Particle Gibbs with Ancestor Sampling

Fredrik Lindsten
Div. of Automatic Control
Linköping University
lindsten@isy.liu.se

Michael I. Jordan
Depts. of EECS and Statistics
University of California, Berkeley
jordan@cs.berkeley.edu

Thomas B. Schön
Dept. of Information Technology
Uppsala University
thomas.schon@it.uu.se

January 6, 2014

Abstract

Particle Markov chain Monte Carlo (PMCMC) is a systematic way of combining the two main tools used for Monte Carlo statistical inference: sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC). We present a novel PMCMC algorithm that we refer to as particle Gibbs with ancestor sampling (PGAS). PGAS provides the data analyst with an off-the-shelf class of Markov kernels that can be used to simulate the typically high-dimensional and highly autocorrelated state trajectory in a state-space model. The ancestor sampling procedure enables fast mixing of the PGAS kernel even when using seemingly few particles in the underlying SMC sampler. This is important as it can significantly reduce the computational burden that is typically associated with using SMC. PGAS is conceptually similar to the existing PG with backward simulation (PGBS) procedure. Instead of using separate forward and backward sweeps as in PGBS, however, we achieve the same effect in a single forward sweep. This makes PGAS well suited for addressing inference problems not only in state-space models, but also in models with more complex dependencies, such as non-Markovian, Bayesian nonparametric, and general probabilistic graphical models.

1 Introduction

Monte Carlo methods are one of the standard tools for inference in statistical models as they, among other things, provide a systematic approach to the problem of computing Bayesian posterior probabilities. Sequential Monte Carlo (SMC) [1,2] and Markov chain Monte Carlo (MCMC) [3,4] methods in particular have found application to a wide range of data analysis problems involving

complex, high-dimensional models. These include state-space models (SSMs) which are used in the context of time series and dynamical systems modeling in a wide range of scientific fields. The strong assumptions of linearity and Gaussianity that were originally invoked for SSMs have indeed been weakened by decades of research on SMC and MCMC.

These methods have not, however, led to a substantial weakening of a further strong assumption, that of Markovianity. It remains a major challenge to develop efficient inference algorithms for models containing a latent stochastic process which, in contrast with the state process in an SSM, is non-Markovian. Such non-Markovian latent variable models arise in various settings, either from direct modeling or via a transformation or marginalization of an SSM. We discuss this further in Section 6; see also [5, Section 4].

In this paper we present a new tool in the family of Monte Carlo methods which is particularly useful for inference in SSMs and, importantly, in non-Markovian latent variable models. However, the proposed method is by no means limited to these model classes. We work within the framework of particle MCMC (PMCMC) [6] which is a systematic way of combining SMC and MCMC, exploiting the strengths of both techniques. More specifically, PMCMC samplers make use of SMC to construct efficient, high-dimensional MCMC kernels with certain invariance properties. These kernels can then be used as off-the-shelf components in MCMC algorithms and other inference strategies relying on Markov kernels, such as Markovian stochastic approximation methods. PMCMC has, in a relatively short period of time, found many applications in areas such as hydrology [7], finance [8], systems biology [9], and epidemiology [10], to mention a few.

Our method builds on the particle Gibbs (PG) sampler proposed by [6]. In PG, the aforementioned Markov kernel is constructed by running an SMC sampler in which one particle trajectory is set deterministically to a reference trajectory that is specified a priori. After a complete run of the SMC algorithm, a new trajectory is obtained by selecting one of the particle trajectories with probabilities given by their importance weights. The effect of the reference trajectory is that the resulting Markov kernel leaves its target distribution invariant, regardless of the number of particles used in the underlying SMC algorithm.

However, PG suffers from a serious drawback, which is that the mixing of the Markov kernel can be very poor when there is path degeneracy in the underlying SMC sampler [5,11]. Unfortunately, path degeneracy is inevitable for high-dimensional problems, which significantly reduces the applicability of PG. This problem has been addressed in the generic setting of SSMs by adding a backward simulation step to the PG sampler, yielding a method denoted as PG with backward simulation (PGBS) [12,13]. It has been found that this considerably improves mixing, making the method much more robust to a small number of particles as well as growth in the size of the data [5,11-13].

Unfortunately, however, the application of backward simulation is problematic for models with more intricate dependencies than in SSMs, such as non-Markovian latent variable models. The reason is that we need to consider com-

plete trajectories of the latent process during the backward simulation pass (see Section 6 for details). The method proposed in this paper, which we refer to as particle Gibbs with ancestor sampling (PGAS), is geared toward this issue. PGAS alleviates the problem with path degeneracy by modifying the original PG kernel with a so called ancestor sampling step, thereby achieving the same effect as backward sampling, but without an explicit backward pass.

The PGAS Markov kernel is constructed in Section 3, extending the preliminary work that we have previously published in [14]. It is also illustrated how ancestor sampling can be used to mitigate the problems with path degeneracy which deteriorates the performance of PG. In Section 4 we establish the theoretical validity of the PGAS approach, including a novel uniform ergodicity result. We then show specifically how PGAS can be used for inference and learning of SSMs and of non-Markovian latent variable models in Sections 5 and 6, respectively. The PGAS algorithm is then illustrated on several numerical examples in Section 7. As part of our development, we also propose a truncation strategy specifically for non-Markovian models. This is a generic method that is also applicable to PGBS, but, as we show in the simulation study in Section 7, the effect of the truncation error is much less severe for PGAS than for PGBS. Indeed, we obtain up to an order of magnitude increase in accuracy in using PGAS when compared to PGBS in this study. Finally, in Section 8 we conclude and point out possible directions for future work.

2 Sequential Monte Carlo

Let $\gamma_{\theta,t}(x_{1:t})$, for $t=1,\ldots,T$, be a sequence of unnormalized densities¹ on the measurable space (X^t, \mathcal{X}^t) , parameterized by $\theta \in \Theta$. Let $\bar{\gamma}_{\theta,t}(x_{1:t})$ be the corresponding normalized probability densities:

$$\bar{\gamma}_{\theta,t}(x_{1:t}) = \frac{\gamma_{\theta,t}(x_{1:t})}{Z_{\theta,t}},\tag{1}$$

where $Z_{\theta,t} = \int \bar{\gamma}_{\theta,t}(x_{1:t}) dx_{1:t}$ and where it is assumed that $Z_{\theta,t} > 0, \forall \theta \in \Theta$. For instance, in the (important) special case of an SSM we have $\bar{\gamma}_{\theta,t}(x_{1:t}) = p_{\theta}(x_{1:t} \mid y_{1:t}), \ \gamma_{\theta,t}(x_{1:t}) = p_{\theta}(x_{1:t}, y_{1:t}), \ \text{and} \ Z_{\theta,t} = p_{\theta}(y_{1:t})$. We discuss this special case in more detail in Section 5.

To draw inference about the latent variables $x_{1:T}$, as well as to enable learning of the model parameter θ , a useful approach is to construct a Monte Carlo algorithm to draw samples from $\bar{\gamma}_{\theta,T}(x_{1:T})$. The sequential nature of the problem suggests the use of SMC methods; in particular, particle filters (PFs) [1, 2, 15].

We start by reviewing a standard SMC sampler, which will be used to construct the PGAS algorithm in the consecutive section. We will refer to the index variable t as time, but in general it might not have any temporal meaning. Let $\{x_{1:t-1}^i, w_{t-1}^i\}_{i=1}^N$ be a weighted particle system targeting $\bar{\gamma}_{\theta,t-1}(x_{1:t-1})$. That is, the weighted particles define an empirical point-mass approximation of the

¹With respect to some dominating measure which we denote simply as $dx_{1:t}$.

target distribution given by

$$\widehat{\gamma}_{\theta,t-1}^{N}(dx_{1:t-1}) = \sum_{i=1}^{N} \frac{w_{t-1}^{i}}{\sum_{l} w_{t-1}^{l}} \delta_{x_{1:t-1}^{i}}(dx_{1:t-1}).$$
(2)

This particle system is propagated to time t by sampling $\{a_t^i, x_t^i\}_{i=1}^N$ independently from a proposal kernel,

$$M_{\theta,t}(a_t, x_t) = \frac{w_{t-1}^{a_t}}{\sum_l w_{t-1}^l} r_{\theta,t}(x_t \mid x_{1:t-1}^{a_t}).$$
(3)

Note that $M_{\theta,t}$ depends on the complete particle system up to time t-1, $\{x_{1:t-1}^i, w_{t-1}^i\}_{i=1}^N$, but for notational convenience we shall not make that dependence explicit. Here, a_t^i is the index of the ancestor particle of x_t^i . In this formulation, the resampling step is implicit and corresponds to sampling these ancestor indices. When we write $x_{1:t}^i$ we refer to the ancestral path of particle x_t^i . That is, the particle trajectory is defined recursively as

$$x_{1:t}^{i} = (x_{1:t-1}^{a_{t}^{i}}, x_{t}^{i}). (4)$$

Once we have generated N ancestor indices and particles from the proposal kernel (3), the particles are weighted according to $w_t^i = W_{\theta,t}(x_{1:t}^i)$ where the weight function is given by

$$W_{\theta,t}(x_{1:t}) = \frac{\gamma_{\theta,t}(x_{1:t})}{\gamma_{\theta,t-1}(x_{1:t-1})r_{\theta,t}(x_t \mid x_{1:t-1})},$$
(5)

for $t \geq 2$. The procedure is initialized by sampling from a proposal density $x_1^i \sim r_{\theta,1}(x_1)$ and assigning importance weights $w_1^i = W_{\theta,1}(x_1^i)$ with $W_{\theta,1}(x_1) =$ $\gamma_{\theta,1}(x_1)/r_{\theta,1}(x_1)$. The SMC sampler is summarized in Algorithm 1.

Algorithm 1 Sequential Monte Carlo (each step is for i = 1, ..., N)

- 1: Draw $x_1^i \sim r_{\theta,1}(x_1)$.
- 2: Set $w_1^i = W_{\theta,1}(x_1^i)$
- 3: for t = 2 to T do
- Draw $\{a_t^i, x_t^i\} \sim M_{\theta,t}(a_t, x_t)$.
- Set $x_{1:t}^i = (x_{1:t-1}^{a_t^i}, x_t^i)$. Set $w_t^i = W_{\theta,t}(x_{1:t}^i)$.

It is interesting to note that the joint law of all the random variables generated by Algorithm 1 can be written down explicitly. Let

$$\mathbf{x}_t = \{x_t^1, \dots, x_t^N\}$$
 and $\mathbf{a}_t = \{a_t^1, \dots, a_t^N\},$

refer to all the particles and ancestor indices, respectively, generated at time t of the algorithm. It follows that the SMC sampler generates a collection of random variables $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}\} \in \mathsf{X}^{NT} \times \{1, \ldots, N\}^{N(T-1)}$. Furthermore, $\{a_t^i, x_t^i\}_{i=1}^N$ are drawn independently (conditionally on the particle system generated up to time t-1) from the proposal kernel $M_{\theta,t}$, and similarly at time t=1. Hence, the joint probability density function (with respect to a natural product of dx and counting measure) of these variables is given by

$$\psi_{\theta}(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}) \triangleq \prod_{i=1}^{N} r_{\theta, 1}(x_1^i) \prod_{t=2}^{T} \prod_{i=1}^{N} M_{\theta, t}(a_t^i, x_t^i).$$
 (6)

3 The PGAS kernel

We now turn to the construction of PGAS, a family of Markov kernels on the space of trajectories (X^T, \mathcal{X}^T) . We will provide an algorithm for generating samples from these Markov kernels, which are thus defined implicitly by the algorithm.

3.1 Particle Gibbs

Before stating the PGAS algorithm, we review the main ideas of the PG algorithm of [6] and we then turn to our proposed modification of this algorithm via the introduction of an *ancestor sampling* step.

PG is based on an SMC sampler, akin to a standard PF, but with the difference that one particle trajectory is specified a priori. This path, denoted as $x'_{1:T} = (x'_1, \ldots, x'_T)$, serves as a reference trajectory. Informally, it can be thought of as guiding the simulated particles to a relevant region of the state space. After a complete pass of the SMC algorithm, a trajectory $x^\star_{1:T}$ is sampled from among the particle trajectories. That is, we draw $x^\star_{1:T}$ with $\mathbb{P}(x^\star_{1:T} = x^i_{1:T}) \propto w^i_T$. This procedure thus maps $x'_{1:T}$ to a probability distribution on \mathcal{X}^T , implicitly defining a Markov kernel on $(\mathsf{X}^T, \mathcal{X}^T)$.

In a standard PF, the samples $\{a_t^i, x_t^i\}$ are drawn independently from the proposal kernel (3) for $i=1,\ldots,N$. When sampling from the PG kernel, however, we condition on the event that the reference trajectory $x'_{1:T}$ is retained throughout the sampling procedure. To accomplish this, we sample according to (3) only for $i=1,\ldots,N-1$. The Nth particle and its ancestor index are then set deterministically as $x_t^N=x_t'$ and $a_t^N=N$. This implies that after a complete pass of the algorithm, the Nth particle path coincides with the reference trajectory, i.e., $x_{1:T}^N=x'_{1:T}$.

