Fast and Numerically Stable Particle-Based Online Additive Smoothing: The AdaSmooth Algorithm

Alessandro Mastrototaro\textsuperscript{a} \@, Jimmy Olsson\textsuperscript{a} \@, and Johan Alenlöv\textsuperscript{b} \@.

\textsuperscript{a}Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden; \textsuperscript{b}Department of Computer and Information Science, Linköping University, Linköping, Sweden

\section*{1. Introduction}

\subsection*{1.1. Background}

We consider a general path-space model comprising general measurable spaces \((X_n, X_n^0)_{n \in \mathbb{N}}\) and unnormalized transition densities \((\ell_n)_{n \in \mathbb{N}}\), where for every \(n \in \mathbb{N}\), \(\ell_n\) is a nonnegative measurable function on \(X_n \times X_{n+1}\) such that \(\text{sup}_{x_n \in X_n} \int \ell_n(x_n, x_{n+1}) \, dx_{n+1} < \infty\), with \(dx_{n+1}\) being some reference measure on \(X_{n+1}\). In addition, we let \(\chi\) be some possibly unnormalized density function on \(X_0\). The transition densities \((\ell_n)_{n \in \mathbb{N}}\), which are assumed to be tractable, induce multivariate probability densities

\[\phi_{0,n}(x_0, x_n) \propto \chi(x_0) \prod_{m=0}^{n-1} \ell_m(x_n, x_{m+1}), \quad n \in \mathbb{N},\]

where \(x_0 = (x_0, \ldots, x_n)\) (being our generic notation for vectors) denotes an element in the Cartesian product \(X_0 \times \cdots \times X_n\). The aim of the present article is the development of \textit{sequential Monte Carlo (SMC) methods} approximating online (in a sense that will be specified below) the expectations

\[\phi_{0,n} h_n = \int h_n(x_0, x_n) \phi_{0,n}(x_0, x_n) \, dx_0, \quad n \in \mathbb{N},\]

for given \textit{additive state functionals} \((h_n)_{n \in \mathbb{N}}\) such that (2) is well defined. Starting with some measurable function \(h_0\) on \(X_0\), these functionals are defined recursively as

\[h_{n+1}(x_0, x_{n+1}) = h_n(x_0, x_n) + \tilde{h}_n(x_{n+1}),\]

where \(\tilde{h}_n\) is some measurable function on \(X_n \times X_{n+1}\).

Our model framework, which was also considered by Gloaguen, Le Corff, and Olsson (2021), has great generality. Among the most common applications we find the so-called \textit{hidden Markov models} (HMMs), which constitute a modeling tool of significant importance in a variety of scientific and engineering disciplines (see Cappé, Moulines, and Rydén 2005). A fully dominated HMM consists of a bivariate Markov chain \((X_n, Y_n)_{n \in \mathbb{N}}\) evolving on some product measurable space \((X \times Y, \mathcal{X} \otimes \mathcal{Y})\) according to Markov transition densities in the form

\[q(x_n, x_{n+1}) q(x_{n+1}, y_{n+1}) \quad (x_n, x_{n+1}, y_{n+1} \in X \times X \times Y),\]

where \(q\) and \(g\) are themselves Markov transition densities (which may depend on \(n\) in the general case) on \(X \times X\) and \(X \times Y\), respectively. The chain is initialized according to \(\chi(x_0) g(x_0, y_0)\) for some density \(\chi\) on \(X\). In this model, only the marginal process \((Y_n)_{n \in \mathbb{N}}\) is observed, whereas \((X_n)_{n \in \mathbb{N}}\) is latent. The construction implies (see Cappé, Moulines, and Rydén 2005, sec. 2.2, for details) that (i) the marginal \textit{state process} \((X_n)_{n \in \mathbb{N}}\) is itself a Markov chain with transition densities \(q\) and that (ii) conditionally on the state process, the observations \((Y_n)_{n \in \mathbb{N}}\) are independent with marginal densities given by \(g(X_n, y_n)\), \(n \in \mathbb{N}\). Now, let \((y_n)_{n \in \mathbb{N}}\) be a fixed sequence of observations and define, for every \(n \in \mathbb{N}\), the transition density \(\ell_n(x_n, x_{n+1}) = q(x_n, x_{n+1}) g(x_{n+1}, y_{n+1}),\)

\[(x_n, x_{n+1}) \in X \times X;\]

then, with these definitions, each density

(1)

corresponds to the \textit{joint-smoothing distribution at time \(n\),}
that is, the conditional density of $X_{0:n}$ given $Y_{0:n} = y_{0:n}$. In the HMM literature, the computation of (1) is referred as joint smoothing, and in the absence of alternative terminology we adopt this term to the more general context considered in the present article. More in general, HMMs can be interpreted as Feynman–Kac models; these models are widely used in, for example, statistics, physics, biology, and signal processing, and we refer to Del Moral (2004) for a comprehensive treatment. 

The transition densities can be decomposed as

$$
\ell_n(x_n, x_{n+1}) = q_n(x_n, x_{n+1})g_{n+1}(x_{n+1}),
$$

where $g_{n+1}$ is some tractable potential function and $q_n$ some Markov transition density.

Inspired by the HMM terminology, we will refer to the above-described problem of computing online the expectations $(\phi_{0:n} h_n)_{n \in \mathbb{N}}$ as online additive smoothing. Additive smoothing is of crucial importance in many applications in statistics and engineering. It is a key ingredient of most approaches to parameter learning in HMMs, for example, when computing log-likelihood gradients (score functions) via the Fisher identity or the intermediate quantity of the expectation-maximization (EM) algorithm (see, e.g., Cappe, Moulines, and Ryden 2005, chap. 10–11). Scenarios of streaming data or limited computing resources call for online versions—such as the recursive maximum likelihood (Le Gland and Mevel 1997) and online EM (Mongillo and Denève 2008; Cappe 2011) methods—of these approaches, which rely entirely on the possibility of computing incrementally expectations of form (2).

However, as the transition densities $(\ell_n)_{n \in \mathbb{N}}$ are typically complicated, the densities (1) are known only up to normalizing constants in the general case, that is, for models outside the classes of finite state-space models or models with a linear Gaussian structure. SMC methods—or, particle methods—constitute a class of powerful genetic-type algorithms sampling recursively from sequences of distributions, defined on spaces of increasing dimension and known only up to normalizing constants, by means of sequential importance sampling and resampling techniques; see Chopin and Papaspiliopoulos (2020) for a recent introduction to this methodology and Kantas et al. (2015) for a survey of its application to parameter inference in general state-space HMMs. In the following we provide an overview of the most popular approaches to SMC-based additive smoothing. Focus is entirely on online algorithms, by which we mean algorithms such that (1) the sequence $(\phi_{0:n} h_n)_{n \in \mathbb{N}}$ is approximated incrementally in a single sweep of the data and (2) the computational cost of each incremental update as well as the total storage demand is uniformly bounded in $n$.

### 1.2. Previous Work

In the following all random variables are assumed to be well defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We aim to approximate the sequence $(\phi_{0:n} h_n)_{n \in \mathbb{N}}$ by propagating recursively a random sample $(\xi^n_{0:i}, \omega^n_{0:i})_{i=1}^N$ of particles (the $\xi^n_{0:i}$) and associated weights (the $\omega^n_{0:i}$). Here $N$ is the Monte Carlo sample size. For each $n$, the sample forms an empirical probability measure $\phi^n_{0:n} \coloneqq \Omega_n^{-1} \sum_{i=1}^N \omega^n_i \delta_{\xi^n_{0:i}}$, where $\Omega_n \coloneqq \sum_{i=1}^N \omega^n_i$ and $\delta_{\xi^n_{0:i}}$ is the Dirac measure located at $\xi^n_{0:i}$, which allows $\phi^n_{0:n} h_n$ to be approximated by $\phi^n_{0:n} h_n = \Omega_n^{-1} \sum_{i=1}^N \omega^n_i h_n(\xi^n_{0:i})$.

