Journal of Machine Learning Research 15 (2014) 2145-2184 Submitted 1/14; Revised 4/14; Published 6/14

Particle Gibbs with Ancestor Sampling

Fredrik Lindsten FREDRIK.LINDSTEN@ENG.CAM.AC.UK
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ, UK, and

Division of Automatic Control
Linköping University
Linköping, 581 83, Sweden

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA

Thomas B. Schön THOMAS.SCHON@IT.UU.SE
Department of Information Technology
Uppsala University
Uppsala, 751 05, Sweden

Editor: Yee Whye Teh

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 new 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, for instance, 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.

Keywords: particle Markov chain Monte Carlo, sequential Monte Carlo, Bayesian inference, non-Markovian models, state-space 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, see, e.g., Doucet and Jo-

hansen, 2011; Del Moral et al., 2006) and Markov chain Monte Carlo (MCMC, see, e.g., Robert and Casella, 2004; Liu, 2001) 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 Lindsten and Schön (2013, 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, Andrieu et al., 2010) 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. These kernels can then be used as off-the-shelf components in MCMC algorithms and other inference strategies relying on Markov kernels. PMCMC has in a relatively short period of time found many applications in areas such as hydrology (Vrugt et al., 2013), finance (Pitt et al., 2012), systems biology (Golightly and Wilkinson, 2011), and epidemiology (Rasmussen et al., 2011), to mention a few.

Our method builds on the particle Gibbs (PG) sampler proposed by Andrieu et al. (2010). 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 (Lindsten and Schön, 2013; Chopin and Singh, 2014). 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, Whiteley, 2010; Whiteley et al., 2010; Lindsten and Schön, 2012). 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 (Lindsten and Schön, 2013; Chopin and Singh, 2014; Whiteley et al., 2010; Lindsten and Schön, 2012).

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 complete trajectories of the latent process

during the backward simulation pass (see Section 6.2 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 (AS) step, thereby achieving the same effect as backward sampling, but without an explicit backward pass.

After giving some background on SMC in Section 2, the PGAS Markov kernel is constructed and analyzed theoretically in Sections 3 and 4, respectively. This extends the preliminary work that we have previously published (Lindsten et al., 2012) with a more straightforward construction, a more complete proof of invariance, and a new 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. 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 a 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. We also evaluate PGAS on a stochastic volatility SSM and on an epidemiological model. 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 densities[1] on the measurable space $(\mathsf{X}^t, \mathcal{X}^t)$, parameterized by $\theta \in \Theta$. Let $\bar{\gamma}_{\theta,t}(x_{1:t})$ be the corresponding normalized probability densities:

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

where $Z_{\theta,t} = \int \gamma_{\theta,t}(x_{1:t})\, dx_{1:t}$ 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 $\bar{\gamma}_{\theta,t}(x_{1:t}) = p_\theta(x_{1:t} \mid y_{1:t})$, $\gamma_{\theta,t}(x_{1:t}) = p_\theta(x_{1:t}, y_{1:t})$, and $Z_{\theta,t} = p_\theta(y_{1:t})$. We discuss this special case in more detail in Section 5.

To make 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); see, e.g., Doucet and Johansen (2011); Del Moral et al. (2006); Pitt and Shephard (1999).

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 $\{x^i_{1:t-1}, w^i_{t-1}\}_{i=1}^N$ 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 target distribution given by

$$\widehat{\gamma}^N_{\theta,t-1}(dx_{1:t-1}) = \sum_{i=1}^N \frac{w^i_{t-1}}{\sum_l w^l_{t-1}} \delta_{x^i_{1:t-1}}(dx_{1:t-1}).$$

1. The dominating measure is denoted simply as $dx_{1:t}$.

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

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

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

$$x_{1:t}^i = (x_{1:t-1}^{a_t^i}, x_t^i).$$

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

$$W_{\theta,t}(x_{1:t}) = \frac{\gamma_{\theta,t}(x_{1:t})}{\gamma_{\theta,t-1}(x_{1:t-1}) r_{\theta,t}(x_t \mid x_{1:t-1})}, \tag{2}$$

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 $i = 1, \ldots, N$)

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**
4: Draw $\{a_t^i, x_t^i\} \sim M_{\theta,t}(a_t, x_t)$.
5: Set $x_{1:t}^i = (x_{1:t-1}^{a_t^i}, x_t^i)$.
6: Set $w_t^i = W_{\theta,t}(x_{1:t}^i)$.
7: **end for**

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

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

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

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

3. The PGAS Kernel

We now turn to the construction of PGAS, a class of Markov kernels on the space of trajectories $(\mathsf{X}^T, \mathcal{X}^T)$. 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 Andrieu et al. (2010) 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 $\{a^i_t, x^i_t\}$ are drawn independently from the proposal kernel (1) 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 (1) only for $i = 1, \ldots, N-1$. The Nth particle and its ancestor index are then set deterministically as $x^N_t = x'_t$ and $a^N_t = N$, respectively. This implies that after a complete pass of the algorithm, the Nth particle path coincides with the reference trajectory, i.e., $x^N_{1:T} = x'_{1:T}$.

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 shown by Andrieu et al. (2010, 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 (Lindsten and Schön, 2013; Chopin and Singh, 2014).