The fact that $x'_{1:T}$ is used as a reference trajectory in the SMC sampler implies an invariance property of the PG kernel which is of key relevance. More precisely, as show by [6, Theorem 5], for any number of particles $N \geq 1$ and for any $\theta \in \Theta$, the PG kernel leaves the exact target distribution $\bar{\gamma}_{\theta,T}$ invariant. We return to this invariance property below, when it is shown to hold also for the proposed PGAS kernel.

3.2 Ancestor sampling

As noted above, the PG algorithm keeps the reference trajectory $x'_{1:T}$ intact throughout the sampling procedure. While this results in a Markov kernel which leaves $\bar{\gamma}_{\theta,T}$ invariant, it has been recognized that the mixing properties of this kernel can be very poor due to path degeneracy [5, 11].

To address this fundamental problem we now turn to our new procedure, PGAS. The idea is to sample a new value for the index variable a_t^N in an ancestor sampling step. While this is a small modification of the algorithm, the improvement in mixing can be quite considerable; see Section 3.3 and the numerical evaluation in Section 7. The ancestor sampling step is implemented as follows.

At time $t \geq 2$, we consider the part of the reference trajectory $x'_{t:T}$ ranging from the current time t to the final time point T. The task is to artificially assign a history to this partial path. This is done by connecting $x'_{t:T}$ to one of the particles $\{x^i_{1:t-1}\}_{i=1}^N$. Recall that the ancestry of a particle is encoded via the corresponding ancestor index. Hence, we can connect the partial reference path to one of the particles $\{x^i_{1:t-1}\}_{i=1}^N$ by assigning a value to the variable $a^N_t \in \{1, \ldots, N\}$. To do this, first we compute the weights

$$\widetilde{w}_{t-1|T}^{i} \triangleq w_{t-1}^{i} \frac{\gamma_{\theta,T}((x_{1:t-1}^{i}, x_{t:T}^{\prime}))}{\gamma_{\theta,t-1}(x_{1:t-1}^{i})}$$

$$(7)$$

for $i=1,\ldots,N$. Here, $(x_{1:t-1}^i,x_{t:T}')$ refers to the point in X^T formed by concatenating the two partial trajectories. Then, we sample a_t^N with $\mathbb{P}(a_t^N=i) \propto \widetilde{w}_{t-1|T}^i$. The expression above can be understood as an application of Bayes' theorem, where the importance weight w_{t-1}^i is the prior probability of the particle $x_{1:t-1}^i$ and the ratio between the target densities in (7) can be seen as the likelihood that $x_{t:T}'$ originated from $x_{1:t-1}^i$. A formal argument for why (7) provides the correct ancestor sampling distribution, in order to retain the invariance properties of the kernel, is detailed in the proof of Theorem 1 in Section 4.

The sampling procedure outlined above is summarized in Algorithm 2 and the family of PGAS kernels is formally defined below. Note that the only difference between PG and PGAS is on line 8 of Algorithm 2 (where, for PG, we would simply set $a_t^N = N$). However, as we shall see, the effect of this small modification on the mixing of the kernel is quite significant.

Definition 1 (PGAS kernels). For any $N \geq 1$ and any $\theta \in \Theta$, Algorithm 2 maps $x'_{1:T}$ stochastically into $x^{\star}_{1:T}$, thus implicitly defining a Markov kernel P^{N}_{θ} on (X^{T}, X^{T}) . The family of Markov kernels $\{P^{N}_{\theta} : \theta \in \Theta\}$, indexed by $N \geq 1$, is referred to as the PGAS family of kernels.

3.3 The effect of path degeneracy on PG and on PGAS

We have argued that ancestor sampling can considerably improve the mixing of PG. To illustrate this effect and to provide an explanation of its cause, we

Algorithm 2 PGAS Markov kernel

```
Require: Reference trajectory x'_{1:T} \in X^T.

1: Draw x_1^i \sim r_{\theta,1}(x_1) for i = 1, ..., N - 1.

2: Set x_1^N = x'_1.

3: Set w_1^i = W_{\theta,1}(x_1^i) for i = 1, ..., N.

4: for t = 2 to T do

5: Draw \{a_t^i, x_t^i\} \sim M_{\theta,t}(a_t, x_t) for i = 1, ..., N - 1.

6: Set x_t^N = x'_t.

7: Compute \{\widetilde{w}_{t-1|T}^i\}_{i=1}^N according to (7).

8: Draw a_t^N with \mathbb{P}(a_t^N = i) \propto \widetilde{w}_{t-1|T}^i.

9: Set x_{1:t}^i = (x_{1:t-1}^{a_{1:t-1}^i}, x_t^i) for i = 1, ..., N.

10: Set w_t^i = W_{\theta,t}(x_{1:t}^i) for i = 1, ..., N.

11: end for

12: Draw k with \mathbb{P}(k = i) \propto w_T^i.

13: return x_{1:T}^* = x_{1:T}^k.
```

consider a simple numerical example. Further empirical evaluation of PGAS is provided in Section 7. Consider the stochastic volatility model,

$$x_{t+1} = ax_t + v_t, v_t \sim \mathcal{N}(0, \sigma^2), (8a)$$

$$y_t = e_t \exp\left(\frac{1}{2}x_t\right), \qquad e_t \sim \mathcal{N}(0, 1),$$
 (8b)

where the state process $\{x_t\}_{t\geq 1}$ is latent and observations are made only via the measurement process $\{y_t\}_{t\geq 1}$. Similar models have been used to generalize the Black-Scholes option pricing equation to allow for the variance to change over time [16,17].

For simplicity, the parameter $\theta = (a, \sigma) = (0.9, 0.5)$ is assumed to be known. A batch of T = 400 observations are simulated from the system. Given these, we seek the joint smoothing density $p(x_{1:T} \mid y_{1:T})$. To generate samples from this density we employ both PG and PGAS with varying number of particles ranging from N = 5 to $N = 1\,000$. We simulate sample paths of length $1\,000$ for each algorithm. To compare the mixing, we look at the update rate of x_t versus t, which is defined as the proportion of iterations where x_t changes value. The results are reported in Figure 1, which reveals that ancestor sampling significantly increases the probability of updating x_t for t far from T.

The poor update rates for PG is a manifestation of the well known path degeneracy problem of SMC samplers (see, e.g., [1]). Consider the process of sampling from the PG kernel for a fixed reference trajectory $x_{1:T}'$. A particle system generated by the PG algorithm (corresponding to Algorithm 2, but with line 8 replaced with $a_t^N=N$) is shown in Figure 2 (left). For clarity of illustration, we have used a small number of particles and time steps, N=20 and T=50, respectively. By construction the reference trajectory (shown by a thick blue line) is retained throughout the sampling procedure. As a consequence, the particle system degenerates toward this trajectory which implies that $x_{1:T}^*$

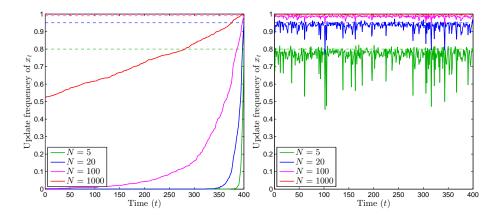


Figure 1: Update rates for x_t versus $t \in \{1, ..., 400\}$ for PG (left) and for PGAS (right). The dashed lines correspond to the ideal rates (N-1)/N. (This figure is best viewed in color.)

(shown as a red line) to a large extent will be identical to $x'_{1:T}$.

What is, perhaps, more surprising is that PGAS is so much more insensitive to the degeneracy issue. To understand why this is the case, we analyze the procedure for sampling from the PGAS kernel $P_{\theta}^{N}(x'_{1:T},\cdot)$ for the same reference trajectory $x'_{1:T}$ as above. The particle system generated by Algorithm 2 (with ancestor sampling) is shown in Figure 2 (right). The thick blue lines are again used to illustrate the reference particles, but now with updated ancestor indices. That is, the blue line segments are drawn between $x_{t-1}^{a_t^N}$ and x'_t for $t \geq 2$. It can be seen that the effect of ancestor sampling is that, informally, the reference trajectory is broken into pieces. It is worth pointing out that the particle system still collapses; ancestor sampling does not prevent path degeneracy. However, it causes the particle system to degenerate toward something different than the reference trajectory. As a consequence, $x_{1:T}^*$ (shown as a red line in the figure) will with high probability be substantially different from $x'_{1:T}$, enabling high update rates and thereby much faster mixing.

4 Theoretical justification

4.1 Stationary distribution

We begin by stating a theorem, whose proof is provided later in this section, which shows that the invariance property of PG is not violated by the ancestor sampling step.

Theorem 1. For any $N \geq 1$ and $\theta \in \Theta$, the PGAS kernel P_{θ}^{N} leaves $\bar{\gamma}_{\theta,T}$

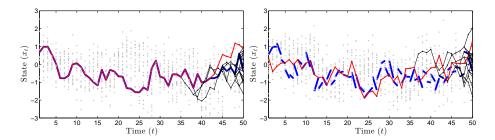


Figure 2: Particle systems generated by the PG algorithm (left) and by the PGAS algorithm (right), for the same reference trajectory $x'_{1:T}$ (shown as a thick blue line in the left panel). The gray dots show the particle positions and the thin black lines show the ancestral dependencies of the particles. The extracted trajectory $x^*_{1:T}$ is illustrated with a red line. Note that, due to path degeneracy, the particles shown as grey dots are not reachable by tracing any of the ancestral lineages from time T and back. In the right panel, ancestor sampling has the effect of breaking the reference trajectory into pieces, causing the particle system to degenerate toward something different than $x'_{1:T}$. (This figure is best viewed in color.)

invariant:

$$\bar{\gamma}_{\theta,T}(B) = \int P_{\theta}^{N}(x'_{1:T}, B) \bar{\gamma}_{\theta,T}(dx'_{1:T}), \quad \forall B \in \mathcal{X}^{T}.$$

An apparent difficulty in establishing this result is that it is not possible to write down a simple, closed-form expression for P_{θ}^{N} . In fact, the PGAS kernel is given by

$$P_{\theta}^{N}(x_{1:T}', B) = \mathbb{E}_{\theta, x_{1:T}'} \left[\mathbb{1}_{B}(x_{1:T}^{k}) \right], \tag{9}$$

where \mathbb{I}_B is the indicator function for the set B and where $\mathbb{E}_{\theta,x'_{1:T}}$ denotes expectation with respect to all the random variables generated by Algorithm 2, i.e., all the particles $\mathbf{x}_{1:T}$ and ancestor indices $\mathbf{a}_{2:T}$, as well as the index k. Computing this expectation is not possible in general. Instead of working directly with (9), however, we can adopt a strategy employed by [6], treating all the random variables generated by Algorithm 2, $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k\}$, as auxiliary variables, thus avoiding an intractable integration. In the following, it is convenient to view x_t^N as a random variable with distribution $\delta_{x'_t}$.

Recall that the particle trajectory $x_{1:T}^k$ is the ancestral path of the particle x_T^k . That is, we can write

$$x_{1:T}^k = x_{1:T}^{b_{1:T}} \triangleq (x_1^{b_1}, \dots, x_T^{b_T}),$$
 (10)

where the indices $b_{1:T}$ are given recursively by the ancestor indices: $b_T = k$ and $b_t = a_{t+1}^{b_{t+1}}$. Let $\Omega \triangleq \mathsf{X}^{NT} \times \{1, \ldots, N\}^{N(T-1)+1}$ be the space of all random

variables generated by Algorithm 2. Following [6], we then define a function $\phi_{\theta}: \Omega \mapsto \mathbb{R}$ as follows:

$$\phi_{\theta}(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k) = \phi_{\theta}(x_{1:T}^{b_{1:T}}, b_{1:T}) \phi_{\theta}(\mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}} \mid x_{1:T}^{b_{1:T}}, b_{1:T})$$

$$\triangleq \underbrace{\frac{\bar{\gamma}_{\theta, T}(x_{1:T}^{b_{1:T}})}{N^{T}} \prod_{\substack{i=1\\i\neq b_{1}}}^{N} r_{\theta, 1}(x_{1}^{i}) \prod_{t=2}^{T} \prod_{\substack{i=1\\i\neq b_{t}}}^{N} M_{\theta, t}(a_{t}^{i}, x_{t}^{i}), \qquad (11)$$
marginal

where we have introduced the notation

$$\mathbf{x}_{t}^{-i} = \{x_{t}^{1}, \dots, x_{t}^{i-1}, x_{t}^{i+1}, \dots, x_{t}^{N}\}, \qquad \mathbf{x}_{1:T}^{-b_{1:T}} = \{\mathbf{x}_{1}^{-b_{1}}, \dots, \mathbf{x}_{T}^{-b_{T}}\}$$

and similarly for the ancestor indices. By construction, ϕ_{θ} is nonnegative and integrates to one, i.e., ϕ_{θ} is a probability density function on Ω . We refer to this density as the *extended target density*.