Algorithm 1 describes how the particle sample is updated recursively in the auxiliary particle filter (APF) introduced by Pitt and Shephard (1999) (generalizing the bootstrap particle filter proposed by Gordon, Salmond, and Smith 1993) and here furnished with adaptive multinomial resampling. Using the APF requires a few algorithmic parameters to be set. The mutation step (Line 7) is determined by some proposal transition density $p_n$ on $X_n \times X_{n+1}$ such that $p_n(x_n, \cdot)$ dominates $\ell_n(x_n, \cdot)$ for all $x_n \in X_n$. As a part of the selection step (Line 3), each particle weight is multiplied by some adjustment multiplier function $\vartheta_n$ allowing information concerning the density $\ell_n$ to be taken into account when selecting the particles. At time zero the particle sample is initialized by standard importance sampling, that is, by drawing independent particles $(\xi^n_0, \omega^n_0)_{n \in \mathbb{N}}$ from some proposal density $\nu$ and assigning each particle the weight $\omega^n_0 := \chi(\xi^n_0) / \nu(\xi^n_0)$. Selection is absolutely essential to counteract weight degeneracy, and hence to stabilize numerically the estimator (see, e.g., Cappe, Moulines, and Ryden 2005, sec. 7.3), but should not be applied unnecessarily; thus, we introduce a sequence of binary-valued random variables $(\rho^n_{\alpha})_{n \in \mathbb{N}}$ indicating whether resampling should be triggered or not. The sequence $(\rho^n_{\alpha})_{n \in \mathbb{N}}$ may be adapted to the output the particle filter, that is, for each $n$ the indicator $\rho^n_{\alpha}$ is possibly a function of the elements in $\{(\xi^n_{0:i})_{i=1}^N, (\xi^n_m, h^n_m)_{m \in [1, n]}, m \in [1, \eta] \}$, implying an adaptive resampling schedule. Alternatively, these may depend on $n$ only, implying a deterministic scheme. In the first case, weight skewness is most commonly assessed using the effective sample size (ESS, Liu 1996) defined by $ESS_{\alpha} = 1 / \sum_{i=1}^N (\omega^n_i / \vartheta_n)^2$, which provides an estimator of the number of active particles at time $n$, taking on the values 1 and $N$ in the cases of maximal (all the weights are equal to zero except one) and minimal (all weights are equal and nonzero) skewness, respectively. Using the ESS, one may let $\rho^n_{\alpha} = 1_{\{ESS_{\alpha} < cN\}}$, where $\alpha \in (0, 1)$ is a design parameter, and this will be our primary choice.

### Algorithm 1 Adaptive APF.

**Require:** $(\xi^n_{0:i}, \omega^n_{0:i})_{i=1}^N$.

1: for $i = 1, \ldots, N$ do
2: if $\rho^n_{\alpha} = 1$ then
   3: draw $I_{n+1} \sim \text{Cat}(t((\omega^n_i \vartheta_n(\xi^n_i)^N)_{i=1}^N))$;
4: else
   5: set $I_{n+1} \gets i$;
6: end if
7: draw $\xi^{n+1}_{i} \sim p_n(\xi^n_{I_{n+1}} , \cdot)$;
8: set $\xi^n_{0:i+1} \gets (\xi^n_{0:i}, \xi^{n+1}_{i}, \omega^n_{0:i})$;
9: weight $\omega^{n+1}_{i} \leftarrow \frac{\ell_n(\xi^n_{0:i}, h^n_{0:i})}{p_n(\xi^n_{0:i}, h^n_{0:i}) \vartheta_n}(\omega^n_i)^{1-\rho^n_{\alpha}}$;
10: end for
11: return $(\xi^n_{0:i+1}, \omega^n_{0:i+1}, I^n_{0:i+1})_{i=1}^N$.

In the case of additive functionals, $\phi^n_{0:n} h_n$ can be updated incrementally and without storing the particle paths, Indeed, assuming that we have, at time $n$, computed the statistics $r^n_{0:i} :=
\( h_n(\xi^i_{n+1}), i \in \{1, N\} \), we can, after having executed Algorithm 1, easily update the same according to
\[
\tau^i_{n+1} = \frac{\rho^i_{n+1} + \tilde{h}_n(\xi^i_{n+1}, s^i_{n+1})}{\tau^i_n + \tilde{h}_n(\xi^i_{n+1}, s^i_{n+1})}.
\]

The procedure is initialized by letting \( \tau^i_0 \leftarrow h_0(\xi^i_0) \). Besides allowing for completely recursive and computationally fast updates, this technique has constant memory requirements; in order to perform (5) and then compute the estimator \( \phi_{\text{FFBS}} h_{n+1} = \Omega^{-1} \sum_{i=1}^{N} \omega_{n+1}^{i} \xi^i_{n+1}, \tau^i_{n+1} \), we only need access to \( (\xi^i_{n+1}, \omega_{n+1}^{i}, \tau^i_{n+1}) \) rather than the whole particle paths, whose dimension increases indefinitely with time.

Despite its ease of use and low computational requirements, the procedure described above is impractical due to the well-known particle-path degeneracy phenomenon caused by the resampling operation. More precisely, every time selection is performed, some particles will be propagated from the same parent; thus, by tracing the genealogical history of the particles we eventually encounter, assuming \( n \) is sufficiently large, a common ancestor for all the particles. In the case of multinomial resampling and under standard strong mixing assumptions on the model, Koskela et al. (2020) showed that the expected number of generations back to the most recent common ancestor is \( O(N) \). This result suggests that as \( n \) grows, all particle paths will largely coincide, affecting greatly the reliability of the approximation and yielding a variance that grows quadratically with \( n \); see, for example, Poyiadjis, Doucet, and Singh (2011) for a discussion. An adaptive strategy based on, say, the ESS would still not prevent this particle-path depletion; in fact, such an approach is only able to defer an inevitable destiny, without ensuring stability for large \( n \). In the light of these shortcomings, we will, following the terminology of Douc, Moulines, and Stoffer (2014), refer to this approach as the poor man’s smoother.

An alternative approach, addressing the particle-path degeneracy, is the fixed-lag smoothing technique, proposed by Kitagawa and Sato (2001) and developed further by Olsson et al. (2008). The method obtains long-term stability at the cost of a bias that depends on the ergodicity properties of the model. The bias is controlled by a lag parameter, which should be neither too small, leading to significant bias, nor too large, leading to increased particle-path collapse and hence increase of the variance. Thus, designing a good lag is nontrivial in general.

Another line of research aims to circumvent the particle-path degeneracy phenomenon using backward-sampling techniques. Assume for a moment a Feynman–Kac model of type (4) and that the particle cloud is propagated using the standard bootstrap particle filter, corresponding to the parameterization \( \rho^N_n = 1, \vartheta_n \equiv 1 \) and \( \rho_n \equiv q_n \) of Algorithm 1 (see Glaoquein, Le Corff, and Olsson 2021, sec. 2.2, for the generalization to our setting). In this case, it is easy seen that the conditional probability \( \mathbf{A}^N_n(i,j) \) that \( \mathbf{I}^n_{n+1} = j \) given \( \xi^i_{n+1} \) and \( (\xi^j_{\ell+1})_{\ell=1}^M \), or, in other words, the probability that \( \xi^i_{n+1} \) is the parent of \( \xi^j_{n+1} \), is
\[
\mathbf{A}^N_n(i,j) \propto \omega_{n+1}^{i} q_{n+1}^{i} (\xi^j_{n+1}, s^j_{n+1}) \propto \omega_{n+1}^{i} \bar{e}_{n+1}^{i} (\xi^j_{n+1}, s^j_{n+1}).
\]

In the case of additive smoothing, Del Moral, Doucet, and Singh (2010) use the conditional backward probabilities (6) to Rao-Blackwellize the update (5), yielding the alternative update
\[
\tau^i_{n+1} = \frac{\sum_{j=1}^{N} \mathbf{A}^N_n(i,j) (\tau^j_n + \tilde{h}_n(\xi^j_{n+1}, s^j_{n+1}))}{\tau^i_n + \tilde{h}_n(\xi^i_{n+1}, s^i_{n+1})}.
\]

It is easily seen that this approach is nothing but a forward-only implementation of the so-called forward-filtering backward-smoothing (FFBSm) algorithm (see, e.g., Doucet, Godsill, and Andrieu 2000). Importantly, the Rao-Blackwellized update (7) avoids genealogical tracing and, as a consequence, the path-degeneracy problem. Still, a significant drawback of this approach is its \( O(N^2) \) complexity, which is due to the fact that each update (7) involves the calculation of two sums of \( N \) terms (including the normalizing constant of \( \mathbf{A}^N_n(i,j) \)).