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^N_t 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 AS 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

$$\widetilde{w}^i_{t-1|T} \triangleq w^i_{t-1} \frac{\gamma_{\theta,T}((x^i_{1:t-1}, x'_{t:T}))}{\gamma_{\theta,t-1}(x^i_{1:t-1})} \tag{3}$$

for $i = 1, \ldots, N$. Here, $(x^i_{1:t-1}, x'_{t:T})$ refers to the point in X^T formed by concatenating the two partial trajectories. Then, we sample a^N_t with $\mathbb{P}(a^N_t = i) \propto \widetilde{w}^i_{t-1|T}$. The expression above can be understood as an application of Bayes' theorem, where the importance weight w^i_{t-1} is the prior probability of the particle $x^i_{1:t-1}$ and the ratio between the target densities in (3) can be seen as the likelihood that $x'_{t:T}$ originated from $x^i_{1:t-1}$. A formal argument for why (3) provides the correct AS 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 class 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 $a^N_t = N$). 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 $(\mathsf{X}^T, \mathcal{X}^T)$. The class of Markov kernels $\{P^N_\theta : \theta \in \Theta\}$, indexed by $N \geq 1$, is referred to as the PGAS class of kernels.*

Algorithm 2 PGAS Markov kernel

Input: Reference trajectory $x'_{1:T} \in \mathsf{X}^T$ and parameter $\theta \in \Theta$.
Output: Sample $x^\star_{1:T} \sim P^N_\theta(x'_{1:T}, \cdot)$ from the PGAS Markov kernel.

1: Draw $x^i_1 \sim r_{\theta,1}(x_1)$ for $i = 1, \ldots, N-1$.
2: Set $x^N_1 = x'_1$.
3: Set $w^i_1 = W_{\theta,1}(x^i_1)$ for $i = 1, \ldots, N$.
4: **for** $t = 2$ **to** T **do**
5: Draw $\{a^i_t, x^i_t\} \sim M_{\theta,t}(a_t, x_t)$ for $i = 1, \ldots, N-1$.
6: Set $x^N_t = x'_t$.
7: Compute $\{\widetilde{w}^i_{t-1|T}\}_{i=1}^N$ according to (3).
8: Draw a^N_t with $\mathbb{P}(a^N_t = i) \propto \widetilde{w}^i_{t-1|T}$.
9: Set $x^i_{1:t} = (x^{a^i_t}_{1:t-1}, x^i_t)$ for $i = 1, \ldots, N$.
10: Set $w^i_t = W_{\theta,t}(x^i_{1:t})$ for $i = 1, \ldots, N$.
11: **end for**
12: Draw k with $\mathbb{P}(k = i) \propto w^i_T$.
13: **return** $x^\star_{1:T} = x^k_{1:T}$.

3.3 The Effect of Path Degeneracy on PG and on PGAS

We have argued that AS can considerably improve the mixing of PG. To illustrate this effect and to provide an explanation of its cause, we consider a simple numerical example. Further

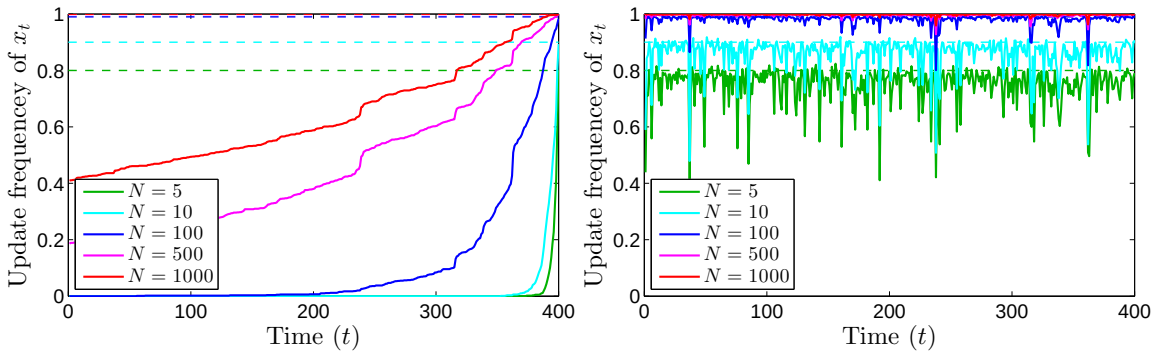

Figure 1: Update rates for x_t versus $t \in \{1, \ldots, 400\}$ for PG (left) and for PGAS (right). The dashed lines correspond to the ideal rates $(N-1)/N$. (This figure is best viewed in color.)

empirical evaluation of PGAS is provided in Section 7. Consider the one-dimensional linear Gaussian state-space (LGSS) model,

$$\begin{aligned} x_{t+1} &= ax_t + v_t, & v_t &\sim \mathcal{N}(0, \sigma_v^2), \\ y_t &= x_t + e_t & e_t &\sim \mathcal{N}(0, \sigma_e^2), \end{aligned}$$

where the state process $\{x_t\}_{t\geq 1}$ is latent and observations are made only via the measurement process $\{y_t\}_{t\geq 1}$. For simplicity, the parameters $\theta = (a, \sigma_v, \sigma_e) = (0.9, 0.32, 1)$ are 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 AS 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., Doucet and Johansen 2011). 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 $x^\star_{1:T}$ (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 AS) is shown in Figure 2 (right). The thick

