M_d \geq 1$, $d \geq 1$, the algorithm is consistent, with respect to some estimates of interest, as N grows. A motivation for using such an approach is that it can potentially provide better estimates for expectations over the complete d-dimensional filtering density $X_n \mid Y_{1:n} = y_{1:n}$, versus a standard filter with $N = 1$, due to an extra selection step that resamples over the $N \geq 1$ local filters. This approach has been motivated by the island particle model of [17], where a

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

related method for standard particle filters (and not related with confronting the dimensionality issue) was developed, but is not a trivial extension of it, so some extra effort is required to ensure correctness of the algorithm. We also explain how to set M_d as a function of d to ensure some stability properties with respect to d in some specific modelling scenarios.

2.1. Time-step $n \geq 1$

We describe separately the local and global filters.

Low level: local filter. We assume availability of a collection of d-dimensional particles $\check{x}_{n-1}^l$, $1 \leq l \leq M_d$, from the end of step $n-1$ (if $n = 1$, all M_d particles at time 0 are equal to the initial position $x_0 \in \mathbb{R}^d$, assumed fixed). At the end of each space-step $0 \leq j \leq d$, a single particle will be comprised of $(x_{n-1},\, x_n(1\colon j))$, under the convention $x_n(1\colon 0) = \varnothing$, that is the algorithm keeps track of the j coordinates at time n and their ancestry at time $n-1$. At space-step j, particle $(x_{n-1},\, x_n(1\colon j-1))$ will be propagated according to a proposal density $q_{n,j}(x_n(j) \mid x_{n-1},\, x_n(1\colon j-1))$ for $1 \leq j \leq d$. Thus, given the factorisation of the target in (1.2) with $\mathcal{A}_{k,j} = \{1\colon j\}$, the incremental weight at step j for particle $(x_{n-1},\, x_n(1\colon j))$ will be equal to

$$G_{n,j}(x_{n-1},\, x_n(1\colon j)) = \frac{\alpha_{n,j}(y_n, x_{n-1},\, x_n(1\colon j))}{q_{n,j}(x_n(j) \mid x_{n-1},\, x_n(1\colon j-1))}.$$

The M_d particles, of dimension $d+j$, will be resampled according to their weights at each step $1 \leq j \leq d$. At the end of all d space-steps, the algorithm will provide M_d particles x_n^l, $1 \leq l \leq M_d$, to be used at the next time-step.

Let $\bar{G}_{n,j}$ denote the average of the M_d weights at step j. We define the product

$$\boldsymbol{G}_n = \prod_{j=1}^{d} \bar{G}_{n,j}. \tag{2.1}$$

High level: global filter. An outer algorithm repeats the above described nth time-step of the local filter N times, independently, with the ith execution initialised by the collection of particles $\check{x}_{n-1}^{i,l}$, $1 \leq l \leq M_d$, for every $1 \leq i \leq N$. Let $\boldsymbol{G}_n^i$ denote the value of the estimate of the normalising constant in (2.1) from the ith execution. The ith execution is assigned weight equal to $\boldsymbol{G}_n^i$ and the N systems will be resampled according to these weights. After resampling, the complete algorithm will provide samples $\check{x}_n^{i,l}$, $1 \leq l \leq M_d$, $1 \leq i \leq N$, to be used as initial positions for the next time-step of the global and local filters.

We call the complete algorithm the STPF. We note here that the normalising constant

$$\int_{\mathbb{R}^{dn}} \left(\prod_{k=1}^{n} g(x_k, y_k) f(x_{k-1}, x_k) \right) \mathrm{d}x_{1\colon n}$$

can be estimated by $\prod_{k=1}^{n} \bar{\boldsymbol{G}}_k$, where we have set $\bar{\boldsymbol{G}}_k = \sum_{i=1}^{N} \boldsymbol{G}_k^i / N$. Also, for $\varphi\colon \mathbb{R}^d \to \mathbb{R}$, the expectation over the filter at time n,

$$\frac{\int_{\mathbb{R}^{nd}} \varphi(x_n) \prod_{k=1}^{n} g(x_k, y_k) f(x_{k-1}, x_k)\, \mathrm{d}x_{1\colon n}}{\int_{\mathbb{R}^{nd}} \prod_{k=1}^{n} g(x_k, y_k) f(x_{k-1}, x_k)\, \mathrm{d}x_{1\colon n}}$$

can be estimated by

$$\frac{1}{N M_d} \sum_{l=1}^{M_d} \sum_{i=1}^{N} \varphi(\check{x}_n^{i,l}). \tag{2.2}$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

In Algorithm 1 below we provide a more precise definition of the method in a pseudocode form.

Algorithm 1. (*STPF targeting the HMM model (1.1) under structure (1.2).*) The algorithm comprises three steps.

Step 0. At time $t = 0$, set $\check{x}_0^{i,l} = x_0$ for $1 \le l \le M_d$ and $1 \le i \le N$. Set $n = 0$.

Step 1. Set $n = n + 1$.

For i in $1 : N$:

Initialise $\check{x}_{n-1,0}^{i,l} = \check{x}_{n-1}^{i,l}$ for $1 \le l \le M_d$.

For j in $1 : d$:

For l in $1 : M_d$:

Propose $x_n^{i,l}(j) \sim q_{n,j}(x_n(j) \mid \check{x}_{n-1,j-1}^{i,l}, x_{n,j-1}^{i,l}(1 : j-1))$.

Assign weight $G_{n,j}^{i,l} = G_{n,j}(\check{x}_{n-1,j-1}^{i,l}, x_{n,j-1}^{i,l}(1 : j-1), x_n^{i,l}(j))$.

Calculate $\bar{G}_{n,j}^{i} = \sum_{l=1}^{M_d} G_{n,j}^{i,l} / M_d$.

Resample from weighted particle population

$$\{G_{n,j}^{i,l}, (\check{x}_{n-1,j-1}^{i,l}, x_{n,j-1}^{i,l}(1 : j-1), x_n^{i,l}(j))\}_{l=1}^{M_d}$$

to obtain M_d equally weighted particles $(\check{x}_{n-1,j}^{i,l}, x_{n,j}^{i,l}(1 : j))_{l=1}^{M_d}$.

Assign weight $\boldsymbol{G}_n^i = \prod_{j=1}^{d} \bar{G}_{n,j}^{i}$.

Resample from weighted island population

$$\{\boldsymbol{G}_n^i, (x_{n,d}^{i,l})_{l=1}^{M_d}\}_{i=1}^{N}$$

to obtain N equally weighted islands $((\check{x}_n^{i,l})_{l=1}^{M_d})_{i=1}^{N}$.

Step 2. Return to step 1.

2.2. Remarks

In terms of the estimate of the filter (2.2), we expect a *path degeneracy* effect for the local filter (see [10]), especially for large d, due to resampling forcing common ancestries for different particles and the generation of the coordinates of a particle at time n requiring, in general, its ancestry at time $n - 1$. For instance, in a worst case scenario, in some algorithmic execution only one of the M_d samples can be a good representation of the target filtering distribution; or one can be left with the same ancestry at time $n - 1$ for all M_d particles before the completion of the d space-steps at time n. However, one can still average over all M_d samples as we have done; one can also select a single sample for estimation for each $1 \le i \le N$, but there is not an obvious advantage to doing so. To an extent, path degeneracy can be somewhat alleviated using dynamic resampling (see, e.g. [9] and the references therein); also, the selection step over the N local filters can have a strong positive effect, as we will see in the numerical applications later on in the paper. In addition, in Section 2.3, we discuss one more approach for potentially dealing with path degeneracy involving particle mutation steps.

Note that we have assumed that

$$g(x_k, y_k) f(x_{k-1}, x_k) = \prod_{j=1}^{d} \alpha_{k,j}(y_k, x_{k-1}, x_k(1 : j)).$$

However, this need not be the case. All one needs is a collection of functions $\alpha_{k,j}$, such that the variance (with respect to the simulated algorithm) of

$$\frac{g(x_k, y_k) f(x_{k-1}, x_k)}{\prod_{j=1}^{d} \alpha_{k,j}(y_k, x_{k-1}, x_k(1 : j))} \tag{2.3}$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

is reasonable (for instance, it is at least subexponential in d), especially as d grows. Then, the M_d particles of the form $(x_{k-1}, x_k) \in \mathbb{R}^{2d}$ obtained at the end of the kth time-step under the employed $\prod_{j=1}^{d} \alpha_{k,j}(y_k, x_{k-1}, x_k(1:j))$ can be used as proposals within an importance sampler targeting $g(x_k, y_k) f(x_{k-1}, x_k)$, with the above ratio giving the relevant weights.

The algorithm is easily parallelised over N, at least between global resampling times. We also note that the idea of using a particle filter within a particle filter has been used, for example, in [11]. In general, the cost of the algorithm is $\mathcal{O}(nNM_d d^2)$, assuming a cost proportional to j or d when sampling the proposal for the jth coordinate and calculating the corresponding weight. The algorithm can also be thought of as a novel generalisation of the island particle filter [17]. In our algorithm, one runs an entire particle filter for d time-steps, as the local filter, whereas it is only one step in [17]; as we will see in Section 3, this appears to be critical in the high-dimensional filtering context.

2.3. Dealing with path degeneracy

The path degeneracy effect may limit the success of the proposed algorithm, that is, produce weights whose variance may be too substantial to provide reliable estimates. We expect the method to be effective in practice when d is maybe too large for the standard particle filter, but not overly large. We cannot prove, for instance, that the number of particles can scale subexponentially with d to control variances, but will present numerical applications showing that in practice one can treat values of d that are far out of the scope of a standard particle filter.

In addition to dynamic resampling and the selection step over the N local filters, one can also attempt the following to reduce the effect of path degeneracy. At time-step $n \geq 1$, one uses the marginal particle filter (see, e.g. [14]) and targets, for each local particle filter at each space-step $1 \leq j \leq d$, the marginal of $x_n(1:j)$ under the model determined by the $\alpha_{n,k}$ functionals via the factorisation (1.2). Such marginals can be estimated, up to a constant, via the Monte Carlo average

$$\sum_{l=1}^{M_d} \prod_{k=1}^{j} \alpha_{n,k}(y_n, \overset{\vee}{x}{}_{n-1}^{i,l}, x_n(1:k)),$$

where $x_{n-1}^{i,l}$ is the collection of particles at the end of the $n-1$ time-step. The complete algorithm will involve both iterative importance sampling targeting the above sequence of marginals (their Monte Carlo estimate) on a space of increasing dimension and mutation Markov chain Monte Carlo (MCMC) steps which will preserve each of the targets and disperse the particles. Thus, the method also requires a proposal kernel $q_{n,j}(x_n(j) \mid x_n(1:j-1))$ for propagating particles across space. The MCMC mutation steps can be applied in all or in only a subset of the algorithmic steps across space.