In order to reduce the computational complexity of forward-only FFBSm, Olsson and Westerborn (2017) propose to replace the update (7) by a Monte Carlo estimate based on \( M \ll N \) conditionally independent draws \( (\xi^i_{n+1})_1^M \) from (6), leading to the update
\[
\tau^i_{n+1} = \frac{1}{M} \sum_{j=1}^{M} \frac{\rho^i_{n+1} + \tilde{h}_n(\xi^j_{n+1}, s^j_{n+1})}{\tau^j_n + \tilde{h}_n(\xi^j_{n+1}, s^j_{n+1})}.
\]

By adopting an accept–reject technique developed by Douc et al. (2011), applicable whenever \( q \) is uniformly bounded, the computational complexity of the resulting algorithm, referred to as the particle-based, rapid incremental smoother (PaRIS), can be shown to be \( O(NM) \). The rejection-sampling approach was originally introduced for the forward-filtering backward-simulation (FFBSi) algorithm (Godsill, Doucet, and West 2004), a batch-mode smoother that avoids the computational overload of FFBSm by means of additional simulation, and the PaRIS can in some sense be viewed as an online version of FFBSi. Importantly, Olsson and Westerborn (2017) establish that the PaRIS is asymptotically consistent (as \( N \) tends to infinity) and numerically stable for any fixed \( M \geq 2 \), while \( M = 1 \) leads to a particle-path degeneracy phenomenon reminiscent of that of the poor man’s smoother. In fact, letting \( M \geq 2 \) in the PaRIS yields an estimator with a linear variance growth in \( n \), which is the optimal rate for a Monte Carlo approximation of additive functions on the path space, since some variance is inevitably added at each step. Even though the accept–reject approach implies an average \( O(MN) \) complexity, which is a significant improvement compared to forward-only FFBSm, backward sampling is still the computational bottleneck of the PaRIS. Indeed, in most applications the computational time of the PaRIS exceeds that of the poor man’s smoother by at least one order of magnitude. The reason is that even if the cost of generating a backward draw is indeed constant on average (and independent of \( N \)), it is, since the time until acceptance may be substantial in some cases, usually considerably higher than that of performing a simple forward update.

### 1.3. Our Contribution

In the next section we propose a novel additive smoothing algorithm which can be viewed as a golden mean between computational speed and stability. If the PaRIS may be viewed as a hybrid between the forward-only FFBSm and the FFBSi, our novel algorithm can rather be viewed as a hybrid between the adaptive poor man’s smoother and the PaRIS. The main idea is to avoid, by adaptation, unnecessary selection in order to reduce the particle-path degeneracy in the poor man’s smoother, while interleaving, possibly adaptively, the evolution of the particles.
with regular backward-sampling operations in order to repopulate, when needed, the support of the estimator. In this way we are able to keep the number of backward-sampling operations at a minimum, yielding an algorithm that is, as demonstrated by our simulations, at least one order of magnitude faster than the PaRIS, but with a fully comparable variance. Moreover, besides proving the consistency and asymptotic normality (as \( N \) tends to infinity) of the estimators produced by the algorithm, we also establish the long-term numerical stability of the algorithm by showing that the asymptotic variance grows at most linearly with \( n \).

The rest of the article is organized as follows. In Section 2 we present our novel algorithm and Section 3 is devoted to the theoretical analysis of the same. Besides benchmarking the proposed algorithm against existing online smoothers, the purpose of the simulation study in Section 4 is also to formulate guidelines on how to set its algorithmic parameters. In Section 5 we conclude the article. The article is furnished with supplementary material providing the proofs of the theoretical results in Section 3, which tend to be quite technical and call for a more advanced notational machinery, and some additional numerical simulations.

### 2. A Novel Adaptive Smoother

In the previous section we introduced the sequence \( \rho_{n}^{N} \) regulating the adaptive selection schedule of the APF. We now introduce another binary-valued random sequence \( \epsilon_{n}^{N} \), where each \( \epsilon_{n}^{N} \) is again assumed to be possibly dependent on the output of the APF until \( n \), as \( \rho_{n}^{N} \), but in addition it may also be a function of the new resampling indices \( (J_{i}^{n+1})_{i=1}^{N} \) and of the history of the indicators \( (\rho_{m}^{N})_{m=0}^{n} \). In all cases we require that \( \epsilon_{n}^{N} = 0 \) whenever \( \rho_{n}^{N} = 0 \). While the sequence \( \rho_{n}^{N} \) determines the resampling times (corresponding to times \( n \) for which \( \rho_{n}^{N} = 1 \)), the sequence \( \epsilon_{n}^{N} \) determines the times for which backward sampling is triggered (\( \epsilon_{n}^{N} = 1 \)). By construction, the backward-sampling times form a subset of the resampling times. As we mentioned in Section 1, we let \( \rho_{n}^{N} = 1 \) (ESS < \( \alpha \)N) for all \( n \in \mathbb{N} \), that is, we resample only when the ESS, estimating the number of active particles, falls below a given threshold \( \alpha \)N for some prescribed \( \alpha \in (0, 1) \). On the other hand, the sequence \( \epsilon_{n}^{N} \) is going to be based on some criterion assessing the degeneracy of the particle paths. In practice, we keep track of the number of distinct ancestors at the last time when the indicator was equal to one and set it again equal to one if such number is below another given threshold. We are now going to present our novel algorithm for a generic sequence \( \epsilon_{n}^{N} \), then at the end of the section we illustrate our strategy for determining it adaptively, as specified by Algorithm 3.

Our approach is, loosely speaking, a poor man’s smoother that regularly executes PaRIS-like updating steps according to the schedule determined by \( \epsilon_{n}^{N} \). As before, the algorithm is propagating a weighted sample \( (\xi_{n}, \omega_{n}^{N}) \) of particles and associated smoothing statistics. Whenever \( \epsilon_{n}^{N} = 0 \), the smoothing statistics \( (\tau_{n}^{N}, \lambda_{n}^{N}) \) are updated according to the Equation (5); when instead \( \epsilon_{n}^{N} = 1 \), implying that resampling has been applied, the statistics are updated by means of a superposition of an update (5) and a PaRIS-like update. More specifically, after selection and mutation, each draw \( \xi_{n+1}^{N} \) is linked to a randomly selected ancestor \( \xi_{n+1}^{N} \) and associated statistic \( \tau_{n+1}^{N} \) in the previous generation, where \( \xi_{n+1}^{N} \) is drawn from \( \Lambda_{n}^{N}(i, \cdot) \); after this, the smoothing statistic is updated according to the equation

\[
\tau_{n+1}^{N} = \frac{1}{2} \left( \tau_{n}^{N} + \tilde{h}(\xi_{n+1}^{N}, \tau_{n}^{N}, \xi_{n+1}^{N}) + \tau_{n}^{N} + \tilde{h}(\xi_{n+1}^{N}, \xi_{n+1}^{N}, \xi_{n+1}^{N}) \right) 
\]

As shown by Gloaguen, Le Corff, and Olsson (2021, sec. 2.2), the index \( J_{i}^{n+1} \) can, using rejection sampling, be generated without calculation of the normalizing constant of \( \Lambda_{n}^{N}(i, \cdot) \), at least under the mild assumption that the exists some positive function \( c_{n} \) on \( X_{n+1} \) such that \( \ell_{n}(x_{n}, x_{n+1}) \leq c_{n}(x_{n}+1) \) for all \( (x_{n}, x_{n+1}) \in X_{n} \times X_{n+1} \). In that case, \( J_{i}^{n+1} \) can be simulated by generating, until acceptance, a candidate \( J^{*} \) from \( \text{Cat}(c_{n}(\omega_{n}^{N}))_{i=1}^{N} \) and accepting the same with probability \( \ell_{n}(\xi_{n}^{N}, \xi_{n+1}^{N})/c_{n}(\xi_{n+1}^{N}) \). This can be shown to yield an overall \( O(N) \) computational complexity (see Douc et al. 2011; Gloaguen, Le Corff, and Olsson 2021 for details).

**Remark 2.1.** Even though the accept–reject technique described above applies to many models of interest, it can be ineffective in some cases. It is easily seen that the expected number of trials needed to sample \( J_{i}^{n+1} \) is given by \( c_{n}(\xi_{n+1}^{N})/\sum_{j=1}^{N}(\omega_{n}^{N}/\Omega_{n})_{j}^{N}(\xi_{n}^{N}, \xi_{n+1}^{N}) \); thus, if \( c_{n} \) is highly peaked and concentrated to a small region of the state space, then most of the values \( (\xi_{n}^{N}, \xi_{n+1}^{N})_{i=1}^{N} \) are expected to be small compared to \( c_{n}(\xi_{n+1}^{N}) \), causing long times until acceptance. It may also happen that the bound \( c_{n}(\xi_{n+1}^{N}) \) is hard to find or does not exist. In such cases the user has always the possibility to sample directly from \( \Lambda_{n}^{N}(i, \cdot) \) at the price of calculating its normalizing constant. Another possibility is to take the MCMC-based approach suggested by Gloaguen, Le Corff, and Olsson (2021, sec. 2.2.2), which does not require \( \ell_{n} \) to be bounded in its first argument.

