A DIVIDE-AND-CONQUER SEQUENTIAL MONTE CARLO APPROACH TO HIGH DIMENSIONAL FILTERING

Francesca R. Crucinio and Adam M. Johansen

Department of Statistics, University of Warwick

Abstract: We propose a divide-and-conquer approach to filtering which decomposes the state variable into low-dimensional components to which standard particle filtering tools can be successfully applied and recursively merges them to recover the full filtering distribution. It is less dependent upon factorization of transition densities and observation likelihoods than competing approaches and can be applied to a broader class of models. Performance is compared with state-of-the-art methods on a benchmark problem and it is demonstrated that the proposed method is broadly comparable in settings in which those methods are applicable, and that it can be applied in settings in which they cannot.

Key words and phrases: data assimilation; marginal particle filter; particle filtering; state-space model; spatio-temporal models

1. Introduction

Particle filters (PFs), an instance of sequential Monte Carlo (SMC) methods, are a popular class of algorithms to perform state estimation for state

space models (SSM) — or general-state-space hidden Markov models as they are sometimes known. We consider the class of SSMs with a latent $\mathbb{R}^d$-valued process $(X_t)_{t \geq 1}$ and conditionally independent $\mathbb{R}^p$-valued observations $(Y_t)_{t \geq 1}$. Such a SSM $(X_t, Y_t)_{t \geq 1}$ is defined by the transition density $f_t(x_{t-1}, x_t)$ of the latent process, with the convention that $f_1(x_0, x_1) \equiv f_1(x_1)$, and the observation likelihood $g_t(y_t|x_t)$. In this work, we are interested in approximating the sequence of filtering distributions, $(p(x_t|y_{1:t}))_{t \geq 1}$, i.e. at each time t the distribution of the latent state at that time given the observations obtained by that time.

Basic PF algorithms are known to suffer from the curse of dimensionality, requiring an exponential increase in computational requirements as the dimension d grows, limiting their applicability to large systems (Rebeschini and Van Handel, 2015; Bengtsson et al., 2008). While the ensemble Kalman filter (Evensen, 2009) can tackle high dimensional problems, it involves approximations which do not disappear even asymptotically and does not perform well if the model is far from linear and Gaussian (Lei et al., 2010).

To extend the use of particle filters to high dimensional problems it is natural to attempt to exploit the fact that dependencies in high dimensional SSMs encountered in practice are often local in space in order to decompose

the filtering problem into a collection of local low-dimensional problems which can somehow be combined; examples of this strategy are the block particle filter (Rebeschini and Van Handel, 2015), space-time particle filters (STPF; Beskos et al. (2017)) and nested sequential Monte Carlo (NSMC) methods (Næsseth et al., 2015, 2019).

We propose a divide-and-conquer approach in which the state space is divided into smaller subsets over which standard particle filtering ideas can be applied, these smaller subsets are then recursively merged in a principled manner to obtain approximations over the full state space. Our method is an extension of the divide-and-conquer sequential Monte Carlo (DaC-SMC) algorithm introduced by Lindsten et al. (2017) to the filtering context, which exploits ideas akin those in Klaas et al. (2005); Lin et al. (2005) to marginalize out the past $x_{1:t-1}$ at a given time t.

In order to apply DaC-SMC to the filtering problem, we define a non-standard sequence of targets evolving both in space and in time: at a given time t we define d univariate targets serving as proxies for the marginals of the filtering distribution, $p(x_t(i)|y_{1:t})$ for $i = 1, \ldots, d$, we then iteratively combine these lower dimensional targets to obtain approximations of higher-dimensional marginals (e.g. $p(x_t(i:i+1)|y_{1:t})$) until we recover the full filtering distribution, $p(x_t|y_{1:t}) \equiv p(x_t(1:d)|y_{1:t})$.

Unlike NSMC and STPF, this approach does not require analytic expressions for marginals of the transition density or observation likelihood, but only point-wise evaluations of f_t and g_t, making it suitable to tackle high dimensional SSM in which the observations are correlated in non-trivial ways (cf. the model in Section 4.2), as is common in real applications (see, e.g., Chib et al. (2009, Section 2)).

We review the basic ideas of particle filtering, its marginal variant and divide-and-conquer SMC in Section 2; we then show how to extend the ideas underlying DaC-SMC to the filtering problem in Section 3, where we also discuss strategies to improve computational cost and accuracy. Finally, in Section 4 we compare the performances of the divide-and-conquer approach with NSMC and STPF on a simple linear Gaussian SSM for which the Kalman filter provides the exact filtering distribution. Our experiments show that the errors in approximating the true filtering distribution obtained with the divide-and-conquer approach are comparable to those of NSMC and STPF. We then consider a spatial model whose correlation structure in the observation model makes it impossible to apply NSMC or STPFs (at least without additional approximations) and empirically show that the proposed approach can recover stable estimates of the filtering distribution which, in small dimensional settings, coincide with those obtained

with a bootstrap PF with a large number of particles.

2. Background

2.1 Particle Filtering

We describe here the basic SMC approach, often referred to as sequential importance resampling (Doucet and Johansen, 2011, p. 15), and refer to Liu (2001); Chopin and Papaspiliopoulos (2020) for a more extensive treatment.

Given the sequence of unnormalized target densities $(\gamma_t)_{t \geq 1}$, with

$$\gamma_t(x_{1:t}) = p(x_{1:t}, y_{1:t}) = \prod_{k=1}^{t} f_k(x_{k-1}, x_k) g_k(y_k | x_k), \qquad (2.1)$$

defined on $(\mathbb{R}^d)^t$, PFs proceed iteratively, and, at time $t - 1$ approximate

$$\pi_{t-1} := \gamma_{t-1} / \int \gamma_{t-1}(x_{1:t-1}) \mathrm{d}x_{1:t-1}$$

with a cloud of particles $\{x_{1:t-1}^n\}_{n=1}^N$. The particles are propagated forward in time using a Markov kernel $K_t(x_{1:t-1}, \cdot)$, reweighted using the weight function $w_t := \gamma_t / \gamma_{t-1} \otimes K_t$ and resampled to obtain a new particle population $\{x_{1:t}^n\}_{n=1}^N$ approximating π_t.

Standard PFs formally target distributions (2.1) whose dimension in-

creases at every time step t, although one is often interested only in the final time marginal of the distributions so approximated, in this case the filtering distribution. An alternative to this approach is given by marginal particle filters (MPFs; Klaas et al. (2005)) and the closely related ideas of Lin et al. (2005). MPFs target the filtering distribution directly

$$\gamma_t(x_t) = p(x_t|y_{1:t}) = g_t(y_t|x_t) \int f_t(x_{t-1}, x_t)p(x_{t-1}|y_{1:t-1})\mathrm{d}x_{t-1}; \qquad (2.2)$$

and, since the integral w.r.t. x_{t-1} is intractable, replace $p(x_{t-1}|y_{1:t-1})$ with its particle approximation obtained at time $t-1$. Given the new sequence of targets, MPFs proceed as standard PFs, with the only difference being that, whenever we need to compute an integral w.r.t. x_{t-1}, this is approximated using π_{t-1}^N, obtained by normalizing γ_{t-1}^N, the particle approximation of $p(x_{t-1}|y_{1:t-1})$. Basic MPFs incur an $O(N^2)$ cost for each time step, because of the presence of the integral w.r.t. x_{t-1} in the weight computations, although lower cost strategies might be employed in some cases (Lin et al. (2005); see also Klaas et al. (2006)).

2.2   Particle Filters for High Dimensional Problems

We briefly summarize three classes of particle filters which make use of space decompositions to tackle the filtering problem which we believe to be the state-of-the-art in Monte Carlo approximation of high dimensional filtering distributions.

The block particle filter (BPF; Rebeschini and Van Handel (2015)) algorithm relies on a decomposition of the state space $\mathbb{R}^d$ into lower dimensional blocks on which, at each t, one step of a standard PF is run. The approximation of the filtering distribution over the whole state space is obtained by taking the product of the lower dimensional approximations on each block. BPFs are inherently biased because of the decomposition into blocks; although this bias can be eliminated asymptotically by allowing the blocks to grow at an appropriate rate with computational effort.