Compared with the main algorithm in Section 2, here the method runs only on the x_n-space, and not the joint (x_{n-1}, x_n)-space. Assuming an effective design of the MCMC step and a good performance of the Monte Carlo estimate of the marginal density, the path degeneracy effect can be alleviated. Each time-step n of this algorithm will still have fixed (but increased) computational complexity. The cost of this modified algorithm, assuming the cost of computing $\alpha_{n,k}$ is $\mathcal{O}(1)$ for each n, k, is $\mathcal{O}(nNM_d^2 d^2)$, where the $d^2 M_d^2$ term is due to requiring the estimate of marginal density for all M_d particles, and the cost for each particle is $j \cdot M_d$ at space-step j. So long as M_d is polynomial in d, the complexity can still be reasonable with regards to computational cost.

We note that, even though we do not analyse this algorithm mathematically, simulation results are provided.

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

3. Theoretical results

3.1. Consistency of the space–time sampler

We now establish that if d, $M_d \geq 1$ are fixed then STPF will provide consistent estimates of quantities of interest of the true filter as N grows. Indeed, one can prove many results about the algorithm in this setting, such as finite-N bounds and central limit theorems (CLTs); however, this is not the focus of this work and the consistency result is provided to validate the use of the algorithm. Throughout, we condition on a fixed data record and we suppose that, for all $1 \leq j \leq d$,

$$\sup_{x \in \mathbb{R}^j} |G_{1,j}(x_0, x)| < +\infty, \quad \sup_{x \in \mathbb{R}^{d+j}} |G_{n,j}(x)| < +\infty, \qquad n \geq 2.$$

Below '$\xrightarrow{\mathbb{P}}$' denotes convergence in probability as N grows, where $\mathbb{P}$ denotes the law under the simulated algorithm. We denote by $\mathcal{B}_b(\mathbb{R}^d)$ the class of bounded and measurable real-valued functions on $\mathbb{R}^d$. We will write, for $n \geq 1$,

$$\pi_n(\varphi) := \frac{\int_{\mathbb{R}^{nd}} \varphi(x_n) \prod_{k=1}^n g(x_k, y_k) f(x_{k-1}, x_k) \, \mathrm{d}x_{1:n}}{\int_{\mathbb{R}^{nd}} \prod_{k=1}^n g(x_k, y_k) f(x_{k-1}, x_k) \, \mathrm{d}x_{1:n}}$$

and

$$p(y_{1:n}) = \int_{\mathbb{R}^{nd}} \left(\prod_{k=1}^n g(x_k, y_k) f(x_{k-1}, x_k) \right) \mathrm{d}x_{1:n},$$

so that π_n corresponds to the filtering density of $X_n \mid y_{1:n}$. The proof of the following theorem is given in Appendix B. It ensures that the N-particle system corresponds to a standard particle filter on an enlarged state space; once this is established standard consistency results for particle filters on general state spaces (see, e.g. [6]) will complete the proof.

Theorem 3.1. *Let d, $M_d \geq 1$ be fixed and let $\varphi \in \mathcal{B}_b(\mathbb{R}^d)$. Then we have, for any $n \geq 1$ and $N \to \infty$,*

$$\frac{1}{NM_d} \sum_{l=1}^{M_d} \sum_{i=1}^{N} \varphi(\check{x}_n^{i,l}(1:d)) \xrightarrow{\mathbb{P}} \pi_n(\varphi), \qquad \prod_{k=1}^n \left(\frac{1}{N} \sum_{i=1}^N \boldsymbol{G}_k^i \right) \xrightarrow{\mathbb{P}} p(y_{1:n}).$$

Remark 3.1. The proof establishes that also $(1/N)\sum_{i=1}^N \varphi(\check{x}_1^{i,1}(1:d))$ is a consistent estimator for the filter; this may be more effective than the estimator given in the statement of the theorem, due to the path degeneracy effect mentioned earlier. In addition, one can assume the context described in (2.3) with the target not having a product structure, but the weights in (2.3) having controlled variance. Even in this more general case, one can follow the arguments in the proof to obtain consistency (assuming the expression in (2.3) is upper bounded).

3.2. Stability in high dimensions for the i.i.d. model

We now come to the main objective of our theoretical analysis. We set N as fixed and consider the algorithm as d grows. In order to facilitate our analysis, we will consider approximating a probability, with density proportional to

$$\prod_{k=1}^n \prod_{j=1}^d \alpha(x_k(j)).$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

We use the STPF with proposals $q_{n,j}(x_n(j) \mid x_{n-1},\, x_n(1\colon j-1)) = q(x_n(j))$. In the case of a state-space model, this would correspond to

$$g(x_k, y_k) f(x_{k-1}, x_k) = \prod_{j=1}^{d} \alpha(x_k(j)),$$

which would seldom occur in a real scenario. However, analysis in this context is expected to be informative for more complex scenarios as in the work of [2]. Note that, because of the loss of dependence on subsequent observation times, we expect that any complexity analysis with respect to d to be slightly over-optimistic; as noted, the path degeneracy effect is expected to play a role in this algorithm in general.

We consider the relative variance of the standard estimate of the normalising constant $p(y_{1\colon n})$, given, for instance, in Theorem 3.1, which can now be written as

$$p^{N,M_d}(y_{1\colon n}) = \prod_{k=1}^{n} \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{d} \frac{1}{M_d} \sum_{l=1}^{M_d} \frac{\alpha(x_k^{i,l}(j))}{q(x_k^{i,l}(j))} \equiv \prod_{k=1}^{n} \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{G}_k^i.$$

The proof of the following result is given in Appendix A. Note that due to the i.i.d. structure along time and space, all variables $x_k^{i,l}(j)$ can be assumed to be i.i.d. from $q(\cdot)$.

Proposition 3.1. *Assume that*

$$\frac{\int \alpha(x)^2/q(x)\,\mathrm{d}x}{(\int \alpha(x)\,\mathrm{d}x)^2} < +\infty.$$

Then

$$\mathbb{E}\left[\left(\frac{p^{N,M_d}(y_{1\colon n})}{p(y_{1\colon n})} - 1\right)^2\right] = \left(\frac{1}{N}\left(\frac{1}{M_d}\frac{\int \alpha(x)^2/q(x)\,\mathrm{d}x}{(\int \alpha(x)\,\mathrm{d}x)^2} + \frac{M_d - 1}{M_d}\right)^d + \frac{N-1}{N}\right)^n - 1.$$

Remark 3.2. The $M_d = 1$ case corresponds, in some sense, to the standard particle filter. In this case, by Jensen's inequality, the right-hand side of the above identity diverges as d grows, unless N is of exponential order in d. As a result, we can stabilise the algorithm with an $\mathcal{O}(nd\kappa^d)$ cost, where $\kappa > 1$. However, if one sets $M_d = d$ then the right-hand side of the above identity stabilises and the cost of the algorithm is $\mathcal{O}(nNd^2)$. This provides some intuition about why our approach may be effective in high dimensions.

In fact, we can say a bit more. We suppose that $\alpha(x)/q(x)$ is upper and lower bounded; this typically implies that x lies only on some compact subset of $\mathbb{R}$. Denoting by '$\xrightarrow{\mathrm{w}}$' weak convergence as $d \to \infty$ and $\mathcal{LN}(\mu, \sigma^2)$ the log-normal distribution of location μ, scale σ, we have the following.

Proposition 3.2. *Let $M_d = d/c$ for some $0 < c < +\infty$ and $N, n \geq 1$ fixed. Suppose that*

$$\sigma^2 := \frac{\int \alpha(x)^2/q(x)\,\mathrm{d}x}{(\int \alpha(x)\,\mathrm{d}x)^2} - 1 < +\infty.$$

Then, as $d \to \infty$, we have $\boldsymbol{G}_k^i/(\int_{\mathbb{R}} \alpha(x)\,\mathrm{d}x)^d \xrightarrow{\mathrm{w}} V_k^i$, and, subsequently,

$$\frac{p^{N,M_d}(y_{1\colon n})}{p(y_{1\colon n})} \xrightarrow{\mathrm{w}} \prod_{k=1}^{n} \frac{1}{N} \sum_{i=1}^{N} V_k^i,$$

where $V_k^i \overset{\text{i.i.d.}}{\sim} \mathcal{LN}(-c\sigma^2/2, c\sigma^2)$.

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

Proof. The result follows from [1, Theorem 1.1] and elementary calculations, which we omit. □

Remark 3.3. The result suggests that the algorithm stabilises as d grows at a $\mathcal{O}(nNd^2)$ cost. Using the continuous mapping theorem, for $N > 1$ one can show that the ESS will also converge to a nontrivial random variable; see, e.g. [2, Proof of Theorem 3.2]. Moreover, based on a personal communication with Pierre Del Moral, we conjecture that setting $M_d = d^{1+\delta}/c$, for some $\delta > 0$, the ESS converges to N; hence suggesting that $M_d = \mathcal{O}(d)$ is an optimal computational effort in this case.

Remark 3.4. An intuition behind the results is that for a standard particle filter, when run for n steps with N particles and under assumptions, the relative variance of the estimate for the normalising constant grows at most linearly in the number of steps n, provided $N = \mathcal{O}(n)$ (see [5] for details). In the algorithm, the weights $\boldsymbol{G}_n$ are estimates of normalising constants for the local filter, so one expects that if $M_d = \mathcal{O}(d)$ then the algorithm should work well for large d. There is, however, an important point to be made. The result above assumes an i.i.d. structure which removes any path degeneracy effect, both within a local filter, and in the time-dependence between observations.