**Algorithm 2.** which we have called AdaSmooth to emphasize its adaptive nature, summarizes all these steps. Clearly, as AdaSmooth operates completely online, without any need of storing the full particle paths, it is enough to input the last particle components and associated weights, \( (\xi_{n}^{N}, \omega_{n}^{N})_{i=1}^{N} \), into the APF (rather than the whole paths) and let it output only the updated sample along with the associated ancestor indices, \( (\xi_{n+1}^{N}, J_{i}^{n+1}, \omega_{n+1}^{N})_{i=1}^{N} \); this operation is expressed compactly as \( \text{APF}(\xi_{n}^{N}, \omega_{n}^{N})_{i=1}^{N} \rightarrow \text{APF}(\xi_{n+1}^{N}, J_{i}^{n+1}, \omega_{n+1}^{N})_{i=1}^{N} \) in Algorithm 2. As explained above, backward sampling is used in Algorithm 2 as a means of guaranteeing the stochastic stability of the resulting estimators, and here the sequences \( \rho_{n}^{N} \) and \( \epsilon_{n}^{N} \) play a critical role; in Section 3 we will discuss the convergence (as \( N \) increases) and stability properties of Algorithm 2, by starting to analyze the case where these sequences are deterministic and then extending the analysis to adaptive policies.

Let us now focus on Line 2 of Algorithm 2. As we mentioned \( \epsilon_{n}^{N} \) may be deterministic or adapted to the output of the APF; we present now our approach in the latter case. Since backward sampling is expensive, our goal is to allow \( \epsilon_{n}^{N} \) to be zero as often as possible without jeopardizing the stability of the estimator. One way to do this is to monitor the number of distinct trajectories by keeping track of the ancestors of the
Algorithm 2 AdaSmooth

Require: \((\tilde{\Omega}^i_n, \tilde{\tau}^i_n, \tilde{\omega}^i_n)_{i=1}^N\)
1: run \((\tilde{\Omega}^i_{n+1}, \tilde{\tau}^i_{n+1}, \tilde{\omega}^i_{n+1})_{i=1}^N \leftarrow \text{APF}(\tilde{\Omega}^i_n, \tilde{\omega}^i_n)_{i=1}^N\);
2: determine \(\varepsilon^N_n\) (using, e.g., Algorithm 3);
3: for \(i = 1, \ldots, N\) do
4: if \(\varepsilon^N_n = 1\) then
5: draw \(j_{n+1} \sim \text{Cat}(\{\omega^i_n\}_{i=1}^N)\) \{using efficient rejection sampling (see discussion following (8))\}
6: set \(\tau^i_{n+1} = 2^{-1}(\tau^i_n+1)+\hat{h}_n(\tilde{\omega}^i_n, \tilde{\tau}^i_n, \tilde{\omega}^i_n) + \tau^i_{n+1} + \tilde{h}_n(\tilde{\omega}^i_{n+1}, \tilde{\tau}^i_{n+1})\);
7: else
8: set \(\tau^i_{n+1} = \tau^i_n + 1 + \hat{h}_n(\tilde{\omega}^i_n, \tilde{\tau}^i_n, \tilde{\omega}^i_n) + \tilde{h}_n(\tilde{\omega}^i_{n+1}, \tilde{\tau}^i_{n+1})\);
9: end if
10: end for
11: return \((\tilde{\Omega}^i_{n+1}, \tilde{\tau}^i_{n+1}, \tilde{\omega}^i_{n+1})_{i=1}^N\).

Algorithm 3 Generation of adaptive backward-sampling schedule \((\varepsilon^N_n)_{n\in\mathbb{N}}\).

Require: \((E^N_n)_{i=1}^N, (\bar{I}^i_n)_{i=1}^N, \beta \in (0, 1)\)
1: set \(E^N_{n+1} \leftarrow \bar{I}^i_{n+1}\) for \(i = 1, \ldots, N\);
2: if \(\rho^N_n = 1\) and \(|E^N_n| < \beta N\) then
3: set \(\varepsilon^N_n = 1\);
4: set \(E^N_{n+1} \leftarrow i\) for \(i = 1, \ldots, N\);
5: else
6: set \(\varepsilon^N_n \leftarrow 0\);
7: end if
8: return \((E^N_{n+1})_{i=1}^N, \varepsilon^N_n\).

Remark 2.2. At this point the reader might wonder why only a single backward draw is considered in the update of the statistics instead of a generic positive integer \(M\) as for the PaRIS. The motivation is that the main purpose of this work is to develop a fast and stable algorithm; one backward draw is sufficient for stability and with a larger number the running time necessarily increases; indeed, the backward sampling operation is the one dominating the complexity algorithm. The motivation behind increasing the number of backward draws would be to reduce the asymptotic variance of the estimates; this is certainly true, nonetheless we have observed that the reduction of the variance obtained by increasing \(M\) does not make worth the slowdown of the algorithm, while it might be more effective to increase the number of particles instead. We demonstrate this with our simulations in Section G of the supplementary materials.

3. Theoretical Results

3.1. Deterministic Selection and Backward-Sampling Schedules

Our initial analysis of Algorithm 2 will be conducted under the assumption that the selection and backward-sampling schedule is deterministic.

Assumption 1. For all \(n \in \mathbb{N}\), \(\rho^N_n = \rho_n\) and \(\varepsilon^N_n = \varepsilon_n\), where the sequences \((\rho_n)_{n\in\mathbb{N}}\) and \((\varepsilon_n)_{n\in\mathbb{N}}\) are deterministic and such that \(\varepsilon_n = 0\) whenever \(\rho_n = 0\).

In this setting we establish two results: the almost-sure convergence (Theorem 3.1) of the estimator \(\Omega_n^{-1} \sum_{i=1}^N \omega^i_n \tilde{\tau}^i_n\), where \((\tilde{\omega}^i_n, \tilde{\tau}^i_n)_{i=1}^N\) is produced by \(n\) steps of Algorithm 2, as well as a central limit theorem (Theorem 3.2), whose asymptotic variance is subject to further investigation regarding the stochastic stability of the algorithm. Proofs are found in the supplementary materials. For every \(n \in \mathbb{N}\) we define the weight function

\[
 w_n(\rho_n) : X_n \times X_{n+1} \ni (x, x') \mapsto \frac{\varepsilon_n(x, x')}{(\partial_n(x))^\rho_n(\rho_n(x, x')).}
\]

In addition, we set

\[
 w_{-1} : X_0 \ni x \mapsto \frac{\chi(x)}{\nu(x)}.
\]

Assumption 2. For all \((\rho_n)_{n\in\mathbb{N}}\) the weight functions \((w_n(\rho_n))_{n\in\mathbb{N}}\) and \(w_{-1}\) are bounded. So are also the auxiliary weight functions \((\partial_n)_{n\in\mathbb{N}}\).

In the following we define, for every \(n \in \mathbb{N}\), \(H_n\) as the set of additive functionals \(h_n\) in the form (3) with bounded terms. In addition, we let \(\rho_{0:n-1} = (\rho_0, \ldots, \rho_{n-1})\) and \(\varepsilon_{0:n-1} = (\varepsilon_0, \ldots, \varepsilon_{n-1})\).
Theorem 3.1 (strong consistency). Let Assumptions 1 and 2 hold. Then for every \( n \in \mathbb{N} \) and \( h_n \in H_n \),
\[
\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} \frac{\omega_i}{\Omega_n} \tau_i = \phi_{0,n} h_n, \quad \text{P-a.s.}
\]

Theorem 3.2 (asymptotic normality). Let Assumptions 1 and 2 hold. Then for every \( n \in \mathbb{N} \) there exists a positive functional \( \sigma_n(h_{n-1}, \varepsilon_{n-1}) \) on \( H_n \) such that for every \( h_n \in H_n \), as \( N \to \infty \),
\[
\sqrt{N} \left( \frac{1}{N} \sum_{i=1}^{N} \frac{\omega_i}{\Omega_n} \tau_i - \phi_{0,n} h_n \right) \xrightarrow{d} \sigma_n(h_{n-1}, \varepsilon_{n-1})(h_n) Z,
\]
where \( Z \) has standard Gaussian distribution.