The factorization into a marginal and a conditional density is intended to reveal some of the structure inherent in the extended target density. In particular, the marginal density of the variables $\{x_{1:T}^{b_{1:T}}, b_{1:T}\}$ is defined to be equal to the original target density $\bar{\gamma}_{\theta,T}(x_{1:T}^{b_{1:T}})$, up to a factor N^{-T} related to the index variables $b_{1:T}$. This has the important implication that if $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k\}$ are distributed according to ϕ_{θ} , then, by construction, the marginal distribution of $x_{1:T}^{b_{1:T}}$ is $\bar{\gamma}_{\theta,T}$.

By constructing an MCMC kernel with invariant distribution ϕ_{θ} , we will thus obtain a kernel with invariant distribution $\bar{\gamma}_{\theta,T}$ (the PGAS kernel) as a byproduct. To prove Theorem 1 we will reinterpret all the steps of the PGAS algorithm as partially collapsed Gibbs steps for ϕ_{θ} . The meaning of partial collapsing will be made precise in the proof of Lemma 2 below, but basically it refers to the process of marginalizing out some of the variables of the model in the individual steps of the Gibbs sampler. This is done in such a way that it does not violate the invariance property of the Gibbs kernel, i.e., each such Gibbs step will leave the extended target distribution invariant. As a consequence, the invariance property of the PGAS kernel follows. First we show that the PGAS algorithm in fact implements the following sequence of partially collapsed Gibbs steps for ϕ_{θ} .

Procedure 1 (Instrumental reformulation of PGAS). Given $x_{1:T}^{\prime,b_{1:T}^{\prime}} \in \mathsf{X}^{T}$ and $b_{1:T}^{\prime} \in \{1,\ldots,N\}^{T}$:

(i)
$$Draw \ \mathbf{x}_{1}^{-b'_{1}} \sim \phi_{\theta}(\cdot \mid x'_{1:T}^{,b'_{1:T}}, b'_{1:T}) \ and, for \ t = 2 \ to \ T, \ draw:$$

$$\{\mathbf{x}_{t}^{-b_{t}}, \mathbf{a}_{t}^{-b_{t}}\} \sim \phi_{\theta}(\cdot \mid \mathbf{x}_{1:t-1}^{-b'_{1:t-1}}, \mathbf{a}_{2:t-1}, x'_{1:T}^{,b'_{1:T}}, b'_{t-1:T}),$$

$$a_{t}^{b_{t}} \sim \phi_{\theta}(\cdot \mid \mathbf{x}_{1:t-1}^{-b'_{1:t-1}}, \mathbf{a}_{2:t-1}, x'_{1:T}^{,b'_{1:T}}, b'_{t-T}),$$

(ii)
$$Draw \ k \sim \phi_{\theta}(\cdot \mid \mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}, \mathbf{x}_{1:T}^{\prime, b_{1:T}^{\prime}}).$$

Lemma 1. Algorithm 2 is equivalent to the partially collapsed Gibbs sampler of Procedure 1, conditionally on $x_{1:T}^{\prime,b_{1:T}^{\prime}} = x_{1:T}^{\prime}$ and $b_{1:T}^{\prime} = (N, ..., N)$.

Proof. From (11) we have, by construction,

$$\phi_{\theta}(\mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1\\i \neq b_1}}^{N} r_{\theta,1}(x_1^i) \prod_{t=2}^{T} \prod_{\substack{i=1\\i \neq b_t}}^{N} M_{\theta,t}(a_t^i, x_t^i).$$
 (12)

By marginalizing this expression over $\{\mathbf{x}_{t+1:T}^{-b_{t+1:T}}, \mathbf{a}_{t+1:T}^{-b_{t+1:T}}\}$ we get

$$\phi_{\theta}(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{2:t}^{-b_{2:t}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1\\i \neq b_1}}^{N} r_{\theta,1}(x_1^i) \prod_{s=2}^{t} \prod_{\substack{i=1\\i \neq b_s}}^{N} M_{\theta,s}(a_s^i, x_s^i), \tag{13}$$

It follows that

$$\phi_{\theta}(\mathbf{x}_{1}^{-b_{1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1\\i \neq b_{1}}}^{N} r_{\theta,1}(x_{1}^{i}), \tag{14a}$$

and, for $t = 2, \ldots, T$,

$$\phi_{\theta}(\mathbf{x}_{t}^{-b_{t}}, \mathbf{a}_{t}^{-b_{t}} \mid \mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}}, x_{1:T}^{b_{1:T}}, b_{1:T})$$

$$= \frac{\phi_{\theta}(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{2:t}^{-b_{2:t}} \mid x_{1:T}^{b_{1:T}}, b_{1:T})}{\phi_{\theta}(\mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T})} = \prod_{\substack{i=1\\i\neq b_{t}}}^{N} M_{\theta,t}(a_{t}^{i}, x_{t}^{i}).$$
(14b)

Hence, we can sample from (14a) and (14b) by drawing $x_1^i \sim r_{\theta,1}(\cdot)$ for $i \in \{1, \ldots, N\} \setminus b_1$ and $\{a_t^i, x_t^i\} \sim M_{\theta,t}(\cdot)$ for $i \in \{1, \ldots, N\} \setminus b_t$, respectively. Consequently, with the choice $b_t = N$ for $t = 1, \ldots, T$, the initialization at line 1 and the particle propagation at line 5 of Algorithm 2 correspond to sampling from (14a) and (14b), respectively.

Next, we consider the ancestor sampling step. Recall that $a_t^{b_t}$ identifies to b_{t-1} . We can thus write

$$\phi_{\theta}(a_{t}^{b_{t}} \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t:T}) \propto \phi_{\theta}(\mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t-1:T})
= \phi_{\theta}(x_{1:T}^{b_{1:T}}, b_{1:T}) \phi_{\theta}(\mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T})
= \frac{\gamma_{\theta, T}(x_{1:T}^{b_{1:T}})}{\gamma_{\theta, t-1}(x_{1:t-1}^{b_{1:t-1}})} \frac{\gamma_{\theta, t-1}(x_{1:t-1}^{b_{1:t-1}})}{Z_{\theta, T}N^{T}} \prod_{\substack{i=1\\i\neq b_{i}}}^{N} r_{\theta, 1}(x_{1}^{i}) \prod_{\substack{s=2\\i\neq b_{i}}}^{t-1} \prod_{\substack{i=1\\i\neq b_{i}}}^{N} M_{\theta, s}(a_{s}^{i}, x_{s}^{i}). \quad (15)$$

To simplify this expression, note first that we can write

$$\gamma_{\theta,t-1}(x_{1:t-1}) = \gamma_{\theta,1}(x_1) \prod_{s=2}^{t-1} \frac{\gamma_{\theta,s}(x_{1:s})}{\gamma_{\theta,s-1}(x_{1:s-1})}.$$
 (16)

By using the definition of the weight function (5), this expression can be expanded according to

$$\gamma_{\theta,t-1}(x_{1:t-1}) = W_{\theta,1}(x_1)r_{\theta,1}(x_1) \prod_{s=2}^{t-1} W_{\theta,s}(x_{1:s})r_{\theta,s}(x_s \mid x_{1:s-1}).$$
 (17)

Plugging the trajectory $x_{1:t-1}^{b_{1:t-1}}$ into the above expression, we get

$$\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}}) = w_1^{b_1} r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} w_s^{b_s} r_{\theta,s}(x_s^{b_s} \mid x_{1:s-1}^{b_{1:s-1}})
= \left(\prod_{s=1}^{t-1} \sum_{l=1}^{N} w_s^l\right) \frac{w_1^{b_1}}{\sum_l w_1^l} r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} \frac{w_s^{b_s}}{\sum_l w_s^l} r_{\theta,s}(x_s^{b_s} \mid x_{1:s-1}^{b_{1:s-1}})
= \frac{w_{t-1}^{b_{t-1}}}{\sum_l w_{t-1}^l} \left(\prod_{s=1}^{t-1} \sum_{l=1}^{N} w_s^l\right) r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} M_{\theta,s}(a_s^{b_s}, x_s^{b_s}).$$
(18)

Expanding the numerator in (15) according to (18) results in

$$\phi_{\theta}(a_{t}^{b_{t}} \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t:T}) \\
\propto \frac{\gamma_{\theta,T}(x_{1:T}^{b_{1:T}})}{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})} \frac{w_{t-1}^{b_{t-1}}}{\sum_{l} w_{t-1}^{l}} \frac{\left(\prod_{s=1}^{t-1} \sum_{l} w_{s}^{l}\right)}{Z_{\theta,T} N^{T}} \prod_{i=1}^{N} r_{\theta,1}(x_{1}^{i}) \prod_{s=2}^{t-1} \prod_{i=1}^{N} M_{\theta,s}(a_{s}^{i}, x_{s}^{i}) \\
\propto w_{t-1}^{b_{t-1}} \frac{\gamma_{\theta,T}((x_{1:t-1}^{b_{1:t-1}}, x_{t:T}^{b_{t:T}}))}{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})}.$$
(19)

Consequently, with $b_t = N$ and $x_{t:T}^{b_{t:T}} = x'_{t:T}$, sampling from (19) corresponds to the ancestor sampling step of line 8 of Algorithm 2. Finally, analogously to (19), it follows that $\phi_{\theta}(k \mid \mathbf{x}_{1:T}, \mathbf{a}_{2:T}) \propto w_T^k$, which corresponds to line 12 of Algorithm 2.

Next, we show that Procedure 1 leaves ϕ_{θ} invariant. This is done by concluding that the procedure is a properly collapsed Gibbs sampler; see [18]. Marginalization, or collapsing, is commonly used within Gibbs sampling to improve the mixing and/or to simplify the sampling procedure. However, it is crucial that the collapsing is carried out in the correct order to respect the dependencies between the variables of the model.

Lemma 2. The Gibbs sampler of Procedure 1 is properly collapsed and thus leaves ϕ_{θ} invariant.

Proof. Consider the following sequence of *complete* Gibbs steps:

(i) Draw
$$\{\mathbf{x}_{1}^{-b'_{1}}, \underline{\mathbf{x}}_{2:T}^{-b'_{2:T}}, \underline{\mathbf{a}}_{2:T}^{-b'_{2:T}}\} \sim \phi_{\theta}(\cdot \mid x'_{1:T}^{,b'_{1:T}}, b'_{1:T})$$
 and, for $t=2$ to T , draw:

$$\{\mathbf{x}_t^{-b_t}, \mathbf{a}_t, \underline{\mathbf{x}}_{t+1:T}^{-b'_{t+1:T}}, \underline{\mathbf{a}}_{t+1:T}^{-b'_{t+1:T}}\} \sim \phi_{\theta}(\,\cdot\mid \mathbf{x}_{1:t-1}^{-b'_{1:t-1}}, \mathbf{a}_{2:t-1}, x'_{1:T}^{,b'_{1:T}}, b'_{t:T}).$$

(ii) Draw
$$k \sim \phi_{\theta}(\cdot \mid \mathbf{x}_{1:T}^{-b'_{1:T}}, \mathbf{a}_{2:T}, x'_{1:T}^{b'_{1:T}}).$$

In the above, all the samples are drawn from conditionals under the full joint density $\phi_{\theta}(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k)$. Hence, it is clear that the above procedure will leave ϕ_{θ} invariant. Note that some of the variables above have been marked by an underline. It can be seen that these variables are in fact never conditioned upon in any subsequent step of the procedure. That is, the underlined variables are never used. Therefore, to obtain a valid sampler it is sufficient to sample all the non-underlined variables from their respective marginals. Furthermore, from (14b) it can be seen that $\{\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t}\}$ are conditionally independent of $a_t^{b_t}$, i.e., it follows that the complete Gibbs sweep above is equivalent to the partially collapsed Gibbs sweep of Procedure 1. Hence, the Gibbs sampler is properly collapsed and it will therefore leave ϕ_{θ} invariant.

Proof (Theorem 1). Let

$$\mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T})$$
(20)

denote the law of the random variables generated by Procedure 1, conditionally on $x_{1:T}^{\prime,b'_{1:T}}=x'_{1:T}$ and on $b'_{1:T}$. Using Lemma 2 and recalling that $\phi_{\theta}(x_{1:T}^{b_{1:T}},b_{1:T})=N^{-T}\bar{\gamma}_{\theta,T}(x_{1:T}^{b_{1:T}})$ we have

$$\bar{\gamma}_{\theta,T}(B) = \int \mathbb{1}_{B}(x_{1:T}^{k}) \mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T})$$

$$\times \delta_{x'_{1}}(dx_{1}^{b'_{1}}) \cdots \delta_{x'_{T}}(dx_{T}^{b'_{T}}) \frac{\bar{\gamma}_{\theta,T}(x'_{1:T})}{N^{T}} dx'_{1:T} db'_{1:T}, \quad \forall B \in \mathcal{X}^{T}. \quad (21)$$

By Lemma 1 we know that Algorithm 2, which implicitly defines P_{θ}^{N} , is equivalent to Procedure 1 conditionally on $x_{1:T}^{\prime,b_{1:T}^{\prime}}=x_{1:T}^{\prime}$ and $b_{1:T}^{\prime}=(N,\ldots,N)$. That is to say,

$$P_{\theta}^{N}(x'_{1:T}, B) = \int \mathbb{1}_{B}(x_{1:T}^{k}) \mathcal{L}(d\mathbf{x}_{1:T}^{-(N, \dots, N)}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, (N, \dots, N))$$

$$\times \delta_{x'_{1}}(dx_{1}^{N}) \cdots \delta_{x'_{T}}(dx_{T}^{N}), \qquad \forall x'_{1:T} \in \mathsf{X}^{T}, \forall B \in \mathcal{X}^{T}.$$
(22)

However, the law of $x_{1:T}^{\star}$ in Algorithm 2 is invariant to permutations of the particle indices. That is, it does not matter if we place the reference particles on the Nth positions, or on some other positions, when enumerating the particles². This implies that for any $b'_{1:T} \in \{1, \ldots, N\}^T$,

$$P_{\theta}^{N}(x'_{1:T}, B) = \int \mathbb{1}_{B}(x_{1:T}^{k}) \mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T})$$

$$\times \delta_{x'_{1}}(dx_{1}^{b'_{1}}) \cdots \delta_{x'_{T}}(dx_{T}^{b'_{T}}), \qquad \forall x'_{1:T} \in \mathsf{X}^{T}, \forall B \in \mathcal{X}^{T}.$$

$$(23)$$

²A formal proof of this statement is given for the PG sampler in [11]. The same argument can be used also for PGAS.