Remark 3.5. In the case of no global resampling, one uses the estimate, for $p(y_{1:n})$,

$$\frac{1}{N}\sum_{i=1}^{N}\prod_{k=1}^{n}\prod_{j=1}^{d}\frac{1}{M_d}\sum_{l=1}^{M_d}\frac{\alpha(x_k^i(j))}{q(x_k^i(j))}.$$

A weak convergence result also holds in this case.

3.3. Stability in high dimensions for the L-Markov model

We now consider a more realistic scenario for our analysis in high dimensions. In order to read this section, one will need to consult Appendices B and C; this section can be skipped with no loss in continuity.

We consider the interaction of the dimension and the time parameter in the behaviour of the algorithm. Let $L \geq 1$ be given, with $L < d$ independent of d. We now list some assumptions and notation needed to describe the result.

(A1) For every $n \geq 1$, we have

$$g(x_n, y_n) f(x_{n-1}, x_n) = \prod_{j=1}^{d} h(y_n, x_n(j)) k(x_n(j-L:j-1), x_n(j)),$$

where $h = h(y_n, \cdot): \mathbb{R} \to \mathbb{R}^+$, $x_n(p) = x_{n-1}(p+d)$, $p \in \{-d+1, \ldots, 0\}$, with $x_n(p)$ null if $p < -d+1$ and, for every $x \in \mathbb{R}^L$, $\int_{\mathbb{R}} k(x, x')\,\mathrm{d}x' = 1$.

It is noted that even under (A1) a standard particle filter which propagates all d coordinates together may degenerate as d grows. However, as we will remark, the STPF can stabilise under assumptions, even if $N = 1$. Our algorithm will use the kernels k as the proposals. Define the semigroup, for $p \geq 1$,

$$\hat{q}_p(x_{p-1}, \mathrm{d}x_p) = f(x_{p-1}, x_p) g_p(x_p)\,\mathrm{d}x_p,$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

where $g_p(x_p) = g(x_p, y_p)$. For $\varphi \in \mathcal{B}_b(\mathbb{R}^d)$ define

$$\hat{q}_{p,n}(\varphi)(x_p) = \int \hat{q}_{p+1}(x_p, \mathrm{d}x_{p+1}) \times \cdots \times \hat{q}_n(x_{n-1}, \mathrm{d}x_n)\varphi(x_n).$$

(A2) There exists a $c < \infty$, such that, for every $1 \leq p < n$ and $d \geq 1$,

$$\sup_{x,y} \frac{\hat{q}_{p,n}(1)(x)}{\hat{q}_{p,n}(1)(y)} \leq c.$$

Note that (A2) is fairly standard in the literature (see, e.g. [7]) and given (A1) it will hold under some simple assumptions on h and k. The scenario considered here is indicative of ones where we expect the STPF to work well; when there is some aspect of spatial mixing, which allows one to transfer the strength of sequential Monte Carlo methods in time to the spatial domain.

Now, we will consider the global filter with N particles, as standard results in the literature can provide immediately CLTs and strong law of large numbers for quantities of interest. We will then investigate the effect of the dimensionality d on the involved terms. Consider the standard estimate for the normalising constant for the global filter

$$\boldsymbol{\gamma}_n^N(1) := \prod_{p=1}^{n-1} \boldsymbol{\eta}_p^N(\boldsymbol{G}_p),$$

where $\boldsymbol{\eta}_p^N(\cdot)$ simply denotes Monte Carlo averages over the N-particle systems at time p; see Appendix B for analytic definitions. From standard particle filtering theory, it follows that $\boldsymbol{\eta}_p^N(\cdot)$ is an unbiased estimator of the corresponding limiting quantity, denoted $\boldsymbol{\gamma}_n(1)$; see, e.g. [6, Theorem 7.4.2]. Also, under our assumptions, we have the following CLT as $N \to \infty$ (see [6, Proposition 9.4.2]):

$$\sqrt{N}\left(\frac{\boldsymbol{\gamma}_n^N(1)}{\boldsymbol{\gamma}_n(1)} - 1\right) \xrightarrow{\mathrm{w}} \mathcal{N}(0, \sigma_n^2), \tag{3.1}$$

where $\mathcal{N}(0, \sigma^2)$ is the one-dimensional normal distribution with 0 mean and variance σ^2, and

$$\sigma_n^2 = \frac{1}{\boldsymbol{\gamma}_n(1)^2} \sum_{p=1}^{n} \boldsymbol{\gamma}_p(1)^2 \boldsymbol{\eta}_p((\boldsymbol{Q}_{p,n}(1) - \boldsymbol{\eta}_p(\boldsymbol{Q}_{p,n}(1)))^2).$$

All maths bold terms correspond to standard Feynman–Kac quantities and are defined in Appendix B. We also show in Appendix B that the normalising constant of the global filter coincides with the one of the original filters of interest, that is

$$\boldsymbol{\gamma}_n(1) \equiv \gamma_n(1) = \int \prod_{p=1}^{n-1} g_p(x_p) f(x_{p-1}, x_p)\, \mathrm{d}x_{1:\,p} = p(y_{1:\,n-1}).$$

Thus, (3.1) provides, in fact, a CLT for the estimate of the STPF for $p(y_{1:\,n-1})$ proposed in Theorem 3.1.

We have the following result, whose proof is in Appendix C.

Theorem 3.2. *Assume that (A1) and (A2) hold. Then there exists a $\bar{c} < \infty$ such that, for any $n, d \geq 1$ and any $M_d \geq \bar{c}d$,*

$$\sigma_n^2 \leq n\bar{c}\left(\frac{d}{M_d} + 1\right).$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

Remark 3.6. Our result establishes that the asymptotic in N-variance of the relative value of the normalising constant estimate grows at most linearly in n and if $M_d = \mathcal{O}(d)$ does not grow with the dimension. The cost of the algorithm is $\mathcal{O}(nNd^2)$. The linear growth in time is a standard result in the literature (see [7]) and one does not expect to do better than this. Note that a particular model structure is chosen and one expects a higher cost in more general problems.

Remark 3.7. We expect that in order to show that the error in the estimation of the filter is time uniform, under (A1), we will need to set $M_d = \mathcal{O}(d^2)$, at least when $L = 1$. This is because we are performing an estimation on the path of the algorithm; see [7, Theorem 15.2.1 and Corollary 15.2.2]. Indeed, we can be even more specific; if $N = 1$ then we can show that, under (A1) and (A2), the L_p-error associated to the estimate of the filter (applied to a bounded test function in $\mathbb{R}^d$) at time n is upper bounded by $c\|\varphi\|_\infty d/\sqrt{M_d}$ (via [7, Theorem 15.2.1, Corollary 15.2.2]) with c independent of d and n. Thus, setting $M_d = \mathcal{O}(d^2)$, the upper bound depends on d only through $\|\varphi\|_\infty$.

4. Numerical results

4.1. Model 1

We consider an autoregressive model of order d. In particular, let $X_n \in \mathbb{R}^d$ be such that we have $X_0 = \mathbf{0}_d$ (the d-dimensional vector of zeros) and

$$X_n(j) = \sum_{i=1}^{j-1} \beta_{d-j+i+1} X_n(i) + \sum_{i=j}^{d} \beta_{i-j+1} X_{n-1}(i) + \epsilon_{n,j}, \tag{4.1}$$

where $\epsilon_{n,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_x^2)$ and $\beta_{1:d}$ are some known static parameters. For the observations, we set

$$Y_n = X_n + \xi_n, \tag{4.2}$$

where $\xi_n(j) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_y^2)$, $j \in \{1, \ldots, d\}$. It is easily shown that this linear Gaussian model has the model structure (1.2) as in this case we have, for $k \geq 1$,

$$g(x_k, y_k) f(x_{k-1}, x_k) = \prod_{j=1}^{d} p(x_k(j) \mid x_{k-1}(j\colon d),\ x_k(1\colon j-1)) p(y_k(j) \mid x_k(j))$$

with the shown conditional densities being analytically available via (4.1) and (4.2).

We consider the standard PF and the STPF. Data are simulated from the model with $\sigma_x^2 = \sigma_y^2 = 1$, $\beta_{1:d} = (1, 1, \ldots, 1)$, $n = 1000$, and various choices of the dimension d. These parameters are also used within the filters. Both filters use the model transitions as the proposal and the likelihood function as the potential. Thus, the standard PF will propose from the d-dimensional law $p(x_n(1\colon d) \mid x_{n-1}(1\colon d))$. The STPF will propose one coordinate at a time from the model dynamics, that is we apply the algorithm described in Section 2 with proposal

$$q_{n,j}(x_n(j) \mid x_{n-1},\ x_n(1\colon j-1)) = p(x_n(j) \mid x_{n-1}(j\colon d),\ x_n(1\colon j-1)).$$

Adaptive resampling is used in all situations (with appropriate adjustment to the formula of calculating the weights for each of the N particles, as well as the estimates).

In Figure 1 we adjusted the number of particles for the PF and STPF so that the methods gave Monte Carlo estimates of $\mathbb{E}[X_n(1) \mid Y_{1:n}]$, $n = 1000$, with RMSE smaller than 0.005 (STPF fixed $N = 100$ and modified M_d; the RMSE averaged individual errors over 100 independent

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

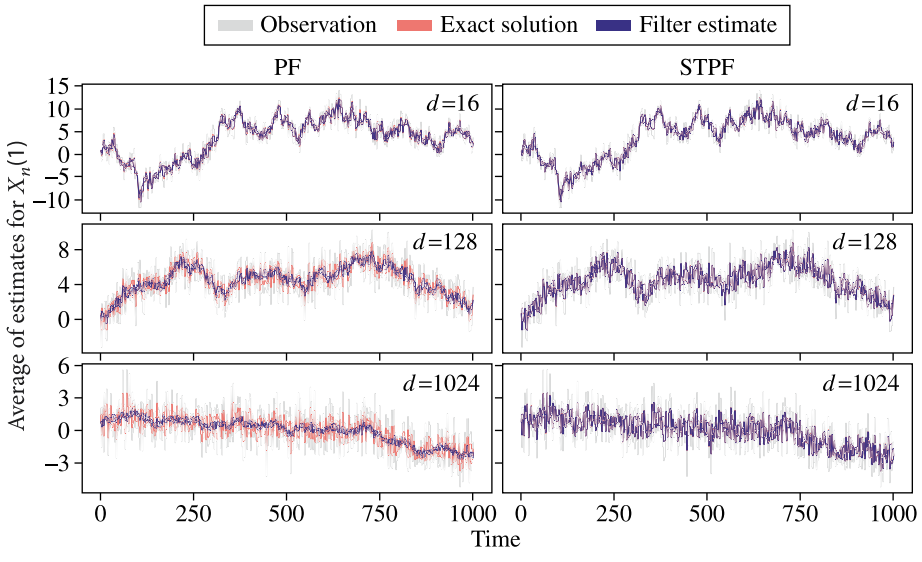

FIGURE 2: Average of estimators of $\mathbb{E}[X_n(1) \mid Y_{1:n}]$ for Model 1 across 100 algorithmic runs.

algorithmic executions, and scaled errors relative to standard deviation of $X_n(1) \mid Y_{1:n}$; both the STPF and PF were parallelised on a four-core PC, with the STPF sending $N/4$ islands at each core and the PF also sending a quarter of used particles at each time-step at each core).