Space-time particle filters (STPF; Beskos et al. (2017)) exploit local dependence structures in the observation y_t to gradually introduce the likelihood term by decomposing the space dimension into smaller subsets and running N independent particle filters on each of the subsets (also called islands) which are then combined using an importance resampling step, the presence of the latter, guarantees asymptotically consistent approximations, contrary to BPFs. Crucial to the implementation of STPF is that the joint

law (2.1) at time t can be factorized so that the marginal of $x_t(i)$ given the observations and the past only depends on a neighbourhood of $x_t(i)$, $\{x_t(j) : j \in \mathcal{A}\}$ for some $\mathcal{A} \subset \{1, \ldots, d\}$, for all $i = 1, \ldots, d$. STPFs are particularly amenable to SSM which are time discretizations of SDEs, since in this case one can build time discretization schemes which guarantee analytical forms for the marginals (Akyildiz et al., 2022). A marginal version of STPFs also exists (Beskos et al., 2017; Xu and Jasra, 2019).

Nested sequential Monte Carlo (NSMC; Næsseth et al. (2015)) treats the problem of recovering the filtering distribution as a smoothing problem, where the time variable is replaced by the dimension d, approximates the fully adapted proposal of Pitt and Shephard (1999) with an inner SMC iteration and then uses the result in the outer level which corresponds to a standard forward filtering backward simulation algorithm. NSMC is particularly well-suited for Markov random fields in which the temporal and the spatial components can be separated, as this makes the backward simulation straightforward.

2.3 Divide and Conquer SMC

Divide-and-Conquer SMC (DaC-SMC; Lindsten et al. (2017)) is an extension of standard SMC in which a collection of (unnormalized) target distri-

butions $(\gamma_u)_{u \in \mathbb{T}}$ is indexed by the nodes of a rooted tree, $\mathbb{T}$, and particles evolve from the leaves to the root, $\mathfrak{R}$, rather than along a sequence of distributions indexed by (algorithmic) time. It shares many of the convergence properties of standard SMC (Kuntz et al., 2021).

The target distributions are defined on spaces whose dimension grows as we progress up the tree: for each u, $\pi_u \propto \gamma_u$ is a density over $\mathbb{R}^{|\mathbb{T}_u|}$ where $\mathbb{T}_u$ denotes the sub-tree of $\mathbb{T}$ rooted at u (obtained by removing all nodes from $\mathbb{T}$ except for u and its descendants) and $|\mathbb{T}_u|$ denotes its cardinality. We focus here on the case in which the state space is $\mathbb{R}^d$, however, essentially the same construction allows for much more general spaces, including those with discrete components.

As in standard SMC, each distribution γ_u is approximated by a particle population $\{x_u^n\}_{n=1}^N$. However, these distributions do not evolve 'linearly' but are merged whenever the corresponding branches of $\mathbb{T}$ merge. For simplicity, we describe here the case in which $\mathbb{T}$ is a binary tree and each non-leaf node u has two children, a left child $\ell(u)$ and a right child $r(u)$.

If u is a leaf node, the algorithm performs a simple importance sampling step with proposal K_u and importance weight $w_u := \gamma_u / K_u$ to obtain a weighted particle population $\{x_u^n, w_u^n\}_{n=1}^N$ approximating γ_u. Otherwise, to obtain a particle population approximating γ_u, we gather the particle

populations associated with each of u's children and compute the (weighted) product form estimator (Kuntz et al., 2022)

$$\gamma_{\mathcal{C}_u}^N = \frac{1}{N^2} \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} w_{\ell(u)}(x_{\ell(u)}^{n_1}) w_{r(u)}(x_{r(u)}^{n_2}) \delta_{(x_{\ell(u)}^{n_1}, x_{r(u)}^{n_2})} \qquad (2.3)$$

to approximate the product of the marginal distributions $\gamma_{\mathcal{C}_u} := \gamma_{\ell(u)} \times \gamma_{r(u)}$. The $O(N^2)$ cost of evaluating $\gamma_{\mathcal{C}_u}^N$ can be prohibitively large, we discuss lower cost alternatives in Section 3.3.

We reweight the particle approximation $\gamma_{\mathcal{C}_u}^N$ of $\gamma_{\mathcal{C}_u}$ to target γ_u; the resulting *mixture* (importance) weights

$$m_u(x_{\ell(u)}, x_{r(u)}) := \frac{\gamma_u(x_{\ell(u)}, x_{r(u)})}{\gamma_{\ell(u)}(x_{\ell(u)}) \gamma_{r(u)}(x_{r(u)})} \qquad (2.4)$$

capture the mismatch between γ_u and $\gamma_{\ell(u)} \times \gamma_{r(u)}$, and are incorporated prior to resampling similarly to the auxiliary "twisting" function in the auxiliary PF (see, e.g., Chopin and Papaspiliopoulos (2020, Chapter 10)). This leads to weights of the form

$$\tilde{w}_u(x_{\ell(u)}, x_{r(u)}) := w_{\ell(u)}(x_{\ell(u)}) w_{r(u)}(x_{r(u)}) m_u(x_{\ell(u)}, x_{r(u)}). \qquad (2.5)$$

Resampling N times from $\tilde{w}_u \gamma_{\mathcal{C}_u}^N$, using any unbiased resampling scheme (cf.

Gerber et al. (2019)), we obtain an equally weighted particle population $\{\tilde{x}_u^n, w_u^n = 1\}_{n=1}^N$ approximating γ_u. If necessary, we can then apply a π_u-invariant Markov kernel K_u, to rejuvenate the particles. Algorithm 1, which is applied to the root node to carry out the sampling process, summarizes this.

The DaC approach in Algorithm 1 is a special case of that considered in Lindsten et al. (2017); Kuntz et al. (2021), in which the target at each non-leaf node is defined on the product of the spaces on which each of its child targets are defined. DaC-SMC is particularly amenable to distributed implementation (Lindsten et al., 2017, Section 5.3).

Algorithm 1 dac_smc(u) for u in $\mathbb{T}$.

1: **if** u is a leaf **then**
2:    *Initialize:* draw $x_u^n \sim K_u$ and compute $w_u^n = \gamma_u/K_u$ for all $n \leq N$.
3: **else**
4:    *Recurse:* set $(\{x_v^n, w_v^n\}_{n=1}^N) := $ dac_smc(v) for v in $\{\ell(u), r(u)\}$ and obtain $\gamma_{\mathcal{C}_u}^N$ in (2.3).
5:    *Merge:* compute $\tilde{w}_u^{(n_1,n_2)}$ in (2.5) for all $n_1, n_2 \leq N$.
6:    *Resample:* draw $\{\tilde{x}_u^n\}_{n=1}^N$ using weights $\tilde{w}_u^{(n_1,n_2)}$ and set $w_u^n = 1$ for all $n \leq N$.
7:    *(Optionally):* draw $x_u^n \sim K_u(\tilde{x}_u^n, \cdot)$ for all $n \leq N$.
     (Otherwise): set $x_u^n = \tilde{x}_u^n$ for all $n \leq N$.
8: **end if**

---

3. Divide and Conquer within Marginal SMC for Filtering

To apply the DaC-SMC algorithm described above to the filtering problem, we need to identify a suitable collection $(\widetilde{\gamma}_{t,u})_{u \in \mathbb{T}}$ indexed by the nodes of the tree $\mathbb{T}$, for each time t. Graphically, this corresponds to a path graph (corresponding to time) in which each node has associated with it a copy of the tree $\mathbb{T}$ (corresponding to space). In this case Algorithm 1 takes as input at the leaves a particle population approximating the filtering distribution at time $t - 1$ and outputs at the root a particle population approximating the filtering distribution at time t.

At a given time t, to build the collection $(\widetilde{\gamma}_{t,u})_{u \in \mathbb{T}}$, we consider spatial decompositions of x_t into low dimensional (often univariate) elements. Here, we consider a simple decomposition obtained by identifying the d components $(x_t(1), \ldots, x_t(d))$ with the leaves of a tree $\mathbb{T}$. As we move up the tree, the components are merged pairwise until $x_t = x_t(1 : d)$ is recovered at the root node $\mathfrak{R}$. For simplicity, we assume that $d = 2^D$ for some $D \in \mathbb{N}$ so that $\mathbb{T}$ is a perfect binary tree; essentially the same construction applies to general d.

