Online Rolling Controlled Sequential Monte Carlo

Liwen Xue, Axel Finke and Adam M. Johansen

August 4, 2025

Abstract

We introduce methodology for real-time inference in general-state-space hidden Markov models. Specifically, we extend recent advances in controlled sequential Monte Carlo (CSMC) methods—originally proposed for offline smoothing—to the online setting via a rolling window mechanism. Our novel online rolling controlled sequential Monte Carlo (ORCSMC) algorithm employs two particle systems to simultaneously estimate twisting functions and perform filtering, ensuring real-time adaptivity to new observations while maintaining bounded computational cost. Numerical results on linear-Gaussian, stochastic volatility, and neuroscience models demonstrate improved estimation accuracy and robustness in higher dimensions, compared to standard particle filtering approaches. The method offers a statistically efficient and practical solution for sequential and real-time inference in complex latent variable models.

Keywords: Sequential Monte Carlo; state-space models; particle filters; controlled SMC; online inference; latent variable models.

1 Introduction

This paper is concerned with inference for general-state-space *hidden Markov models (HMMs)*, a.k.a. *state-space models*. These models are used in diverse applications ranging from control systems to financial analysis, navigation, and biological monitoring. In such applications, efficient and accurate estimation of the hidden states (given noisy and partial observations) is crucial. Three inferential tasks are of particular interest:

1. *Filtering:* characterizing the conditional law of the most recent latent state given all the observations received thus far.
2. *Smoothing:* characterizing the conditional law of all latent states until the current time point given all the observations received thus far.
3. *Marginal-likelihood evaluation:* evaluating the density of all available observations with the states marginalised out. This is typically needed for performing likelihood-based estimation of unknown model parameters.

Such inference is typically *sequential* in the sense that beliefs about the latent states are to be updated whenever new observations become available—often at short intervals; and may even need to be *online* in the sense that both the space and time costs of incorporating one new observation are bounded uniformly in time.

Obtaining the filtering and smoothing distributions and marginal likelihoods involves computing integrals which are typically intractable, except in a limited number of scenarios such as linear Gaussian models—which give rise to the celebrated Kalman filter and related recursions (Kalman, 1960)—and a few others (Vidoni, 1999). Consequently, Monte Carlo methods are the state of the art for general inference in these methods.

Particle filters (PFs) are a class of *sequential Monte Carlo (SMC)* methods adapted to conducting inference in general-state-space HMMs (with a particular focus on online solutions of the filtering problem). They first came to prominence in the form of the *bootstrap particle filter (BPF)* introduced in Gordon et al. (1993). Since then, there have been numerous innovations in the area. Those most critical to the current work are summarised in the next section; see, e.g., Doucet and Johansen (2011) for a survey of approaches to filtering and smoothing, Kantas et al. (2015) for an overview of parameter estimation techniques or Chopin and Papaspiliopoulos (2020) for a book-length introduction to SMC methods including their applications to general-state-space HMMs.

PFs perform poorly in challenging scenarios such as higher-dimensional latent states. To alleviate this problem, Guarniero et al. (2017); Heng et al. (2020) proposed *controlled sequential Monte Carlo (CSMC)* methods (a.k.a., *iterated auxiliary particle filters*) which leverage insights from optimal control theory to improve the efficiency of the PF. Unfortunately, as CSMC must repeatedly browse through the entire observation sequence, it is limited to *offline* inference (i.e., settings in which the full data record is available before inference is conducted).

In this work, we propose a novel method, *online rolling controlled SMC (ORCSMC)*, which aims to extend the benefits of CSMC to online inference. ORCSMC propagates a particle system using a finite-horizon modifications of the CSMC approach. At each time point, aditional simulations temporarily extend this system to the time of the current observation in order to compute efficient approximations of filtering and smoothing distributions and marginal likelihoods in real time. ORCSMC is also applicable in contexts in which storing on processing the entire data set (as required for CSMC) is impractical.

Section 2 reviews existing SMC approaches for filtering and smoothing, as well as CSMC; Section 3 presents the novel ORCSMC method. Section 4 demonstrates the performance of our proposed approach on different models; Section 5 concludes.

Throughout this work, we use the shorthand $[\![m, n]\!] := \{k \in \mathbb{Z} \mid m \leq k \leq n\}$, for $m, n \in \mathbb{Z} \cup \{\infty\}$ with $m \leq n$. If $n \geq 1$, we also write $[\![n]\!] := [\![1, n]\!]$. If m and n are finite, we further write $z_{m:n} := (z_m, z_{m+1}, \dots, z_n)$ and $z^{m:n} := (z^m, z^{m+1}, \dots, z^n)$.

2 Background

This section reviews SMC methods for inference in general-state-space HMMs, with a focus on controlled methods and techniques designed for approximating smoothing distributions.

2.1 General-state-space HMMs

For definiteness, consider an $\mathbb{R}^d$-valued discrete-time Markov process $\{X_t\}_{t\geq 1}$ such that

$$X_1 \sim \mu \quad \text{and} \quad X_t | (X_{t-1} = x_{t-1}) \sim f_t(\,\cdot\,| x_{t-1}),$$

where $t > 1$, "$\sim$" means distributed according to, $\mu(x_1)$ is a probability density function and $f_t(x_t|x_{t-1})$ denotes the probability density associated with moving from $X_{t-1} = x_{t-1}$ at time $t-1$ to $X_t = x_t$ at time t. To simplify the notation, any additional 'static' model parameters parametrising these densities are omitted from the notation.

We are interested in performing inference about $\{X_t\}_{t\geq 1}$ but only have access to $\mathbb{R}^{d'}$-valued observations $\{Y_t\}_{t\geq 1}$, assumed to be conditionally independent given $\{X_t\}_{t\geq 1}$ with densities

$$Y_t | (X_t = x_t) \sim g_t(\,\cdot\,| x_t).$$

The main inference problems for this model class is characterizing the filtering and smoothing densities and marginal likelihoods:

$$\begin{aligned} p(x_t|y_{1:t}) &\propto \int \left[\mu(x_1) g_1(y_1|x_1) \prod_{s=2}^{t} f_s(x_s|x_{s-1}) g_s(y_s|x_s) \right] \mathrm{d}x_{1:t-1}, \\ p(x_{1:t}|y_{1:t}) &\propto \mu(x_1) g_1(y_1|x_1) \prod_{s=2}^{t} f_s(x_s|x_{s-1}) g_s(y_s|x_s), \qquad (1) \\ p(y_{1:t}) &= \int \left[\mu(x_1) g_1(y_1|x_1) \prod_{s=2}^{t} f_s(x_s|x_{s-1}) g_s(y_s|x_s) \right] \mathrm{d}x_{1:t}. \end{aligned}$$

If online inference is required, then the most we can generally hope to achieve is characterizing fixed-dimensional marginals of the smoothing distributions from (1).

2.2 Particle Filtering

The *bootstrap particle filter (BPF)* (Gordon et al., 1993) proceeds by sampling a collection of Monte Carlo samples ('particles') from the initial distribution μ, weighting them in accordance with the likelihood associated with the first observation, y_1, and then—at time t—iteratively sampling from the resulting weighted empirical distribution, extending the sampled paths according to the transition density f_t and weighting them with the likelihood of y_t.