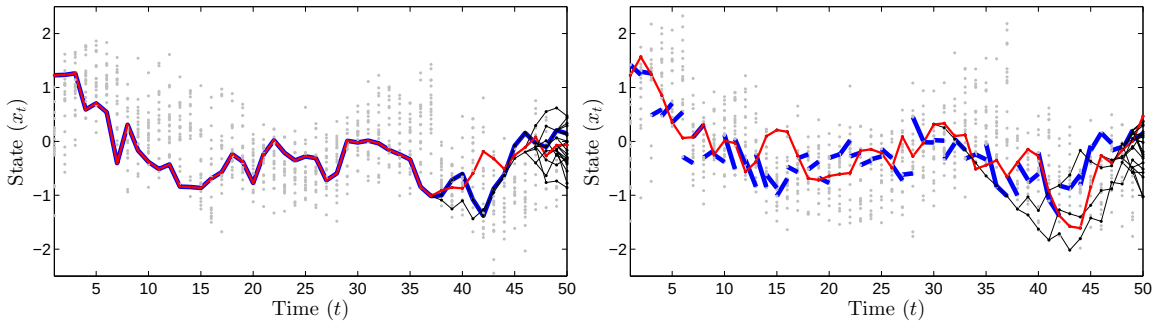

Figure 2: Particle systems generated by the PG algorithm (left) and by the PGAS algorithm (right), for the same reference trajectory $x'_{1:T}$ (shown as a thick blue line in the left panel, partly underneath the red line). The gray dots show the particle positions and the thin black lines show the ancestral dependencies of the particles. The extracted trajectory $x^\star_{1:T}$ is illustrated with a red line. In the right panel, AS 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.)

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 AS is that, informally, the reference trajectory is broken into pieces. It is worth pointing out that the particle system still collapses; AS does not prevent path degeneracy. However, it causes the particle system to degenerate toward something different than the reference trajectory. As a consequence, $x^\star_{1:T}$ (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

In this section we investigate the invariance and ergodicity properties of the PGAS kernel.

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 AS step.

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

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

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

$$P_\theta^N(x'_{1:T}, B) = \mathbb{E}_{\theta, x'_{1:T}}\left[\mathbb{1}_B(x_{1:T}^k)\right], \tag{4}$$

where $\mathbb{1}_B$ is the indicator function for the set $B \in \mathcal{X}^T$ 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 (4), however, we can adopt the strategy employed by Andrieu et al. (2010). That is, we treat 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

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

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 Andrieu et al. (2010), we then define a probability density function $\phi_\theta : \Omega \mapsto \mathbb{R}$ as follows:

$$\begin{aligned}\phi_\theta(\mathbf{x}_{1:T}, \mathbf{a}_{2:T}, k) &= \phi_\theta(x_{1:T}^{b_{1:T}}, b_{1:T})\phi_\theta(\mathbf{x}_{1:T}^{-b_{1:T}}, \mathbf{a}_{2:T}^{-b_{2:T}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) \\ &\triangleq \underbrace{\frac{\bar{\gamma}_{\theta,T}(x_{1:T}^{b_{1:T}})}{N^T}}_{\text{marginal}} \underbrace{\prod_{\substack{i=1 \\ i\neq b_1}}^{N} r_{\theta,1}(x_1^i) \prod_{t=2}^{T} \prod_{\substack{i=1 \\ i\neq b_t}}^{N} M_{\theta,t}(a_t^i, x_t^i)}_{\text{conditional}},\end{aligned} \tag{5}$$

where we have introduced the notation

$$\mathbf{x}_t^{-i} = \{x_t^1, \ldots, x_t^{i-1}, x_t^{i+1}, \ldots, x_t^N\}, \qquad \mathbf{x}_{1:T}^{-b_{1:T}} = \{\mathbf{x}_1^{-b_1}, \ldots, \mathbf{x}_T^{-b_T}\},$$

and similarly for the ancestor indices. By construction, ϕ_θ is nonnegative and integrates to one, i.e., ϕ_θ is indeed 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} corresponding to a uniform distribution over 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* $x'^{,b'_{1:T}}_{1:T} \in \mathsf{X}^T$ *and* $b'_{1:T} \in \{1, \ldots, N\}^T$*:*

(i) *Draw* $\mathbf{x}_1^{-b'_1} \sim \phi_\theta(\,\cdot \mid x'^{,b'_{1:T}}_{1:T}, b'_{1:T})$ *and, for* $t = 2$ *to* T*, draw:*

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

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

Lemma 1. *Algorithm 2 is equivalent to the partially collapsed Gibbs sampler of Procedure 1, conditionally on* $x'^{,b'_{1:T}}_{1:T} = x'_{1:T}$ *and* $b'_{1:T} = (N, \ldots, N)$.

Proof. From (5) we have, by construction,

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

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

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

It follows that

$$\phi_\theta(\mathbf{x}_1^{-b_1} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) = \prod_{\substack{i=1\\ i\neq b_1}}^{N} r_{\theta,1}(x_1^i), \tag{6a}$$

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

$$\begin{aligned}\phi_\theta(\mathbf{x}_t^{-b_t}, \mathbf{a}_t^{-b_t} \mid \mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}}, x_{1:T}^{b_{1:T}}, b_{1:T}) \\ = \frac{\phi_\theta(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{2:t}^{-b_{2:t}} \mid x_{1:T}^{b_{1:T}}, b_{1:T})}{\phi_\theta(\mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T})} = \prod_{\substack{i=1\\ i\neq b_t}}^{N} M_{\theta,t}(a_t^i, x_t^i).\end{aligned} \tag{6b}$$

Hence, we can sample from (6a) and (6b) 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 (6a) and (6b), respectively.