We denote the set of components associated with node u by $\mathcal{V}_u$, its cardinality increases from leaves to root: at the level of the leaves $|\mathcal{V}_u| = 1$, while $|\mathcal{V}_{\mathfrak{R}}| = d$. Figure 1 shows the space decomposition for $d = 8$.
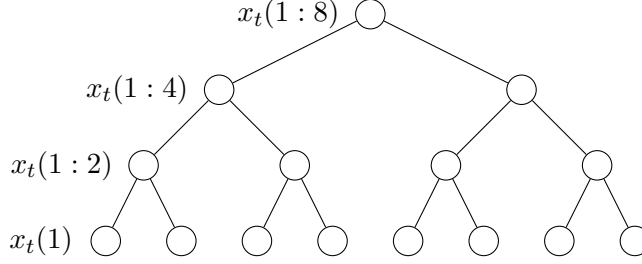
Figure 1: Space decomposition for $d = 8$.

As the filtering problem has an inherent (temporal) sequential structure, the collection $(\widetilde{\gamma}_{t,u})_{u \in \mathbb{T}}$ at time t is most easily specified in terms of the filtering distribution at time $t-1$, $\widetilde{\gamma}_{t-1,\mathfrak{R}}$ — as shown below in (3.6). Similarly to MPFs, we deal with this dependence by approximately marginalizing out the previous timestep using the existing sample approximation. We introduce auxiliary functions $f_{t,u} : \mathbb{R}^d \times \mathbb{R}^{|\mathcal{V}_u|} \to \mathbb{R}$ and $g_{t,u} : \mathbb{R}^{|\mathcal{V}_u|} \times \mathbb{R}^{|\mathcal{V}_u|} \to \mathbb{R}$ for $t \geq 1$ and $u \in \mathbb{T}$, such that $f_{t,\mathfrak{R}} = f_t$, $g_{t,\mathfrak{R}} = g_t$ and for $u \in \mathbb{T} \backslash \mathfrak{R}$, $f_{t,u}$ and $g_{t,u}$ serve as proxies for marginals of the transition density and observation likelihood, respectively. These auxiliary functions are used to define our collection of target densities $(\widetilde{\gamma}_{t,u})_{t \geq 1, u \in \mathbb{T}}$ over $\mathbb{R}^{|\mathcal{V}_u|}$:

$$\widetilde{\gamma}_{t,u}(z_{t,u}) = g_{t,u}(z_{t,u}, (y_t(i))_{i \in \mathcal{V}_u}) \int f_{t,u}(x_{t-1}, z_{t,u})\widetilde{\gamma}_{t-1,\mathfrak{R}}(x_{t-1}) \mathrm{d}x_{t-1} \quad (3.6)$$

where $z_{t,u} = (x_t(i))_{i \in \mathcal{V}_u}$ are the components of x_t associated with node

u and x_{t-1} denotes the previous state of the system. The requirement that $g_{t,\mathfrak{R}} = g_t, f_{t,\mathfrak{R}} = f_t$ ensures that at the root we obtain the distribution in (2.2). The integral w.r.t. $\widetilde{\gamma}_{t-1,\mathfrak{R}}$ in (3.6) cannot be computed analytically, hence, as in MPFs, we use a sample approximation of this integral, as described in the next section.

If the marginals of f_t, g_t are available, one could use them to define $g_{t,u}, f_{t,u}$, but this is not essential: these intermediate distributions can be essentially arbitrary up to the absolute continuity required to justify the importance sampling steps although, of course, the variance of the estimator will be influenced by this choice. The issue of specifying these distributions is closely related to that of choosing the sequence of artificial targets in a standard SMC sampler when one is only interested in the final target distribution (see Del Moral et al. (2006), where they note that optimizing this sequence is a very difficult problem). See Kuntz et al. (2021, Section 4.2) for a theoretical perspective in the divide-and-conquer context, which suggests that the optimal choice of intermediate target distribution is the appropriate marginal of the root target, hence the suggestion to use the marginals of f_t and g_t where these are available. In practice, when doing this exactly is not feasible, this perspective suggests that we should seek to approximate these distributions with distributions with comparable tail

behaviour to keep all importance weights well controlled. If there is a sub-stantial mismatch between the children of a node and itself then choosing a parametric path between the two, adaptively specifying a tempering se-quence along that path and using SMC techniques to approximate each distribution in turn may mitigate some difficulties (See Appendix S2).

While we believe that the choice of the best $f_{t,u}, g_{t,u}$ will be model dependent, we found that, in some cases, some choices are preferable. As an example, take f_t to be a Gaussian distribution with mean μ and covariance Σ. Then, a possible choice for $\gamma_{t,u}$ are Gaussian distributions with mean μ_u equal to the restriction of μ to the components corresponding to node u and covariance Σ_u obtained by subsetting Σ and discarding all components not in u. In this context, we found that setting $\Sigma_u = \Sigma(\mathcal{V}_u)^{-1}$, i.e. selecting the components of Σ corresponding to node u and then inverting this matrix, instead of $\Sigma_u = \Sigma^{-1}(\mathcal{V}_u)$ has lower computational cost and leads to more diffuse distributions and thus better behaved mixture weights (3.10).

3.1   The Algorithm

For each time t, having identified the space decomposition over $\mathbb{T}$ and the collection of distributions $(\widetilde{\gamma}_{t,u})_{u \in \mathbb{T}}$, we can apply Algorithm 1 to the root $\mathfrak{R}$ of $\mathbb{T}$. However, since the integral in (3.6) is not analytically tractable, we

replace $\widetilde{\gamma}_{t-1,\mathfrak{R}}$ with an approximation provided by the particle population at the root of the tree corresponding to $t-1$, $\{z_{t-1,\mathfrak{R}}^n\}_{n=1}^N$, i.e. its particle approximation obtained at the previous time step, as is normally done in MPFs, and define

$$\gamma_{t,u}(z_{t,u}) := g_{t,u}(z_{t,u}, (y_t(i))_{i \in \mathcal{V}_u}) \frac{1}{N} \sum_{n=1}^N f_{t,u}(z_{t-1,\mathfrak{R}}^n, z_{t,u}). \qquad (3.7)$$

Given $\{z_{t-1,\mathfrak{R}}^n\}_{n=1}^N$, at each leaf node of the tree we sample one component of x_t per node from $N^{-1} \sum_{n=1}^N K_{t,u}(z_{t-1,\mathfrak{R}}^n, \cdot)$, the importance weights are then given by

$$w_{t,u}(z_{t,u}, x_{1:t-1,u}) = \frac{g_{t,u}(z_{t,u}, (y_t(i))_{i \in \mathcal{V}_u}) \sum_{n=1}^N f_{t,u}(z_{t-1,\mathfrak{R}}^n, z_{t,u})}{\sum_{n=1}^N K_{t,u}(z_{t-1,\mathfrak{R}}^n, z_{t,u})}. \qquad (3.8)$$

As in the MPF, if we choose $K_{t,u} = f_{t,u}$, (3.8) simplifies dramatically to $w_{t,y}(z_{t,u}, x_{1:t-1,u}) = g_{t,u}(z_{t,u}, (y_t(i))_{i \in \mathcal{V}_u})$, considerably reducing the cost of evaluating the weights at the leaves.

For any non-leaf node u we gather the particle populations $\{z_{t,\ell(u)}^n\}_{n=1}^N$ and $\{z_{t,r(u)}^n\}_{n=1}^N$ on its left and right child, respectively, and obtain an approximation of the product measure $\gamma_{t,\mathcal{C}_u} := \gamma_{t,\ell(u)} \times \gamma_{t,r(u)}$ using the

weighted product form estimator (2.3)

$$\gamma_{t,\mathcal{C}_u}^N = N^{-2} \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} w_{t,\ell(u)}(z_{t,\ell(u)}^{n_1}) w_{t,r(u)}(z_{t,r(u)}^{n_2}) \delta_{(z_{t,\ell(u)}^{n_1}, z_{t,r(u)}^{n_2})}, \qquad (3.9)$$

or one of the lower cost alternatives discussed in Section 3.3.