The almost sure convergence established by Theorem 3.1 is in fact a direct consequence of a stronger result in the form of a Hoeffding-type exponential concentration inequality for finite sample sizes \( N \); see Section C.2 in the supplementary materials for details. An explicit expression of the asymptotic variance \( \sigma_n^2(h_{n-1}, \varepsilon_{n-1}) / n \) of Theorem 3.2 is provided in Section C.3 of the supplementary materials. Next, we establish, again under Assumption 1, the stochastic stability of Algorithm 2 by bounding \( \sigma_n(h_{n-1}, \varepsilon_{n-1}) / n \) uniformly in \( n \). Again, the proof is provided in the supplementary materials. The analysis proceeds in two steps, where we in the first step analyze the algorithm in the case of systematic selection at each time point, and then, in the second step, extend these results to nonsystematic, but still deterministic, selection schedules using an auxiliary model extension. In the first step, our proofs build upon recent works on the PaRIS by Olsson and Westerborn (2017) and Gloaguen, Le Corff, and Olsson (2021); however, the fact that the updating rule (8) combines forward as well as backward indices induces a complex dependence structure that makes the adaptation highly nontrivial.

The following assumption is used to control the stochastic stability of the marginal approximations produced by the APF, by bounding uniformly the distance between any two consecutive resampling times.

Assumption 3. There exists \( d \in \mathbb{N}^+ \) such that for all \( n \in \mathbb{N} \),
\[
\min\{k \in \mathbb{N}^+ : \rho_{n+k} = 1\} \leq d, \quad \text{that is, the distance between two resampling times is always less than or equal to } d.
\]

For any bounded measurable function \( h \), let \( \|h\|_\infty \) denote the supnorm of \( h \). Our stability analysis will be carried through under the following—now classical—strong mixing assumption, which typically require the state spaces to be compact sets (see, e.g., Del Moral 2004, sec. 4).

Assumption 4. There exist constants \( 0 < \underline{\varepsilon} < \bar{\varepsilon} < \infty \) such that for every \( n \in \mathbb{N} \) and \((x, x') \in X_n \times X_{n+1}, \underline{\varepsilon} \leq \ell_n(x, x') \leq \bar{\varepsilon} \). Moreover, there exist positive constants \( \delta, \bar{\delta} \) and \( \|\theta\|_\infty \) such that for all \( n \in \mathbb{N} \) and \( \rho \in \{0, 1\} \),
\[
\|w_n(\rho)\|_\infty \leq \delta \quad \text{and} \quad \|\theta_n\|_\infty \leq \|\theta\|_\infty.
\]
In addition, \( \|w_1 - w_n\|_\infty \leq \bar{\delta} \).

For every \( n \in \mathbb{N}^+ \) and \( j \in \mathbb{N} \), we define \( r_n := \sum_{m=0}^{n-1} \rho_m \), that is, the number of selection operations before time \( n \), and \( n_j := \min\{n \in \mathbb{N} : r_{n+1} = j + 1\} \), the time of the \((j+1)\)th selection operation.

Theorem 3.3. Let Assumptions 1, 3, and 4 hold. Then there exist positive constants \( C_1 \) and \( C_2 \), both depending on \( \underline{\varepsilon}, \bar{\varepsilon}, \delta, \) and \( d \), such that for all additive functionals \( (h_n)_{n \in \mathbb{N}} \) in the form (3) for which there exists \( \|h\|_\infty > 0 \) such that for all \( n \in \mathbb{N}, \|h_n\|_\infty \leq \|h\|_\infty \) and \( \|h_0 + h_1\|_\infty \leq \|h\|_\infty \),
\[
\limsup_{n \to \infty} \frac{1}{n^2} \rho_{n-1} (h_{n-1}, \varepsilon_{n-1})(h_n) \leq \int d^2 \|h\|^2_\infty |\vartheta|_\infty \left( C_1 + C_2 \int \sum \sum (1 + \varepsilon_n)^{-1} \right).
\]
As discussed above, our aim is to establish the stability of Algorithm 2 by bounding the right-hand side of (9) uniformly in \( n \). However, such a bound is not possible for all sequences \( (\varepsilon_n)_{n \in \mathbb{N}} \); indeed, in the case where \( \varepsilon_n = 0 \) for all \( j \in \mathbb{N} \) it holds that
\[
\lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} \sum_{\ell=0}^m (1 + \varepsilon_n)^{-1} \leq \lim_{n \to \infty} \frac{(r_n + 1)}{n} \to 1 = \infty,
\]
which is not surprising since Algorithm 2 coincides with the poor man’s smoother (with adaptive resampling) when the backward simulation mechanism is de-activated. Still, as established by the following theorem, a regular backward sampling schedule is sufficient to obtain a linearly increasing asymptotic variance. We define \( \Delta_j := \min\{k \in \mathbb{N}^+ : \varepsilon_{nk+1} = 1\}, j \in \mathbb{N} \cup \{-1\} \), which corresponds to the distance, in terms of the number of selection operations, between any selection time \( n_j \) and the first subsequent backward-sampling time. If these distances are uniformly bounded, then we may obtain the desired linear bound.

Proposition 3.4. Assume that there exists \( \Delta \in \mathbb{N}^+ \) such that \( \Delta_j \leq \Delta \) for all \( j \geq -1 \). Then
\[
\lim_{n \to \infty} \frac{1}{r_n} \sum_{m=0}^{n-1} \sum_{\ell=0}^m (1 + \varepsilon_n)^{-1} \leq \frac{3\Delta - 1}{2},
\]
with equality if \( \Delta_j = \Delta \) for all \( j \geq -1 \).

The proof of Proposition 3.4 is given in the supplementary materials.

3.2. Adaptive Selection and Backward-Sampling Schedules

Next we will show that the central limit theorem in Theorem 3.2 can be extended to the case where the selection schedule is random and adapted to the values of the ESS. In order to guarantee the stability of the algorithm, we will still assume that selection is performed at least every \( d \in \mathbb{N}^+ \) steps. In addition, instead of using the same parameter \( \alpha \) for all iterations, we will, following Del Moral, Doucet, and Jasra (2012, sec. 5.2), in order to deal with some technicalities in the proofs, consider a sequence \( (\alpha_n)_{n \in \mathbb{N}} \) of parameters being realizations of random variables with state space \((0, 1)\). These technical assumptions can be relaxed in practice.

Assumption 5. Let \( (\alpha_n)_{n \in \mathbb{N}} \) be realizations of absolutely continuous independent random variables \( (\alpha_n)_{n \in \mathbb{N}} \) with values in \((0, 1)\).
For a given \( d \in \mathbb{N}^* \), let \( (\rho_n^N)_{n \in \mathbb{N}} \) be defined recursively as
\[
\rho_0^N := 1_{\{\text{ESS}_0 < c_0 N\}} \quad \text{and} \quad \rho_{n+1}^N := 1 - 1_{\{\text{ESS}_{n+1} \geq c_{n+1} N\}} \rho_{n+1}^N \cdot \rho_n^N < d_{n+1} < d, \quad n \in \mathbb{N},
\]
with \( (d_n^N)_{n \in \mathbb{N}} \) being also recursively defined through
\[
d_0^N := 1 - \rho_0^N \quad \text{and} \quad d_{n+1}^N := (1 - \rho_{n+1})^N (1 + \rho_n^N), \quad n \in \mathbb{N}.
\]

Note that \( d_n^N \) counts the number of consecutive times, including \( n \), for which resampling has not been performed. The following lemma is proved in the Section D.1 of the supplementary materials.

**Lemma 3.5.** Let Assumption 5 hold. Then for all \( n \in \mathbb{N} \) and almost all \( \alpha_{0:n} \in (0, 1)^{n+1} \) there exists \( \rho_n^{a,d} \in \{0, 1\} \) such that, as \( N \to \infty \),
\[
\rho_n^N \overset{P}{\to} \rho_n^{a,d}.
\]

**Assumption 6.** For every \( n \in \mathbb{N} \), \( \epsilon_n^N \) is \( \sigma(\rho_{0:n}^N) \)-measurable, which implies the existence of a measurable function \( f_n : [0, 1]^{n+1} \to [0, 1] \) such that \( \epsilon_n^N = f_n(\rho_{0:n}^N) \). Moreover, we assume that \( \epsilon_0^N = 0 \) whenever \( \rho_0^N = 0 \).

Assumption 6, which requires that the backward sampling schedule only depends on the selection schedule, covers, for example, the approach where backward sampling is triggered repeatedly after a fixed, deterministic number of intermediate resampling operations. It does not cover the more delicate backward sampling strategy in Algorithm 3, where backward sampling is triggered on the basis of a criterion measuring particle-path degeneracy.