Sampling from the weighted empirical distribution can be decomposed as a selection step, known as *resampling*, in which particles with larger weights tend to produce more offspring followed by a mutation step in which particles move according to the underlying dynamic model. Seen in this way, the algorithm associates particle values at each time with its predecessor and introduces a genealogical relationship allowing us to think about the ancestor of a given particle at any number of generations earlier. Tracing ancestries back in this way is central to the most basic uses of PFs in order to address smoothing problems.

The BPF is effective in many settings, but has a number of limitations which more recent work has sought to overcome. More general sequential importance resampling PFs (see, e.g., Doucet et al., 2000) follow from recognising that the BPF is essentially carrying out a sequence of importance sampling operations and hence that one can adjust the proposal distribution to better take into account the observation at the current time, with a proposal distribution of $p(x_t|x_{t-1}, y_t)$ providing the smallest conditional variance for the importance weights at step t, the appropriate weights being proportional to $p(y_t|x_{t-1})$ in this case.

The *auxiliary particle filter (APF)* approach is a further enhancement of standard PFs. Originally introduced using auxiliary variables, it improves the effectiveness of state estimation by incorporating a look-ahead mechanism into the resampling procedure which predicts the likelihood that particles will be in regions of high probability at the next time step (Pitt and Shephard, 1999; Carpenter et al., 1999). This algorithm can also be understood as a variant of sequential importance resampling (Johansen and Doucet, 2008) which targets a slightly different sequence of distributions and corrects for the discrepancy between those and the filtering distributions with an additional importance sampling correction.

While simple PFs are often able to provide adequate approximations of filtering distributions, more specialised techniques are generally required in smoothing contexts. PFs suffer from limitations such as particle impoverishment in which repeated resampling operations lead to only a small number of distinct state values at early times having descendants at later times and weight degeneracy in which the importance weights have very high variance with only a small number differing substantially from zero; these and related phenomena are particularly problematic when the goal is to approximate smoothing distributions.

2.3 Particle Smoothing: Fixed-lag and Other Schemes

In general, one seeks good sample approximations of the joint distribution $p(x_{1:T}|y_{1:T})$ and its marginals $p(x_t|y_{1:T})$, for $t \in [\![T]\!]$, and in the online context one similarly seeks approximations of $p(x_{1:t}|y_{1:t})$ and its marginals $p(x_s|y_{1:t})$, for $s \in [\![t]\!]$, which can be updated as the tth observation arrives, for $t = 1, \ldots, T$. *Online filtering* as described in the previous section focuses on real-time estimation, while *offline smoothing* involves utilizing the complete set of observations to refine state estimates. *Online smoothing*, as distinct from offline approaches, is concerned with making estimates as new data arrives while retaining a focus on both current and prior states.

Online smoothing in which we obtain good approximations of the smoothing distributions iteratively as we obtain observations at a computational cost which does not grow unboundedly with time is a challenging problem and perhaps the main focus of this work. Often, one may wish to approximate the marginals $p(x_s|y_{1:t})$ for $s \in [\![t]\!]$. However, when t is large, the simple approach (sometimes termed the "smoothing mode of the particle filter") in which one just traces back the histories of the surviving particles at time t to obtain this approximation generally fails also a result of the degeneracy of this sample approximation. Although Kitagawa (2014) demonstrates that one can obtain good estimation if one uses very large numbers of particles in this context, approaches which require less substantial computational resources are needed.

The fixed-lag SMC approach from Kitagawa and Sato (2001) is a practical method for smoothing in dynamic systems, which aims to balance computational efficiency with estimation accuracy. It was introduced to tackle the inefficiency of standard smoothing in scenarios where observations continue to arrive over time. For certain HMMs with 'forgetting' properties, for a sufficiently large lag $L > 1$, observations beyond $s + L$ provide minimal additional information about the trajectory $X_{1:s}$. We may then approximate $p(x_{1:s}|y_{1:t})$ by $p(x_{1:s}|y_{1:\min\{s+L,t\}})$. However, the choice of L involves a bias–variance trade-off: If L is too small, the model's memory has not dissipated, causing bias. Conversely, an excessively large L leads to path degeneracy in the PF and high variance. Olsson et al. (2008) proposes an optimal lag choice of $\lceil c \log(s) \rceil$ under specific mixing conditions, noting that in practice, L values between 20 and 50 often suffice. This method approximates only marginal distributions $p(x_s|y_{1:t})$, not the joint distribution $p(x_{1:t}, y_{1:t})$, and does not converge to the true distribution as $N \to \infty$, but this fixed-lag approach lays the foundation for more sophisticated methods.

Motivated by similar considerations, Lin et al. (2013) incorporate information from subsequent observations into proposal distributions in a principled manner. This can substantially improve performance but limits use in online settings as estimation is delayed until a number of additional observations are received; it also requires careful tuning of approximations of certain conditional distributions associated with the algorithm which is challenging for most realistic models. Doucet et al. (2006) address the first of these issues by developing a related algorithm—amenable to online application—in which a block of recent states are refreshed as each new observation becomes available. But again good approximations of some complex conditional distributions must be obtained by hand in order to implement this algorithm.

Finally, considerable effort has been dedicated to developing Monte Carlo schemes for addressing the smoothing problem. However, these methods tend to either involve a forward-backward structure making them unsuitable for online applications (such as the methods of Briers et al. (2010)), or provide approximations of only single time marginals of the smoothing distribution (e.g., Fearnhead et al., 2010). The PaRIS algorithm of Olsson and Westerborn (2017) and related methods are one promising recent approach but are restricted to approximating additive functionals.

2.4 ψ-APF

Before describing controlled sequential Monte Carlo, the algorithm we seek to adapt to an online context, we introduce the algorithm that sits at its heart which was introduced as the *ψ-auxiliary* *particle filter (ψ-APF)* by Guarniero et al. (2017). Underlying this algorithm is the construction of a class of *twisted* HMMs, parameterized by a sequence $\psi := \{\psi_t\}_{t\geq 1}$ of continuous and positive functions $\psi_t \colon \mathbb{R}^d \to (0,\infty)$. In describing these, it is convenient to introduce the integral notation $\mu(\psi_1) := \int \mu(x_1)\psi_1(x_1)\,\mathrm{d}x_1$ and $f_t(\psi_t)(x_{t-1}) := \int f_t(x_t|x_{t-1})\psi_t(x_t)\,\mathrm{d}x_t$, for $t > 1$. These twisted models adjust the dynamics and observation likelihoods of the original HMM to account for future information via a form of importance weighting.

To simplify notation in the boundary case $t = 1$, we define $f_1(x_1|x_0) := \mu(x_1)$ with the convention that any quantity with time subscript 0 should be ignored. Thus, we can write $\mu(\psi_1) = f_1(\psi_1)(x_0) = f_1(\psi_1)$ which makes the presentation of algorithms simpler.