Some results for $d \in \{16, 128, 1024\}$ are presented in Figures 2–4. For these figures, the STPF used $N = 100$ and $M_d = d$, and the standard PF used NM_d particles. The analytical solution for this model can be obtained via the Kalman filter and it is shown alongside the filter results in Figure 2.

The averages of estimators per time-step for the posterior mean of the first coordinate given all data up to time n, $\mathbb{E}[X_n(1) \mid Y_{1:n}]$, across 100 separate algorithmic runs are illustrated in Figure 2. For the STPF, each estimator corresponds to the double average over M_d, N as shown in Section 2. In Figure 2 we see that the standard particle filter collapses when the dimension becomes moderate or large. We see that there are no meaningful estimates when $d = 1024$ (as the estimates completely lose track of the observations and the analytical mean). In contrast, the STPF performs reasonably well in all three cases. In Figure 3 we show the ESS (calculated over the global filter for the STPF, and scaled by the number of particles for both algorithms) for each time-step of the two algorithms. The standard filter struggles significantly even in the $d = 16$ case and collapses when $d = 128$. The performance of the new algorithm is stable up to the dimension considered here. These conclusions are further supported in Figure 4 where the MSE (using the Kalman filter to obtain correct values) per time-step for the estimators of $\mathbb{E}[X_n(1) \mid Y_{1:n}]$, as estimated via 100 independent algorithmic runs, is displayed. To further analyse the stability and behaviour of the STPF, we consider the RMSE as a measure of accuracy. We consider the case $N = 100$ and $M_d = d$, with increasing dimension d. Results in Figure 5 suggest that in this case adjusting $M_d = \mathcal{O}(d)$ provides a robust algorithm for the choices of d under consideration.

The runtime cost of the STPF is, in general, $\mathcal{O}(nNd^3)$ when $M_d = \mathcal{O}(d)$. However, in many cases, the algorithm can be implemented with smaller cost. For example, in this case, it is possible to apply the algorithm with $\mathcal{O}(nNd^2)$ cost, as one can keep track of the conditional mean

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

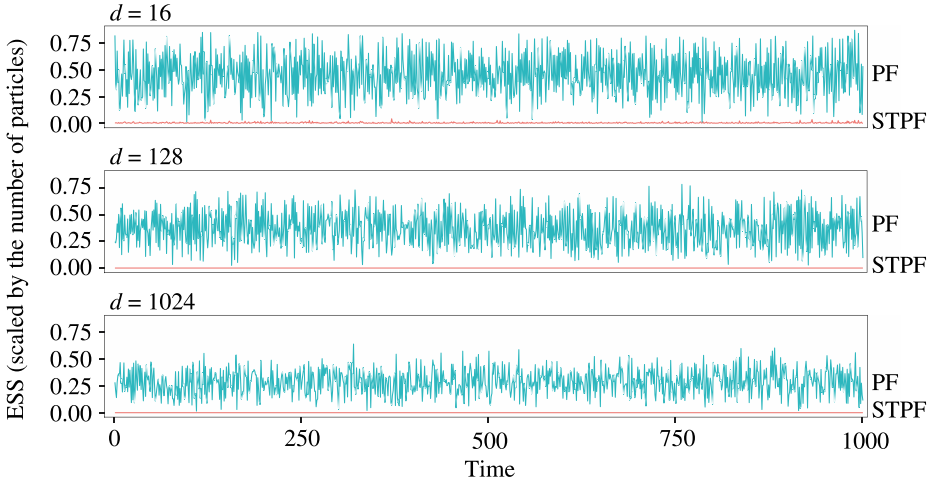

FIGURE 3: ESS plots of the PF and STPF algorithms for Model 1 from a single algorithmic run.

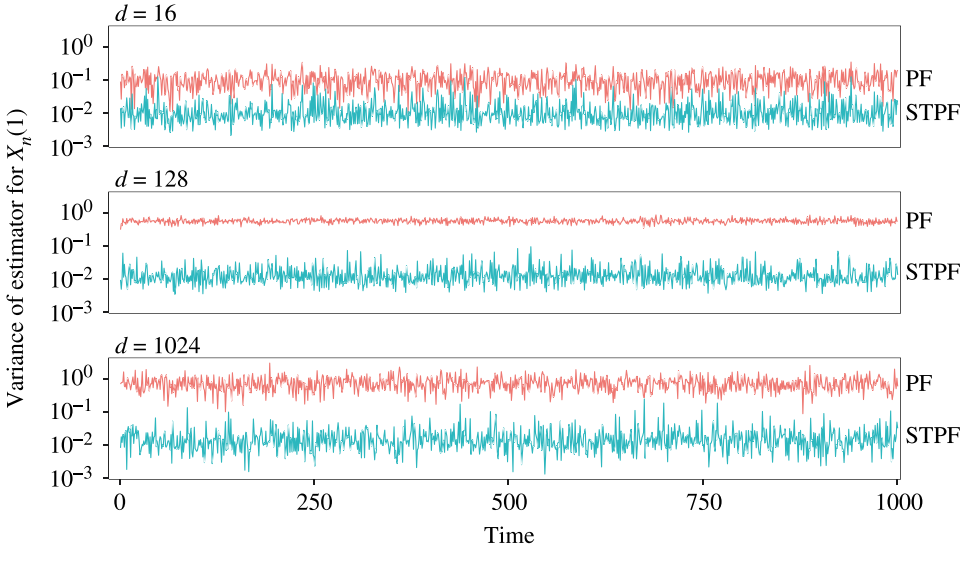

FIGURE 4: The MSE (on logarithm scale) of the PF and STPF algorithms for estimators of $\mathbb{E}[X_n(1) \mid Y_{1:n}]$ for Model 1 across 100 runs.

for the proposal for the jth coordinate and the evolving particle $(x_{k-1}(j+1:d), x_k(1:j))$ with $\mathcal{O}(1)$ calculations, $1 \leq j \leq d$. As the particles in the local filters now need to be resampled only after all d dimension steps, the cost of resampling is reduced to $O(nNM_d d)$ from $O(nNM_d d^2)$. This is illustrated in Figure 6: the slopes for the generic and the 'special' implementation with the $\mathcal{O}(1)$ calculation mentioned above, for this example are 3.026 and 1.981, respectively.

4.2. Model 2

4.2.1. *Model and simulation settings.* We consider a model on a two-dimensional graph, which follows one described in [15]. Let the components of state X_n be indexed by vertices $v \in V$,

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

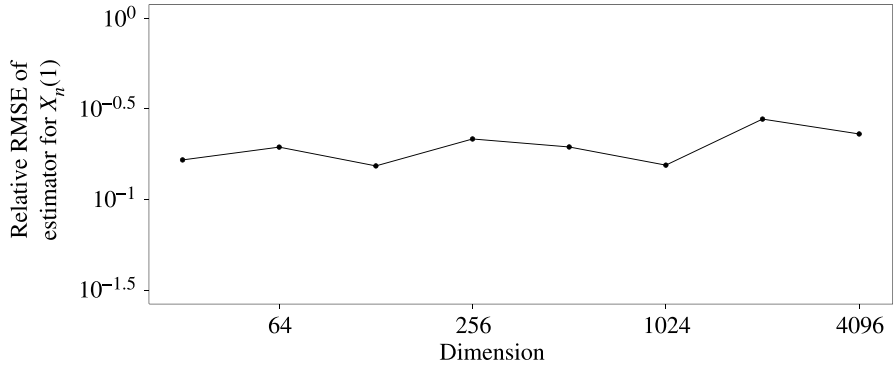

FIGURE 5: The RMSE of estimator of $\mathbb{E}[X_n(1) \mid Y_{1:n}]$ with $n = 1000$ for Model 1 over 100 algorithmic runs (MSE is scaled relatively to the variance of $X_n(1) \mid Y_{1:n}$). We varied d and applied the STPF with $N = 100$, $M_d = d$.

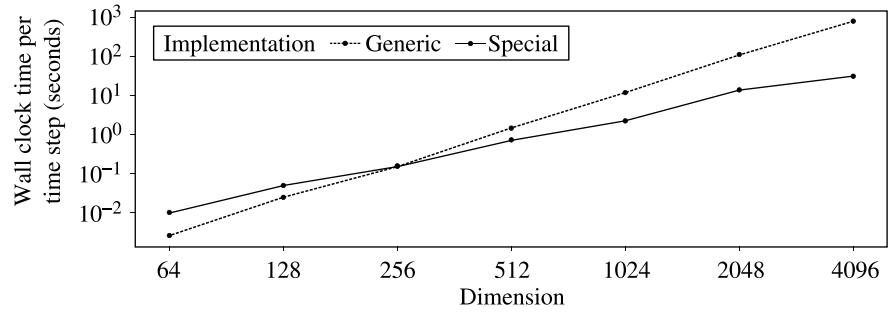

FIGURE 6: Runtime cost of the SFTP applied on Model 1 against dimension for a generic and a 'special' implementation of smaller cost. In both cases the SFTP used $N = 100$, $M_d = d$.

where $V = \{1, \ldots, s\}^2$. The dimension of the state space is thus $d = s^2$. The distance between two vertices, $v = (a, b)$, $u = (c, d)$, is calculated in the usual Euclidean sense, $D(v, u) = \sqrt{(a-c)^2 + (b-d)^2}$. At time n, conditionally on X_{n-1}, positions $X_n(v)$, $v \in V$, are independent and, for given $v \in V$, the variable $X_n(v)$ follows a mixture distribution

$$p(x_n(v) \mid x_{n-1}) = \sum_{u \in N(v)} w_u(v)\, p(x_n(v) \mid x_{n-1}(u)), \tag{4.3}$$

where $N(v) = \{u : D(v, u) \leq r\}$ for $r \geq 1$ is the neighbourhood of vertex v. For observations, we have the model

$$Y_n = X_n + \xi_n,$$

where $\xi_n(v)$, $v \in V$, are i.i.d. errors.