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

$$\begin{aligned}
\phi_\theta(a_t^{b_t} \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t:T}) &\propto \phi_\theta(\mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t-1:T}) \\
&= \phi_\theta(x_{1:T}^{b_{1:T}}, b_{1:T}) \phi_\theta(\mathbf{x}_{1:t-1}^{-b_{1:t-1}}, \mathbf{a}_{2:t-1}^{-b_{2:t-1}} \mid x_{1:T}^{b_{1:T}}, b_{1:T}) \\
&= \frac{\gamma_{\theta,T}(x_{1:T}^{b_{1:T}})}{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})} \frac{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})}{Z_{\theta,T} N^T} \prod_{\substack{i=1 \\ i \neq b_1}}^{N} r_{\theta,1}(x_1^i) \prod_{s=2}^{t-1} \prod_{\substack{i=1 \\ i \neq b_s}}^{N} M_{\theta,s}(a_s^i, x_s^i). \qquad (7)
\end{aligned}$$

To simplify this expression, note first that we can write

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

By using the definition of the weight function (2), this expression can be expanded according to

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

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

$$\begin{aligned}
\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}}) &= w_1^{b_1} r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} w_s^{b_s} r_{\theta,s}(x_s^{b_s} \mid x_{1:s-1}^{b_{1:s-1}}) \\
&= \left(\prod_{s=1}^{t-1} \sum_{l=1}^{N} w_s^l \right) \frac{w_1^{b_1}}{\sum_l w_1^l} r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} \frac{w_s^{b_s}}{\sum_l w_s^l} r_{\theta,s}(x_s^{b_s} \mid x_{1:s-1}^{b_{1:s-1}}) \\
&= \frac{w_{t-1}^{b_{t-1}}}{\sum_l w_{t-1}^l} \left(\prod_{s=1}^{t-1} \sum_{l=1}^{N} w_s^l \right) r_{\theta,1}(x_1^{b_1}) \prod_{s=2}^{t-1} M_{\theta,s}(a_s^{b_s}, x_s^{b_s}). \qquad (8)
\end{aligned}$$

Expanding the numerator in (7) according to (8) results in

$$\begin{aligned}
&\phi_\theta(a_t^{b_t} \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}, x_{t:T}^{b_{t:T}}, b_{t:T}) \\
&\quad \propto \frac{\gamma_{\theta,T}(x_{1:T}^{b_{1:T}})}{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})} \frac{w_{t-1}^{b_{t-1}}}{\sum_l w_{t-1}^l} \frac{\left(\prod_{s=1}^{t-1} \sum_l w_s^l \right)}{Z_{\theta,T} N^T} \prod_{i=1}^{N} r_{\theta,1}(x_1^i) \prod_{s=2}^{t-1} \prod_{i=1}^{N} M_{\theta,s}(a_s^i, x_s^i) \\
&\quad \propto w_{t-1}^{b_{t-1}} \frac{\gamma_{\theta,T}((x_{1:t-1}^{b_{1:t-1}}, x_{t:T}^{b_{t:T}}))}{\gamma_{\theta,t-1}(x_{1:t-1}^{b_{1:t-1}})}. \qquad (9)
\end{aligned}$$

Consequently, with $b_t = N$ and $x_{t:T}^{b_{t:T}} = x'_{t:T}$, sampling from (9) corresponds to the AS step of line 8 of Algorithm 2. Finally, analogously to (9), 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 Dyk and Park (2008). 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 $\{\mathbf{x}_1^{-b'_1}, \underline{\mathbf{x}}_{2:T}^{-b'_{2:T}}, \underline{\mathbf{a}}_{2:T}^{-b'_{2:T}}\} \sim \phi_\theta(\,\cdot \mid x_{1:T}^{\prime,b'_{1:T}}, b'_{1:T})$ and, for $t = 2$ to T, draw:

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

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

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 (6b) 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 $\mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T})$ 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

$$\begin{aligned}\bar{\gamma}_{\theta,T}(B) = \int \mathbb{1}_B(x_{1:T}^k)\mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T}) \\ \times \delta_{x'_1}(dx_1^{b'_1}) \cdots \delta_{x'_T}(dx_T^{b'_T}) \frac{\bar{\gamma}_{\theta,T}(x'_{1:T})}{N^T} dx'_{1:T} db'_{1:T}, \quad \forall B \in \mathcal{X}^T. \end{aligned} \tag{10}$$

By Lemma 1 we know that Algorithm 2, which implicitly defines P_θ^N, is equivalent to Procedure 1 conditionally on $x_{1:T}^{\prime,b'_{1:T}} = x'_{1:T}$ and $b'_{1:T} = (N, \ldots, N)$. That is to say,

$$\begin{aligned}P_\theta^N(x'_{1:T}, B) = \int \mathbb{1}_B(x_{1:T}^k)\mathcal{L}(d\mathbf{x}_{1:T}^{-(N,\ldots,N)}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, (N, \ldots, N)) \\ \times \delta_{x'_1}(dx_1^N) \cdots \delta_{x'_T}(dx_T^N),\end{aligned}$$