Specifically, for $t > 1$, the twisted models are:

$$\mu^{\psi}(x_1) = f_1^{\psi}(x_1) := \frac{\mu(x_1)\psi_1(x_1)}{\mu(\psi_1)}, \qquad f_t^{\psi}(x_t|x_{t-1}) := \frac{f_t(x_t|x_{t-1})\psi_t(x_t)}{f_t(\psi_t)(x_{t-1})},$$
$$g_1^{\psi}(x_1) := \frac{g_1(y_1|x_1)\mu(\psi_1)}{\psi_1(x_1)} f_2(\psi_2)(x_1), \qquad g_t^{\psi}(x_t) := \frac{g_t(y_t|x_t)}{\psi_t(x_t)} f_{t+1}(\psi_{t+1})(x_t). \tag{2}$$

Given the sequence ψ, one simply applies a BPF to the twisted model defined by $\{(f_t^{\psi}, g_t^{\psi})\}_{t\geq 1}$. The resulting algorithm, the ψ-APF, is obtained by applying Algorithm 1 recursively for $t = 1, 2, \ldots$

Algorithm 1 ψ-APF(t, ψ_t, H_{t-1})

```
Input: Time index $t \in \mathbb{N}$.
Input: Twisting function $\psi_t$.
Input: Particle system $H_{t-1}$ as in (3), if $t > 1$; if $t = 1$, then $H_0$ is as in (4).
 1: Set $v^n := W_{t-1}^n f_t(\psi_t)(X_{t-1}^n)$, for $n \in [\![N]\!]$.
 2: Set $Z_t^N := Z_{t-1}^N \sum_{n=1}^N v^n$ and self-normalise: $V^n := v^n / \sum_{m=1}^N v^m$, for $n \in [\![N]\!]$.
 3: if $\mathrm{ESS}(V^{1:N}) < \kappa N$ then          ▷ $\mathrm{ESS}(V^{1:N}) := 1/\sum_{n=1}^N (V^n)^2$
 4:     $A_{t-1}^{1:N} \leftarrow \mathrm{RESAMPLE}(V^{1:N})$,
 5:     $V^n \leftarrow 1/N$, for $n \in [\![N]\!]$;
 6: else
 7:     $A_{t-1}^{1:N} \leftarrow (1, \ldots, N)$.
 8: Sample $X_t^n \sim f_t^{\psi}(\cdot\,|X_{t-1}^{A_{t-1}^n})$, for $n \in [\![N]\!]$.
 9: Set $w_t^n := V^n g_t(y_t|X_t^n)/\psi_t(X_t^n)$, for $n \in [\![N]\!]$.
10: Set $Z_t^N \leftarrow Z_t^N \sum_{n=1}^N w_t^n$ and self-normalise: $W_t^n := w_t^n / \sum_{m=1}^N w_t^m$, for $n \in [\![N]\!]$.
Output: Particle system $H_t = (X_t^{1:N}, W_t^{1:N}, A_{t-1}^{1:N}, Z_t^N)$.
```

Algorithm 1 can be viewed as (the tth step of) a form of APF (Pitt and Shephard, 1999). It also generalises the BPF (with $\psi_t \equiv 1$, for $t \geq 1$) and the fully-adapted APF (with $\psi_t := g_t(y_t|\cdot)$, for $t \geq 1$). Some comments about Algorithm 1 are in order.

Firstly, to keep the notation concise throughout, we have defined the *particle system* generated by the ψ-APF at time t as

$$H_t = (X_t^{1:N}, W_t^{1:N}, A_{t-1}^{1:N}, Z_t^N), \tag{3}$$

where $X_t^n \in \mathbb{R}^d$ is the nth particle, $W_t^n \in [0,1]$ is its self-normalised weight, $A_{t-1}^n \in [\![N]\!]$ the index of its ancestor particle which is generated by the resampling procedure (in other words, $X_{t-1}^{A_{t-1}^n}$ is nth resampled particle from time $t-1$), and Z_t^N is used to construct an estimate of the normalising constant of the twisted HMM; specifically, if the ψ-APF is iterated up to any finite final time T, in the sense that the twisting functions include information from observations only over $t \in [\![T]\!]$, Z_T^N is an unbiased estimate of $p(y_{1:T})$ (Guarniero et al., 2017, Proposition 1). Actually, as written, Z_t^N provides an unbiased estimate of $p(y_{1:t})$ for *any* t, but the variance of this estimate will typically be large if the twisting functions have incorporated information from subsequent observations, e.g., from y_{t+1}. For the particle system at time $t = 0$, we use the convention

$$W_0^1 = \cdots = W_0^N \equiv 1/N, \quad \text{and} \quad Z_0^N := 1. \tag{4}$$

The values of $X_0^{1:N}$ and $A_0^{1:N}$ can be arbitrary as they are not used by the algorithm.

Secondly, Lines 1–2 update the time-$(t-1)$ particle weights (and normalising-constant estimate) to account for the second term, $f_t(\psi_t)(\,\cdot\,)$, in (2) (recall that this term is simply the scalar $\mu(\psi_1)$ if $t = 1$). This differs from the presentation of the ψ-APF in Guarniero et al. (2017); Heng et al. (2020) which views this update as happening at the end of Step $(t-1)$ rather than at the beginning of Step t but will be convenient for the online methodology developed in the next section. Specifically, this formulation allows us to carry out the $(t-1)$th step of the algorithm even if ψ_t is not yet known. However, we stress that the difference is purely presentational, i.e., combining Lines 1–2 from Step t with Lines 9–10 from Step $(t-1)$ results exactly in the ψ-APF as presented in Guarniero et al. (2017); Heng et al. (2020).

Thirdly, Lines 3–7 perform adaptive resampling Kong et al. (1994); Liu and Chen (1995) based on the weights $V^{1:N}$. That is, whenever the *effective sample size (ESS)* drops below κN, for some user-defined threshold $0 < \kappa \leq 1$, $\textsc{resample}(V^{1:N})$ samples the ancestor indices $A_{t-1}^{1:N}$ via some suitable resampling scheme based on a set of weights $V^{1:N}$; numerous resampling schemes have been proposed over the years, in our numerical experiments we use residual–multinomial resampling (Liu and Chen, 1998).

Finally, there exists an optimal sequence ψ which yields zero-variance estimates of final-time normalising constants Z_T^N (Guarniero et al., 2017, Proposition 2), but this sequence is generally not computable or usable in practice. We are constrained in the models to which this approach can be applied and the twisting functions that can be employed by the fact that we must be able to sample from f_t^ψ for each t and also to evaluate the corresponding g_t^ψ; one surprisingly common setting in which this is possible is that in which the transition densities are Gaussian and the twisting functions employed are exponentials of a quadratic form.

2.5 (Offline) Controlled SMC

The *controlled sequential Monte Carlo (CSMC)* algorithm—also known as *iterated APF*–introduced by Guarniero et al. (2017); Heng et al. (2020), excels in offline contexts by leveraging a complete data set $y_{1:T}$ to provide accurate approximations of smoothing distributions of the form $p(x_{1:T}|y_{1:T})$ and the marginal likelihood $p(y_{1:T})$ for a class of models.

A key insight underlying CSMC is that the problem of incorporating the influence of future observations can be cast as a function approximation problem at each time point. In particular, Heng et al. (2020) cast this problem as one of optimal control in which one seeks to learn the policy. As summarised in Algorithm 2, the idea is to repeatedly apply the ψ-APF (from time 1 to time T), learning a better sequence of twisting functions during each iteration to use it in the next.