As in the DaC-SMC setting, we reweight the particle approximation of $\gamma_{t,\mathcal{C}_u}$ to target $\gamma_{t,u}$. In this case, the mixture weights are given by

$$m_{t,u}(z_{t,\mathcal{C}_u}) = \frac{\gamma_{t,u}(z_{t,\mathcal{C}_u})}{\gamma_{t,\mathcal{C}_u}(z_{t,\mathcal{C}_u})} \qquad (3.10)$$

$$= \frac{g_{t,u}(z_{t,\mathcal{C}_u}, (y_t(i))_{i \in \mathcal{V}_u})}{g_{t,\ell(u)}(z_{t,\ell(u)}, (y_t(i))_{i \in \mathcal{V}_{\ell(u)}}) g_{t,r(u)}(z_{t,r(u)}, (y_t(i))_{i \in \mathcal{V}_{r(u)}})} \times$$

$$\frac{N^{-1} \sum_{n=1}^{N} f_{t,u}(z_{t-1,\mathfrak{R}}^n, z_{t,\mathcal{C}_u})}{N^{-1} \sum_{n=1}^{N} f_{t,\ell(u)}(z_{t-1,\mathfrak{R}}^n, z_{t,\ell(u)}) N^{-1} \sum_{n=1}^{N} f_{t,r(u)}(z_{t-1,\mathfrak{R}}^n, z_{t,r(u)})},$$

where we defined $z_{t,\mathcal{C}_u} := (z_{t,\ell(u)}, z_{t,r(u)})$ the vector obtained by merging the components on the left and on the right child of u.

For each pair in (3.9) we obtain the incremental mixture weights $m_{t,u}^{(n_1,n_2)} :=$ $m_{t,u}((z_{t,\ell(u)}^{n_1}, z_{t,r(u)}^{n_2}))$ in (3.10) and the updated weights

$$\tilde{w}_{t,u}^{(n_1,n_2)} = \tilde{w}_{t,u}((z_{t,\ell(u)}^{n_1}, z_{t,r(u)}^{n_2})) := w_{t,\ell(u)}(z_{t,\ell(u)}^{n_1}) w_{t,r(u)}(z_{t,r(u)}^{n_2}) m_{t,u}((z_{t,\ell(u)}^{n_1}, z_{t,r(u)}^{n_2})),$$

for $n_1, n_2 = 1, \ldots, N$. To avoid unbounded growth in the number of parti-

cles, the weights $\tilde{w}_{t,u}$ are then used to resample a population of N particles approximating $\gamma_{t,u}$, $\{\tilde{z}_{t,u}^n, w_{t,u} = 1\}_{n=1}^N$ — although one could allow the number of particles retained to grow as the simulation approaches the root of the tree to accommodate the growing dimension of the space. Algorithm 2 summarizes the procedure described above with the natural modification at $t = 1$, where it is not necessary to "marginalize" over previous states and simple importance sampling can be used at the leaf nodes. Contrary to Algorithm 1, we do not include an additional MCMC step in this statement of the algorithm, but one could easily be added as discussed in Section 3.2. In this case, given a $\pi_{t,u}$-invariant kernel $Q_{t,u}$, with $\pi_{t,u} \propto \gamma_{t,u}$, line 8 should be replaced by: draw $z_{t,u}^n \sim Q_{t,u}(\tilde{z}_{t,u}^n, \cdot)$ for all $n \leq N$.

Algorithm 2 dac_smc(t) for $t \geq 1$. Given $(\{z_{t-1,\mathfrak{R}}^n\}_{n=1}^N) := $ dac_smc($t - 1$).

1: **for** u leaf node **do**
2:    *Initialize:* draw $z_{t,u}^n \sim N^{-1} \sum_{n=1}^N K_{t,u}(z_{t-1,\mathfrak{R}}^n, \cdot)$ and compute $w_{t,u}^n$ as in (3.8) for all $n \leq N$.
3: **end for**
4: **for** u non-leaf node **do**
5:    *Recurse:* set $(\{z_{t,v}^n, w_{t,v}^n\}_{n=1}^N) := $ dac_smc(t, v) for v in $\{\ell(u), r(u)\}$ and obtain $\gamma_{\mathcal{C}_u}^N$ in (3.9).
6:    *Merge:* compute the mixture weights $m_{t,u}^{(n_1,n_2)}$ in (3.10) and $\tilde{w}_{t,u}^{(n_1,n_2)}$ for all $n_1, n_2 \leq N$.
7:    *Resample:* draw $\{\tilde{z}_{t,u}^n\}_{n=1}^N$ using weights $\tilde{w}_{t,u}^{(n_1,n_2)}$ and set $w_u^n = 1$ for all $n \leq N$.
8:    *Update:* set $z_{t,u}^n = \tilde{z}_{t,u}^n$ for all $n \leq N$.
9: **end for**
10: *Output* $(\{z_{t,\mathfrak{R}}^n\}_{n=1}^N)$.

The mixture resampling strategy described above requires evaluating the mixture weights (3.10) for each of the N^2 pairs in (3.9). The $O(N^2)$ cost of this operation is often prohibitive for large N; we describe smaller cost alternatives in Section 3.3 and demonstrate that these approaches have good performance in Section 4.

3.2   Choice of Proposals

Algorithm 2 describes a general strategy to perform filtering using DaC-SMC; as in the case of standard SMC the performances of the algorithm are heavily influenced by the choice of the proposals at the leaf nodes. We discuss here a simple strategy to select $K_{t,u}$.

We assume that we can sample from $f_{t,u}$ in (3.6) and set $K_{t,u}(z^n_{t-1,\mathfrak{R}}, \cdot) = f_{t,u}(z^n_{t-1,\mathfrak{R}}, \cdot)$ so that the importance weights (3.8) reduce to $w_{t,u} = g_{t,u}$. This choice corresponds to the proposal used in the bootstrap PF of Gordon et al. (1993), (locally) optimal proposals also exist (see e.g., Chopin and Papaspiliopoulos (2020, Chapter 10)) and are expected to lead to better performances but incur a higher computational cost.

While picking $K_{t,u}(z^n_{t-1,\mathfrak{R}}, \cdot) = f_{t,u}(z^n_{t-1,\mathfrak{R}}, \cdot)$ causes standard marginal PFs to reduce to the bootstrap PF (as described in Klaas et al. (2005, Section 3.3)), this is not true for our marginal DaC-SMC, since the integral

w.r.t. $z_{t-1,\mathfrak{R}}$ still appears in the mixture weights (3.10).

If needed, to avoid particle impoverishment, one might consider apply-ing a Markov kernel $Q_{t,u}$ which leaves $\pi_{t,u} \propto \gamma_{t,u}$ invariant after the re-sampling step in line 7 of Algorithm 2. These $\pi_{t,u}$-invariant kernels can be selected exploiting the vast literature on sequential MCMC methods (e.g., Gilks and Berzuini (2001); Septier et al. (2009); Carmi et al. (2012); Septier and Peters (2016); Pal and Coates (2018); Han and Nakamura (2021)) to employ proposals whose cost is not $O(N)$ as it would be for some naïve choices.

3.3   Adaptive Lightweight Mixture Resampling

The mixture resampling in line 6–7 of Algorithm 2 becomes computation-ally impractical for large N, since it requires evaluating the mixture weights for N^2 particles (Lindsten et al., 2017; Kuntz et al., 2021; Corneflos et al., 2022). Several strategies have been proposed to alleviate this cost by only constructing a subset of the N^2 combinations in (3.9), e.g. the multiple matching strategy of Lin et al. (2005) which gives rise to the lightweight mixture resampling of Lindsten et al. (2017) in this context, strategies bor-rowed from the literature on incomplete U-statistics (Kuntz et al., 2021) and lazy resampling schemes (Corneflos et al., 2022).

We consider the lightweight version of mixture resampling proposed in Lindsten et al. (2017) which only considers a subset θN with $\theta \ll N$ of the N^2 possible pairs. However, instead of setting θ to some pre-specified value (e.g. $\theta = \lceil \sqrt{N} \rceil$), we propose a simple strategy to select θ adaptively based on the effective sample size (ESS; Kong et al. (1994)),

$$\text{ESS} := \left(\sum_n \tilde{w}_{t,u}^n \right)^2 / \sum_n (\tilde{w}_{t,u}^n)^2, \tag{3.11}$$

where the sum is over all pairs $n = (n_1, n_2)$ obtained from (3.9), which is similar in spirit to the adaptive tempering strategies commonly encountered in the SMC literature (see, e.g., Jasra et al. (2010); Johansen (2015)) and aiming to do just enough computation to obtain a good N-sample approximation.