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 particles.[2] This implies that for any $b'_{1:T} \in \{1, \ldots, N\}^T$,

$$P_\theta^N(x'_{1:T}, B) = \int \mathbb{1}_B(x^k_{1:T}) \mathcal{L}(d\mathbf{x}_{1:T}^{-b'_{1:T}}, d\mathbf{a}_{2:T}, dk \mid x'_{1:T}, b'_{1:T}) \delta_{x'_1}(dx_1^{b'_1}) \cdots \delta_{x'_T}(dx_T^{b'_T}). \quad (11)$$

Plugging (11) into (10) gives the desired result,

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

■

4.2 Ergodicity

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

$$\mathcal{S}_{\theta,t} = \{x_{1:t} \in \mathsf{X}^t : \bar{\gamma}_{\theta,t}(x_{1:t}) > 0\},$$
$$\mathcal{Q}_{\theta,t} = \{x_{1:t} \in \mathsf{X}^t : r_{\theta,t}(x_t \mid x_{1:t-1}) \bar{\gamma}_{\theta,t-1}(x_{1:t-1}) > 0\},$$

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

(A1) For any $\theta \in \Theta$ and $t \in \{1, \ldots, T\}$ 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. (2010). The same argument can be applied also to PGAS.

Theorem 2 (Andrieu et al. (2010, 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,*

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

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 stronger version of assumption (A1).

(A2) For any $\theta \in \Theta$ and $t \in \{1, \ldots, T\}$, 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*

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

2. A formal proof of this statement is given for the PG sampler by Chopin and Singh (2014). The same argument can be used also for PGAS.

Proof. We show that P_θ^N satisfies a Doeblin condition,

$$P_\theta^N(x'_{1:T}, B) \geq \varepsilon_\theta \bar{\gamma}_{\theta,T}(B), \qquad \forall x'_{1:T} \in \mathsf{X}^T, \forall B \in \mathcal{X}^T, \tag{12}$$

for some constant $\varepsilon_\theta > 0$. Uniform ergodicity then follows from Tierney (1994, Proposition 2). To prove (12) we use the representation of the PGAS kernel in (4),

$$\begin{aligned} P_\theta^N(x'_{1:T}, B) &= \mathbb{E}_{\theta, x'_{1:T}}\left[\mathbb{1}_B(x^k_{1:T})\right] = \sum_{j=1}^N \mathbb{E}_{\theta, x'_{1:T}}\left[\frac{w_T^j}{\sum_l w_T^l}\mathbb{1}_B(x^j_{1:T})\right] \\ &\geq \frac{1}{N\kappa_\theta}\sum_{j=1}^{N-1}\mathbb{E}_{\theta, x'_{1:T}}\left[w_T^j \mathbb{1}_B(x^j_{1:T})\right] = \frac{N-1}{N\kappa_\theta}\mathbb{E}_{\theta, x'_{1:T}}\left[W_{\theta,T}(x^1_{1:T})\mathbb{1}_B(x^1_{1:T})\right]. \end{aligned} \tag{13}$$

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 $\{x^i_{1:T}\}_{i=1}^{N-1}$ are equally distributed under Algorithm 2. Let $h_{\theta,t} : \mathsf{X}^t \mapsto \mathbb{R}_+$ and consider

$$\begin{aligned} \mathbb{E}_{\theta, x'_{1:T}}\left[h_{\theta,t}(x^1_{1:t})\right] &= \mathbb{E}_{\theta, x'_{1:T}}\left[\mathbb{E}_{\theta, x'_{1:T}}\left[h_{\theta,t}(x^1_{1:t}) \mid \mathbf{x}_{1:t-1}, \mathbf{a}_{2:t-1}\right]\right] \\ &= \mathbb{E}_{\theta, x'_{1:T}}\left[\sum_{j=1}^N \int h_{\theta,t}((x^j_{1:t-1}, x_t))\frac{w^j_{t-1}}{\sum_l w^l_{t-1}} r_{\theta,t}(x_t \mid x^j_{1:t-1})\,dx_t\right] \\ &\geq \frac{N-1}{N\kappa_\theta}\mathbb{E}_{\theta, x'_{1:T}}\left[\int h_{\theta,t}((x^1_{1:t-1}, x_t)) W_{\theta,t-1}(x^1_{1:t-1}) r_{\theta,t}(x_t \mid x^1_{1:t-1})\,dx_t\right], \end{aligned} \tag{14}$$

where the inequality follows analogously to (13). Now, let

$$\begin{aligned} h_{\theta,T}(x_{1:T}) &= W_{\theta,T}(x_{1:T})\mathbb{1}_B(x_{1:T}), \\ h_{\theta,t-1}(x_{1:t-1}) &= \int h_{\theta,t}(x_{1:t}) W_{\theta,t-1}(x_{1:t-1}) r_{\theta,t}(x_t \mid x_{1:t-1})\,dx_t, \qquad t \leq T. \end{aligned}$$