In this example we use a Gaussian mixture for (4.3) with component mean $X_{n-1}(u)$ and unit variance. The weights are set to be $w_u(v) \propto 1/(D(v, u) + \delta)$ and $\sum_{u \in N(v)} w_u(v) = 1$. In other words, when $\delta \to 0$, each vertex evolves as an independent Gaussian random walk, with more interesting spatial dependencies arising when $\delta > 0$. The i.i.d. observation errors are t-distributed with degrees of freedom ν. We simulated data from the model with $r = 1$, $\delta = 1$, $\nu = 10$, and various choices of dimension $d = s^2$. These true parameter values are also used in the filters.

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

We will compare the standard PF, the STPF, the marginal STPF algorithm (as described in Section 2.3), and the block PF in [15] in the $s = 32$ case, so $d = 1024$. For the STPF, we add the vertices one by one when applying the space-steps, in the order $(1, 1), \ldots, (1, s), (2, 1), \ldots, (2, s), \ldots, (s, s)$. The simulations for the STPF versions were performed with $N = M_d = 100$. The number of particles for the standard particle filter and block PF are NM_d. For the marginal algorithm, we also used $N = 1$ and $M_d = 1000$. For the marginal algorithm, MCMC mutation steps are applied in all d space-steps and are comprised of Gaussian random walk proposals with variance (the scale) $0.5/j$ and the corresponding Metropolis–Hastings correction; also, particles are propagated across space according to a kernel $q_{n,j}(x_n(j) \mid x_n(1\colon j-1)) \propto \sum_{l=1}^{M_d} p(x_n(j) \mid x_{n-1}^{i,l})$ (for island i), meant to be close to the marginal law of $x_n(j)$. The block size of the block PF is set to be b^2, $b \in \{1, \ldots, s\}$, and the domain is partitioned such that each block is itself a square. The optimal block size in [15] is about $b = 7$ for 10^4 particles and a two-dimensional graph. Thus, we considered the cases $b = 4$ and 8, the two nearest integers such that s is divisible by b.

Remark 4.1. The block PF partitions V into subsets $V_1, V_2, \ldots, V_\kappa$ for some $\kappa \geq 1$. Given particles $\{x_{n-1}^l\}_{l=1}^M$ approximating $p(x_{n-1} \mid y_{1\colon n-1})$, the block PF samples $x_n^l \sim p(x_n \mid x_{n-1}^l)$, $1 \leq l \leq M$. Each particle x_n^l is split into subparticles $x_n^l(V_1), x_n^l(V_2), \ldots, x_n^l(V_\kappa)$. Importance sampling is performed at all subparticles $\{x_n^l(V_j)\}_{l=1}^M$ using only the subset of data $Y_n(V_j)$ to provide the weighted subparticle population $\{x_n^l(V_j), W_n^l(V_j)\}_{l=1}^M$; this is repeated independently for $j = 1, 2, \ldots, \kappa$. Complete particles $\{x_n^l\}_{l=1}^M$ are obtained by resampling each of subparticle $x_n^l(V_j)$ from $\{x_n^l(V_j), W_n^l(V_j)\}_{l=1}^M$ for $1 \leq j \leq \kappa$. This process is iterated in the next time-steps. Due to its nature, the block PF is characterised by bias, which is spread nonuniformly across the vertices: vertices on the boundaries of the partition of V will have higher bias, while one can control the bias in the centre of, say, V_j, by taking the boundary of V_j far enough from its centre. In the context of our numerical example, V is split into blocks corresponding to equally sized squares, with side length equal to b.

4.2.2. *Results.* A single run takes around two minutes for the standard particle filter and the block filter on an Intel® Xeon™ W3550 CPU, with four cores and eight threads. It takes around 10 minutes for the STPF. For the marginal algorithm, it takes about 40 minutes when $N = 1$ and $M_d = 1000$, and about seven hours when $N = M_d = 100$.

The standard particle filter performs poorly and cannot provide adequate estimates (similar to the $d = 1024$ case in the previous example). In Figure 7 we observe the variance per time-step of the estimators for two vertices, across 30 runs. The first vertex, $X_n(3, 3)$ is in the interior of a block PF for both block sizes $b = 4$ and 8, whereas the second vertex considered, $X_n(8, 8)$, is on a block boundary for both sizes. In either case, the STPF significantly outperforms the block filter, albeit under slightly longer run times. The STPF does not collapse in high dimensions, but perhaps does not have excellent performance. The marginal STPF performs very well, but the computational time is substantially higher than all of the other algorithms. However, with $N = 1$ and $M_d = 1000 (= \mathcal{O}(d))$, the marginal STPF provides a good balance between performance and computational cost in challenging situations where the path degeneracy may hinder successful application of the new algorithm. The block filter variance is large already from the version of the algorithm with $b = 4$.

In Figure 8 we show the RMSE versus model dimension for estimates of $\mathbb{E}[X_n(1, 1) \mid Y_{1\colon n}]$, with $n = 1000$. This quantity is found for a number of choices of dimension d. The ground truth values were obtained from a very expensive simulation. In all cases, the STPF uses $N = 100$ and $M_d = d/8$. The runtime costs, again, of a generic implementation and a 'special' one exploiting

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

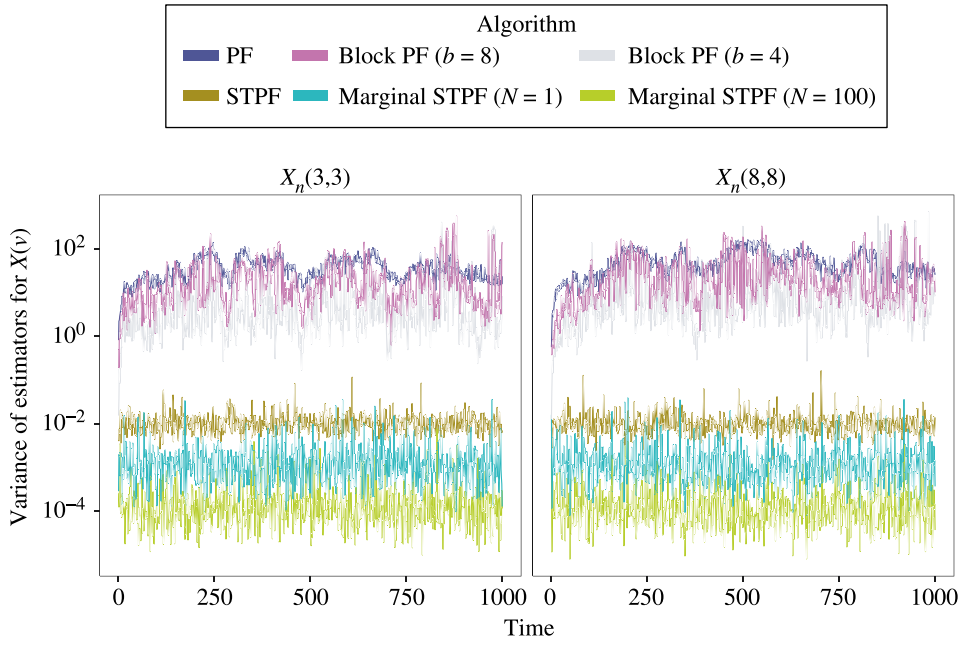

FIGURE 7: Variance plots (on logarithm scale) for estimators of $X_n(3,3)$ and $X_n(8,8)$ for Model 2. Variances are estimated from 100 independent runs for each algorithm. The STPF uses $N = M_d = 100$; the standard PF and block PF use $NM_d = 10^4$ particles; the two marginal algorithms have used $N = M_d = 100$ and $N = 1$, $M_d = 1000$. Note that the upper three plots are the PF and block PFs and the lower three plots are the STPF and marginal STPFs.

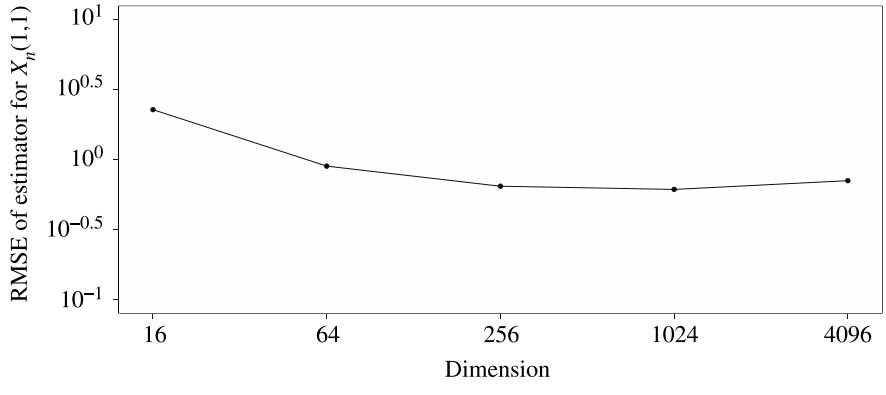

FIGURE 8: The RMSE of estimators of $\mathbb{E}[X_n(1,1) \mid Y_{1:n}]$ with $n = 1000$, for Model 2, over 100 algorithmic runs. We applied the STPF with $N = 100$, $M_d = d/8$, and varying dimension d.

the particular structure of this model that reduces the cost to $\mathcal{O}(nNd^2)$, are shown in Figure 9 (we omit details about the special implementation; the method is straightforward and we can give details upon request). Indeed, the slopes of fitted lines are 3.013 and 1.964, respectively.