Plugging (23) into (21) gives the desired result,

$$\bar{\gamma}_{\theta,T}(B) = \int P_{\theta}^{N}(x'_{1:T}, B) \bar{\gamma}_{\theta,T}(x'_{1:T}) \underbrace{\left(\sum_{b'_{1:T}} \frac{1}{N^{T}}\right)}_{=1} dx'_{1:T}, \quad \forall B \in \mathcal{X}^{T}.$$

4.2 Ergodicity

To show ergodicity of the PGAS kernel we need to characterize the support of the target and the proposal densities. Let,

$$S_{\theta,t} = \{ x_{1:t} \in \mathsf{X}^t : \bar{\gamma}_{\theta,t}(x_{1:t}) > 0 \}, \tag{24a}$$

$$Q_{\theta,t} = \{ x_{1:t} \in \mathsf{X}^t : r_{\theta,t}(x_t \mid x_{1:t-1}) \bar{\gamma}_{\theta,t-1}(x_{1:t-1}) > 0 \}, \tag{24b}$$

with obvious modifications for t = 1. The following is a minimal assumption.

(A1) For any $\theta \in \Theta$ and $t \in \{1, ..., T\}$ we have $\mathcal{S}_t^{\theta} \subseteq \mathcal{Q}_t^{\theta}$.

Assumption (A1) basically states that the support of the proposal density should cover the support of the target density. Ergodicity of PG under Assumption (A1) has been established by Andrieu et al. [6]. The same argument can be applied also to PGAS.

Theorem 2 (Andrieu et al. [6, Theorem 5]). Assume (A1). Then, for any $N \geq 2$ and $\theta \in \Theta$, P_{θ}^{N} is $\bar{\gamma}_{\theta,T}$ -irreducible and aperiodic. Consequently,

$$\lim_{n \to \infty} \|(P_{\theta}^N)^n(x'_{1:T}, \cdot) - \bar{\gamma}_{\theta, T}(\cdot)\|_{\text{TV}} = 0, \qquad \forall x'_{1:T} \in \mathsf{X}^T.$$

To strengthen the ergodicity results for the PGAS kernel, we use a boundedness condition for the importance weights, given in assumption (A2) below. Such a condition is typical also in classical importance sampling and is, basically, a slightly stronger version of assumption (A1).

(A2) For any $\theta \in \Theta$ and $t \in \{1, ..., T\}$, there exists a constant $\kappa_{\theta} < \infty$ such that $||W_{\theta,t}||_{\infty} \leq \kappa_{\theta}$.

Theorem 3. Assume (A2). Then, for any $N \geq 2$ and $\theta \in \Theta$, P_{θ}^{N} is uniformly ergodic. That is, there exist constants $R_{\theta} < \infty$ and $\rho_{\theta} \in [0,1)$ such that

$$\|(P_{\theta}^N)^n(x'_{1:T},\cdot) - \bar{\gamma}_{\theta,T}(\cdot)\|_{\text{TV}} \le R_{\theta} \rho_{\theta}^n, \qquad \forall x'_{1:T} \in \mathsf{X}^T.$$

Proof. We show that P_{θ}^{N} satisfies a Doeblin condition,

$$P_{\theta}^{N}(x_{1:T}', B) \ge \varepsilon_{\theta} \bar{\gamma}_{\theta, T}(B), \qquad \forall x_{1:T}' \in \mathsf{X}^{T}, \forall B \in \mathcal{X}^{T}, \tag{25}$$

for some constant $\varepsilon_{\theta} > 0$. Uniform ergodicity then follows from [19, Proposition 2]. To prove (25) we use the representation of the PGAS kernel in (9),

$$P_{\theta}^{N}(x'_{1:T}, B) = \mathbb{E}_{\theta, x'_{1:T}} \left[\mathbb{1}_{B}(x_{1:T}^{k}) \right] = \sum_{j=1}^{N} \mathbb{E}_{\theta, x'_{1:T}} \left[\frac{w_{T}^{j}}{\sum_{l} w_{T}^{l}} \mathbb{1}_{B}(x_{1:T}^{j}) \right]$$

$$\geq \frac{1}{N \kappa_{\theta}} \sum_{j=1}^{N-1} \mathbb{E}_{\theta, x'_{1:T}} \left[w_{T}^{j} \mathbb{1}_{B}(x_{1:T}^{j}) \right] = \frac{N-1}{N \kappa_{\theta}} \mathbb{E}_{\theta, x'_{1:T}} \left[W_{\theta, T}(x_{1:T}^{1}) \mathbb{1}_{B}(x_{1:T}^{1}) \right].$$
(26)

Here, the inequality follows from bounding the weights in the normalization by κ_{θ} and by simply discarding the Nth term of the sum (which is clearly nonnegative). The last equality follows from the fact that the particle trajectories $\{x_{1:T}^i\}_{i=1}^{N-1}$ are equally distributed under Algorithm 2.

Let $h_{\theta,t}: X^t \mapsto \mathbb{R}_+$ and consider

$$\mathbb{E}_{\theta,x'_{1:T}}\left[h_{\theta,t}(x_{1:t}^{1})\right] = \mathbb{E}_{\theta,x'_{1:T}}\left[\mathbb{E}_{\theta,x'_{1:T}}\left[h_{\theta,t}(x_{1:t}^{1}) \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}\right]\right] \\
= \mathbb{E}_{\theta,x'_{1:T}}\left[\sum_{j=1}^{N} \int h_{\theta,t}((x_{1:t-1}^{j}, x_{t})) \frac{w_{t-1}^{j}}{\sum_{l} w_{t-1}^{l}} r_{\theta,t}(x_{t} \mid x_{1:t-1}^{j}) dx_{t}\right] \\
\geq \frac{N-1}{N\kappa_{\theta}} \mathbb{E}_{\theta,x'_{1:T}}\left[\int h_{\theta,t}((x_{1:t-1}^{1}, x_{t})) W_{\theta,t-1}(x_{1:t-1}^{1}) r_{\theta,t}(x_{t} \mid x_{1:t-1}^{1}) dx_{t}\right], \tag{27}$$

where the inequality follows analogously to (26). Now, let

$$h_{\theta,T}(x_{1:T}) = W_{\theta,T}(x_{1:T}) \mathbb{1}_B(x_{1:T}),$$

$$h_{\theta,t-1}(x_{1:t-1}) = \int h_{\theta,t}(x_{1:t}) W_{\theta,t-1}(x_{1:t-1}) r_{\theta,t}(x_t \mid x_{1:t-1}) dx_t, \quad t \le T.$$

Then, by iteratively making use of (27) and changing the order of integration, we can bound (26) according to

$$\left(\frac{N-1}{N\kappa_{\theta}}\right)^{-T} P_{\theta}^{N}(x'_{1:T}, B) \geq \mathbb{E}_{\theta, x'_{1:T}} \left[h_{\theta, 1}(x_{1}^{1})\right]
= \int W_{\theta, 1}(x_{1}) r_{\theta, 1}(x_{1}) \prod_{t=2}^{T} \left(W_{\theta, t}(x_{1:t}) r_{\theta, t}(x_{t} \mid x_{1:t-1})\right) \mathbb{1}_{B}(x_{1:T}) dx_{1:T}
= \int \gamma_{\theta, 1}(x_{1}) \prod_{t=2}^{T} \left(\frac{\gamma_{\theta, t}(x_{1:t})}{\gamma_{\theta, t-1}(x_{1:t-1})}\right) \mathbb{1}_{B}(x_{1:T}) dx_{1:T}
= \int \gamma_{\theta, T}(x_{1:T}) \mathbb{1}_{B}(x_{1:T}) dx_{1:T} = Z_{\theta, T} \bar{\gamma}_{\theta, T}(B).$$
(28)

With $N \geq 2$ and since $Z_{\theta,T} > 0$ the result follows.

Algorithm 3 PGAS for Bayesian learning of SSMs

- 1: Set $\theta[0]$ and $x_{1:T}[0]$ arbitrarily.
- 2: **for** $n \ge 1$ **do**
- Draw $x_{1:T}[n] \sim P_{\theta[n-1]}^{N}(x_{1:T}[n-1], \cdot)$. /* By running Algorithm 2 */ Draw $\theta[n] \sim p(\theta \mid x_{1:T}[n], y_{1:T})$. 3:
- 4:
- 5: end for

5 PGAS for state-space models

Learning of state-space models with PGAS

SSMs comprise an important special case of the model class treated above. In this section, we illustrate how PGAS can be used for inference and learning of these models. We consider the nonlinear/non-Gaussian SSM

$$x_{t+1} \sim f_{\theta}(x_{t+1} \mid x_t),$$
 (29a)

$$y_t \sim g_\theta(y_t \mid x_t),\tag{29b}$$

and $x_1 \sim \mu_{\theta}(x_1)$, where $\theta \in \Theta$ is a static parameter, x_t is the latent state and y_t is the observation at time t, respectively. Given a batch of measurements $y_{1:T}$, we wish to make inferences about θ and/or about the latent states $x_{1:T}$.

Consider first the Bayesian setting where a prior distribution $\pi(\theta)$ is assigned to θ . We seek the parameter posterior $p(\theta \mid x_{1:T})$ or, more generally, the joint state and parameter posterior $p(\theta, x_{1:T} \mid y_{1:T})$. Gibbs sampling can be used to simulate from this distribution by sampling the state variables $\{x_t\}$ one at a time and the parameters θ from their respective posteriors. However, it has been recognized that this can result in poor mixing, due to the often high autocorrelation of the state sequence. The PGAS kernel offers a different approach, namely to sample the complete state trajectory $x_{1:T}$ in one block. This can considerably improve the mixing of the sampler [20]. Due to the invariance property of the kernel (Theorem 1), the validity of the Gibbs sampler is not violated. We summarize the procedure in Algorithm 3.

PGAS is also useful for maximum-likelihood-based learning of SSMs. A popular strategy for computing the maximum likelihood estimator

$$\widehat{\theta}_{\mathrm{ML}} = \operatorname*{arg\,max} \log p_{\theta}(y_{1:T}) \tag{30}$$

is to use the expectation maximization (EM) algorithm [21, 22]. EM is an iterative method, which maximizes $\log p_{\theta}(y_{1:T})$ by iteratively maximizing an auxiliary quantity: $\theta[n] = \arg\max_{\theta \in \Theta} Q(\theta, \theta[n-1])$, where

$$Q(\theta, \theta[n-1]) = \int \log p_{\theta}(x_{1:T}, y_{1:T}) p_{\theta[n-1]}(x_{1:T} \mid y_{1:T}) dx_{1:T}.$$
 (31)

When the above integral is intractable to compute, one can use a Monte Carlo approximation or a stochastic approximation of the intermediate quantity, leading to the MCEM [23] and the SAEM [24] algorithms, respectively. When the

Algorithm 4 PGAS for frequentist learning of SSMs

```
1: Set \theta[0] and x_{1:T}[0] arbitrarily. Set \widehat{Q}_0(\theta) \equiv 0.

2: for n \geq 1 do

3: Draw x_{1:T}[n] \sim P_{\theta[n-1]}^N(x_{1:T}[n-1], \cdot). /* By running Algorithm 2 */

4: Compute \widehat{Q}_n(\theta) according to (32).

5: Compute \theta[n] = \arg\max_{\theta \in \Theta} \widehat{Q}_n(\theta).

6: if convergence criterion is met then

7: return \theta[n].

8: end if

9: end for
```

underlying Monte Carlo simulation is computationally involved, SAEM is particularly useful since it makes efficient use of the simulated values. The SAEM approximation of the auxiliary quantity is given by

$$\widehat{Q}_n(\theta) = (1 - \alpha_n)\widehat{Q}_{n-1}(\theta) + \alpha_n \left(\log p_{\theta}(x_{1:T}[n], y_{1:T})\right), \tag{32}$$

where α_n is the step size and, in the vanilla form of SAEM, $x_{1:T}[n]$ is drawn from the joint smoothing density $p_{\theta[n-1]}(x_{1:T} \mid y_{1:T})$. In practice, the stochastic approximation update (32) is typically made on some sufficient statistic for the complete data log-likelihood; see [24] for details. While the joint smoothing density is intractable for a general nonlinear/non-Gaussian SSM, it has been recognized that it is sufficient to sample from a uniformly ergodic Markov kernel, leaving the joint smoothing distribution invariant [25, 26]. A practical approach is therefore to compute the auxiliary quantity according to the stochastic approximation (32), but where $x_{1:T}[n]$ is simulated from the PGAS kernel $P_{\theta[n-1]}^N(x_{1:T}[n-1], \cdot)$. This particle SAEM algorithm, previously presented in [27], is summarized in Algorithm 4.