Then, by iteratively making use of (14) and changing the order of integration, we can bound (13) according to

$$\begin{aligned} \left(\frac{N-1}{N\kappa_\theta}\right)^{-T} P_\theta^N(x'_{1:T}, B) &\geq \mathbb{E}_{\theta, x'_{1:T}}\left[h_{\theta,1}(x^1_1)\right] \\ &= \int W_{\theta,1}(x_1) r_{\theta,1}(x_1) \prod_{t=2}^T \left(W_{\theta,t}(x_{1:t}) r_{\theta,t}(x_t \mid x_{1:t-1})\right) \mathbb{1}_B(x_{1:T})\,dx_{1:T} \\ &= \int \gamma_{\theta,1}(x_1) \prod_{t=2}^T \left(\frac{\gamma_{\theta,t}(x_{1:t})}{\gamma_{\theta,t-1}(x_{1:t-1})}\right) \mathbb{1}_B(x_{1:T})\,dx_{1:T} \\ &= \int \gamma_{\theta,T}(x_{1:T})\mathbb{1}_B(x_{1:T})\,dx_{1:T} = Z_{\theta,T}\bar{\gamma}_{\theta,T}(B). \end{aligned}$$

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

5. PGAS for State-Space Models

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.

5.1 Sampling from the Joint Smoothing Distribution with PGAS

Consider the (possibly) nonlinear/non-Gaussian SSM

$$x_{t+1} \sim f_\theta(x_{t+1} \mid x_t), \tag{15a}$$
$$y_t \sim g_\theta(y_t \mid x_t), \tag{15b}$$

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 $x_{1:T}$. In the subsequent section we will provide both a Bayesian and a frequentist learning algorithm based on the PGAS kernel. However, we start by discussing how to implement the PGAS algorithm for this specific model.

For an SSM the target distribution of interest is typically the joint smoothing distribution $p_\theta(x_{1:T} \mid y_{1:T})$. Consequently, since $p_\theta(x_{1:T} \mid y_{1:T}) \propto p_\theta(x_{1:T}, y_{1:T})$, the sequence of *unnormalized* target densities is given by

$$\gamma_{\theta,t}(x_{1:t}) = p_\theta(x_{1:t}, y_{1:t}), \qquad t = 1, \ldots, T. \tag{16}$$

As we have previously discussed, the process of sampling from the PGAS kernel is similar to running a PF. The only non-standard (and nontrivial) operation is the AS step. By plugging the specific choice of unnormalized target densities (16) into the general expression for the AS weights (3), we get

$$\widetilde{w}^i_{t-1|T} = w^i_{t-1} \frac{p_\theta((x^i_{1:t-1}, x'_{t:T}), y_{1:T})}{p(x^i_{1:t-1}, y_{1:t-1})} = w^i_{t-1} p_\theta(x'_{t:T}, y_{t:T} \mid x^i_{t-1}) \propto w^i_{t-1} f_\theta(x'_t \mid x^i_{t-1}). \tag{17}$$

This expression can be understood as an application of Bayes' theorem. Recall that we want to assign an ancestor at time $t-1$ to the reference particle x'_t. 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}, which gives us the AS probability.

Expression (17) can also be recognized as the backward sampling weights in a backward simulator; see Godsill et al. (2004); Lindsten and Schön (2013). Consequently, the AS step corresponds to a one-step backward simulation, which highlights the close relationship between PGAS and PGBS for SSMs. The latter method is conceptually similar to PGAS, but it makes use of an explicit backward simulation pass; see Whiteley (2010); Whiteley et al. (2010) or Lindsten and Schön (2013, Section 5.4). We discuss this relationship in more detail in Appendix A. In particular, we show that PGAS and PGBS are in fact probabilistically equivalent under certain conditions when applied to SSMs. Note, however, that this equivalence *does not* hold in general for models outside the class of SSMs. In particular, for the class of non-Markovian models, discussed in the subsequent section, we have found that PGAS and PGBS have quite different properties.

For concreteness we provide a restatement of the PGAS algorithm, specifically for the case of SSMs, in Algorithm 3. To highlight the similarities between PGAS and a standard PF, we have chosen to present Algorithm 3 using a notation and nomenclature that is common in the particle filtering literature, but that differs slightly from our previous notation. However, we emphasize that Algorithm 3 is completely equivalent to Algorithm 2 when the target distributions are given by (16). Note that the computational cost of the AS step is $O(N)$ per time step, i.e., of the same order as the PF. Consequently, for an SSM, the computational complexity of PGAS is the same as for PG, in total $O(NT)$.

Algorithm 3 PGAS Markov kernel for the joint smoothing distribution $p_\theta(x_{1:T} \mid y_{1:T})$

Input: Reference trajectory $x'_{1:T} \in \mathsf{X}^T$ and parameter $\theta \in \Theta$.
Output: Sample $x^\star_{1:T} \sim P^N_\theta(x'_{1:T}, \cdot)$ from the PGAS Markov kernel.

1: Draw $x^i_1 \sim r_{\theta,1}(x_1 \mid y_1)$ for $i = 1, \ldots, N-1$.
2: Set $x^N_1 = x'_1$.
3: Set $w^i_1 = g_\theta(y_1 \mid x^i_1)\mu_\theta(x^i_1)/r_{\theta,1}(x^i_1 \mid y_1)$ for $i = 1, \ldots, N$.
4: **for** $t = 2$ **to** T **do**
   / Resampling and ancestor sampling */*
5: Generate $\{\widetilde{x}^i_{1:t-1}\}_{i=1}^{N-1}$ by sampling $N-1$ times with replacement from $\{x^i_{1:t-1}\}_{i=1}^N$ with probabilities proportional to the importance weights $\{w^i_{t-1}\}_{i=1}^N$.
6: Draw J with

$$\mathbb{P}(J = i) = \frac{w^i_{t-1} f_\theta(x'_t \mid x^i_{t-1})}{\sum_{l=1} w^l_{t-1} f_\theta(x'_t \mid x^l_{t-1})}, \qquad i = 1, \ldots, N$$

   and set $\widetilde{x}^N_{1:t-1} = x^J_{1:t-1}$.