The precise implementation of Line 2 in Algorithm 3 varies between Guarniero et al. (2017) and Heng et al. (2020). Both methodologies converge on solving an optimization problem which is designed to refine the current policy towards an optimal one, utilizing information obtained from current sample values. Our specific approach is detailed in Section 4.1 below.

Empirical results have demonstrated that CSMC performs significantly better than the standard BPF in challenging scenarios, and can significantly outperform the fully-adapted APF, often regarded as a gold standard in the online setting, at least in some simple models for which it can be implemented.

Algorithm 2 controlled sequential Monte Carlo (CSMC)

```
Input: Initial particle system $H_0$ as in (4).
1: Set $\psi_1 = \ldots = \psi_{T+1} \equiv 1$.
2: for $k = 1, \ldots, K$ do
3:     for $t = 1, \ldots, T$ do
4:         sample $H_t \leftarrow \psi\text{-APF}(t, \psi_t, H_{t-1})$,          ▷ Algorithm 1
5:     for $t = T, \ldots, 1$ do
6:         set $\psi_t \leftarrow \text{LEARN-}\psi(t, \psi_{t+1}, H_t)$.          ▷ Algorithm 3
Output: Approximations $p(y_{1:T}) \approx Z_T^N$ and $p(x_{1:T}|y_{1:T}) \approx \sum_{n=1}^N W_T^n \delta_{X_{1:T}^{(n)}}$ (based on $H_{1:T}$ available at the end of the algorithm), where $X_{1:T}^{(n)}$ denotes the $n$th particle lineage at time $T$, recursively defined as $X_{1:t}^{(n)} := (X_{1:(t-1)}^{(A_{t-1}^n)}, X_t^n)$, with initial condition $X_1^{(n)} := X_t^n$.
```

Algorithm 3 LEARN-$\psi(t, \psi_{t+1}, H_t)$

```
Input: Time index $t \in \mathbb{N}$.
Input: Twisting function $\psi_{t+1}$.
Input: Particle system $H_t$ as in (3).
1: Set $\psi_t^n \leftarrow g_t(y_t|X_t^n) f_{t+1}(\psi_{t+1})(X_t^n)$, for $n \in [\![N]\!]$.
2: Choose $\psi_t$ on the basis of $X_t^{1:N}$ and $\psi_t^{1:N}$.
Output: $\psi_t$.
```

3 Online Rolling Controlled SMC

Unfortunately, the application of CSMC in online settings is limited because it requires processing of the whole data sequence. Although warm-starting would be possible and one could simply re-run the entire algorithm as observations become available, the cost of processing each new observation would grow with the length of the data sequence, making the computational complexity of such an approach prohibitive.

In this section, we present an online CSMC scheme—termed *online rolling controlled SMC (ORCSMC)* and summarised in Algorithm 4—that transfers the high accuracy of CSMC to real-time inference through a rolling-window, dual-filter design scenario. Specifically, the underlying strategy is to consider two concurrent ψ-APFs at time t:

1. The *learning filter* (Lines 6–11) (whose particle systems are indicated by a '$\sim$'-accent) repeatedly runs over the rolling window $[\![t_0, t]\!]$, where $t_0 := \max\{1, t - L + 1\}$, to obtain approximations, $\psi_{t_0:t}$, of optimal twisting functions. These updates are warm-started by initialising $\psi_{t_0:(t-1)}$ to the twisting functions obtained based on the learning filter at time $t-1$; ψ_t is initialised to the unit function.
2. The *estimation filter* (Line 13) runs over the time steps $s = t_0, \ldots, t$ using the latest values of the twisting functions $\psi_{t_0:t}$ obtained in Line 9. Note that it thus overwrites the values of $H_{t_0:(t-1)}$ it had generated at time $t-1$. The filtering estimates output by the algorithm at time t are then computed using the quantities $X_t^{1:N}$, $W_t^{1:N}$, $A_{t-1}^{1:N}$ and Z_t^N which are assumed to come from the (most recent values of the) particle systems $H_{1:t}$ available at time t. Specifically, Z_t^N obtained from H_t in Line 14 is an unbiased estimate of $p(y_{1:t})$. If the user requires filtering estimates only at *some* times $\mathcal{T} \subseteq [\![T]\!]$, then the above-mentioned 'overwriting' can be skipped at other times. More precisely, the sampling operation in Line 13 then (a) only needs to be fully performed if $t \in \mathcal{T} \cup \{T\}$; (b) can be skipped entirely if $t < L$ and $t \notin \mathcal{T}$; (c) only needs to be carried out for $s = t_0 = t - L + 1$, at all other times t.

Line 5 of Algorithm 4 is marked as 'optional' because it is only needed to ensure that the memory cost is bounded in t. Of course, this implicitly assumes that we are not interested in

Algorithm 4 online rolling controlled SMC (ORCSMC)

```
Input: Time horizon $T \in \mathbb{N} \cup \{\infty\}$.
Input: Initial particle systems $H_0 = \tilde{H}_0$ whose components are defined in (4).
 1: Set $\psi_1 = \ldots = \psi_{T+1} \equiv 1$.
 2: for $t = 1, \ldots, T$ do
 3:     set $t_0 := \max\{1, t - L + 1\}$,
 4:     if $t - 1 > L$ then
 5:         discard $\psi_{t-L-1}$, $\tilde{H}_{t-L-1}$, $H_{t-L-1}$,                ▷ optional
 6:     sample $\tilde{H}_t \leftarrow \psi$-APF$(t, \psi_t, \tilde{H}_{t-1})$;                ▷ Algorithm 1
 7:     for $k = 1, \ldots, K$ do
 8:         for $s = t, t-1, \ldots, t_0$ do
 9:             set $\psi_s \leftarrow$ LEARN-$\psi(s, \psi_{s+1}, \tilde{H}_s)$,                ▷ Algorithm 3
10:         for $s = t_0, \ldots, t$ do
11:             sample $\tilde{H}_s \leftarrow \psi$-APF$(s, \psi_s, \tilde{H}_{s-1})$;                ▷ Algorithm 1
12:     for $s = t_0, \ldots, t$ do
13:         sample $H_s \leftarrow \psi$-APF$(s, \psi_s, H_{s-1})$;                ▷ Algorithm 1
14:     Output: Approximations $p(y_{1:t}) \approx Z_t^N$ and $p(x_{1:t}|y_{1:t}) \approx \sum_{n=1}^N W_t^n \delta_{X_{1:t}^{(n)}}$ (based on $H_{1:t}$).
    Here, $X_{1:t}^{(n)} := (X_{1:(t-1)}^{(A_{t-1}^n)}, X_t^n)$ is recursively defined with initial condition $X_1^{(n)} := X_t^n$.
```

outputting approximations of (marginals of) $p(x_{1:(t-L-1)}|y_{1:t})$ in Line 14.

This framework is roughly as flexible as the (offline) CSMC algorithm which it extends; see Section 4.1 for details of the implementation employed in our empirical study.

4 Experiments

In this section, we illustrate that a number of models used to illustrate the behaviour of offline algorithms by Guarniero et al. (2017); Heng et al. (2020) also admit efficient online inference using the controlled approaches developed in this paper.