The merge step in lines 6–7 of Algorithm 2 is replaced by Algorithm 3: after building all the N pairs obtained by concatenating the particles associated with each of the two children, further permutations are added until the ESS achieves a pre-specified value $\text{ESS}^\star$ (e.g. $\text{ESS}^\star = N$). To avoid θ getting too large, we stop adding permutations when $\theta = \lceil \sqrt{N} \rceil$ thereby allowing us to bound the worst-case computational cost by $O(N^{3/2})$. We empirically compare several mixture resampling approaches in Appendix S1.

Algorithm 3 Adaptive lightweight mixture resampling.

1: *Correct:* compute the mixture weights $m_{t,u}(x^n_{\ell(u)}, x^n_{r(u)})$ as in (3.10) and $\tilde{w}_{t,u}(x^n_{\ell(u)}, x^n_{r(u)})$ for all $n \leq N$ and the ESS (3.11).

2: *Set:* $\theta \leftarrow 1$ and $\tilde{x}^n_u = (x^n_{\ell(u)}, x^n_{r(u)})$ for $n \leq N$.

3: **while** $\mathrm{ESS}((\tilde{x}^n)^{\theta N}_{n=1}) < \mathrm{ESS}^\star$ and $\theta < \lceil \sqrt{N} \rceil$ **do**

4:     *Set:* $\theta \leftarrow \theta + 1$.

5:     *Permute:* draw one permutation of N, $\pi(N)$, set $\tilde{x}^{N(\theta-1)+n}_u = (x^n_{\ell(u)}, x^{\pi(n)}_{r(u)})$, compute the mixture weights $m_{t,u}(\tilde{x}^{N(\theta-1)+n}_u)$ in (3.10) and the updated weights $\tilde{w}_{t,u}(\tilde{x}^{N(\theta-1)+n}_u)$ for $n \leq N$ and update the ESS.

6: **end while**

7: *Resample:* draw $\{z^n_{t,u}\}^N_{n=1}$ from $\tilde{x}^{1:N\theta}_u$ with weights $\tilde{w}_{t,u}(\tilde{x}^{1:N\theta}_u)$ and set $w^n_{t,u} = 1$ for all $n \leq N$.

Algorithm 3 can be implemented in a space-efficient manner by storing the permutations $\pi(N)$ corresponding to each value of θ rather than building the θN pairs $\tilde{x}^{1:N\theta}_u$.

We report in Figure 2 the distribution of the number of permutation θ selected by Algorithm 3 for the linear Gaussian model in Section 4. We observe that the highest values of θ are chosen at the level above the leaves (panel 1), indeed, at the leaf level the observation y_t is incorporated, causing a larger adjustment to the distribution at the first mixture resampling step than that needed as we move up the tree. The spike at $\theta = \lceil \sqrt{N} \rceil$ at level 1 shows that sometimes the target ESS is not reached, which suggests that the product of the proposal distributions over the children might be a poor proposal for $\gamma_{t,u}$ — this is in part likely to be due to the use of a
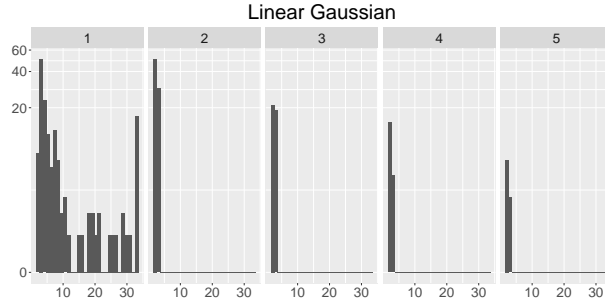
Figure 2: Distribution of the number of permutations θ selected by Algorithm 3 for the simple linear Gaussian model in Section 4.1 with $d = 32$, $N = 10^3$ and 10 time steps; the 5 panels correspond to the 5 levels of the tree from the level above the leaves (level 1) to the root (level 5).

"bootstrap proposal" and could be mitigated in the same way as standard particle filters by seeking to design (marginal) proposal distributions which incorporate the influence of observations. In addition, this phenomenon can be mitigated with the use of (adaptive) tempering, as shown in, e.g., Jasra et al. (2010); Johansen (2015); Wang et al. (2020); Zhou et al. (2016) for standard SMC and Lindsten et al. (2017, Section 4.2) for DaC-SMC. However, naïve implementation of this approach will bear a substantial computational cost in the marginal context; designing MCMC kernels that are efficient in this context has been explored in the sequential MCMC context — see, e.g., Septier and Peters (2016, Section III-B) — and in marginal STPFs (Xu and Jasra, 2019). We give details of a tempering strategy for Algorithm 3 in Appendix S2.

3.4 Computational Cost

At a given node, u, the runtime cost of Algorithm 2 with lightweight mixture resampling given the particle approximations for its children, is $O(h(N,d)\theta N)$ where $h(N,d)$ denotes the cost of obtaining the mixture weights (3.10).

In general, the dependence of $h(N,d)$ on the number of particles N is $O(N)$ as for MPFs. Similarly to MPFs, one could try to reduce the cost of computing (3.10) making use of techniques from N-body learning (e.g. Gray and Moore (2000); Lang et al. (2005)) as shown in Klaas et al. (2005). Alternatively, one could consider efficient implementations using GPUs (Charlier et al., 2021) as shown in (Clarté et al., 2019, Section 4) for sums of the form of those in (3.10).

For some models, it might be possible to pick the auxiliary functions $f_{t,u}$ so that the dependence on the past (i.e. $z_{t-1,\mathfrak{R}}$ in (3.7)) vanishes and obtain an $O(1)$ cost w.r.t. N. However, we expect that this type of decompositions will require larger corrections at the root, where $f_{t,\mathfrak{R}} = f_t$, which might offset the cost savings.

In the adaptive case, worst case costs are given by $O(h(N,d)\theta N)$ with θ replaced by the upper bound imposed upon the number of permutations considered. For instance, for the examples in Section 4 we have an $h(N,d) =$

$O(N)$ cost to obtain the mixture weights and we set the upper bound for θ to be $N^{1/2}$ leading to a cost of order $O(N^{5/2})$ w.r.t. the number of particles.

Denoting $C_u(d, \theta_u, N)$ the cost of running Algorithm 2 at node u, we can then bound the total cost of serial implementations Algorithm 2 applied at the root node $\mathfrak{R}$ by $O(dt \sup_u C_u)$, where the supremum is taken over all nodes in $\mathbb{T}$; a lower running cost of $O(t \sup_u C_u \log_2 d)$, can be achieved parallelizing the computations over each level of the tree (Lindsten et al., 2017, Section 5.3).

In the adaptive case, this upper bound is far from being tight, since, as shown in the histogram in Figure 2, θ tends to be high when the observation is incorporated (level 1) but stabilizes as we move up the tree. Additionally, for large N one expects the number of permutations required to obtain a good N-particle approximation to converge to some fixed integer and hence the cost for sufficiently large N will with high probability be of smaller order than these bounds. Furthermore the constants multiplying the $N^{5/2}$ contribution arising from the level above the leaves are sufficiently small that this is not the dominant cost in our experiments — and is likely to be typical in high-dimensional settings in which it is rarely feasible to employ very large numbers of particles and the objective is to obtain a good approximation at acceptable time *and space* costs.

4. Experiments

We compare the results obtained with DaC with those of NSMC and STPF; we do not include simpler strategies because both standard PF and BPF have been shown to have worse performances than NSMC and STPF for the model considered here (Næsseth et al., 2015, 2019; Beskos et al., 2017), and the marginal version of STPF because of the higher cost for large d.

The functions $g_{t,u}, f_{t,u}$ in (3.6) are obtained from g_t, f_t, respectively, by discarding all the terms in those functions involving components $i \notin \mathcal{V}_u$, further details are given in Appendix S3. For Algorithm 2 we use the proposals discussed in Section 3.2 and the lightweight mixture resampling strategies described in Section 3.3. All resampling steps are performed using stratified resampling (Kitagawa, 1996).