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

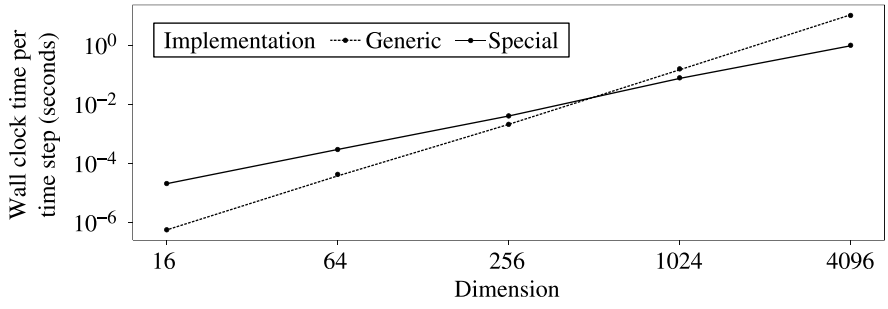

FIGURE 9: Runtime cost of the SFTP applied on Model 2 against dimension, for a generic and a 'special' implementation of smaller cost. In both cases, the SFTP used $N = 100$, $M_d = d/8$.

5. Summary

In this paper we have considered a novel class of particle algorithms for high-dimensional filtering problems and investigated both theoretical and practical aspects of the algorithm. We believe this paper opens new directions in an important and challenging contemporary Monte Carlo problem; several aspects of the method remain to be investigated in future research. There are indeed many possible extensions to the work in this paper. In particular, a theoretical analysis of the algorithm when the structure of the state-space model is more complex than the structures considered in this paper. Empirical results are encouraging, but may not tell the entire story with regards to dimension dependence. In addition, the interaction of dimension and time behaviour is of particular interest. Finally, an interesting approach in [13] has appeared subsequently to the first versions of this paper, also investigating algorithmic behaviour in filtering problems in high dimensions; it would be of interest to understand the relative theoretical benefits of both approaches.

Appendix A.

Proof of Proposition 3.1. We set

$$X = \frac{1}{M_d}\sum_{l=1}^{M_d}\frac{\alpha(x_1^{i,l}(1))}{q(x_1^{1,l}(1))}\Big/\int \alpha(x)\,\mathrm{d}x,$$

$$I = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d}\frac{1}{M_d}\sum_{l=1}^{M_d}\frac{\alpha(x_1^{i,l}(j))}{q(x_1^{i,l}(j))}\Big/\int \alpha(x)\,\mathrm{d}x.$$

Note that $\mathbb{E}[I] = \mathbb{E}[X] = 1$, so that, due to the i.i.d. structure along j, we have

$$\mathbb{E}[I^2] = \frac{1}{N}(\mathbb{E}[X^2])^d + \frac{N-1}{N}.$$

Also, due to the i.i.d. structure along j, l, we have

$$\mathbb{E}[X^2] = \frac{1}{M_d}\frac{\int a^2(x)/q(x)\,\mathrm{d}x}{(\int a(x)\,\mathrm{d}x)^2} + \frac{M_d - 1}{M_d}.$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

Finally, it follows that, due to the i.i.d. structure along n,

$$\mathbb{E}\left[\left(\frac{p^{N,M_d}(y_{1:n})}{p(y_{1:n})}-1\right)^2\right]=\mathbb{E}\left[\left(\frac{p^{N,M_d}(y_{1:n})}{(\int\alpha(x)\,\mathrm{d}x)^{nd}}\right)^2\right]-1=(\mathbb{E}[I^2])^n-1.$$

A synthesis of the above three equations gives the required result. □

Appendix B. Proof of Theorem 3.1

B.1. Further notation

To prove Theorem 3.1, we will first introduce another round of notation. Let $(E_n,\mathcal{E}_n)_{n\geq 0}$ be a sequence of measurable spaces endowed with a countably generated σ-field $\mathcal{E}_n$. The set $\mathcal{B}_b(E_n)$ denotes the class of bounded $\mathcal{E}_n/\mathbb{B}(\mathbb{R})$-measurable functions on E_n, where $\mathbb{B}(\mathbb{R})$ is the Borel σ-algebra on $\mathbb{R}$. We will consider nonnegative operators $K\colon E_{n-1}\times\mathcal{E}_n\to\mathbb{R}_+$ such that, for each $x\in E_{n-1}$, the mapping $A\mapsto K(x,A)$ is a finite nonnegative measure on $\mathcal{E}_n$ and, for each $A\in\mathcal{E}_n$, the function $x\mapsto K(x,A)$ is $\mathcal{E}_{n-1}/\mathbb{B}(\mathbb{R})$-measurable; the kernel K is Markovian if $K(x,\mathrm{d}y)$ is a probability measure for every $x\in E_{n-1}$. For a finite measure μ on $(E_{n-1},\mathcal{E}_{n-1})$ and Borel test function $f\in\mathcal{B}_b(E_n)$, we define

$$\mu K\colon A\mapsto\int K(x,A)\mu(\mathrm{d}x),\qquad Kf\colon x\mapsto\int f(y)K(x,\mathrm{d}y).$$

B.2. The Feynman–Kac model on an enlarged space

We will define a Feynman–Kac model on an appropriate enlarged space. That is, one Markov transition on the enlarged space will correspond to one observation time and will collect all d space-steps of the local filter for this time-step. Some care is needed with the notation as we need to keep track of the development of the coordinates at time n, together with the states at time $n-1$, as the latter are involved in the proposal.

Time-step 1. Consider observation time 1 of the algorithm. We define a sequence of random variables $Z_{1,j}^l$ with $j\in\{1,\dots,d+1\}$, $1\leq l\leq M_d$, such that $Z_{1,j}^l\in\mathbb{R}^j$, for $j\in\{1,\dots,d\}$, and $Z_{1,d+1}^l\in\mathbb{R}^d$. For $j\in\{1,\dots,d\}$, we will write the coordinates of $Z_{1,j}^l$ as $(Z_{1,j}^l(1),\dots,Z_{1,j}^l(j))$, with the obvious extension for the $j=d+1$ case. As x_0 is fixed, we will drop it from our notation, as will become clear below. Also, for simplicity, we simply write $q(\cdot)$ instead of the analytical $q_{1,j}(\cdot)$ as the subscripts are implied by those of $Z_{1,j}$. We follow this convention throughout Appendix B. We define the following sequence of Markov kernels corresponding to the proposal for the coordinates at the first time-step:

$$\begin{aligned}M_{1,1}(\mathrm{d}z_{1,1})&=q(z_{1,1})\,\mathrm{d}z_{1,1},\qquad j=1,\\ M_{1,j}(z_{1,j-1},\mathrm{d}z_{1,j})&\\ &=q(z_{1,j}(j)\mid z_{1,j-1})\,\mathrm{d}z_{1,j}(j)\delta_{\{z_{1,j-1}\}}(\mathrm{d}z_{1,j}(1\colon j-1)),\qquad j\in\{2,\dots,d\},\\ M_{1,j}(z_{1,j-1},\mathrm{d}z_{1,j})&=\delta_{\{z_{1,j-1}\}}(\mathrm{d}z_{1,j}),\qquad j=d+1.\end{aligned}$$

Next, we will take under consideration the weights and the resampling. For $j\in\{1,\dots,d\}$ and a probability measure μ on $\mathbb{R}^j$, define

$$\Phi_{1,j+1}(\mu)(\mathrm{d}z)=\frac{\int_{\mathbb{R}^j}\mu(\mathrm{d}z')G_{1,j}(z')M_{1,j+1}(z',\mathrm{d}z)}{\int_{\mathbb{R}^j}\mu(\mathrm{d}z')G_{1,j}(z')}.$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

For the local PF in observation time 1, write the unweighted empirical measure as

$$\eta_{1,j}^{M_d}(\mathrm{d}z) = \frac{1}{M_d}\sum_{l=1}^{M_d}\delta_{z_{1,j}^l}(\mathrm{d}z), \quad j \in \{1,\dots,d\}.$$

We also consider all random variables involved at time-step 1 and set

$$\boldsymbol{z}_1 = (z_{1,1}^{1:M_d},\dots,z_{1,d+1}^{1:M_d}).$$

The joint law of the samples required by the local filter is

$$\boldsymbol{\eta}_1(\mathrm{d}\boldsymbol{z}_1) = \Bigg(\prod_{l=1}^{M_d} M_{1,1}(\mathrm{d}z_{1,1}^l)\Bigg)\Bigg(\prod_{j=2}^{d+1}\prod_{l=1}^{M_d}\Phi_{1,j}(\eta_{1,j-1}^{M_d})(\mathrm{d}z_{1,j}^l)\Bigg). \tag{B.1}$$

Note that in the notation we have established herein, the potential $\boldsymbol{G}_1$ defined in the main text can now equivalently be expressed as

$$\boldsymbol{G}_1(z_1) = \prod_{j=1}^{d}\eta_{1,j}^{M_d}(G_{1,j}). \tag{B.2}$$

We also set $z_{1,d+1}^l(1) = z_{1,d+1}^l$.

Time-step $n \geq 2$. At subsequent observation times, $n \geq 2$, we again work with variables denoted $Z_{n,j}^l$, with $j \in \{1,\dots,d+1\}$, but this time we have to keep track of the corresponding paths at time $n-1$, thus we will use the notation $Z_{n,j}^l = (Z_{n,j}^{l,+}, Z_{n,j}^{l,-})$, with $Z_{n,j}^{l,+} \in \mathbb{R}^j$, $Z_{n,j}^{l,-} \in \mathbb{R}^d$, with the latter component referring to the 'tail' at time $n-1$ of the path found at $Z_{n,j}^+$ at time n and space position j. So, we have $Z_{n,j}^l \in \mathbb{R}^{j+d}$, $j \in \{1,\dots,d\}$ and $Z_{n,d+1}^l \in \mathbb{R}^{2d}$. We define the following sequence of kernels:

$$\begin{aligned}
M_{n,1}(z_{n-1,d+1}^+, \mathrm{d}z_{n,1}) &= q(z_{n,1}^+ \mid z_{n-1,d+1}^+)\,\mathrm{d}z_{n,1}^+\delta_{\{z_{n-1,d+1}^+\}}(\mathrm{d}z_{n,1}^-), \quad j=1,\\
M_{n,j}(z_{n,j-1}, \mathrm{d}z_{n,j}) &= q(z_{n,j}^+(j) \mid z_{n,j-1})\,\mathrm{d}z_{n,j}^+(j)\delta_{\{z_{n,j-1}^+\}}(\mathrm{d}z_{n,j}^+(1\!:\!j-1))\\
&\quad \times \delta_{\{z_{n,j-1}^-\}}(\mathrm{d}z_{n,j}^-), \qquad j \in \{2,\dots,d\},\\
M_{n,d+1}(z_{n,d}, \mathrm{d}z_{n,d+1}) &= \delta_{\{z_{n,d}\}}(\mathrm{d}z_{n,d+1}), \qquad j = d+1.
\end{aligned}$$