   / Particle propagation */*
7: Simulate $x^i_t \sim r_{\theta,t}(x_t \mid \widetilde{x}^i_{t-1}, y_t)$ for $i = 1, \ldots, N-1$.
8: Set $x^N_t = x'_t$.
9: Set $x^i_{1:t} = (\widetilde{x}^i_{1:t-1}, x^i_t)$ for $i = 1, \ldots, N$.

   / Weighting */*
10: Set $w^i_t = g_\theta(y_t \mid x^i_t) f_\theta(x^i_t \mid \widetilde{x}^i_{t-1})/r_{\theta,t}(x^i_t \mid \widetilde{x}^i_{t-1}, y_t)$ for $i = 1, \ldots, N$.
11: **end for**
12: Draw k with $\mathbb{P}(k = i) \propto w^i_T$.
13: **return** $x^\star_{1:T} = x^k_{1:T}$.

5.2 Learning Algorithms for State-Space Models

We now turn to the problem of learning the model parameter θ in the SSM (15), given a batch of observations $y_{1:T}$. Consider first the Bayesian setting where a prior distribution $\pi(\theta)$ is assigned to θ. We seek the parameter posterior $p(\theta \mid y_{1:T})$ 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 conditionals. However, it has been recognized that this can result in

Algorithm 4 PGAS for Bayesian learning of SSMs

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

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 (de Jong and Shephard, 1995). Due to the invariance and ergodicity properties of the kernel (Theorems 1– 3), the validity of the Gibbs sampler is not violated. We summarize the procedure in Algorithm 4.

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

$$\widehat{\theta}_{\mathrm{ML}} = \arg\max_{\theta \in \Theta} \log p_\theta(y_{1:T})$$

is to use the expectation maximization (EM) algorithm (Dempster et al., 1977; McLachlan and Krishnan, 2008). 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

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

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 (Wei and Tanner, 1990) and the SAEM (Delyon et al., 1999) algorithms, respectively. When the 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

$$\widehat{Q}_n(\theta) = (1 - \alpha_n)\widehat{Q}_{n-1}(\theta) + \alpha_n \log p_\theta(x_{1:T}[n], y_{1:T}), \tag{18}$$

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 (18) is typically made on some sufficient statistic for the complete data log-likelihood; see Delyon et al. (1999) 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 (Benveniste et al., 1990; Andrieu et al., 2005). A practical approach is therefore to compute the auxiliary quantity according to the stochastic approximation (18), but where $x_{1:T}[n]$ is simulated from the PGAS kernel $P^N_{\theta[n-1]}(x_{1:T}[n-1], \cdot)$. This particle SAEM algorithm, previously presented by Lindsten (2013), is summarized in Algorithm 5.

6. Beyond State-Space Models

For SSMs, the Markovianity implies a simple expression for the AS weights, depending only of the one-step transition density according to (17). For models with more intricate

Algorithm 5 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 $x_{1:T}[n] \sim P^N_{\theta[n-1]}(x_{1:T}[n-1], \cdot)$. */* By running Algorithm 3 */*
4: Compute $\widehat{Q}_n(\theta)$ according to (18).
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**

dependencies between the latent variables, however, this is not the case and the general expression (3) needs to be used. In this section we consider the computational aspects of the AS step, first in a very general setting and then specifically for the class of non-Markovian latent variable models.

6.1 Modifications and Mixed Strategies

The interpretation of the PGAS algorithm as a standard MCMC sampler on an extended space opens up for straightforward modifications of the algorithm while still making sure that it retains its desirable theoretical properties. In particular, for models where the computation of the AS weights in (3) is costly—that is, when evaluating the unnormalized joint target density $\gamma_{\theta,T}$ is computationally involved—it can be beneficial to modify the AS step to reduce the overall computational cost of the algorithm. Let

$$\rho(i) = \frac{\widetilde{w}^i_{t-1|T}}{\sum_{l=1}^N \widetilde{w}^l_{t-1|T}}, \qquad i = 1, \ldots, N, \tag{19}$$

denote the law of the ancestor index a^N_t, 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 (5). To retain the correct limiting distribution of the PGAS kernel, it is therefore sufficient that a^N_t is sampled from a Markov kernel leaving (19) invariant (resulting in a standard combination of MCMC kernels; see, e.g., Tierney 1994).