**Lemma 3.6.** Let Assumptions 5 and 6 hold. Then for all \( n \in \mathbb{N} \) and almost all \( \alpha_{0:n} \in (0, 1)^{n+1} \) there exists \( \epsilon_n^{a,d} \in \{0, 1\} \) for which it holds as \( N \to \infty \),
\[
\epsilon_n^N \overset{P}{\to} \epsilon_n^{a,d}.
\]

**Corollary 3.7.** Let Assumptions 2, 5, and 6 hold and \( (\omega_i^N, \tau_{i}^N)_{i=1}^{N} \) be generated by \( n \) iterations of Algorithm 2. Then for every \( n \in \mathbb{N}^* \), almost all \( \alpha_{0:n-1} \in (0, 1)^n \) and all \( h_n \in \mathbb{H}_n \), as \( N \to \infty \),
\[
\sqrt{N} \left( \sum_{i=1}^{N} \frac{\omega_i^N \phi_i^N}{\Omega_n^N} \right) \overset{D}{\to} \sigma_n^{a,d} \sigma_n^{d,a} \epsilon_{n-1}^{a,d}(h_n) Z,
\]
where \( Z \) has standard Gaussian distribution and \( \sigma_n^{a,d} \sigma_n^{d,a} \epsilon_{n-1}^{a,d}(h_n) \) is the asymptotic variance of Theorem 3.2 with selection schedule \( \rho_{0:n-1}^N \) and backward-sampling schedule \( \epsilon_{n-1}^{a,d} \) given by Lemmas 3.5 and 3.6, respectively.

The proofs of Lemma 3.6 and Corollary 3.7 are provided in the Sections D.2 and D.3 of the supplementary materials, respectively. Note that an immediate consequence of Corollary 3.7 is that \( \Omega_n^{N-1} \sum_{i=1}^{N} \omega_i^N \tau_{i}^N \overset{P}{\to} \phi_0^N h_N \) as \( N \to \infty \). The stochastic stability of the adaptive algorithm depends on the asymptotic variance \( \sigma_n^{a,d} \sigma_n^{d,a} \epsilon_{n-1}^{a,d}(h_n) \), more specifically on the limit sequence \( (\epsilon_n^{a,d})_{n \in \mathbb{N}} \). Proposition 3.4 guarantees a linear growth of the variance with respect to \( n \) for any adaptation schedule that allows the number of selection operations between each backward-sampling operation to be uniformly bounded; for such schedules, this property will be transferred to the limit schedule, providing an \( O(n) \) bound on the asymptotic variance.

A thorough analysis of the setting where also the backward-sampling mechanism is activated adaptively using the technology described in Algorithm 3 is beyond the scope of the present article. Instead, we limit ourselves to justifying heuristically that triggering, as in Algorithm 3, backward sampling only when the proportion of distinct Enoch indices falls below a given threshold leads, in accordance with Proposition 3.4, to regular distances on average between the times of backward sampling. This justification, which is based on results by Koskela et al. (2020), is presented in Section F in the supplementary materials. As we will see in the next section, our simulations indicate that the distance between two backward sampling operations in terms of the number of intermediate resampling operations stays close to regular and constant on average with respect to the sample size \( N \).

### 4. Numerical Results

We demonstrate numerically our algorithm on two different state-space models: a linear Gaussian HMM and a stochastic volatility model with correlated noise.

#### 4.1. Linear Gaussian HMM

We first consider a linear Gaussian HMM on \( \mathbb{R} \), described by the equations
\[
\begin{align*}
X_{n+1} &= aX_n + \sigma_U U_{n+1}, \\
Y_n &= bX_n + \sigma_V V_n,
\end{align*}
\]
where \( (U_n)_{n \in \mathbb{N}^*} \) and \( (V_n)_{n \in \mathbb{N}} \) are independent sequences of mutually independent standard normally distributed noise variables and \( (a, b, \sigma_U, \sigma_V) \in \mathbb{R}^2 \) are model parameters. If \( |a| < 1 \), the unobserved state process \( (X_n)_{n \in \mathbb{N}} \) has a stationary distribution given by the zero-mean Gaussian distribution with variance \( \sigma_U^2 / (1 - a^2) \), according to which \( X_0 \) is initialized. In this section, focus is set on the problem of estimating the expectation of the state sum \( h_n(x_{0:n}) = \sum_{m=0}^{n} x_m \) under the joint-smoothing distribution \( \phi_0^N \) on the basis of \( n + 1 = 501 \) observations generated by simulation of the model parameterized by \( (a, b, \sigma_U, \sigma_V) = (0.7, 1, 0.2, 1) \). The main reason for considering a linear Gaussian HMM and this particular state functional is that these allow exact solutions to the additive smoothing problem to be calculated using disturbance smoothing (see, e.g., Cappé, Moulines, and Rydén 2005, sec. 5.2). Having access to the exact solution, we may study the convergence and accuracy of AdaSmooth and benchmark the same against existing algorithms.

In order to investigate how the choices of \( \alpha \) and \( \beta \), governing the adaptation criteria for resampling through the ESS and the backward-sampling through Algorithm 3, respectively, affect the performance of the algorithm, we run the algorithm for varying combinations of \((\alpha, \beta) \in [0, 1]^2 \) and \( N \in \mathbb{N}^* \).
Figure 1. Efficiencies of AdaSmooth operating on the linear Gaussian HMM with different combinations of \((\alpha, \beta) \in [0, 1]^2\) and \(N = \{50, 100, 200, 500\}\). Each estimate is based on 100 replicates.

Figure 2. Boxplots of estimates (divided by \(\sqrt{n}\)) of smoothed expectations of \(h_n(x_0) = \sum_{m=0}^{n} x_m\), for \(n \in \{100, 500, 1000\}\), in the linear Gaussian HMM in Section 4.1. Each algorithm was rerun 100 times with \(N = 50\) and \(N = 500\). The black-dashed lines represent exact solutions provided by the disturbance smoother.

Table 1. Efficiencies of the forward-only FFBSm, the PaRIS, and AdaSmooth parameterized by \((\alpha, \beta) = (0.6, 0.5)\), operating on the linear Gaussian model in Section 4.1 with different sample sizes \(N\).

| \(N\)     | 50   | 100  | 200  | 500  |
|-----------|------|------|------|------|
| FFBSm     | 0.86 | 0.45 | 0.33 | 0.16 |
| PaRIS      | 5.09 | 5.62 | 6.08 | 5.86 |
| AdaSmooth (0.6, 0.5) | 31.17 | 38.01 | 58.32 | 68.92 |

\([50, 100, 200, 500]\). For each combination, we replicated 100 independent estimates of \(\phi_{0:n}h_n\) for each of the AdaSmooth, PaRIS and forward-only FFBSm algorithms, all running with the same number \(N\) of particles. For simplicity, the underlying particles were mutated according to the dynamics of the state process and selected without any adjustment of the particle weights.

In our comparison we first evaluate the efficiency of each algorithm, which we define as the ratio of inverse sample variance to computational time, scaled further by \(1/\sqrt{N}\); that is, efficiency = \(1/(\sqrt{N} \times \text{sample variance} \times \text{computational time})\). Figure 1 displays efficiencies of AdaSmooth for different combinations of the algorithmic parameters \(\alpha\) and \(\beta\), and in these plots there is clearly a region in the parameter space, with \(\alpha\) and \(\beta\) being around 0.6 and 0.5, respectively, varying slightly with \(N\), for which the efficiency is maximal. For a comparison, Table 1 shows the efficiencies also of the PaRIS and the forward-only FFBSm algorithms, which are outperformed by AdaSmooth by about one and two orders of magnitude, respectively.

Next, we illustrate that the output of AdaSmooth converges, as \(N\) increases and for any combination of \(\alpha\) and \(\beta\), to the exact solution provided by the disturbance smoother and compare the same to the outputs of the competitors. Figure 2 displays boxplots of independent estimates obtained with AdaSmooth for a selection of parameterizations as well as with the forward-only FFBSm, the PaRIS and the poor man’s smoother for an observation record comprising \(n + 1 = 1001\) observations. The figure also displays exact solutions provided by the disturbance smoother. Each box is based on 100 replicates and for each particle sample size \(N \in \{50, 500\}\), estimates of \(\phi_{0:n}h_n\) for \(n \in \{100, 500, 1000\}\) are reported. The estimates are divided by \(\sqrt{n}\) with the purpose of illustrating the different smoothers’ stability properties as \(n\) increases. In short: completely in line with the theoretical results obtained in Section 3, the range of the boxes decrease with \(N\) and stay, with exception of the poor man’s smoother, close to constant in \(n\). Interestingly, the algorithm parameterized by \((\alpha, \beta) = (1, 0.1)\) (corresponding to systematic selection and infrequent backward sampling), exhibiting the...
largest variance among the AdaSmooth estimators, still does not show the quadratic variance growth of the poor man’s smoother. This is even clearer from Figure 3, displaying time-normalized empirical variances, produced by AdaSmooth (with different parameterizations), the forward-only FFBSm, the PaRIS, and the poor man’s smoother, of estimates of smoothed expectations of $h_n(x_{2T}) = \sum_{m=0}^n x_m$ for different $n$, in the linear Gaussian HMM in Section 4.1. The empirical variances were obtained by rerunning each algorithm 100 times with $N = 500$ particles.