For $j \in \{2,\dots,d\}$ and a probability measure μ on $\mathbb{R}^{j+d}$, define the measure on $\mathbb{R}^{\min\{j+1,d\}+d}$ as

$$\Phi_{n,j+1}(\mu)(\mathrm{d}z) = \frac{\int \mu(\mathrm{d}z')G_{n,j}(z')M_{n,j+1}(z',\mathrm{d}z)}{\int \mu(\mathrm{d}z')G_{n,j}(z')}.$$

For the local PF at space-step j, we write the empirical measure

$$\eta_{n,j}^{M_d}(\mathrm{d}z) = \frac{1}{M_d}\sum_{l=1}^{M_d}\delta_{z_{n,j}^l}(\mathrm{d}z), \qquad j \in \{1,\dots,d\}.$$

Set $\boldsymbol{z}_n = (z_{n,1}^{1:M_d},\dots,z_{n,d+1}^{1:M_d})$. The transition law of all involved samples in the local particle filter is

$$\boldsymbol{M}_n(\boldsymbol{z}_{n-1},\mathrm{d}\boldsymbol{z}_n) = \Bigg(\prod_{l=1}^{M_d} M_{n,1}(z_{n-1,\,d+1}^{l,+},\mathrm{d}z_{n,1}^l)\Bigg)\Bigg(\prod_{j=2}^{d+1}\prod_{l=1}^{M_d}\Phi_{n,j}(\eta_{n,j-1}^{M_d})(\mathrm{d}z_{n,j}^l)\Bigg). \tag{B.3}$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

Then, we will work with the potential

$$\boldsymbol{G}_n(z_n) = \prod_{j=1}^{d} \eta_{n,j}^{M_d}(G_{n,j}). \tag{B.4}$$

Algorithm 1 corresponds to a standard PF approximation (with N particles) of a Feynman–Kac model specified by the initial distribution (B.1), the Markovian transitions (B.3), and the potentials in (B.2), (B.4). Thus, for the Monte Carlo algorithm with N particles, set $\boldsymbol{\eta}_n^N$ for the N-empirical measure of $\boldsymbol{z}_n^{1:N}$ and set, for μ a probability measure, $n \geq 2$,

$$\boldsymbol{\Phi}_n(\mu)(\mathrm{d}z) = \frac{\int \mu(\mathrm{d}z')\boldsymbol{G}_{n-1}(z')\boldsymbol{M}_n(z', \mathrm{d}z)}{\int \mu(\mathrm{d}z')\boldsymbol{G}_{n-1}(z')}.$$

Then our global filter samples, from the path measure up to observation time n,

$$\left(\prod_{i=1}^{N} \eta_1(\mathrm{d}z_1^i)\right)\left(\prod_{k=2}^{n}\prod_{i=1}^{N} \boldsymbol{\Phi}_k(\boldsymbol{\eta}_{k-1}^N)(\mathrm{d}z_k^i)\right)$$

do not include resampling at observation time n. We use the standard definition of the normalising constant, for any $n \geq 1$,

$$\boldsymbol{\gamma}_n(\varphi) = \int \eta_1(\mathrm{d}z_1) \prod_{p=2}^{n} \boldsymbol{G}_{p-1}(z_{p-1})\boldsymbol{M}_p(z_{p-1}, \mathrm{d}z_p)\varphi(z_n) \tag{B.5}$$

and set $\boldsymbol{\eta}_n(\varphi) = \boldsymbol{\gamma}_n(\varphi)/\boldsymbol{\gamma}_n(1)$; thus $\boldsymbol{\eta}_n$ corresponds to the predictive distribution at time n for the global filter. Note that, from (B.5), we can equivalently write for the unnormalised measure

$$\boldsymbol{\gamma}_n(\boldsymbol{\varphi}) = \eta_1(\boldsymbol{G}_1\boldsymbol{M}_2(\boldsymbol{G}_2\boldsymbol{M}_3 \cdots (\boldsymbol{G}_{n-1}\boldsymbol{M}_n(\boldsymbol{\varphi})))). \tag{B.6}$$

B.3. Calculation of quantities for the global filter

We consider functions of the particular form

$$\boldsymbol{\phi}(z_p) = \frac{1}{M_d}\sum_{l=1}^{M_d} \phi(z_{p,d+1}^{l,+}), \qquad \phi \in \mathcal{B}_b(\mathbb{R}^d).$$

For functions of this type, we write $\boldsymbol{\phi} \in \mathcal{A}_p$. We will illustrate that, upon application on this family, several Feynman–Kac quantities of the global model (with signal dynamics $\boldsymbol{\eta}_1, \boldsymbol{M}_2, \ldots$, and potentials $\boldsymbol{G}_1, \boldsymbol{G}_2 \ldots$) coincide with those of the original model of interest (with signal dynamics $f_1, f_2, \ldots$ and potentials $g_1, g_2, \ldots$). In particular, we calculate $\boldsymbol{M}_p(\boldsymbol{G}_p\boldsymbol{\phi})$ as, from (B.6), it is the building block for other expressions. Note that, we can write

$$\begin{aligned}
\boldsymbol{M}_p(\boldsymbol{G}_p\boldsymbol{\phi}) &= \int \boldsymbol{M}_p(z_{p-1}, \mathrm{d}z_p)\boldsymbol{G}_p(z_p)\frac{1}{M_d}\sum_{l=1}^{M_d}\phi(z_{p,d+1}^{l,+}) \\
&= \int \left(\prod_{l=1}^{M_d} M_{p,1}(z_{p-1,d+1}^{l,+}, \mathrm{d}z_{p,1}^l)\right)\left(\prod_{j=2}^{d+1}\prod_{l=1}^{M_d}\Phi_{p,j}(\eta_{p,j-1}^{M_d})(\mathrm{d}z_{p,j}^l)\right) \\
&\quad \times \prod_{j=1}^{d}\eta_{p,j}^{M_d}(G_{p,j}) \cdot \eta_{p,d+1}^{M_d}(\phi).
\end{aligned}$$

So, the integral now concerns the local PF with weights $G_{p,j}$ and Markov kernels $M_{q,j}$.

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

In particular, the integral corresponds to the expected value of the particle approximation of the standard Feynman–Kac unnormalised estimator with standard unbiasedness properties [6, Theorem 7.4.2]. That is, the integral is equal to

$$\frac{1}{M_d}\sum_{l=1}^{M_d}\mathbb{E}[\phi(z_{p,d+1}^l)G_{p,d}(z_{p,d}^l)\cdots G_{p,2}(z_{p,2}^l)G_{p,1}(z_{p,1}^l)\mid z_{p-1,d+1}^{l,+}]$$

(note that here, for each l, the process $z_{p,1}^l, z_{p,2}^l, \ldots, z_{p,d+1}^l$ is a Markov chain evolving via $M_{p,1}(z_{p-1,d+1}^{l,+}, \mathrm{d}z_{p,1}^l)$, $M_{p,2}(z_{p,1}^l, \mathrm{d}z_{p,2}^l), \ldots, M_{p,d+1}(z_{p,d}^l, \mathrm{d}z_{p,d+1}^l)$, respectively). From the analytical definition of the kernels and the weights, this latter quantity is easily seen to be equal to

$$\begin{aligned}
&\frac{1}{M_d}\sum_{l=1}^{M_d}\int\phi(z)\prod_{j=1}^{d}\alpha_{p,j}(y_p, z_{p-1,d+1}^{l,+}, z(1\colon j))\,\mathrm{d}z(1\colon j)\\
&\quad=\frac{1}{M_d}\sum_{l=1}^{M_d}\int\phi(z)f_p(z_{p-1,d+1}^{l,+}, \mathrm{d}z)g_p(z, y_p)\,\mathrm{d}z\\
&\quad=\eta_{p-1,d+1}^{M_d}(f_p(g_p\phi)).
\end{aligned}$$

So, we have obtained

$$\boldsymbol{M}_p(\boldsymbol{G}_p\boldsymbol{\phi})=\eta_{p-1,d+1}^{M_d}(f_p(g_p\phi))\in\mathcal{A}_{p-1}. \tag{B.7}$$

Thus, applying the above result recursively, we obtain, from (B.6),

$$\boldsymbol{\gamma}_n(\boldsymbol{G}_n\boldsymbol{\phi})=\int\prod_{p=1}^{n}f_p(x_{p-1}, \mathrm{d}x_p)g_p(x_p, y_p)\phi(x_p). \tag{B.8}$$

Using the standard Feynman–Kac notation, this latter integral can be denoted as $\gamma_n(g_n\phi)$ for the unnormalised measure γ_n. Thus, for instance, for the normalising constants, we have

$$\boldsymbol{\gamma}_n(\boldsymbol{G}_n)=\gamma_n(g_n)\equiv p(y_{1\colon n}).$$

Proof of Theorem 3.1. We have established that the algorithm is a standard PF approximation of a Feynman–Kac formula on an extended space. Thus, standard results, e.g. in [6], will give consistency for Monte Carlo estimates on the enlarged state-space. It remains only to show that indeed the quantities in the statement of Theorem 3.1 correspond to Monte Carlo averages of the global filter in the enlarged space. We look at the two quantities in the statement of the theorem. For the first, we set (we assume that $n\geq 2$ as the $n=1$ case is treated in an entirely similar manner, with small changes in the notation)

$$\boldsymbol{\varphi}(z_n)=\frac{1}{M_d}\sum_{l=1}^{M_d}\varphi(z_{n,d+1}^{l,+})\in\mathcal{A}_n,$$

and we immediately have (denoting by $\check{z}_n^i$ the resampled islands, under the weights $\boldsymbol{G}_n(z_n^i)$)

$$\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\varphi}(\check{z}_n^i)\xrightarrow{\mathbb{P}}\frac{\int\boldsymbol{\eta}_n(\mathrm{d}z_n)\boldsymbol{G}_n(z_n)\boldsymbol{\varphi}(z_n)}{\int\boldsymbol{\eta}_n(\mathrm{d}z_n)\boldsymbol{G}_n(z_n)}=\frac{\boldsymbol{\gamma}_n(\boldsymbol{G}_n\boldsymbol{\varphi})}{\boldsymbol{\gamma}_n(\boldsymbol{G}_n)}.$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

Note that now the quantity on the left is precisely the double average in the statement of the theorem and the quantity on the right, from (B.8), is equal to $\gamma_n(g_n\varphi)/\gamma_n(g_n) = \pi_n(\varphi)$. For the last statement in the theorem, the quantity on the left is $\boldsymbol{\gamma}_n^N(\boldsymbol{G}_n)$ which, from standard PF theory converges in probability to $\boldsymbol{\gamma}_n(\boldsymbol{G}_n) = \gamma_n(g_n) = p(y_{1:n})$. □

Appendix C.