4.1 Setup

Throughout all experiments, we assume homogeneous linear-Gaussian dynamics with $\mu(x_1) := \mathcal{N}(x_1; m, \Sigma)$ and $f_t(x_t|x_{t-1}) := \mathcal{N}(x_t; Ax_{t-1}, B)$, for $m \in \mathbb{R}^d$, $A, B, \Sigma \in \mathbb{R}^{d \times d}$ with B and Σ positive definite.

To iteratively optimize the twisting functions ψ_t within both CSMC and ORCSMC, we use the *approximate dynamic programming (ADP)* approach from Heng et al. (2020) to implement Line 2 of Algorithm 3. This entails minimising a squared error on the logarithmic scale based on the terms ψ_t^n from Line 1 of Algorithm 3. For this purpose, our experiments adopt a quadratic function class for optimization: the family of functions of the form $\psi_t(x_t) = \exp(x_t^T \mathsf{A}_t x_t + \mathsf{b}_t^T x_t + \mathsf{c}_t)$ for $\mathsf{A}_t \in \mathbb{S}_d = \{\mathsf{A} \in \mathbb{R}^{d \times d} : \mathsf{A} = \mathsf{A}^T\}$, $\mathsf{b}_t \in \mathbb{R}^d$ and $\mathsf{c}_t \in \mathbb{R}$ and x_t is a multivariate state vector. By treating the negative logarithm of the ADP target values as the dependent variable, we transform the optimization into a linear least squares problem. The coefficients A_t and b_t derived from this regression then directly forms the parameters of a Gaussian density, which serves as our optimized policy, defining the learned ψ_t. To ensure that the number of parameters of the twisting functions grows only linearly with d, we restrict the matrices A_t to be diagonal.

Throughout all our experiments, both CSMC and ORCSMC use $K = 5$ iterations for learning the twisting functions. We focus in particular on the estimates Z_T^N of the final-time normalising constant $Z_T = p(y_{1:T})$ as these are well known to be a good indicators of the approximation quality of PFs. All results are based on 100 independent replicates.

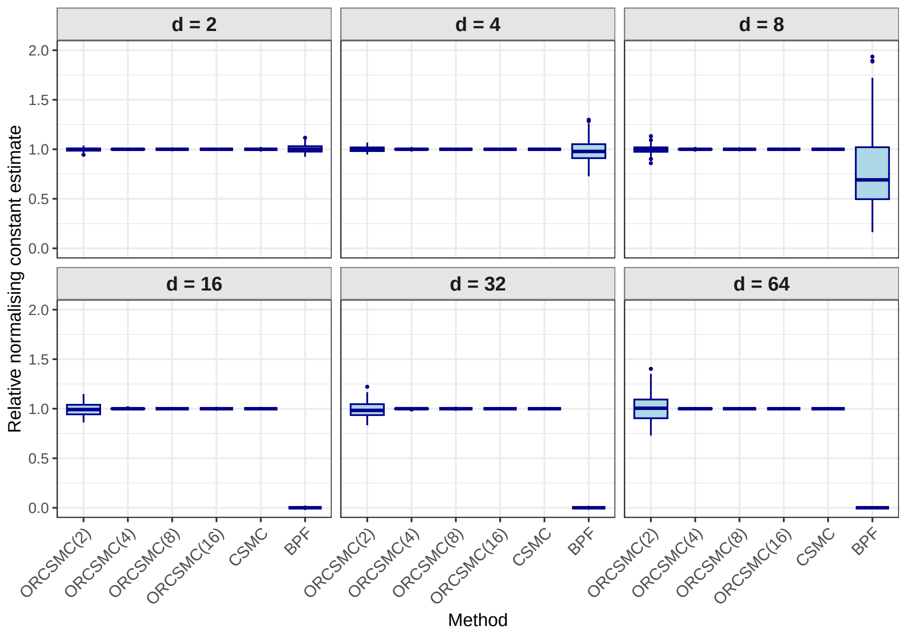


Figure 1: Relative normalising constant estimates for the diagonal Gaussian model obtained via ORCSMC (where parenthetical numbers indicate lag), CSMC and BPF.

4.2 Linear-Gaussian Model

To illustrate the behavior of ORCSMC, we begin with a simple linear-Gaussian toy model in which $g_t(\cdot|x_t) = \mathcal{N}(\cdot; Cx_t, D)$, where $m = \mathbf{0}_d$ is the zero vector in $\mathbb{R}^d$, and $\Sigma = B = C = D = I_d$, where I_d is the $(d \times d)$-identity matrix. This model admits exact inference via the Kalman filter and hence provides a convenient testbed.

Throughout, we fix $T = 100$ and employ $N = 1000$ particles for inference within ORCSMC. We also consider state dimensions $d \in \{2, 4, 8, 16, 32, 64\}$ and lags $L \in \{2, 4, 8, 16\}$. To balance the computational cost, where comparisons are conducted, we use $N = 320\,000$ particles for the BPF and $N = 14\,000$ for CSMC. Since $Z_T = p(y_{1:T})$ is analytically tractable in this model, we consider the more meaningful *relative* estimates of the normalising constant, Z_T^N/Z_T, throughout.

We consider two different linear Gaussian models to explore the impact of restricting the class of twisting functions to a diagonal class, where $\alpha := 0.415$:

1. in the *diagonal case,* we set $A = \alpha I_d$;
2. in the *non-diagonal case,* we follow Guarniero et al. (2017), in setting $[A]_{ij} = \alpha^{|i-j|+1}$, for $(i, j) \in [\![d]\!]^2$.

4.2.1 Diagonal Case

We first consider the case that the diagonal structure of the transition matrix $A = 0.415 I_d$ is compatible with the form of the twisting function ψ_t (recall that we restrict the parameter A_t of ψ_t to be a diagonal matrix).

Figure 1 compares the normalising constant estimates obtained by ORCSMC, the BPF and the (offline) CSMC algorithm.

4.2.2 Non-diagonal Case

We now consider the case that the transition matrix A has the above-mentioned non-diagonal structure previously used in Guarniero et al. (2017), which is less favourable to the diagonal twisting functions which we employ.

Figure 2 illustrates that ORCSMC outperforms the BPF across nearly all dimensions in terms of both accuracy and stability of the normalising-constant estimates.

While CSMC exhibits some improvement over ORCSMC, this advantage is marginal if a sufficient lag length is used by the latter, and is expected as offline methods inherently offer certain benefits over their online counterparts.