5.2 Sampling from the PGAS kernel

Sampling from the PGAS kernel, i.e., running Algorithm 2, is similar to running a PF. The only non-standard (and nontrivial) operation is the ancestor sampling step. For the learning algorithms discussed above, the distribution of interest is the joint smoothing distribution. Consequently, the unnormalized target density is given by $\gamma_{\theta,T}(x_{1:T}) = p_{\theta}(x_{1:T}, y_{1:T})$. The ancestor sampling weights in (7) are thus given by

$$\widetilde{w}_{t-1|T}^{i} = w_{t-1}^{i} p_{\theta}(x_{t:T}', y_{t:T} \mid x_{1:t-1}^{i}, y_{1:t-1}) \propto w_{t-1}^{i} f_{\theta}(x_{t}' \mid x_{t-1}^{i}).$$
 (33)

This expression can be understood as an application of Bayes' theorem. The importance weight w^i_{t-1} is the prior probability of the particle x^i_{t-1} and the factor $f_{\theta}(x'_t \mid x^i_{t-1})$ is the likelihood of moving from x^i_{t-1} to x'_t . The product of these two factors is thus proportional to the posterior probability that x'_t originated from x^i_{t-1} .

The expression (33) can also be recognized as a one-step backward simulation; see [5, 28]. This highlights the close relationship between PGAS and PGBS. The latter method is conceptually similar to PGAS, but it make use of an explicit backward simulation pass; see [12, 29] or [5, Section 5.4]. More precisely, to generate a draw from the PGBS kernel, we first run a particle filter with reference trajectory $x'_{1:T}$ without ancestor sampling (i.e., in Algorithm 2, we replace line 8 with $a_t^N = N$, as in the basic PG sampler). Thereafter, we extract a new trajectory by running a backward simulator. That is, we draw $j_{1:T}$ with $\mathbb{P}(j_T = i) \propto w_T^i$ and then, for t = T - 1 to 1,

$$\mathbb{P}(j_t = i \mid j_{t+1}) \propto w_t^i f_{\theta}(x_{t+1}^{j_{t+1}} \mid x_t^i), \tag{34}$$

and take $x_{1:T}^{\star} = x_{1:T}^{j_{1:T}}$ as the output from the algorithm. In the above, the conditioning on the forward particle system $\{\mathbf{x}_{1:T}, \mathbf{a}_{2:T}\}$ is implicit.

Let the Markov kernel on (X^T, \mathcal{X}^T) defined by this procedure be denoted as $P_{\mathsf{BS},\theta}^N$. An interesting question to ask is whether or not the PGAS kernel P_{θ}^N and the PGBS kernel $P_{\mathsf{BS},\theta}^N$ are probabilistically equivalent. It turns out that, in some specific scenarios, this is indeed the case.

Proposition 1. Assume that PGAS and PGBS both target the joint smoothing distribution for an SSM and that both methods use the bootstrap proposal kernel in the internal particle filters, i.e., $r_{\theta,t}(x_t \mid x_{1:t-1}) = f_{\theta}(x_t \mid x_{t-1})$. Then, for any $x'_{1:T} \in \mathsf{X}^T$ and $B \in \mathcal{X}^T$, $P^N_{\theta}(x'_{1:T}, B) = P^N_{\mathsf{BS},\theta}(x'_{1:T}, B)$.

Proof. See Appendix A.
$$\Box$$

Proposition 1 builds upon [30, Proposition 5], where the equivalence between a (standard) bootstrap PF and a backward simulator is established. In Appendix A, we adapt their argument to handle the case with conditioning on a reference trajectory and ancestor sampling. The conditions of Proposition 1 imply that the weight functions (5) in the internal particle filters are independent of the ancestor indices. This is key in establishing the above result and we emphasize that the equivalence between the samplers does not hold in general for models outside the class of SSMs. In particular, for the class of non-Markovian latent variable models, discussed in the subsequent section, we have found that the samplers have quite different properties.

6 PGAS for non-Markovian models

6.1 Non-Markovian latent variable models

A very useful generalization of SSMs is the class of non-Markovian latent variable models,

$$x_{t+1} \sim f_{\theta}(x_{t+1} \mid x_{1:t}),$$
 (35a)

$$y_t \sim g_\theta(y_t \mid x_{1:t}). \tag{35b}$$

Similarly to the SSM (29), this model is characterized by a latent process $x_t \in \mathsf{X}$ and an observed process $y_t \in \mathsf{Y}$. However, it does not share the conditional independence properties that are central to SSMs. Instead, both the transition density f_θ and the measurement density g_θ may depend on the entire past history of the latent process. In Sections 6.2 and 6.3, we discuss the ancestor sampling step of the PGAS algorithm specifically for these non-Markovian models. We consider two approaches for efficient implementation of this step, first by using Metropolis-Hastings within PGAS and then by using a truncation strategy for the ancestor sampling weights. First, however, to motivate the present development we review some application areas in which this type of models arise.

In Bayesian nonparametrics [31] the latent random variables of the classical Bayesian model are replaced by latent stochastic processes, which are typically non-Markovian. This includes popular models based on the Dirichlet process, e.g., [32, 33], and Gaussian process regression and classification models [34]. These processes are also commonly used as components in hierarchical Bayesian models, which then inherit their non-Markovianity. An example is the Gaussian process SSM [35, 36], a flexible nonlinear dynamical systems model, for which PGAS has been successfully applied [36].

Another typical source of non-Markovianity is by marginalization over part of the state vector (i.e., Rao-Blackwellization [12,37,38]) or by a change of variables in an SSM. This type of operations typically results in a loss of the Markov property, but they can, however, be very useful. For instance, by expressing an SSM in terms of its "innovations" (i.e., the driving noise of the state process), it is possible to use backward and ancestor sampling in models for which the state transition density is not available to us. This includes many models for which the transition is implicitly given by a simulator [39–42] or degenerate models where the transition density does not even exist [43, 44]. We illustrate these ideas in Section 7. See also [5, Section 4] for a more in-depth discussion on reformulations of SSMs as non-Markovian models.

Finally, it is worth to point out that many statistical models which are not sequential "by nature" can be conveniently viewed as non-Markovian latent variable models. This includes, among others, probabilistic graphical models such as Markov random fields; see [5, Section 4].

6.2 Forced move Metropolis-Hastings

To employ PGAS (or in fact any backward-simulation-based method; see [5]) we need to evaluate the ancestor sampling weights (7) which depend on the ratio,

$$\frac{\gamma_{\theta,T}(x_{1:T})}{\gamma_{\theta,t-1}(x_{1:t-1})} = \frac{p_{\theta}(x_{1:T}, y_{1:T})}{p_{\theta}(x_{1:t-1}, y_{1:t-1})} = \prod_{s=t}^{T} g_{\theta}(y_s \mid x_{1:s}) f_{\theta}(x_s \mid x_{1:s-1}). \tag{36}$$

Assuming that g_{θ} and f_{θ} can both be evaluated in constant time, the computational cost of computing the backward sampling weights (7) will thus be O(NT).

This step can easily become the computational bottleneck when applying the PGAS algorithm to a non-Markovian model.

A simple way to reduce the complexity is to employ Metropolis-Hastings (MH) within PGAS. Let

$$\rho(k) = \frac{\widetilde{w}_{t-1|T}^k}{\sum_{l=1}^N \widetilde{w}_{t-1|T}^l} \tag{37}$$

denote the law of the ancestor index a_t^N , sampled at line 8 of Algorithm 2. From Lemma 1, we know that this step of the algorithm in fact corresponds to a Gibbs step for the extended target distribution (11). To retain the correct limiting distribution of the PGAS kernel, it is therefore sufficient that a_t^N is sampled from a Markov kernel leaving (37) invariant (resulting in a standard combination of MCMC kernels; see, e.g., [19]).

Let $q(k' \mid k)$ be an MH proposal kernel on $\{1, \ldots, N\}$. We can thus propose a move for the ancestor index a_t^N , from N to k', by simulating $k' \sim q(\cdot \mid N)$. With probability

$$1 \wedge \frac{\widetilde{w}_{t-1\mid T}^{k'}}{\widetilde{w}_{t-1\mid T}^{N}} \frac{q(N\mid k')}{q(k'\mid N)} \tag{38}$$

the sample is accepted and we set $a_t^N = k'$, otherwise we keep the ancestry $a_t^N = N$. Using this approach, we only need to evaluate the ancestor sampling weights for the proposed values, bringing the total computational cost down from $O(NT^2)$ to $O(NT+T^2)$. While still quadratic in T, this reduction can be very useful whenever N is moderately large.

Since the variable a_t^N is discrete-valued, it is recommended to use a forced move proposal in the spirit of [46]. That is, q is constructed so that $q(k \mid k) = 0, \forall k$, ensuring that the current state of the chain is not proposed anew, which would be a wasteful operation. One simple choice is to let $q(k' \mid k)$ be uniform over $\{1, \ldots, N\} \setminus k$. In the subsequent section, we discuss a different strategy for reducing the complexity of the ancestor sampling step, which can also be used to design a better proposal for the forced move MH sampler.

6.3 Truncation of the ancestor sampling weights

The quadratic computational complexity in T for the forced move MH sampler may still be prohibitive if T is large. To make progress, we consider non-Markovian models in which there is a decay in the influence of the past on the present, akin to that in Markovian models but without the strong Markovian assumption. Hence, it is possible to obtain a useful approximation of the ancestor sampling weights by truncating the product (36) to a smaller number of

factors, say ℓ . We can thus replace (7) with the approximation

$$\widetilde{w}_{t-1|T}^{\ell,i} \triangleq w_{t-1}^{i} \frac{\gamma_{\theta,t-1+\ell}((x_{1:t-1}^{i}, x_{t:t-1+\ell}^{\prime}))}{\gamma_{\theta,t-1}(x_{1:t-1}^{i})} \\
= w_{t-1}^{i} \prod_{s=t}^{t-1+\ell} g_{\theta}(y_{s} \mid x_{1:t-1}^{i}, x_{t:s}^{\prime}) f_{\theta}(x_{s}^{\prime} \mid x_{1:t-1}^{i}, x_{t:s-1}^{\prime}).$$
(39)

Let $\hat{\rho}_{\ell}(k)$ be the probability distribution defined by the truncated ancestor sampling weights (39), analogously to (37). The following proposition formalizes our assumption.

Proposition 2. Let

$$h_s(k) = g_{\theta}(y_{t-1+s} \mid x_{1:t-1}^k, x_{t:t-1+s}') f_{\theta}(x_{t-1+s}' \mid x_{1:t-1}^k, x_{t:t-1+s}')$$

and assume that $\max_{k,l} (h_s(k)/h_s(l) - 1) \leq A \exp(-cs)$, for some constants A and c > 0. Then, $D_{\mathsf{KLD}}(\rho \| \widehat{\rho}_\ell) \leq C \exp(-c\ell)$ for some constant C, where D_{KLD} is the Kullback-Leibler (KL) divergence.

Proof. See Appendix A.
$$\Box$$

Using the approximation given by (39), the ancestor sampling weights can be computed in constant time within the PGAS framework. The resulting approximation can be quite useful; indeed, in our experiments we have seen that even $\ell=1$ can lead to very accurate inferential results. In general, however, it will not be known a priori how to set the truncation level ℓ . To address this problem, we propose to use an adaption of the truncation level. Since the approximative weights (39) can be evaluated sequentially, the idea is to start with $\ell=1$ and then increase ℓ until the weights have, in some sense, converged. In particular, in our experimental work, we have used the following simple approach.

At time t, let $\varepsilon_\ell = D_{\mathsf{TV}}(\widehat{\rho}_\ell,\widehat{\rho}_{\ell-1})$ be the total variation (TV) distance between the approximative ancestor sampling distributions for two consecutive truncation levels. We then compute the exponentially decaying moving average of the sequence ε_ℓ , with forgetting factor $v \in [0, 1]$, and stop when this falls below some threshold $\tau \in [0, 1]$. This adaption scheme removes the requirement to specify ℓ directly, but instead introduces the design parameters v and τ . However, these parameters are much easier to reason about—a small value for v gives a rapid response to changes in ε_ℓ whereas a large value gives a more conservative stopping rule, improving the accuracy of the approximation at the cost of higher computational complexity. A similar tradeoff holds for the threshold τ as well. Most importantly, we have found that the same values for v and τ can be used for a wide range of models, with very different mixing properties.