**4.2. Stochastic Volatility Model**

In order to investigate the performance of AdaSmooth in a nonlinear setting, we consider a modification of the stochastic volatility model proposed by Hull and White (1987). The observed stock returns $(Y_n)_{n \in \mathbb{N}}$ and the unobserved log-volatility $(X_n)_{n \in \mathbb{N}}$ are modeled as $\mathbb{R}$-valued processes evolving recursively according to

$$X_{n+1} = aX_n + \sigma U_{n+1},$$

$$Y_n = b \exp(X_n/2) V_n,$$

where $a \in \mathbb{R}$, $b > 0$ and $\sigma > 0$ are model parameters. Here $V_0$ has standard Gaussian distribution, independent of $X_0$, while $(U_{n}, V_{n})_{n \in \mathbb{N}}$ is a sequence of independent bivariate Gaussian random variables, with standard marginals and correlation $\rho \in (-1, 1)$. All parameters of the model are assumed to be known, with $a = 0.975$, $b = 0.641$, $\sigma = 0.165$, values which appear frequently in the literature, and $\rho = -0.1$; here the negative correlation reflects the fact that stock returns tend to be lower than average and sometimes negative in high-risk environments (high volatility). With this parameterization, the log-volatility has a stationary distribution given by the zero-mean Gaussian distribution with variance $\sigma^2/(1-a^2)$, according to which $X_0$ is initialized. It immediately follows that for all $n \in \mathbb{N}^*$, we may write $V_n = \rho U_n + \sqrt{1-\rho^2} W_n$, where $W_n$ has standard Gaussian distribution and is independent of $U_n$. Thus, for $n \in \mathbb{N}$ the observation process can be alternatively expressed as $Y_{n+1} = b \exp(X_{n+1}/2)(\rho X_{n+1} - a X_n)/\sigma + \sqrt{1-\rho^2} W_{n+1}$, corresponding to a transition density $g(x_n, x_{n+1}, y_{n+1})$. Note that model $(X_n, Y_n)_{n \in \mathbb{N}}$ is not an HMM, since the correlation of the noise variables induces a conditional correlation between $Y_{n+1}$ and $X_n$, given $X_{n+1}$. Still, this does not cause any problem for us, since the general setting of Section 1.1 does not presuppose the densities $(\ell_n)_{n \in \mathbb{N}}$ to satisfy a Feynman–Kac-type decomposition (4), and we may simply set $\ell_n(x_n, x_{n+1}) = q(x_n, x_{n+1})g((x_n, x_{n+1}), y_{n+1})$, where $q$ is the transition density of the log-volatility and $y_{n+1}$ given data at time $n + 1$. (Using the modified model $(\tilde{X}_n, \tilde{Y}_n)_{n \in \mathbb{N}}$, with $\tilde{X}_n := X_{n-1/2}$ being compound states, which is indeed an HMM, would not be an option, since the fact that the transition kernel of $(\tilde{X}_n)_{n \in \mathbb{N}}$ involves a Dirac mass implies that this HMM is not fully dominated.)

For this model we consider online additive smoothing for three different additive functional with terms given by $\tilde{h}^{(1)}(x_n, x_{n+1}) = x_{n+1}^2$, $\tilde{h}^{(2)}(x_n, x_{n+1}) = x_{n+1}$ and $\tilde{h}^{(3)}(x_n, x_{n+1}) = x_n x_{n+1}$, and compare the performance of AdaSmooth, for different parameterizations $(\alpha, \beta)$, to the poor man’s smoother with adaptive selection, the PaRIS and the forward-only FFBSm. Each of these algorithms was rerun 100 times for $n = 1000$ time steps. As in the previous example, the underlying particles were mutated using $q$ and selected without any adjustment of the particle weights. This choice is obviously sub-optimal, since evolving the particles “blindly,” without taking information concerning subsequent observations into account, may cause faster weight degeneration. Hence, with more sophisticated adaptive proposals and adjustment functions we would expect even better results than those we are about to report, since a slower weight degeneration requires selection and backward-sampling to be applied less frequently. Like in the previous example, AdaSmooth outperforms by far its competitors. In Figure 4 we observe that with suitable choices of $\alpha$ and $\beta$, AdaSmooth is not only significantly faster—by one to two orders of magnitude—that the PaRIS and the forward-only FFBSm, it also exhibits lower variance. In fact, its computational complexity is of the same order as that of the poor man’s smoother, whose stochastic instability is evident from the plots. Figure 5, which displays time-normalized empirical variances over time, confirms perfectly well our theoretical results in that AdaSmooth exhibits a linear increase of variance with $n$ for any parameterization. The fact that AdaSmooth provides the lowest variance in some cases is due to the adaptation of the selection schedule. In this example, the choice $(\alpha, \beta) = (0.8, 0.6)$ leads to a doubled computational complexity compared to $(0.6, 0.5)$, however, without increasing notably the accuracy. Like in the previous example, we observed that an optimal tradeoff between variance and computational effort was obtained by setting $\alpha$ and $\beta$ to values around 0.5, with $\alpha \geq \beta$. In all these simulations, any backward-sampling operation in AdaSmooth and the PaRIS was performed using the rejection-sampling technique described in

![Figure 3. Time-normalized empirical variances, produced by AdaSmooth (with different parameterizations), the forward-only FFBSm, the PaRIS, and the poor man's smoother, of estimates of smoothed expectations of $h_n(x_{2T}) = \sum_{m=0}^n x_m$ for different $n$, in the linear Gaussian HMM in Section 4.1. The empirical variances were obtained by rerunning each algorithm 100 times with $N = 500$ particles.](image-url)
Figure 4. Boxplots of estimates (divided by $\sqrt{n}$) of smoothed expectations of three distinct smoothed functionals for $n \in \{300, 600, 900\}$ in the stochastic volatility model in Section 4.2. For each algorithm, 100 estimates were produced using $N = 1000$ particles. Depending on the chosen parameters $\alpha$ and $\beta$, AdaSmooth was $10^{-70}$ times faster than the PaRIS, $35-250$ times faster than the forward-only FFBSm, and $3.5-25$ times slower than the poor man’s smoother.

Figure 5. Time-normalized empirical variances, produced by AdaSmooth (with different parameterizations), the forward-only FFBSm, the PaRIS, and the poor man’s smoother, of estimates of smoothed expectations of $h_n(X_{n-1:n}) = X_n^2$ for different $n$, in the stochastic volatility model in Section 4.2. The empirical variances were obtained by rerunning each algorithm 100 times with $N = 250$ particles.

Section 2; on the other hand, when the backward probabilities were instead computed explicitly, the computational time of the PaRIS became similar to that of the FFBSm, while AdaSmooth slowed down by a factor $10^{-20}$.

Finally, Table 2 reports the average time duration between adaptive resampling operations as well as the number of selection operations on average between subsequent backward-sampling operations for different parameterizations. We observe that these averages stay basically constant when $N$ is varied, suggesting that the operations are triggered regularly for given $\alpha$ and $\beta$. This supports our heuristic arguments outlined in Section F of the supplementary materials. We have also observed that the parameter $\beta$ may become useless if greater than $\alpha$, especially for $\alpha$ being greater than about 0.5, since selection is likely to automatically trigger backward sampling in that case.

5. Conclusions

The presented algorithm, AdaSmooth, aims to combine the best of standard adaptive sequential importance sampling with resampling—which is computationally fast but numerically unstable—and the best of the PaRIS—whose long-term numerical stability is obtained via computationally costly backward sampling. As only limited code extensions of the standard particle filter is needed, AdaSmooth is very easily implemented or at least not significantly more complicated than the PaRIS.
Table 2. Average time duration between adaptive resampling operations (left columns) and the number of selection operations on average between subsequent backward-sampling operations (right columns) for different parameterizations of AdaSmooth in the stochastic volatility model in Section 4.2.

| N    | 50   | 100  | 250  | 500  | 1000 | 2000 | 5000 | 10,000 | 50,000 | 100,000 |
|------|------|------|------|------|------|------|------|--------|--------|---------|
| α = 1.0, β = 0.1 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 |
| α = 0.8, β = 0.6 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 |
| α = 0.6, β = 0.5 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 |
| α = 0.3, β = 0.2 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 |
| α = 0.5, β = 0.6 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 | 10.0 |

NOTE: For each parameterization and particle sample size, the values are based on a single run of the algorithm across n = 10,000 time steps.

in this respect. Still, the estimator is function-specific in that the implementation depends on the additive functional under consideration.