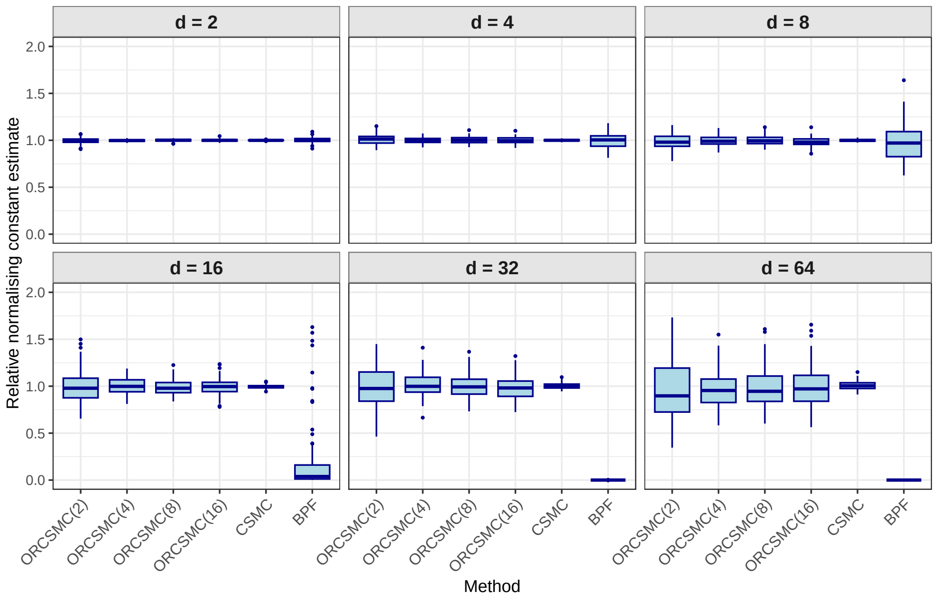


Figure 2: Relative normalising constant estimates for the non-diagonal Gaussian model obtained via ORCSMC (where parenthetical numbers indicate lag), CSMC and BPF.

Figure 3 indicates that the approximation error of ORCSMC decays with the lag; and this finding appears to hold across all considered dimensions.

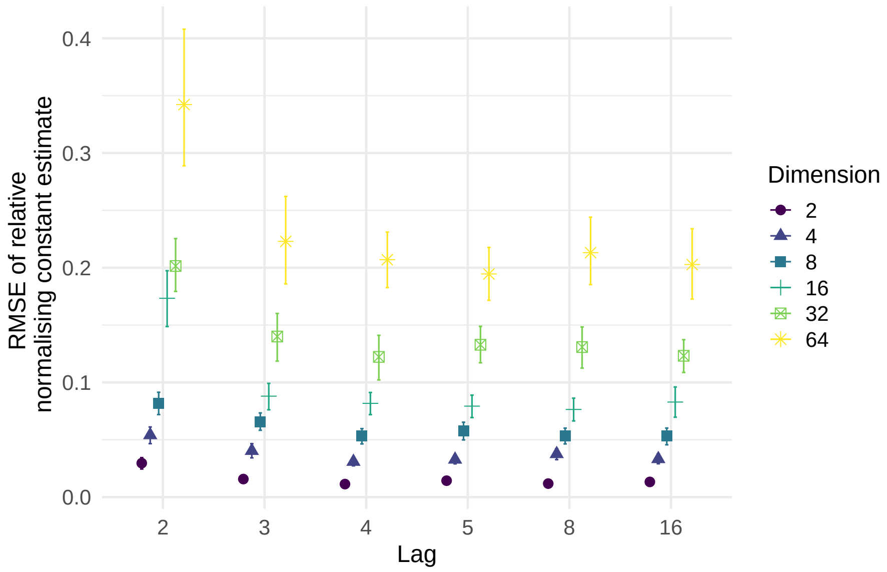


Figure 3: Root mean-square error (RMSE), computed across the 100 independent replicates, of the normalising constant estimates for the non-diagonal Gaussian model in different dimensions.

Figure 4 investigates how accurately ORCSMC approximates the marginal smoothing distribution $p(x_t|y_{1:T})$ at time t. Specifically, at each time t, the true mean and standard deviation of the first coordinate of the state variable, computed using a Kalman filter–smoother, are used to standardize the empirical distribution obtained from the particles generated by ORCSMC. The resulting standardized empirical distribution is then compared with the standard normal distribution. This plot reveals a high degree of concordance between these two distributions, indicating that the algorithm is capable of providing accurate smoothing results.

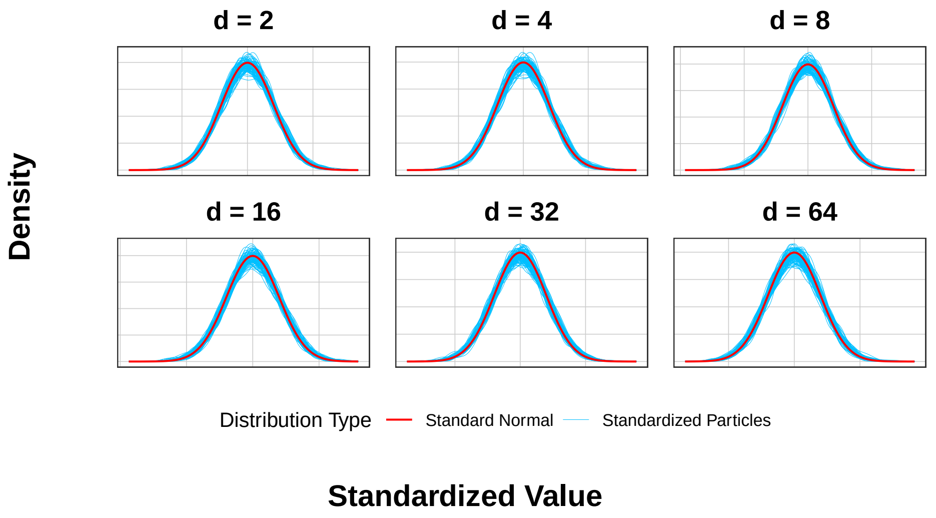


Figure 4: The distribution comparison between the standardized first coordinate marginals of the empirical distribution and the optimal smoother in the non-diagonal Gaussian model at each time t—obtained from a single run of ORCSMC for each value of d.

Figure 5 compares the empirical marginal cumulative distribution function (CDF) $\hat{F}_{t,j}$ to the true Gaussian CDF $F_{t,j}$ (of $p(x_t|y_{1:T})$) at times $t \in \{1, T/2, T\}$. For each state coordinate $j \in [\![d]\!]$, this comparison is made using the L_1 distance between their cumulative distribution functions, which coincides with the Wasserstein-1 distance between their distributions via the Kantorovich–Rubinstein duality:

$$L_1(t,j) = \int_{-\infty}^{\infty} \left|\hat{F}_{t,j}(x) - F_{t,j}(x)\right| \mathrm{d}x.$$

Thus, $L_1(t,j)$ represents the Wasserstein-1 distance between the empirical distribution function of the jth coordinate of the particle approximation of $p(x_t|y_{1:T})$ and its true (Gaussian) distribution at time t. The results show stability over time; the average marginal L_1 error, calculated as the mean of $L_1(t,j)$ across all state coordinates $j \in [\![d]\!]$, remains stable for all t, which indicates accurate marginal smoothing performance across coordinates.

4.3 Stochastic Volatility Model

We consider a simple univariate stochastic volatility model as in Kim et al. (1998), $\mu(\cdot) = \mathcal{N}(\cdot; 0, \sigma^2/(1-\alpha^2))$, $f_t(\cdot|x_t) = \mathcal{N}(\cdot; \alpha x_t, \sigma^2)$ and $g_t(\cdot|x_t) = \mathcal{N}(\cdot; 0, \beta^2 \exp(x_t))$ for $\alpha \in (0,1)$, $\beta > 0$ and $\sigma^2 > 0$. The parameters were estimated from weekday close exchange rates data from 1/10/81 to 28/6/85 ($T = 945$ observations) by taking the approximate posterior mode (under a weakly informative prior) from Guarniero et al. (2017, Figure 5) giving $\alpha = 0.986, \sigma = 0.13$ and $\beta = 0.69$; these values are also close to the MLE provided in the same paper. We run the ORCSMC with $N = 200$ particles.