To illustrate the effect of the adaption rule, and how the distribution $\widehat{\rho}_{\ell}$ typically evolves as we increase ℓ , we provide two examples in Figure 3. These examples are taken from the simulation study provided in Section 7.2. Note that the untruncated distribution ρ is given for the maximal value of ℓ , i.e.,

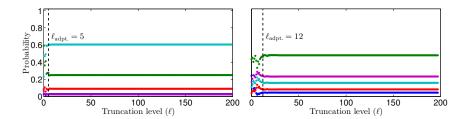


Figure 3: Probability under $\hat{\rho}_{\ell}$ as a function of the truncation level ℓ for two different systems; one 5 dimensional (left) and one 20 dimensional (right). The N=5 dotted lines correspond to $\hat{\rho}_{\ell}(k)$ for $k \in \{1, \ldots, N\}$, respectively (N.B. two of the lines overlap in the left figure). The dashed vertical lines show the value of the truncation level $\ell_{\text{adpt.}}$, resulting from the adaption scheme with v=0.1 and $\tau=10^{-2}$. See Section 7.2 for details on the experiments.

furthest to the right in the figures. By using the adaptive truncation, we can stop the evaluation of the weights at a much earlier stage, and still obtain an accurate approximation of ρ .

The approximation (39) can be used in a few different ways. First, as discussed above, we can simply replace ρ with $\hat{\rho}_{\ell}$ in the PGAS algorithm, resulting in a total computational cost of $O(NT\ell)$. This is the approach that we have favored, owing to its simplicity and the fact that we have found the truncation to lead to very accurate approximations. Another approach, however, is to use $\hat{\rho}_{\ell}$ as an efficient proposal distribution for the MH algorithm suggested in Section 6.2. The MH accept/reject decision will then compensate for the approximation error caused by the truncation. A third approach is to use the MH algorithm, but to make use of the approximation (39) when evaluating the acceptance probability (38). By doing so, the algorithm can be implemented with $O(NT + T\ell)$ computational complexity.

7 Numerical evaluation

In this section we illustrate the properties of PGAS in a simulation study. First, in Section 7.1 we consider a simple linear Gaussian SSM and investigate the improvement in mixing offered by ancestor sampling when PGAS is compared with PG. We do not consider PGBS in this example since, by Proposition 1, PGAS and PGBS are probabilistically equivalent in this scenario.

When applied to non-Markovian models, however, Proposition 1 does not apply since the weight function will depend on the complete history of the particles. PGAS and PGBS will then have different properties as is illustrated empirically in Section 7.2 where we consider inference in degenerate SSMs reformulated as non-Markovian models. Finally, in Section 7.3 we use a similar reformulation and apply PGAS for identification of an epidemiological model for which the transition kernel is not available.

7.11st order LGSS model

Consider a first-order linear Gaussian state-space (LGSS) model,

$$x_{t+1} = ax_t + v_t, v_t \sim \mathcal{N}(0, q), (40a)$$

$$y_t = x_t + e_t, e_t \sim \mathcal{N}(0, r), (40b)$$

$$y_t = x_t + e_t, e_t \sim \mathcal{N}(0, r), (40b)$$

with initial state $p(x_1) = \mathcal{N}(x_1; 0, q/(1-a^2))$ and unknown parameters $\theta =$ (a,q,r). This system is used as a proof of concept to illustrate the superior mixing of PGAS when compared to PG. For this system it is possible to implement an *ideal Gibbs* sampler, i.e., by iteratively sampling from the posterior parameter distribution $p(\theta \mid x_{1:T}, y_{1:T})$ and from the full joint smoothing distribution $p_{\theta}(x_{1:T} \mid y_{1:T})$. This is useful for comparison, since the ideal Gibbs sampler is the baseline for both PG samplers.

We simulate the system (40) for T = 100 time steps with $\theta = (0.8, 1, 0.5)$. We then run the PG [6] and the PGAS (Algorithm 3) samplers with different number of particles, $N \in \{5, 20, 100, 1000\}$, as well as the ideal Gibbs sampler. All methods are initialized at $\theta[0] = (-0.8, 0.5, 1)$ and simulated for 50 000 iterations, whereafter the first 10000 samples are discarded as burn-in. To evaluate the mixing of the samplers, we compute the autocorrelation functions (ACFs) for the sequences $\theta[n] - \mathbb{E}[\theta \mid y_{1:T}]^3$. The results for the parameter q are reported in Figure 4 (top row). Similar results hold for a and r as well. We see that the PG sampler requires a large N to obtain good mixing. For N=100 the ACF drops off much slower than for the ideal sampler and for $N \leq 20$ the ACF is more or less constant. For PGAS, on the other hand, the ACF is much more robust to the choice of N. Indeed, we obtain a mixing which is comparable to that of the ideal Gibbs sampler for any number of particles N > 5.

To further investigate the robustness of PGAS we repeat the same experiment with a larger data batch consisting of $T=2\,000$ samples. The results are given in Figure 4 (bottom row). The effect can be seen even more clearly in this more challenging scenario. The big difference in mixing between the two samplers can be understood as a manifestation of how they are affected by path degeneracy. These results are in agreement with the discussion in Section 3.3.

7.2Degenerate LGSS models

Many dynamical systems are most naturally modeled as degenerate in the sense that the transition kernel of the state-process does not admit any density with respect to a dominating measure. It is problematic to use (particle-filter-based) backward sampling methods for these models, owing to the fact that the backward kernel of the state process will also be degenerate. As a consequence, it is not possible to approximate the backward kernel using the forward filter

 $^{^3}$ The "true" posterior mean is computed as the sample mean obtained from the ideal Gibbs sampler.

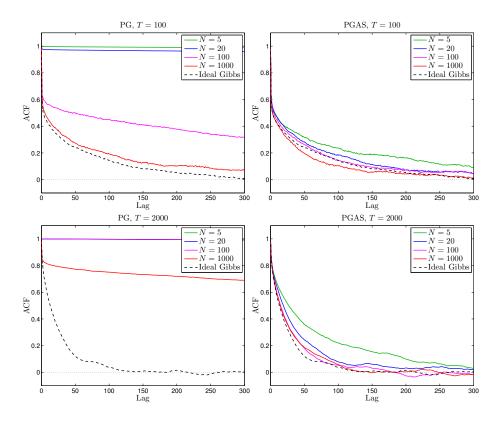


Figure 4: ACFs the parameter q for PG (left column) and for PGAS (right column) for T=100 (top row) and $T=2\,000$ (bottom row). The results are reported for different number of particles N, as well as for the ideal Gibbs sampler. (This figure is best viewed in color.)

To illustrate how this difficulty can be remedied by a change of variables, consider an LGSS model of the form

$$\begin{pmatrix} x_{t+1} \\ z_{t+1} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_t \\ z_t \end{pmatrix} + \begin{pmatrix} v_t \\ 0 \end{pmatrix}, \qquad v_t \sim \mathcal{N}(0, Q), \qquad (41a)$$
$$y_t = C \begin{pmatrix} x_t \\ z_t \end{pmatrix} + e_t, \qquad e_t \sim \mathcal{N}(0, R). \qquad (41b)$$

$$y_t = C \begin{pmatrix} x_t \\ z_t \end{pmatrix} + e_t,$$
 $e_t \sim \mathcal{N}(0, R).$ (41b)

Since the Gaussian process noise enters only on the first part of the state vector (or, equivalently, the process noise covariance matrix is rank deficient) the state transition kernel is degenerate. However, for the same reason, the state component z_t is $\sigma(x_{1:t})$ -measurable and we can write $z_t = z_t(x_{1:t})$. Therefore, it is possible to rephrase (41) as a non-Markovian model with latent process given by $\{x_t\}_{t>1}$.

As a first illustration, we simulate T = 200 samples from a fourth-order, single output system with poles at -0.65, -0.12, and $0.22 \pm 0.10i$. We let $\dim(x_t) = 1$ and Q = R = 0.1. For simplicity, we assume that the system parameters are known and seek the joint smoothing distribution $p(x_{1:T} \mid y_{1:T})$. In the non-Markovian formulation it is possible to apply backward-simulationbased methods, such as PGAS and PGBS, as described in Section 6. The problem, however, is that the non-Markovianity gives rise to an $O(T^2)$ computational complexity. To obtain more practical inference algorithms we employ the weight truncation strategy (39).

First, we consider the coarse approximation $\ell = 1$. We run PGAS and PGBS, both with N=5 particles for 10000 iterations (with the first 1000 discarded as burn-in). We then compute running means of the latent variables $x_{1:T}$ and, from these, we compute the running root mean squared errors (RMSEs) ϵ_n relative to the true posterior means (computed with a modified Bryson-Frazier smoother [47]). Hence, if no approximation would have been made, we would expect $\epsilon_n \to 0$, so any static error can be seen as the effect of the truncation. The results for five independent runs are shown in Figure 5. First, we note that both methods give accurate results. Still, the error for PGAS is significantly lower than for PGBS. For further comparison, we also run an untruncated forward filter/backward simulator (FFBS) particle smoother [28], using $N=10\,000$ particles and M = 1000 backward trajectories (with a computational cost of $O(NMT^2)$). The resulting RMSE value is shown as a thick gray line in Figure 5. This result suggest that PGAS can be a serious competitor to more "classical" particle smoothers, even when there are no unknown parameters of the model. Already with $\ell = 1$, PGAS outperforms FFBS in terms of accuracy and, due to the fact that ancestor sampling allows us to use as few as N=5 particles at each iteration, at a much lower computational cost.

To see how the samplers are affected by the choice of truncation level ℓ and by the mixing properties of the system, we consider randomly generated systems of the form (41) of different orders. We generate 150 random systems, using the matlab function drss from the Control Systems Toolbox, with model orders 2, 5 and 20 (50 systems for each model order). The number of outputs are taken

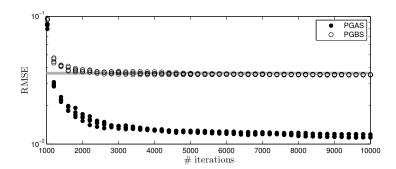


Figure 5: Running RMSEs for $x_{1:T}$ for five independent runs of PGAS (\bullet) and PGBS (\circ), respectively. The truncation level is set to $\ell = 1$. The thick gray line corresponds to a run of an untruncated FFBS particle smoother.

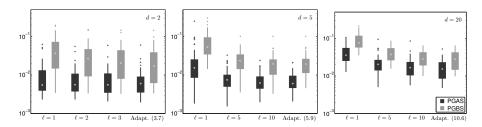


Figure 6: Box plots of the RMSE errors for PGAS (black) and PGBS (gray), for 150 random systems of different dimensions d (left, d=2; middle, d=5; right, d=20). Different values for the truncation level ℓ are considered. The rightmost boxes correspond to an adaptive truncation and the values in parentheses are the average truncation levels over all systems and MCMC iterations (the same for both methods). The dots within the boxes show the median errors.

as 1, 2 and 4 for the different model orders, respectively. We consider different fixed truncation levels ($\ell \in \{1, 2, 3\}$ for 2nd order systems and $\ell \in \{1, 5, 10\}$ for 5th and 20th order systems), as well as an adaptive level with v = 0.1 and $\tau = 10^{-2}$ (see Section 6.3). All other settings are as above.

Again, we compute the posterior means of $x_{1:T}$ (discarding 1000 samples) and RMSE values relative the true posterior mean. Box plots over the different systems are shown in Figure 6. Since the process noise only enters on one of the state components, the mixing tends to deteriorate as we increase the model order. Figure 3 shows how the probability distributions on $\{1, \ldots, N\}$ change as we increase the truncation level, in two representative cases for a 5th and a 20th order system, respectively. By using an adaptive level, we can obtain accurate results for systems of different dimensions, without having to change any settings between the runs.

7.3 Epidemiological model

As a final numerical illustration, we consider identification of an epidemiological model using PGAS. Seasonal influenza epidemics each year cause millions of severe illnesses and hundreds of thousands of deaths world-wide [48]. Furthermore, new strains of influenza viruses can possibly cause pandemics with very severe effects on the public health. The ability to accurately predict disease activity can enable early response to such epidemics, which in turn can reduce their impact.

We consider a susceptible/infected/recovered (SIR) model with environmental noise and seasonal fluctuations [10,49]. The model, specified by a stochastic differential equation, is discretized according to the Euler-Maruyama method, yielding

$$S_{t+dt} = S_t + \mu \mathcal{P}dt - \mu S_t dt - (1 + Fv_t) \beta_t \frac{S_t}{\mathcal{P}} I_t dt, \qquad (42a)$$

$$I_{t+dt} = I_t - (\gamma + \mu)I_t dt + (1 + Fv_t) \beta_t \frac{S_t}{\mathcal{P}} I_t dt, \tag{42b}$$

$$R_{t+dt} = R_t + \gamma I_t dt - \mu R_t dt, \tag{42c}$$

where $v_t \sim \mathcal{N}(0, 1/\sqrt{dt})$ and dt is the sampling time. Here, S_t , I_t and R_t represent the number of susceptible, infected and recovered individuals at time t (months), respectively. The total population size $\mathcal{P} = 10^6$ and the host birth/death rate $\mu = 0.0012$ are assumed known. The seasonally varying transmission rate is given by $\beta_t = R_0(\gamma + \mu)(1 + \alpha \sin(2\pi t/12))$ where R_0 is the basic reproductive ratio, γ is the rate of recovery and α is the strength of seasonality.