First, we consider a simple linear Gaussian SSM, and compare the results obtained by the three algorithms with the exact filtering distribution given by the Kalman filter. Then, we consider a spatial model with simple latent dynamics but non-trivial spatial correlations structure between observations, moving away from the assumption of i.i.d. observations which is convenient from a computational perspective, but rarely satisfied in practice (Chib et al., 2009).

All the experiments have been executed in serial using a single core of

a Intel(R) Xeon(R) CPU E5-2440 0 @ 2.40GHz using R 4.1.0.

4.1   Simple Linear Gaussian Model

We start by considering a simple linear Gaussian SSM taken from Næsseth et al. (2015), for which the filtering distributions can be computed exactly with the Kalman filter. The model is given by $f_t(x_{t-1}, x_t) = \mathcal{N}(x_t; Ax_{t-1}, \Sigma)$, and $g_t(x_t, y_t) = \mathcal{N}(y_t; x_t, \sigma_y^2 \mathrm{Id}_d)$, with $A \in \mathbb{R}^{d \times d}$, $\sigma_y^2 > 0$, $\Sigma \in \mathbb{R}^{d \times d}$ a tridiagonal covariance matrix and Id_d the d-dimensional identity matrix (see Appendix S3.1 for full details and computation of the mixture weights (3.10)).

We compare DaC with both non-adaptive and adaptive lightweight mixture resampling with 2-level NSMC with fully adapted outer level and STPF on data simulated from the model for $d = 2^5, 2^8, 2^{11}$ for $t = 100$ time steps. We use different number of particles $N = 100, 500, 1000$ for Algorithm 2 and the outer level of NSMC and STPF, while the number of particles for the inner level of NSMC and the number of particles for each island of STPF is fixed to $M = 100$ as suggested in Næsseth et al. (2015); Beskos et al. (2017).

To evaluate the results, we consider two global measures of accuracy for each of the d marginals, the Wasserstein-1 distance (see, e.g., Vallender

(1974)), and the Kolmogorov-Smirnov distance

$$W_{1,i} := \int \left| F_{t,i}(x) - \widehat{F}_{t,i}(x) \right| \mathrm{d}x, \qquad \mathrm{KS}_i := \max_{x} \left| F_{t,i}(x) - \widehat{F}_{t,i}(x) \right|,$$

where $F_{t,i}$ denotes the 1-dimensional cumulative distribution function of marginal i at time t and $\widehat{F}_{t,i}$ its particle approximation. Further comparisons, which demonstrate that the mean squared error (MSE) of the filtering mean behaves similarly, are collected in Appendix S4.1.

For lower-dimensional problems (e.g. $d = 32$) STPF achieves the best results both in terms of Wasserstein-1 distance and in terms of KS distance (Figure 3) and the relative MSE of the reconstructions is considerably smaller (almost one order of magnitude smaller; see Appendix S4.1). The results provided by STPF deteriorate quickly as d grows, for $d = 256$ the estimates of W_1 and KS distance are significantly worse than those provided by NSMC or DaC without adaptation.

The cost of STPF grows quadratically with d, and becomes unmanageable for large d, it is therefore not included in Figure 3 bottom panels. STPF has the higher cost also for lower dimension (Figure 3 top panel), but in this case STPF provides the best results. STPF has higher variability than the other methods, and even when the average results are better than
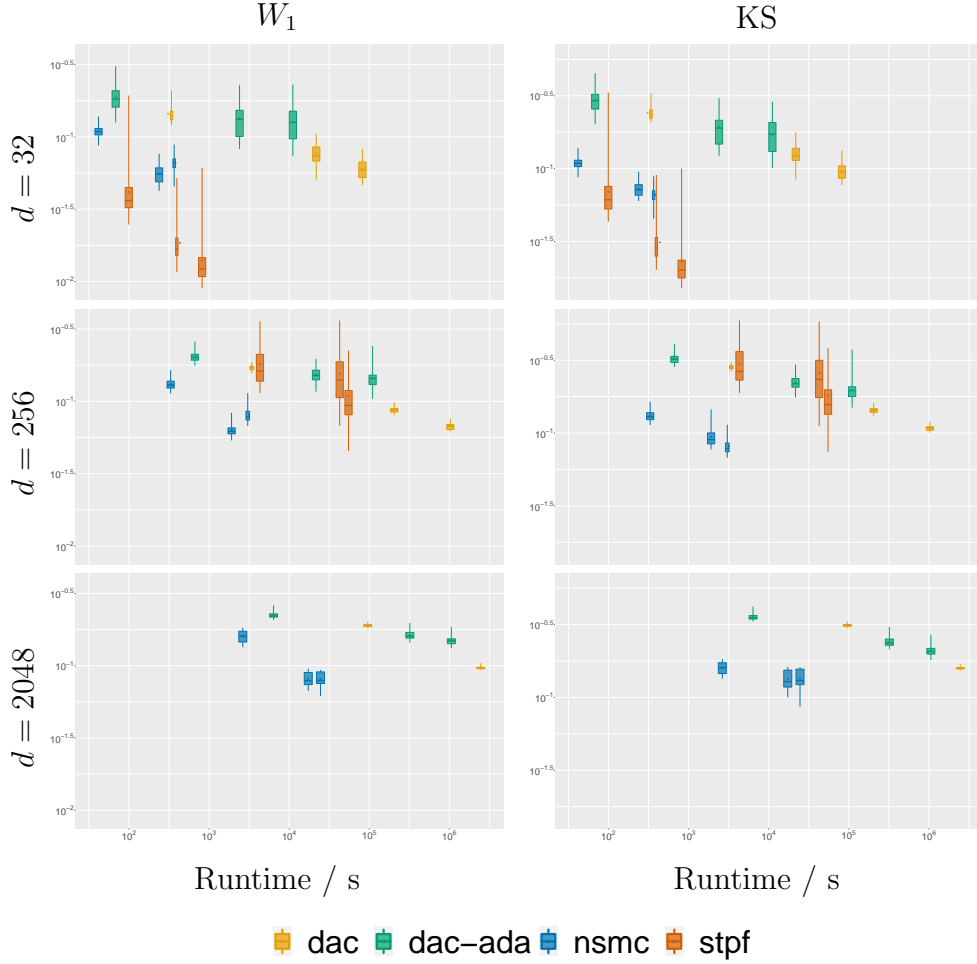
Figure 3: Comparison of DaC, NSMC and STPF for $d = 2^5, 2^8, 2^{11}$. Distribution of the average (over dimension) W_1 and KS distance at the last time step $t = 100$ for 50 runs; the boxes, form left to right, correspond to increasing number of particles ($N = 100, 500, 1000$). Due to their excessive cost, we do not include the results for STPF with $d = 2048$ and those of the non-adaptive version of DaC $d = 2048, N = 1000$.

DaC and NSMC (e.g. $d = 32$), W_1 and KS can take considerably high values.

The results in terms of KS are generally more variable, this is likely due to the fact that KS is a measure of the worst case mismatch between $F_{t,i}(x)$ and $\widehat{F}_{t,i}(x)$, while for W_1 the mismatch is averaged over locations. For large d, DaC has the smallest variability among the three algorithms.

DaC with fixed-cost lightweight mixture resampling generally gives better results than the adaptive lightweight mixture resampling, however, the cost of the latter is considerably smaller, making the adaptive version still manageable for large N whereas the fixed-cost lightweight mixture resampling becomes too costly for large N and large d. As discussed in Section 3.4, the computational cost of both versions of DaC could be reduced using GPUs. In particular, both W_1 and KS decay more quickly with N for DaC without adaptation than for DaC with adaptive lightweight mixture resampling. The decay with N is less evident for NSMC.