Proof of Theorem 3.2. Recall the notation for the global filter from Appendix B. We define the semi-group

$$\boldsymbol{Q}_{p+1}(z_p, \mathrm{d}z_{p+1}) = \boldsymbol{G}_p(z_p)\boldsymbol{M}_{p+1}(z_p, \mathrm{d}z_{p+1})$$

and we also set

$$\boldsymbol{Q}_{p,n}(\varphi) = \int \boldsymbol{Q}_{p+1}(z_p, \mathrm{d}z_{p+1}) \times \cdots \times \boldsymbol{Q}_n(z_{n-1}, \mathrm{d}z_n)\boldsymbol{\varphi}(z_n).$$

Recall from the main result in (B.7) in Appendix B, connecting the global with the local filter, that $\boldsymbol{M}_p(\boldsymbol{G}_p) = \eta_{p-1,d+1}^{M_d}(f_p(g_p))$, and upon an iterative application of this result

$$\boldsymbol{Q}_{p,n}(1) = \boldsymbol{G}_p(z_p)\eta_{p,d+1}^{M_d}(\hat{q}_{p+1,n-1}(1)).$$

We also have $\boldsymbol{\gamma}_n(1) = \gamma_n(1) = \gamma_p(g_p\hat{q}_{p+1,n-1}(1)) = \pi_p(\hat{q}_{p+1,n-1}(1))\gamma_p(g_p)$ and, finally, that $\gamma_p(g_p) = \pi_{p-1}(f_p(g_p))\gamma_p(1)$. Using all these expressions, simple calculations reveal that

$$\sigma_n^2 = \sum_{p=1}^{n} \frac{\boldsymbol{\gamma}_p(1)^2}{\boldsymbol{\gamma}_n(1)^2}\boldsymbol{\eta}_p((\boldsymbol{Q}_{p,n}(1) - \boldsymbol{\eta}_p(\boldsymbol{Q}_{p,n}(1)))^2) = \sum_{p=1}^{n} \boldsymbol{\eta}_p\left(\left(\frac{\boldsymbol{G}_p(z_p)}{\boldsymbol{M}_p(\boldsymbol{G}_p)}A_p - 1\right)^2\right), \quad \text{(C.1)}$$

where we have defined

$$A_p = \frac{\eta_p^{M_d}(\hat{q}_{p+1,n-1}(1))}{\pi_p(\hat{q}_{p+1,n-1}(1))}\frac{\eta_{p-1}^{M_d}(f_p(g_p))}{\pi_{p-1}(f_p(g_p))}.$$

The main thing to note now is that $\boldsymbol{G}_p(z_p)/\boldsymbol{M}_p(\boldsymbol{G}_p)$ corresponds to the standard estimate of the normalising constant for the pth local filter divided with its expected value, and we can use standard results from the literature to control its second moment. Indeed, by assumptions (A1), (A2), and [7, Theorem 16.4.1] (see Remark C.1), there exists $\tilde{c} < \infty$ (which does not depend on p or $\boldsymbol{z}_p$) such that, for any $d \geq 1$ and any $M_d \geq \tilde{c}d$,

$$\boldsymbol{M}_p\left(\left(\frac{\boldsymbol{G}_p(z_p)}{\boldsymbol{M}_p(\boldsymbol{G}_p)} - 1\right)^2\right) \leq \frac{\tilde{c}(2+e)\,\mathrm{d}}{M_d},$$

where the upper bound depends only on d via the term d/M_d. Note also that $f_p(g_p) \equiv \hat{q}_{p-1,p}(1)$, so by (A2) and Jensen's inequality (so that $M_d/\sum_{i=1}^{M_d} x_i \leq \sum_{i=1}^{M_d} 1/x_i M_d$ for positive x_i), we have

$$0 \leq A_p \leq c^4.$$

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

Thus, returning to (C.1), and using the last two equations, we obtain, starting with the C_2-inequality,

$$\begin{aligned}
&\boldsymbol{\eta}_p\left(\left(\frac{\boldsymbol{G}_p(z_p)}{\boldsymbol{M}_p(\boldsymbol{G}_p)}A_p - 1\right)^2\right)\\
&\quad\le 2\boldsymbol{\eta}_p\left(\left(\frac{\boldsymbol{G}_p(z_p)}{\boldsymbol{M}_p(\boldsymbol{G}_p)} - 1\right)^2 c^8\right) + 2\boldsymbol{\eta}_p((A_p-1)^2)\\
&\quad= 2c^8\boldsymbol{\gamma}_{p-1}\left(\boldsymbol{G}_{p-1}\boldsymbol{M}_p\left(\frac{\boldsymbol{G}_p(z_p)}{\boldsymbol{M}_p(\boldsymbol{G}_p)} - 1\right)^2\right)\Big/\boldsymbol{\gamma}_p(1) + 2\boldsymbol{\eta}_p((A_p-1)^2)\\
&\quad\le \frac{2c^8\tilde{c}(2+e)\,\mathrm{d}}{M_d} + 2c^8.
\end{aligned}$$

From here, one can easily complete the proof and, hence, we conclude. □

Remark C.1. In the proof of Theorem 3.2 we have used [7, Theorem 16.4.1]. This is a result on the relative variance of the particle estimate of the normalising constant, and as stated in [7] does not include a function, that is, an estimate of the form $\prod_{j=1}^{d}\eta_{p,j}^{M_d}(G_{p,j})\eta_{p,j}^{M_d}(\varphi)$ for some $\varphi\in\mathcal{B}_b(\mathbb{R}^d)$. Based on a personal communication with Pierre Del Moral, [7, Theorem 16.4.1] can be extended to include a function, by modification of the potential functions and the use of the final formula in [7, p. 484].

Acknowledgements

Ajay Jasra and Yan Zhou were supported by a Singapore Government AcRF Tier 2 grant R-155-000-143-112. Alexandros Beskos was supported by the Leverhulme Trust Prize. We thank Pierre Del Moral for many useful conversations on this work.

References

[1] Bérard, J., Del Moral, P. and Doucet, A. (2014). A lognormal central limit theorem for particle approximations of normalizing constants. *Electron. J. Prob.* **19,** 1–28.

[2] Beskos, A., Crisan, D. and Jasra, A. (2014). On the stability of sequential Monte Carlo methods in high dimensions. *Ann. Appl. Prob.* **24,** 1396–1445.

[3] Beskos, A., Crisan, D., Jasra, A. and Whiteley, N. P. (2014). Error bounds and normalising constants for sequential Monte Carlo samplers in high dimensions. *Adv. Appl. Prob.* **46,** 279–306.

[4] Bickel, P., Li, B. and Bengtsson, T. (2008). 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* (Inst. Math. Statist. (IMS) Collect. **3**), eds B. Clarke and S. Ghosal, Institute of Mathematical Statistics, Beachwood, OH, pp. 318–329.

[5] Cérou, F., Del Moral, P. and Guyader, A. (2011). A nonasymptotic theorem for unnormalized Feynman–Kac particle models. *Ann. Inst. H. Poincaré Prob. Statist.* **47,** 629–649.

[6] Del Moral, P. (2004). *Feynman–Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, New York.

[7] Del Moral, P. (2013). *Mean Field Simulation for Monte Carlo Integration* (Monogr. Statist. Appl. Prob. **126**). CRC Press, Boca Raton, FL.

[8] Del Moral, P., Doucet, A. and Jasra, A. (2006). Sequential Monte Carlo samplers. *J. R. Statist. Soc. B* **68,** 411–436.

[9] Del Moral, P., Doucet, A. and Jasra, A. (2012). On adaptive resampling procedures for sequential Monte Carlo methods. *Bernoulli* **18,** 252–278.

[10] Doucet, A. and Johansen, A. M. (2011). A tutorial on particle filtering and smoothing: fifteen years later. In *The Oxford Handbook of Nonlinear Filtering*, eds D. Crisan and B. Rozovsky, Oxford University Press, pp. 656–704.

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press

[11] Johansen, A. M., Whiteley, N. and Doucet, A. (2012). Exact approximation of Rao–Blackwellised particle filters. In *Proc. 16th IFAC Symp. on System Identification* The International Federation of Automatic Control, Brussels, pp. 488–493.
[12] Kantas, N., Beskos, A. and Jasra, A. (2014). Sequential Monte Carlo methods for high-dimensional inverse problems: a case study for the Navier–Stokes equations. *SIAM/ASA J. Uncertain. Quantif.* **2,** 464–489.
[13] Naesseth, C. A., Lindsten, F. and Schön, T. B. (2015). Nested sequential Monte Carlo methods. In *Proc. 32nd Internat. Conf. on Machine Learning*, pp. 1292–1301.
[14] Poyiadjis, G., Doucet, A. and Singh, S. S. (2011). Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika* **98,** 65–80.
[15] Rebeschini, P. and Van Handel, R. (2015). Can local particle filters beat the curse of dimensionality? *Ann. Appl. Prob.* **25,** 2809–2866.
[16] Rubin, D. (1988). Using the SIR algorithm to simulate posterior distributions. In *Bayesian Statistics 3*, eds J. M. Bernado *et al.*, Oxford University Press, pp. 395–402.
[17] Vergé, C., Dubarry, C., Del Moral, P. and Moulines, E. (2015). On parallel implementation of Sequential Monte Carlo methods: the island particle model. *Statist. Comput.* **25,** 243–260.

https://doi.org/10.1017/apr.2016.77 Published online by Cambridge University Press