Furthermore, we consider an observation model which is inspired by the Google Flu Trends project [48]. The idea is to use the frequency of influenzarelated search engine queries to infer knowledge of the dynamics of the epidemic. Let Q_k be the proportion of influenza-related queries counted during a time interval $(\Delta(k-1), \Delta k]$. Following [48], we use a linear relationship between the log-odds of the relative query counts and the log-odds of the proportion of infected individuals,

$$y_k \triangleq \text{logit}(Q_k) = \rho \, \text{logit}(\bar{I}_k/\mathcal{P}) + e_k, \qquad e_k \sim \mathcal{N}(0, \sigma^2),$$
 (43)

where \bar{I}_k is the mean value of I_t during the time interval $(\Delta(k-1), \Delta k]$ and $\log \operatorname{it}(p) = \log(p/(1-p))$. As in [48] we consider weekly query counts, i.e., $\Delta = 7/30$ (assuming for simplicity that we have 30 days in each month). Using this value of Δ as sampling time will, however, result in an overly large discretization errors. Instead, we sample the model (42) m=7 times per week: $dt = \Delta/m$.

In [10], the particle marginal MH sampler [6] is used to identify a similar SIR model, though with a different observation model. A different Monte Carlo strategy, based on a particle filter with an augmented state space, for identification of an SIR model is proposed in [50]. We suggest to use the PGAS algorithm for joint state and parameter inference in the model (42)–(43). However, there are two difficulties in applying PGAS directly to this

model. Firstly, the transition kernel of the state process, as defined between consecutive observation time points $\Delta(k-1)$ and Δk , is not available in closed form. Secondly, since the state is three-dimensional, whereas the driving noise v_t is scalar, the transition kernel is degenerate. To cope with these difficulties we (again) suggest collapsing the model to the driving noise variables. Let $V_k = \begin{pmatrix} v_{\Delta(k-1)} & v_{\Delta(k-1)+dt} & \cdots & v_{\Delta k-dt} \end{pmatrix}^T$. It follows that the model (42)–(43) can be equivalently expressed as the non-Markovian latent variable model,

$$V_k \sim \mathcal{N}(0, I_m / \sqrt{dt}),$$
 (44a)

$$y_k \sim g_\theta(y_k \mid V_{1:k}),\tag{44b}$$

for some likelihood function g_{θ} (see (46)). A further motivation for using this reformulation is that the latent variables V_k are a priori independent of the model parameters θ . This can result in a significant improvement in mixing of the Gibbs sampler, in particular when there are strong dependencies between the system state the parameters [41,51].

The parameters of the model are $\theta = (\gamma, R_0, \alpha, F, \rho, \sigma)$, with the true values given by $\gamma = 3$, $R_0 = 10$, $\alpha = 0.16$, F = 0.03, $\rho = 1.1$ and $\sigma = 0.224$. We place a normal-inverse-gamma prior on the pair (ρ, σ^2) , i.e., $p(\rho, \sigma^2) = \mathcal{N}(\rho; \mu_\rho, c_\rho \sigma^2) \mathcal{I} \mathcal{G}(\sigma^2; a_\sigma, b_\sigma)$. The hyperparameters are chosen as $\mu_\rho = 1$, $c_\rho = 0.5$ and $a_\sigma = b_\sigma = 0.01$. For the remaining parameters, we use improper flat priors on \mathbb{R}_+ .

We generate 8 years of data with weekly observations. The number of infected individuals I_t over this time period is shown in Figure 8. The first half of the data batch is used for estimation of the model parameters. We run PGAS with N=10 for 50 000 iterations (discarding the first 10 000). For sampling the system parameters (γ, R_0, α, F) , we use Metropolis-Hastings steps with a Gaussian random walk proposal, tuned according to an initial trial run. For (ρ, σ^2) , we exploit the conjugacy of the normal-inverse-gamma prior to the likelihood (43) and sample the variables from their true posterior. The innovation variables $V_{1:T}$ are sampled from the PGAS kernel by Algorithm 2 (no truncation is used for the ancestor sampling weights). Since the latter step is the computational bottleneck of the algorithm, we execute ten MH steps for θ , for each draw from the PGAS kernel.

It is worth pointing out that, while the sampler effectively targets the collapsed model (44), it is most straightforwardly implemented using the original state variables from (42). With $x_k = (S_{\Delta k}, I_{\Delta k}, R_{\Delta k})^{\mathsf{T}}$ we can simulate x_{k+1} given x_k according to (42) which is used in the underlying particle filter. The innovation variables V_k need only be taken into account for the ancestor sampling step. Let $V'_{1:T}$ be the reference innovation trajectory. To compute the ancestor sampling weights (7) we need to evaluate the ratios,

$$\frac{p_{\theta}((V_{1:k-1}^{i}, V_{k:T}^{\prime}), y_{1:T})}{p_{\theta}(V_{1:k-1}^{i}, y_{1:k-1})} \propto \prod_{\ell=k}^{T} g_{\theta}(y_{\ell} \mid V_{1:k-1}^{i}, V_{k:\ell}^{\prime}). \tag{45}$$

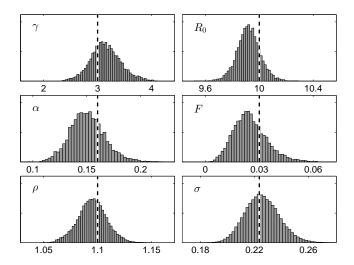


Figure 7: Posterior densities for the parameters of model (42)–(43). The true values are marked by vertical dashed lines.

Using (43), the observation likelihood can be written as

$$g_{\theta}(y_{\ell} \mid V_{1:k-1}^{i}, V_{k:\ell}') = \mathcal{N}(y_{\ell} \mid \rho \operatorname{logit}(\bar{I}_{\ell}\{x_{k-1}^{i}, V_{k:\ell}'\}/\mathcal{P}), \sigma^{2}),$$
 (46)

where $I_{\ell}\{x_{k-1}^i, V_{k:\ell}'\}$ is obtained by simulating the system (42) from time $\Delta(k-1)$ to time $\Delta\ell$, initialized at x_{k-1}^i and using the innovation sequence $V_{k:\ell}'$.

Histograms representing the estimated posterior parameter distributions are shown in Figure 7. As can be seen, the true system parameters fall well within the credible regions. Finally, the identified model is used to make one-month-ahead predictions of the disease activity for the subsequent four years, as shown in Figure 8. The predictions are made by sub-sampling the Markov chain and, for each sample, running a particle filter on the validation data using 100 particles. As can be seen, we obtain an accurate prediction of the disease activity, which falls within the estimated 95 % credibility intervals, one month in advance.

8 Discussion

PGAS is a novel approach to PMCMC that provides the statistician with an off-the-shelf class of Markov kernels which can be used to simulate, for instance, the typically high-dimensional and highly autocorrelated state trajectory in a state-space model. This opens up for using PGAS as a key component in different inference algorithms, enabling both Bayesian and frequentist parameter inference as well as state inference. However, PGAS by no means limited to inference in state-space models. Indeed, we believe that the method can be

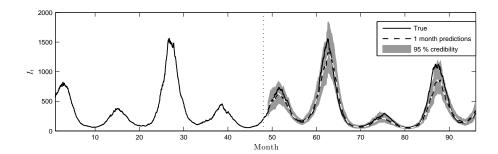


Figure 8: Disease activity (number of infected individuals I_t) over an eight year period. The first four years are used as estimation data, to find the unknown parameters of the model. For the consecutive four years, one-month-ahead predictions are computed using the estimated model.

particularly useful for models with more complex dependencies, such as non-Markovian, nonparametric, and graphical models.

The PGAS Markov kernels are built upon two main ideas. First, by conditioning the underlying SMC sampler on a reference trajectory the correct stationary distribution of the kernel is enforced. Second, *ancestor sampling* enables movement around the reference trajectory which drastically improves the mixing of the sampler. In particular, we have shown empirically that ancestor sampling makes the mixing of the PGAS kernels robust to a small number of particles as well as to large data records.

Ancestor sampling is basically a way of exploiting backward simulation ideas without needing an explicit backward pass. Compared to PGBS, a conceptually similar method that does require an explicit backward pass, PGAS has several advantages, most notably for inference in non-Markovian models. When using the proposed truncation of the backward weights, we have found PGAS to be more robust to the approximation error than PGBS, yielding up to an order-of-magnitude improvement in accuracy. An interesting topic for future work is to further investigate the effect on these samplers by errors in the backward weights, whether these errors arise from a truncation or some other approximation of the transition density function. It is also worth pointing out that for non-Markovian model PGAS is simpler to implement than PGBS as it requires less bookkeeping. It can also be more memory efficient since it does not require storage of intermediate quantities that are needed for a separate backward simulation pass. See [52] for a related discussion on path storage in the particle filter.

Other directions for future work include further analysis of the ergodicity of PGAS. While the established uniform ergodicity result is encouraging, it does not provide information about how fast the mixing rate improves with the number of particles. Finding informative rates with an explicit dependence on N is an interesting, though challenging, topic for future work. It would also be

interesting to further investigate empirically the convergence rate of PGAS for different settings, such as the number of particles, the amount of data, and the dimension of the latent process.

A Proofs

A.1 Proof of Proposition 1

In this appendix we prove Proposition 1. For improved readability we provide the complete proof, though it should be noted that the main part is due to [30]. For ease of notation, we write \mathbb{E} for $\mathbb{E}_{\theta,x_{1:T}'}$. First, note that for a bootstrap proposal kernel, the weight function (5) is given by $W_{\theta,t}(x_t) = g_{\theta}(y_t \mid x_t)$, i.e., it depend only on the current state and not on its ancestor. As a consequence, the law of the forward particle system is independent of the ancestor variables $\{a_t^N\}_{t=2}^T$, meaning that the particle systems (excluding $\{a_t^N\}_{t=2}^T$) are equally distributed for PGAS and for PGBS.

Let $B \in \mathcal{X}^T$ be a measurable rectangle: $B = \times_{t=1}^T B_t$ with $B_t \in \mathcal{X}$ for t = 1, ..., T. Then,

$$P_{\theta}^N(x_{1:T}',B) = \mathbb{E}\left[\prod_{t=1}^T \mathbbm{1}_{B_t}(x_t^{b_t})\right], \quad \text{and} \quad P_{\mathsf{BS},\theta}^N(x_{1:T}',B) = \mathbb{E}\left[\prod_{t=1}^T \mathbbm{1}_{B_t}(x_t^{j_t})\right].$$

Since the measurable rectangles form a π -system generating \mathcal{X}^T , it is by the π - λ theorem sufficient to show that $\mathbb{E}[h(x_{1:T}^{b_{1:T}})] = \mathbb{E}[h(x_{1:T}^{j_{1:T}})]$ for all bounded, multiplicative functionals, $h(x_{1:T}) = \prod_{t=1}^T h_t(x_t)$. As in [30], we establish this result by induction. Hence, for t < T, assume that

$$\mathbb{E}\left[\prod_{s=t+1}^{T} h_s(x_s^{b_s})\right] = \mathbb{E}\left[\prod_{s=t+1}^{T} h_s(x_s^{j_s})\right]. \tag{47}$$

For t = T - 1, the induction hypothesis holds since b_T and j_T are equally distributed (both are drawn from the discrete distribution induced by the weights $\{w_T^i\}_{i=1}^N$). Let

$$\Lambda_{t}(x_{t+1}^{j_{t+1}}, h) \triangleq \mathbb{E}\left[h(x_{t}^{j_{t}}) \mid x_{t+1}^{j_{t+1}}\right] = \mathbb{E}\left[\mathbb{E}\left[h(x_{t}^{j_{t}}) \mid \mathbf{x}_{t}, x_{t+1}^{j_{t+1}}\right] \mid x_{t+1}^{j_{t+1}}\right] \\
= \mathbb{E}\left[\sum_{i=1}^{N} h(x_{t}^{i}) \frac{w_{t}^{i} f_{\theta}(x_{t+1}^{j_{t+1}} \mid x_{t}^{i})}{\sum_{l} w_{t}^{l} f_{\theta}(x_{t+1}^{j_{t+1}} \mid x_{t}^{l})} \mid x_{t+1}^{j_{t+1}}\right],$$
(48)

where we recall that $w_t^i = W_{\theta,t}(x_t^i)$ and where the last equality follows from (34). Consider,

$$\mathbb{E}\left[\prod_{s=t}^{T} h_s(x_s^{b_s})\right] = \mathbb{E}\left[\mathbb{E}\left[h_t(x_t^{b_t}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] \prod_{s=t+1}^{T} h_s(x_s^{b_s})\right]. \tag{49}$$