4.2   Spatial Model

We consider a model on a 2D-lattice in which the latent dynamics are simple but the observation structure is challenging. The components of X_t are indexed by the vertices $v \in V$ of a lattice, where $V = \{1, \ldots, d\}^2$, and

follow a simple linear evolution $X_t(v) = X_{t-1}(v) + U_t(v)$, where $U_t(v) \overset{\text{i.i.d}}{\sim}$

$\mathcal{N}(0, \sigma_x^2)$. The observations model is $Y_t = X_t + V_t$, where we take V_t to

be jointly t-distributed with $\nu = 10$ degrees of freedom, mean zero and

precision structure encapsulating a spatial component. Let D denote the

graph distance, then the entry in row v and column j of the precision matrix

Σ^{-1} is given by $(\Sigma^{-1})_{vj} = \tau^{D(j,v)}$ if $D(j, v) \le r_y$ and 0 otherwise. We obtain

data from the model above with $\sigma_x^2 = 1$, $\tau = -0.25$, $r_y = 1$ and $t = 10$.

The observation density does not factorize, and therefore NSMC and STPF

cannot be applied (at least without approximating g with e.g. a Gaussian

or discarding the covariance information). To validate the correctness of

the algorithm, we compared the results obtained by DaC with those of the

standard bootstrap PF in Appendix S4 on a small lattice and found the

agreement to be excellent.

To decompose the 2D lattice into a binary tree we use the decomposition

described in Lindsten et al. (2017, Section 5.1), which recursively connects

the vertices first horizontally and then vertically. To evaluate the perfor-

mances of the algorithm we consider the filtering means obtained with 50

repetitions of DaC-SMC on a 8×8 and a 16×16 grid for $N = 100, 500, 1000$

and 5000. To show how the standard bootstrap particle filter struggles with

higher dimensional problems we run a bootstrap PF with $N = 10^5$ particles

for the 8×8 grid. Figure 4 reports the filtering means for a corner node and an interior node of the lattice. Both DaC approaches are in agreement, however the adaptive version of DaC seems to provide slightly less variable results. The behaviour for different nodes is similar. As observed for the linear Gaussian model, DaC with adaptive lightweight mixture resampling has lower cost than the non-adaptive counterpart and remains feasible for large N (e.g. $N = 5000$).

Unsurprisingly, the bootstrap PF struggles to recover the filtering means and provides high variance estimates for node $(1,1)$ while collapses completely for node $(8,6)$ failing to recover the filtering mean.

The size of the boxplots in Figure 4 gives an indication of the variance of the estimator provided by DaC, for small N the decay in variance seems to be more pronounced (at least for the adaptive version of DaC) than for large N, this is consistent with the decay of the standard deviation which would be expected from the variance expansions in Kuntz et al. (2021) where for small N the higher order contributions to the variance are not yet negligible (we anticipate that a central limit theorem could be obtained by combining the results of Kuntz et al. (2021) with those for marginal PFs).
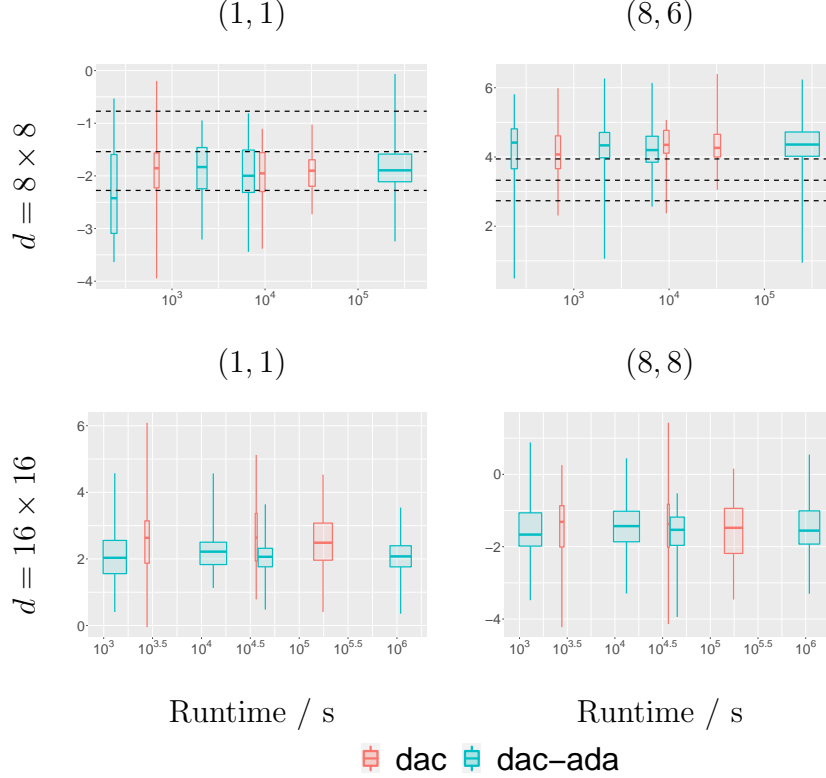
Figure 4: Filtering mean estimates for a corner node and a node in the middle of the grid for a 8×8 and a 16×16 lattice at time $t = 10$. The boxplots from left to right report the distributions over 50 repetitions for $N = 100, 500, 1000$ and 5000. The results for the non-adaptive version of DaC are not included for $N = 5000$ due to the excessive cost. The reference lines for the 8×8 grid show the average value of the filtering mean estimate and the interquartile range obtained with 50 repetitions of a bootstrap PF with $N = 10^5$ particles.

5. Discussion

We introduced a novel sequential Monte Carlo algorithm, combining ideas from marginal PFs and divide-and-conquer SMC to extend the latter to the filtering context. This algorithm is based on a novel space decomposition for high dimensional SSM which allows to recursively merge low dimensional marginals of the filtering distribution to obtain the full filtering distribution, taking into account the mismatch between product of marginals and joint distributions using importance sampling. In principle, the DaC-SMC approach is amenable to distributed implementation, although the marginalization technique employed herein would necessitate significant communication from the node which computes the overall filtering distribution at time $t - 1$ to all nodes involved in computing at time t and we have not explored that direction here.

In contrast with Nested SMC and space-time PFs, the DaC-SMC approach to filtering can be applied when the marginals of the joint density (2.1) are not available analytically. The computational cost of this new approach grows polynomially with the number of particles N, however, this cost can be reduced exploiting GPU routines to reduce the cost of computing the weights as discussed in Section 3.4.

The experiments in Section 4.1 show that DaC-SMC achieves com-

parable performances of NSMC and STPF with a runtime that remains competitive even for large d (but small N), contrary to STPFs. In addition, DaC-SMC can be applied to filtering problems which do not allow for factorization as shown in Section 4.2 and Appendix S4. The variance decay in Figure 4 and Figure 2 in Appendix S4.1 suggest that this extended DaC-SMC achieves the same convergences rates as DaC-SMC (Kuntz et al., 2021) for sufficiently large N and we anticipate that techniques used to analyze the marginal particle filter could be combined with those in order to provide formal convergence results for the method developed herein. The adaptive lightweight mixture resampling discussed in Section 3.3 is a promising route to further reduce the computational cost of DaC-SMC for filtering, however, as the experiments in Section 4 and Appendix S1 suggest, selecting the value at which the target ESS should be set to obtain the best trade-off between computational cost and accuracy is likely to be problem dependent and raises interesting theoretical questions that we leave for future work.

For challenging problems it is likely that tempering and MCMC kernels would be required to give good performance. As discussed in Johansen (2015) and Guarniero et al. (2017) the smoothing and filtering distributions (i.e. $p(x_{1:t}|y_{1:t})$ and $p(x_t|y_{1:t})$, respectively) have significantly different support in the presence of informative observations, especially in high di-

mensional settings and so we expect that including the influence of future observations in the proposals and targets in Section 3.2 (as in lookahead methods, e.g., Lin et al. (2013); Guarniero et al. (2017); Ruzayqat et al. (2022)) would lead to considerable improvements in the accuracy of the estimates and might ultimately be essential in the development of good general purpose filters for high dimensional problems.

This work focuses on obtaining approximations of the filtering distribution for high dimensional SSM. In recent years there has been a lot of interest in obtaining approximations of the smoothing distribution which is a necessary component of parameter estimation algorithms (e.g., Finke and Singh (2017); Guarniero et al. (2017)); we anticipate that the DaC-SMC approach to filtering could be extended to tackle smoothing and parameter estimation dealing with the marginalization in (3.6). In principle, algorithms which directly approximate only marginals of smoothing distributions can be adapted to these settings (see, e.g. Gerber and Chopin (2017)). We leave this for future work.