The experiment shows that even in this non-Gaussian, strongly heteroskedastic setting, ORCSMC is capable of producing stable marginal-likelihood estimates. As the lag L increases, the variance of (the logarithm of) Z_T^N falls substantially, demonstrating its robustness for stochastic volatility models.

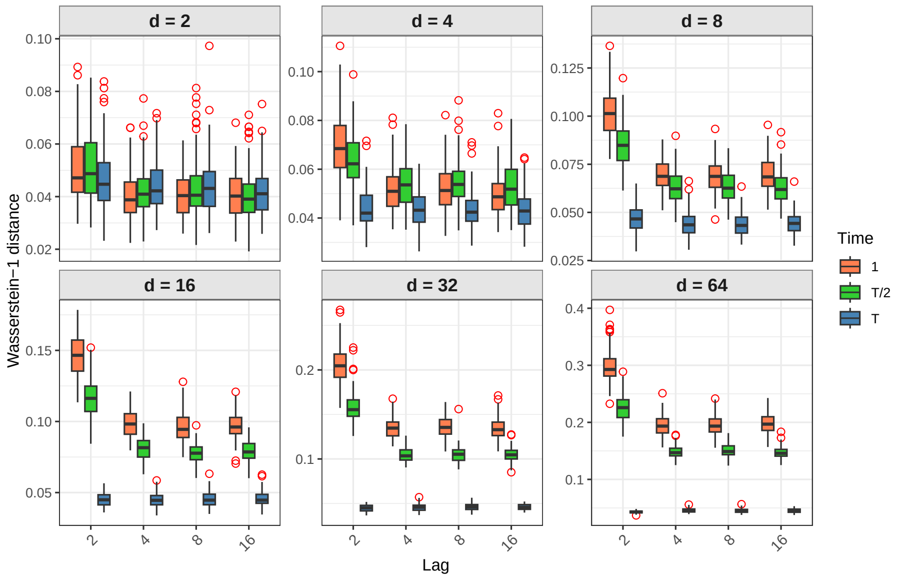


Figure 5: The Wasserstein-1 distance between empirical and exact smoothing distributions in the non-diagonal Gaussian model.

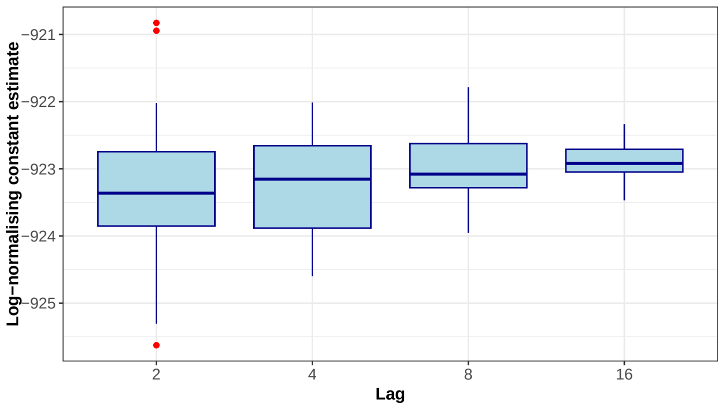


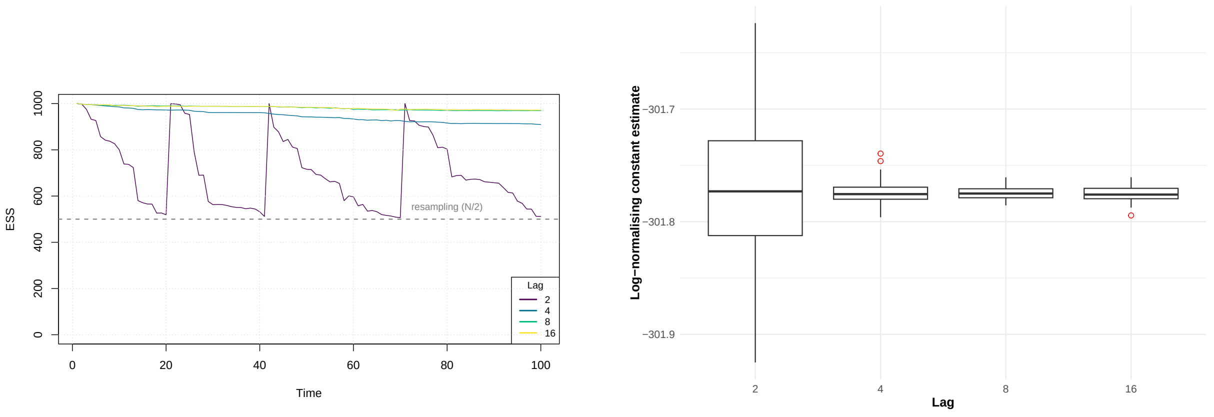
Figure 6: The normalising constant estimates for the stochastic volatility model.

4.4 Neuroscience Model

Finally, we evaluate ORCSMC using a neuroscience-inspired model, following the setup in Heng et al. (2020). This defines a time-homogeneous state-space model on $\mathbb{R}$. At each time step t, the observation $y_t \in [0, M]$ represents the number of activated neurons out of $M = 50$ repeated experiments. The observation likelihood for the univariate case is given by $g_t(y_t|x_t) = \mathrm{Bin}(y_t; M, \kappa(x_t))$, where $\kappa(\cdot)$ is the logistic link function, defined as $\kappa(z) := 1/(1 + \exp(-z))$. The dynamics are specified as $\mu(\cdot) := \mathcal{N}(\cdot\,; 0, 1)$ and $f_t(\cdot\,|x_{t-1}) := \mathcal{N}(\cdot\,; \alpha x_{t-1}, \sigma^2)$, for $\alpha = 0.99$ and $\sigma^2 = 0.11$.

To allow us to also investigate the behaviour in higher dimensions, we extend this model to a multivariate setting in which both the states $X_t := X_{t,1:d}$ and observations $Y_t := Y_{t,1:d}$ are d-dimensional. To that end, we set $m = \mathbf{0}_d, \Sigma = I_d, A = \alpha I_d, B = \sigma^2 I_d$. The observation densities are $g_t(y_t|x_t) = \prod_{j=1}^{d} \mathrm{Bin}(y_{t,j}; M, \kappa(x_{t,j}))$.

Figures 7 and 8 illustrate the performance of ORCSMC in this model in terms of the evolution of the ESS over time and the stability of log-normalising constant estimates.



(a) Evolution of ESS over time. (b) Log-normalising constant estimates.

Figure 7: Results for univariate binomial model for various lags.

5 Discussion

We have presented an approach for performing online inference in a class of state-space models which exploits adaptive guiding methods. It extents the applicability of CSMC methods to settings in which one wishes to approximate filtering or smoothing distributions, or associated normalising constants, online as observations become available.