Using the Markov property of the generated particle system and the tower property of conditional expectation, we have

$$\mathbb{E}\left[h_{t}(x_{t}^{b_{t}}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] = \mathbb{E}\left[\mathbb{E}\left[h_{t}(x_{t}^{b_{t}}) \mid \mathbf{x}_{t}, x_{t+1}^{b_{t+1}}, b_{t+1}\right] \mid x_{t+1}^{b_{t+1}}, b_{t+1}\right].$$
(50)

Recall that $b_t = a_{t+1}^{b_{t+1}}$. Consider first the case $b_{t+1} < N$. From (3), we have that $\mathbb{P}(b_t = i \mid \mathbf{x}_t) \propto w_t^i$ and $x_{t+1}^{b_{t+1}} \mid x_t^{b_t} \sim f_{\theta}(\cdot \mid x_t^{b_t})$. If follows from Bayes' theorem that $\mathbb{P}(b_t = i \mid \mathbf{x}_t, x_{t+1}^{b_{t+1}}) \propto w_t^i f_{\theta}(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})$. However, by the ancestor sampling procedure (Algorithm 2, line 8), the same expression holds also for $b_{t+1} = N$. We can thus write (50) as

$$\mathbb{E}\left[h_{t}(x_{t}^{b_{t}}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] = \mathbb{E}\left[\sum_{i=1}^{N} h_{t}(x_{t}^{i}) \frac{w_{t}^{i} f_{\theta}(x_{t+1}^{b_{t+1}} \mid x_{t}^{i})}{\sum_{l} w_{t}^{l} f_{\theta}(x_{t+1}^{b_{t+1}} \mid x_{t}^{l})} \mid x_{t+1}^{b_{t+1}}, b_{t+1}\right] \\
= \Lambda_{t}(x_{t+1}^{b_{t+1}}, h_{t}), \tag{51}$$

Hence, since the function $x_{t+1} \mapsto \Lambda_t(x_{t+1}, h_t)$ is bounded, we can use the induction hypothesis to write (49) as

$$\mathbb{E}\left[\prod_{s=t}^{T} h_s(x_s^{b_s})\right] = \mathbb{E}\left[\Lambda_t(x_{t+1}^{b_{t+1}}, h_t) \prod_{s=t+1}^{T} h_s(x_s^{b_s})\right] = \mathbb{E}\left[\Lambda_t(x_{t+1}^{j_{t+1}}, h_t) \prod_{s=t+1}^{T} h_s(x_s^{j_s})\right]$$

$$= \mathbb{E}\left[\mathbb{E}\left[h_t(x_t^{j_t}) \mid x_{t+1:T}^{j_{t+1:T}}, j_{t+1:T}\right] \prod_{s=t+1}^{T} h_s(x_s^{j_s})\right] = \mathbb{E}\left[\prod_{s=t}^{T} h_s(x_s^{j_s})\right].$$

A.2 Proof of Proposition 2

With M = T - t + 1 and $w(k) = w_{t-1}^k$, the distributions of interest are given by

$$\rho(k) = \frac{w(k) \prod_{s=1}^{M} h_s(k)}{\sum_{l} w(l) \prod_{s=1}^{M} h_s(l)} \quad \text{and} \quad \widehat{\rho}_{\ell}(k) = \frac{w(k) \prod_{s=1}^{\ell} h_s(k)}{\sum_{l} w(l) \prod_{s=1}^{\ell} h_s(l)},$$

respectively. Let $\varepsilon_s \triangleq \max_{k,l} (h_s(k)/h_s(l) - 1) \leq A \exp(-cs)$ and consider

$$\left(\sum_{l} w(l) \prod_{s=1}^{\ell} h_s(l)\right) \prod_{s=\ell+1}^{M} h_s(k) \leq \sum_{l} \left(w(l) \prod_{s=1}^{\ell} h_s(l) \prod_{s=\ell+1}^{M} h_s(l)(1+\varepsilon_s)\right)$$
$$= \left(\sum_{l} w(l) \prod_{s=1}^{M} h_s(l)\right) \prod_{s=\ell+1}^{M} (1+\varepsilon_s).$$

It follows that the KL divergence is bounded according to,

$$\begin{split} D_{\mathsf{KLD}}(\rho \| \widehat{\rho}_{\ell}) &= \sum_{k} \rho(k) \log \frac{\rho(k)}{\widehat{\rho}_{\ell}(k)} \\ &= \sum_{k} \rho(k) \log \left(\frac{\prod_{s=\ell+1}^{M} h_{s}(k) \left(\sum_{l} w(l) \prod_{s=1}^{\ell} h_{s}(l) \right)}{\sum_{l} w(l) \prod_{s=1}^{M} h_{s}(l)} \right) \\ &\leq \sum_{k} \rho(k) \sum_{s=\ell+1}^{M} \log(1 + \varepsilon_{s}) \leq \sum_{s=\ell+1}^{M} \varepsilon_{s} \leq A \sum_{s=\ell+1}^{M} \exp(-cs) \\ &= A \frac{e^{-c(\ell+1)} - e^{-c(M+1)}}{1 - e^{-c}}. \end{split}$$

References

[1] A. Doucet and A. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*, D. Crisan and B. Rozovskii, Eds. Oxford University Press, 2011.

- [2] P. Del Moral, A. Doucet, and A. Jasra, "Sequential Monte Carlo samplers," *Journal of the Royal Statistical Society: Series B*, vol. 68, no. 3, pp. 411–436, 2006.
- [3] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2004.
- [4] J. S. Liu, Monte Carlo Strategies in Scientific Computing. Springer, 2001.
- [5] F. Lindsten and T. B. Schön, "Backward simulation methods for Monte Carlo statistical inference," Foundations and Trends in Machine Learning, vol. 6, no. 1, pp. 1–143, 2013.
- [6] C. Andrieu, A. Doucet, and R. Holenstein, "Particle Markov chain Monte Carlo methods," *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 3, pp. 269–342, 2010.
- [7] J. A. Vrugt, J. F. ter Braak, C. G. H. Diks, and G. Schoups, "Hydrologic data assimilation using particle Markov chain Monte Carlo simulation: Theory, concepts and applications," *Advances in Water Resources*, vol. 51, pp. 457–478, 2013.
- [8] M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn, "On some properties of Markov chain Monte Carlo simulation methods based on the particle filter," *Journal of Econometrics*, vol. 171, pp. 134–151, 2012.

- [9] A. Golightly and D. J. Wilkinson, "Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo," *Interface Focus*, vol. 1, no. 6, pp. 807–820, 2011.
- [10] D. A. Rasmussen, O. Ratmann, and K. Koelle, "Inference for nonlinear epidemiological models using genealogies and time series," *PLoS Comput Biology*, vol. 7, no. 8, 2011.
- [11] N. Chopin and S. S. Singh, "On the particle Gibbs sampler," arXiv.org, arXiv:1304.1887, Apr. 2013.
- [12] N. Whiteley, C. Andrieu, and A. Doucet, "Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods," Bristol Statistics Research Report 10:04, Tech. Rep., 2010.
- [13] F. Lindsten and T. B. Schön, "On the use of backward simulation in the particle Gibbs sampler," in *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012.
- [14] F. Lindsten, M. I. Jordan, and T. B. Schön, "Ancestor sampling for particle Gibbs," in Advances in Neural Information Processing Systems (NIPS) 25, P. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 2600–2608.
- [15] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999.
- [16] M. Chesney and L. Scott, "Pricing european currency options: A comparison of the modified Black-Scholes model and a random variance model," The Journal of Financial and Quantitative Analysis, vol. 24, no. 3, pp. 267–284, 1989.
- [17] A. Melino and S. M. Turnbull, "Pricing foreign currency options with stochastic volatility," *Journal of Econometrics*, vol. 45, no. 1–2, pp. 239–265, 1990.
- [18] D. A. V. Dyk and T. Park, "Partially collapsed Gibbs samplers: Theory and methods," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 790–796, 2008.
- [19] L. Tierney, "Markov chains for exploring posterior distributions," *The Annals of Statistics*, vol. 22, no. 4, pp. 1701–1728, 1994.
- [20] P. de Jong and N. Shephard, "The simulation smoother for time series models," *Biometrika*, vol. 82, no. 2, pp. 339–350, 1995.
- [21] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, *Series B*, vol. 39, no. 1, pp. 1–38, 1977.

- [22] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, 2nd ed., ser. Whiley Series in Probability and Statistics. New York, USA: John Wiley & Sons, 2008.
- [23] G. C. G. Wei and M. A. Tanner, "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 699–704, 1990.
- [24] B. Delyon, M. Lavielle, and E. Moulines, "Convergence of a stochastic approximation version of the EM algorithm," *The Annals of Statistics*, vol. 27, no. 1, pp. 94–128, 1999.
- [25] A. Benveniste, M. Métivier, and P. Priouret, *Adaptive algorithms and stochastic approximations*. New York, USA: Springer-Verlag, 1990.
- [26] C. Andrieu, E. Moulines, and P. Priouret, "Stability of stochastic approximation under verifiable conditions," SIAM Journal on Control and Optimization, vol. 44, no. 1, pp. 283–312, 2005.
- [27] F. Lindsten, "An efficient stochastic approximation EM algorithm using conditional particle filters," in *Proceedings of the 38th IEEE International* Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, Canada, May 2013.
- [28] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 156–168, Mar. 2004.
- [29] N. Whiteley, "Discussion on Particle Markov chain Monte Carlo methods," Journal of the Royal Statistical Society: Series B, 72(3), p 306–307, 2010.
- [30] J. Olsson and T. Rydén, "Rao-Blackwellization of particle Markov chain Monte Carlo methods using forward filtering backward sampling," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4606–4619, 2011.
- [31] N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker, Eds., *Bayesian Non-parametrics*. Cambridge University Press, 2010.
- [32] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical Dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.
- [33] M. D. Escobar and M. West, "Bayesian density estiamtion and inference using mixtures," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 577–588, 1995.
- [34] C. E. Rasmussen and C. K. I. Williams, Gaussian Processes for Machine Learning. MIT Press, 2006.

- [35] R. Turner and C. E. Deisenroth, M. P. Rasmussen, "State-space inference and learning with Gaussian processes," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.
- [36] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian inference and learning in Gaussian process state-space models with particle MCMC," in *Advances in Neural Information Processing Systems (NIPS)* 26 (to appear), Dec. 2013.
- [37] R. Chen and J. S. Liu, "Mixture Kalman filters," *Journal of the Royal Statistical Society: Series B*, vol. 62, no. 3, pp. 493–508, 2000.
- [38] F. Lindsten, P. Bunch, S. J. Godsill, and T. B. Schön, "Rao-Blackwellized particle smoothers for mixed linear/nonlinear state-space models," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.
- [39] M. P. S. Gander and D. A. Stephens, "Stochastic volatility modelling in continuous time with general marginal distributions: Inference, prediction and model selection," *Journal of Statistical Planning and Inference*, vol. 137, no. 10, pp. 3068–3081, 2007.
- [40] P. Fearnhead, O. Papaspiliopoulos, and G. O. Roberts, "Particle filters for partially observed diffusions," *Journal of the Royal Statistical Society: Series B*, vol. 70, no. 4, pp. 755–777, 2008.
- [41] A. Golightly and D. J. Wilkinson, "Bayesian inference for nonlinear multivariate diffusion models observed with error," *Computational Statistics & Data Analysis*, vol. 52, no. 3, pp. 1674–1693, 2008.
- [42] L. M. Murray, E. M. Jones, and J. Parslow, "On collapsed state-space models and the particle marginal Metropolis-Hastings sampler," arXiv.org, arXiv:1202.6159v1, Feb. 2012.
- [43] B. Ristic, S. Arulampalam, and N. Gordon, Beyond the Kalman filter: particle filters for tracking applications. London, UK: Artech House, 2004.
- [44] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.
- [45] D. Koller and N. Friedman, *Probabilistic Graphical Models*. MIT Press, 2009.
- [46] J. S. Liu, "Peskun's theorem and a modified discrete-state Gibbs sampler," Biometrika, vol. 83, no. 3, pp. 681–682, 1996.
- [47] G. J. Bierman, "Fixed interval smoothing with discrete measurements," *International Journal of Control*, vol. 18, no. 1, pp. 65–75, 1973.

- [48] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant, "Detecting influenza epidemics using search engine query data," *Nature*, vol. 457, pp. 1012–1014, 2009.
- [49] M. Keeling and P. Rohani, *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press, 2007.
- [50] A. Skvortsov and B. Ristic, "Monitoring and prediction of an epidemic outbreak using syndromic observations," arXiv.org, arXiv:1110.4696v1, Oct. 2011.
- [51] O. Papaspiliopoulos, G. O. Roberts, and M. Sköld, "Non-centered parameterisations for hierarchical models and data augmentation," in *Bayesian Statistics 7*, J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, Eds. Oxford University Press, 2003, pp. 307–326.
- [52] P. E. Jacob, L. Murray, and S. Rubenthaler, "Path storage in the particle filter," arXiv.org, arXiv:1307.3180, Jul. 2013.