Supplementary Materials

Supplementary materials contains details of the tempering approach, additional details and results on the experiments. An `R` package reproducing

the experiments is available at `https://github.com/FrancescaCrucinio/`
`Dac4filtering`.

Acknowledgements

References

Akyildiz, D., D. Crisan, and J. Miguez (2022). Space-sequential particle filters for high-dimensional dynamical systems described by stochastic differential equations. *arXiv preprint arXiv:2204.07680*.

Bengtsson, T., P. Bickel, and B. Li (2008). 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*, pp. 316–334. Institute of Mathematical Statistics.

Beskos, A., D. Crisan, A. Jasra, K. Kamatani, and Y. Zhou (2017). A stable particle filter for a class of high-dimensional state-space models. *Advances in Applied Probability 49*(1),

24–48.

Carmi, A., F. Septier, and S. J. Godsill (2012). The Gaussian mixture MCMC particle algorithm for dynamic cluster tracking. *Automatica 48*(10), 2454–2467.

Charlier, B., J. Feydy, J. A. Glaunes, F.-D. Collin, and G. Durif (2021). Kernel operations on the GPU, with autodiff, without memory overflows. *Journal of Machine Learning Research 22*(74), 1–6.

Chib, S., Y. Omori, and M. Asai (2009). Multivariate stochastic volatility. In *Handbook of Financial Time Series*, pp. 365–400. Springer.

Chopin, N. and O. Papaspiliopoulos (2020). *An Introduction to Sequential Monte Carlo*. Cham: Springer.

Clarté, G., A. Diez, and J. Feydy (2022). Collective proposal distributions for nonlinear MCMC samplers: Mean-field theory and fast implementation. *Electronic Journal of Statistics 22*(2), 6395–6460.

Corneflos, A., N. Chopin, and S. Särkkä (2022). De-Sequentialized Monte Carlo: a parallel-in-time particle smoother. *Journal of Machine Learning Research 23*(283), 1–39.

Del Moral, P., A. Doucet, and A. Jasra (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 68*(3), 411–436.

Doucet, A. and A. M. Johansen (2011). A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky (Eds.), *The Oxford Handbook of Nonlinear*

Filtering, pp. 656–704. Oxford University Press.

Evensen, G. (2009). *Data Assimilation* (2nd ed.). Berlin, Heidelberg: Springer.

Finke, A. and S. S. Singh (2017). Approximate smoothing and parameter estimation in high-dimensional state-space models. *IEEE Transactions on Signal Processing 65*(22), 5982–5994.

Gerber, M. and N. Chopin (2017). Convergence of sequential quasi-Monte Carlo smoothing algorithms. *Bernoulli 23*(4B), 2951–2987.

Gerber, M., N. Chopin, and N. Whiteley (2019). Negative association, ordering and convergence of resampling methods. *Annals of Statistics 47*(4), 2236–2260.

Gilks, W. R. and C. Berzuini (2001). Following a moving target—Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63*(1), 127–146.

Gordon, N. J., D. J. Salmond, and A. F. Smith (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F-Radar and Signal Processing*, Volume 140, pp. 107–113. IEE.

Gray, A. and A. Moore (2000). 'N-body' problems in statistical learning. In *Advances in Neural Information Irocessing Systems*, pp. 521–527.

Guarniero, P., A. M. Johansen, and A. Lee (2017). The iterated auxiliary particle filter. *Journal of the American Statistical Association 112*(520), 1636–1647.

Han, Y. and K. Nakamura (2021). The application of Zig-Zag sampler in sequential Markov chain Monte Carlo. *arXiv preprint arXiv:2111.10210*.

Jasra, A., D. A. Stephens, A. Doucet, and T. Tsagaris (2010). Inference for Lévy-driven stochastic volatility models via adaptive sequential Monte Carlo. *Scandinavian Journal of Statistics 38*(1), 1–22.

Johansen, A. M. (2015). On blocks, tempering and particle MCMC for systems identification. *IFAC-PapersOnLine 48*(28), 969–974.

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

Klaas, M., M. Briers, N. De Freitas, A. Doucet, S. Maskell, and D. Lang (2006). Fast particle smoothing: If I had a million particles. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 481–488.

Klaas, M., N. De Freitas, and A. Doucet (2005). Toward practical N^2 Monte Carlo: The marginal particle filter. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pp. 308–315.

Kong, A., J. S. Liu, and W. H. Wong (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association 89*(425), 278–288.

Kuntz, J., F. R. Crucinio, and A. M. Johansen (2021). The divide-and-conquer sequential Monte Carlo algorithm: theoretical properties and limit theorems. *arXiv preprint arXiv:2110.15782*.

Kuntz, J., F. R. Crucinio, and A. M. Johansen (2022). Product-form estimators: exploiting independence to scale up Monte Carlo. *Statistics and Computing 32*(1), 1–22.

Lang, D., M. Klaas, and N. de Freitas (2005). Empirical testing of fast kernel density estimation algorithms. *Technical Report TR2005-03, Department of Computer Science, University of British Columbia*.

Lei, J., P. Bickel, and C. Snyder (2010). Comparison of ensemble Kalman filters under non-Gaussianity. *Monthly Weather Review 138*(4), 1293–1306.

Lin, M., R. Chen, J. S. Liu, et al. (2013). Lookahead strategies for sequential Monte Carlo. *Statistical Science 28*(1), 69–94.

Lin, M. T., J. L. Zhang, Q. Cheng, and R. Chen (2005). Independent particle filters. *Journal of the American Statistical Association 100*(472), 1412–1421.

Lindsten, F., A. M. Johansen, C. A. Næsseth, B. Kirkpatrick, T. B. Schön, J. A. D. Aston, and A. Bouchard-Côté (2017). Divide-and-Conquer with sequential Monte Carlo. *Journal of Computational and Graphical Statistics 26*(2), 445–458.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*. New York: Springer.

Næsseth, C. A., F. Lindsten, and T. B. Schön (2015). Nested sequential Monte Carlo methods. In *Proceedings of the 32nd International Conference on Machine Learning*, Volume 37, pp. 1292–1301. Proceedings of Machine Learning Research.

Næsseth, C. A., F. Lindsten, and T. B. Schön (2019). High-dimensional filtering using nested

sequential Monte Carlo. *IEEE Transactions on Signal Processing 67*(16), 4177–4188.

Pal, S. and M. Coates (2018). Sequential MCMC with the discrete bouncy particle sampler. In *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 663–667. IEEE.

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

Rebeschini, P. and R. Van Handel (2015). Can local particle filters beat the curse of dimensionality? *Annals of Applied Probability 25*(5), 2809–2866.

Ruzayqat, H., A. Er-Raiy, A. Beskos, D. Crisan, A. Jasra, and N. Kantas (2022). A lagged particle filter for stable filtering of certain high-dimensional state-space models. *SIAM/ASA Journal on Uncertainty Quantification 10*(3).

Septier, F., S. K. Pang, A. Carmi, and S. Godsill (2009). On MCMC-based particle methods for Bayesian filtering: Application to multitarget tracking. In *3rd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 360–363. IEEE.

Septier, F. and G. W. Peters (2016). Langevin and Hamiltonian based sequential MCMC for efficient Bayesian filtering in high-dimensional spaces. *IEEE Journal of Selected Topics in Signal Processing 10*(2), 312–327.

Vallender, S. S. (1974). Calculation of the Wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications 18*(4), 784–786.

Wang, L., S. Wang, and A. Bouchard-Côté (2020). An annealed sequential Monte Carlo method for Bayesian phylogenetics. *Systematic Biology 69*(1), 155–183.

Xu, Y. and A. Jasra (2019). Particle filters for inference of high-dimensional multivariate stochastic volatility models with cross-leverage effects. *Foundations of Data Science 1*(1), 61–85.

Zhou, Y., A. M. Johansen, and J. A. Aston (2016). Toward automatic model comparison: an adaptive sequential Monte Carlo approach. *Journal of Computational and Graphical Statistics 25*(3), 701–726.

Department of Statistics, University of Warwick

E-mail: francesca.crucinio@gmail.com

Department of Statistics, University of Warwick

E-mail: a.m.johansen@warwick.ac.uk