Proof-of-concept numerical results illustrate the performance of the method on some linear Gaussian toy models, a stochastic volatility model and a model inspired by a problem in neuroscience. Within the family of models with Gaussian transitions for which the particular numerical method used herein might find use lie many models in which the underlying dynamics are obtained from the time discretisation of an underlying stochastic differential equation (see Guarniero (2017) for the case of simple Euler–Maruyama discretisations or Huang et al. (2025) for approaches based on splitting schemes and diffusion bridges) and the approach developed here should allow good *online* inference for those models.

As with all CSMC methods in the literature, the breadth of applicability of the method depends upon the class of models which can be handled. While the randomization approach of Bon et al. (2022) provides one avenue to extend the applicability of methods, and exploiting broader conjugacy properties than those of Gaussian families (such as those described in Vidoni (1999)) would be feasible, there is scope for further development in this area.

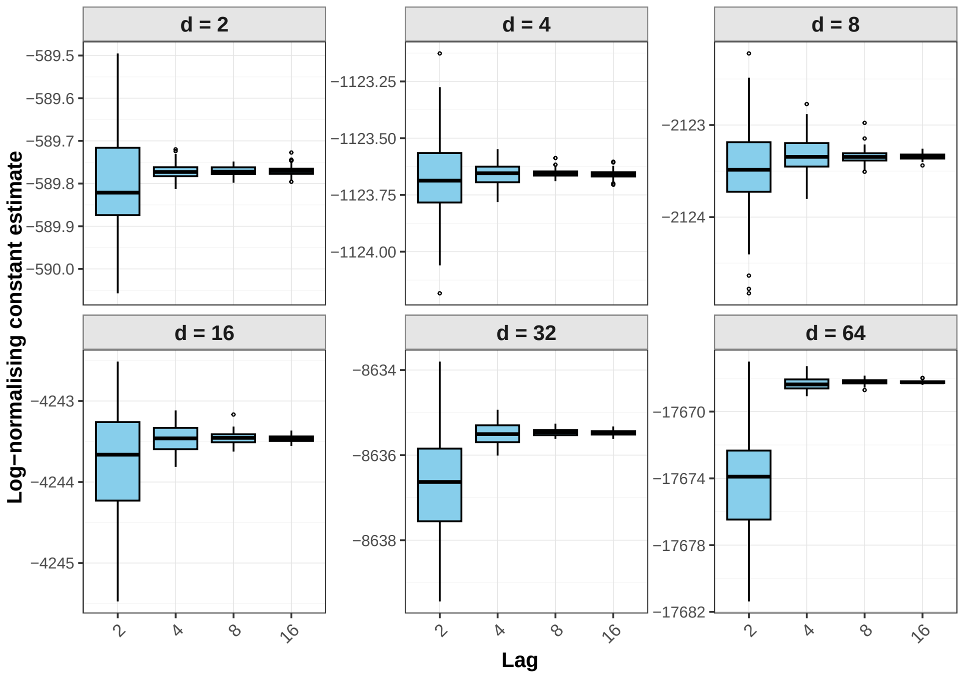


Figure 8: Log-normalising constant estimates obtained from ORCSMC (with different lags) for the multivariate binomial model in different dimensions.

Acknowledgements

We are grateful to the Scientific Computing Research Technology Platform (SCRTP) at the University of Warwick for the provision of computational resources. LX acknowledges support from China Scholarship Council via grant number 202208060348. AMJ acknowledges support from the United Kingdom Engineering and Physical Sciences Research Council (EPSRC; grants EP/R034710/1 and EP/T004134/1) and from United Kingdom Research and Innovation (UKRI) via grant number EP/Y014650/1, as part of the ERC Synergy project OCEAN.
Data availability: All data and code used in this paper can be obtained from `https://github.com/Sempreteamo/orc.smc/tree/main`.

References

Bon, J. J., Drovandi, C., and Lee, A. (2022), "Monte Carlo twisting for particle filters," e-print 2208.04288, arXiv.

Briers, M., Doucet, A., and Maskell, S. (2010), "Smoothing algorithms for state space models," *Annals of the Institute of Statistical Mathematics*, 62, 61–89.

Carpenter, J., Clifford, P., and Fearnhead, P. (1999), "An improved particle filter for non-linear problems," *IEE Proceedings on Radar, Sonar and Navigation*, 146, 2–7.

Chopin, N. and Papaspiliopoulos, O. (2020), *An introduction to sequential Monte Carlo*, Springer.

Doucet, A., Briers, M., and Sénécal, S. (2006), "Efficient block sampling strategies for sequential Monte Carlo methods," *Journal of Computational and Graphical Statistics*, 15, 693–711.

Doucet, A., Godsill, S., and Andrieu, C. (2000), "On sequential simulation-based methods for Bayesian filtering," *Statistics and Computing*, 10, 197–208.

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

Fearnhead, P., Wyncoll, D., and Tawn, J. (2010), “A sequential smoothing algorithm with linear computational cost,” *Biometrika*, 97, 447–464.

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

Guarniero, P. (2017), “The iterated auxiliary particle filter and applications to state space models and diffusion processes.” PhD thesis, University of Warwick.

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

Heng, J., Bishop, A. N., Deligiannidis, G., Doucet, A., et al. (2020), “Controlled sequential Monte Carlo,” *Annals of Statistics*, 48, 2904–2929.

Huang, S., Everitt, R., Tamborrino, M., and Johansen, A. M. (2025), “Inference for diffusion processes via controlled sequential Monte Carlo and splitting schemes,” e-print 2507.14535, arXiv.

Johansen, A. M. and Doucet, A. (2008), “A note on the auxiliary particle filter,” *Statistics and Probability Letters*, 78, 1498–1504.

Kalman, R. (1960), “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, 82, 35–42.

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J. M., and Chopin, N. (2015), “On particle methods for parameter estimation in general state-space models,” *Statistical Science*, 30, 328–351.

Kim, S., Shephard, N., and Chib, S. (1998), “Stochastic volatility: likelihood inference and comparison with ARCH models,” *The Review of Economic Studies*, 65, 361–393.

Kitagawa, G. (2014), “Computational aspects of sequential Monte Carlo filter and smoother,” *Annals of the Institute of Statistical Mathematics*, 66, 443–471.

Kitagawa, G. and Sato, S. (2001), “Monte Carlo smoothing and self-organising state-space model,” in *Sequential Monte Carlo Methods in Practice*, Springer, pp. 177–195.

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

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

Liu, J. S. and Chen, R. (1995), “Blind deconvolution via sequential imputations,” *Journal of the American Statistical Association*, 90, 567–576.

— (1998), “Sequential Monte Carlo methods for dynamic systems,” *Journal of the American Statistical Association*, 93, 1032–1044.

Olsson, J., Cappé, O., Douc, R., and Moulines, E. (2008), “Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models,” *Bernoulli*, 14, 155–179.

Olsson, J. and Westerborn, J. (2017), “Efficient particle-based online smoothing in general hidden Markov models: the PaRIS algorithm,” *Bernoulli*, 23, 1951–1996.

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

Vidoni, P. (1999), “Exponential family state space models based on a conjugate latent process,” *Journal of the Royal Statistical Society B*, 61, 213–221.