Even though the recursive updating step (8) of AdaSmooth, combining the forward and backward indices produced by the selection and backward-sampling operations, respectively, gives the estimator a very complex intrinsic dependence structure, we have been able to, by adapting existing theoretical analyses of the PaRIS, furnish the proposed algorithm with solid convergence and stability results, at least as long as the backward sampling schedule is adapted to the forward sampling schedule. As indicated by our numerical examples, AdaSmooth provides a tremendous improvement, of about one and two orders of magnitude, in terms of accuracy and computational efficiency compared to the PaRIS and the forward-only FFBSm algorithms, respectively. The improvement depends on the algorithmic parameters α and β, and in the models we tested it was observed that having both values around 0.5 with α larger than β provides the best results. Even if we do not exclude that other combinations could work better on other examples, we dare to elevate this to a general rule of thumb when it comes to selecting these parameters.

The theoretical analysis of the backward-sampling schedule $(ε_n^n)$ proposed in Algorithm 3 remains an open—and possibly very complex—problem that we leave as future research. Other possible directions of future research are the extension of AdaSmooth beyond additive functionals and the further improvement of the algorithm via adaptation of the proposal kernels and adjustment multipliers of the APF.

Supplementary Materials

Supplement: Theoretical analyses and proofs of the results in Section 3 and further numerical simulations (supplement.pdf).

Python code: Python code for replicating the simulations of Section 4 (adasmooth_code.zip).

Acknowledgments

The authors would like to thank Dr. Sumeetpal Singh for valuable discussions. The authors also wish to thank the associate editor and the two anonymous reviewers for all their precious observations and thoughtful comments, which have improved the quality of the article.

Disclosure Statement

The authors report there are no competing interests to declare.

Funding

J. Olsson and A. Mastrototaro are supported by the Swedish Research Council, grant 2018-05230.

ORCID

Alessandro Mastrototaro  http://orcid.org/0000-0001-9380-1197
Jimmy Olsson  http://orcid.org/0000-0003-0772-846X
Johan Alenlöv  http://orcid.org/0000-0001-9565-7686

References

Cappé, O. (2011), “Online EM Algorithm for Hidden Markov Models,” Journal of Computational and Graphical Statistics, 20, 728–749. [357]
Cappé, O., Moulines, E., and Rydén, T. (2005), *Inference in Hidden Markov Models*, New York: Springer. [356,357,362]

Chopin, N., and Papaspiliopoulos, O. (2020), *An Introduction to Sequential Monte Carlo Methods*, Cham: Springer. [357]

Del Moral, P. (2004), *Feynman-Kac Formulae. Genealogical and Interacting Particle Systems with Applications*, New York: Springer. [357,361]

Del Moral, P., Doucet, A., and Jara, A. (2012), “On Adaptive Resampling Strategies for Sequential Monte Carlo Methods,” *Bernoulli*, 18, 252–278. [361]

Del Moral, P., Doucet, A., and Singh, S. S. (2010), “A Backward Interpretation of Feynman-Kac Formulae,” *ESAIM: Mathematical Modelling and Numerical Analysis*, 44, 947–975. [358]

Douc, R., Garivier, A., Moulines, E., and Olsson, J. (2011), “Sequential Monte Carlo Smoothing for General State Space Hidden Markov Models,” *Annals of Applied Probability*, 21, 1201–2145. [358,359]

Douc, R., Moulines, E., and Stoffer, D. (2014), *Nonlinear Time Series: Theory, Methods and Applications with R Examples*, Boca Raton, FL: CRC Press. [358]

Doucet, A., Godsill, S., and Andrieu, C. (2000), “On Sequential Monte-Carlo Sampling Methods for Bayesian Filtering,” *Statistics and Computing*, 10, 197–208. [358]

Gloaguen, P., Le Corff, S., and Olsson, J. (2021), “A Pseudo-Marginal Sequential Monte Carlo Online Smoothing Algorithm,” available at https://arxiv.org/abs/1908.07254. [356,358,359,361]

Godsil, S. J., Doucet, A., and West, M. (2004), “Monte Carlo Smoothing for Non-linear Time Series,” *Journal of the American Statistical Association*, 50, 438–449. [358]

Gordon, N., Salmond, D., and Smith, A. F. (1993), “Novel Approach to Nonlinear/non-Gaussian Bayesian State Estimation,” *IEE Proceedings F (Radar and Signal Processing)*, 140, 107–113. [357]

Hull, J., and White, A. (1987), “The Pricing of Options on Assets with Stochastic Volatilities,” *The Journal of Finance*, 42, 281–300. [364]

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., and Chopin, N. (2015), “On Particle Methods for Parameter Estimation in State-Space Models,” *Statistical Science*, 30, 328–351. [357]

Kitagawa, G., and Sato, S. (2001), “Monte Carlo Smoothing and Self-Organising State-Space Model,” in *Sequential Monte Carlo Methods in Practice*, Information Science and Statistics, eds. A. Doucet, N. Freitas, and N. Gordon, pp. 177–195, New York: Springer. [358]

Koskelo, J., Jenkins, P. A., Johansen, A. M., and Spanò, D. (2020), *Asymptotic Genealogies of Interacting Particle Systems with an Application to Sequential Monte Carlo*, *The Annals of Statistics*, 48, 560–583. [358,362]

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., Chopin, N. (2015), “On Particle Methods for Parameter Estimation in State-Space Models,” *Statistical Science*, 30, 328–351. [357]

Kitagawa, G., and Sato, S. (2001), “Monte Carlo Smoothing and Self-Organising State-Space Model,” in *Sequential Monte Carlo Methods in Practice*, Information Science and Statistics, eds. A. Doucet, N. Freitas, and N. Gordon, pp. 177–195, New York: Springer. [358]

Koskelo, J., Jenkins, P. A., Johansen, A. M., and Spanò, D. (2020), *Asymptotic Genealogies of Interacting Particle Systems with an Application to Sequential Monte Carlo*, *The Annals of Statistics*, 48, 560–583. [358,362]

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., Chopin, N. (2015), “On Particle Methods for Parameter Estimation in State-Space Models,” *Statistical Science*, 30, 328–351. [357]

Kitagawa, G., and Sato, S. (2001), “Monte Carlo Smoothing and Self-Organising State-Space Model,” in *Sequential Monte Carlo Methods in Practice*, Information Science and Statistics, eds. A. Doucet, N. Freitas, and N. Gordon, pp. 177–195, New York: Springer. [358]

Koskelo, J., Jenkins, P. A., Johansen, A. M., and Spanò, D. (2020), *Asymptotic Genealogies of Interacting Particle Systems with an Application to Sequential Monte Carlo*, *The Annals of Statistics*, 48, 560–583. [358,362]

Le Gland, F., and Mevel, L. (1997), “Recursive Estimation in HMMs,” in *Proceedings of the IEEE Conference on Decision and Control*, pp. 3468–3473. [357]

Liu, J. S. (1996), “Metropolized Independent Sampling with Comparisons to Rejection Sampling and Importance Sampling,” *Statistics and Computing*, 6, 113–119. [357]

Mongillo, G., and Denève, S. (2008), “Online Learning with Hidden Markov Models,” *Neural Computation*, 20, 1706–1716. [357]

Olsson, J., Cappé, O., Douc, R., and Moulines, E. (2008), “Sequential Monte Carlo Smoothing with Application to Parameter Estimation in Non-linear State Space Models,” *Bernoulli*, 14, 155–179. [358]

Olsson, J., and Douc, R. (2019), “Numerically Stable Online Estimation of Variance in Particle Filters,” *Bernoulli*, 25, 1504–1535. [360]

Olsson, J., and Westerborn, J. (2017), “Efficient Particle-Based Online Smoothing in General Hidden Markov Models: The PaRIS Algorithm,” *Bernoulli*, 23, 1951–1996. [358,361]

Pitt, M. K., and Shephard, N. (1999), “Filtering via Simulation: Auxiliary Particle Filters,” *Journal of the American Statistical Association*, 94, 590–599. [357]

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. [358]