A simple modification is to carry out the AS step only for a fraction of the time steps. For instance, we can generate the ancestor index a^N_t according to:

$$\begin{cases} \text{With probability } 1-\eta, \text{ set } a^N_t = N, \\ \text{Otherwise, simulate } a^N_t \text{ with } \mathbb{P}(a^N_t = i) = \rho(i), \end{cases} \tag{20}$$

where $\eta \in [0, 1]$ is a user specified parameter, controlling the probability of executing the AS step. This strategy results in a mix between PG and PGAS; for $\eta = 0$ we recover the original PG algorithm and for $\eta = 1$ we obtain the basic PGAS algorithm. For complex models, this modification can be quite useful. In fact, there is no immediate gain in changing the ancestry of the reference trajectory as long as the particle trajectories have not degenerated. That is, it is sufficient to carry out the AS step "once in a while" to obtain high update

rates for the complete trajectory (cf. Figure 1 where we get high update rates for PG for the last few time steps, even when using a small number of particles). We illustrate this empirically in the simulation study in Section 7.3.

Another modification, that can be used either on its own or in conjunction with (20), is to use MH to simulate from (19). Let $q(i' \mid i)$ 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 i', by simulating $i' \sim q(\cdot \mid N)$. With probability

$$1 \wedge \frac{\widetilde{w}_{t-1|T}^{i'}}{\widetilde{w}_{t-1|T}^{N}} \frac{q(N \mid i')}{q(i' \mid N)} \tag{21}$$

the sample is accepted and we set $a_t^N = i'$, otherwise we keep the ancestry $a_t^N = N$. Using this approach, we avoid computing the normalizing constant in (19), i.e., we only need to evaluate the AS weights for the proposed values. This will reduce the computational cost of the AS step by, roughly, a factor N which 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 Liu (1996). That is, q is constructed so that $q(i \mid i) = 0, \forall i$, ensuring that the current state of the chain is not proposed anew, which would be a wasteful operation.

6.2 Non-Markovian Latent Variable Models

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

$$\begin{aligned} x_{t+1} &\sim f_\theta(x_{t+1} \mid x_{1:t}), \\ y_t &\sim g_\theta(y_t \mid x_{1:t}). \end{aligned}$$

Similarly to the SSM (15), 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. Below we discuss the AS step of the PGAS algorithm specifically for these non-Markovian models and derive a truncation strategy for the AS weights. First, however, to motivate the present development we review some application areas in which this type of models arise.

In Bayesian nonparametrics (Hjort et al., 2010) 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., Teh et al. (2006); Escobar and West (1995), and Gaussian process regression and classification models (Rasmussen and Williams, 2006). 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 (Turner and Deisenroth, 2010; Frigola et al., 2013), a flexible nonlinear dynamical systems model, for which PGAS has been successfully applied (Frigola et al., 2013).

Another typical source of non-Markovianity is by marginalization over part of the state vector, i.e., Rao-Blackwellization, (Chen and Liu, 2000; Whiteley et al., 2010; Lindsten et al., 2013) or by a change of variables in an SSM. This type of operation typically results in a loss of the Markov property, but 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. This includes many models for which the transition is implicitly given by a simulator (Gander and Stephens, 2007; Fearnhead et al., 2008; Golightly and Wilkinson, 2008; Murray et al., 2013) or degenerate models where the transition density does not even exist (Ristic et al., 2004; Gustafsson et al., 2002). We illustrate these ideas in Section 7. See also Lindsten and Schön (2013, 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 Lindsten and Schön (2013, Section 4).

To employ PGAS, or in fact any backward-simulation-based method (see Lindsten and Schön 2013), we need to evaluate the AS weights (3) which depend on the ratio

$$\frac{\gamma_{\theta,T}(x_{1:T})}{\gamma_{\theta,t-1}(x_{1:t-1})} = \frac{p_\theta(x_{1:T}, y_{1:T})}{p_\theta(x_{1:t-1}, y_{1:t-1})} = \prod_{s=t}^{T} g_\theta(y_s \mid x_{1:s}) f_\theta(x_s \mid x_{1:s-1}). \tag{22}$$

Assuming that g_θ and f_θ can both be evaluated in constant time, the computational cost of computing the backward sampling weights (3) will thus be $O(NT)$. This implies that the overall computational complexity of the PGAS kernel will scale quadratically with T which can be prohibitive in some cases. The general strategies discussed in Section 6.1, i.e., using AS sporadically and/or using MH within PGAS, can of course be used to mitigate this issue. Nonetheless, the AS step can easily become the computational bottleneck when applying the PGAS algorithm to a non-Markovian model.

To make further 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 AS weights by truncating the product (22) to a smaller number of factors, say ℓ. We can thus replace (3) with the approximation

$$\begin{aligned}\widetilde{w}^{\ell,i}_{t-1|T} &\triangleq w^i_{t-1} \frac{\gamma_{\theta,t-1+\ell}((x^i_{1:t-1}, x'_{t:t-1+\ell}))}{\gamma_{\theta,t-1}(x^i_{1:t-1})} \\ &= w^i_{t-1} \prod_{s=t}^{t-1+\ell} g_\theta(y_s \mid x^i_{1:t-1}, x'_{t:s}) f_\theta(x'_s \mid x^i_{1:t-1}, x'_{t:s-1}).\end{aligned} \tag{23}$$

Let $\widehat{\rho}_\ell(k)$ be the probability distribution defined by the truncated AS weights (23), analogously to (19). The following proposition formalizes our assumption.

Proposition 1. *Let* $h_s(k) = g_\theta(y_{t-1+s} \mid x^k_{1:t-1}, x'_{t:t-1+s}) f_\theta(x'_{t-1+s} \mid x^k_{1:t-1}, x'_{t:t-1+s})$ *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 B. ■

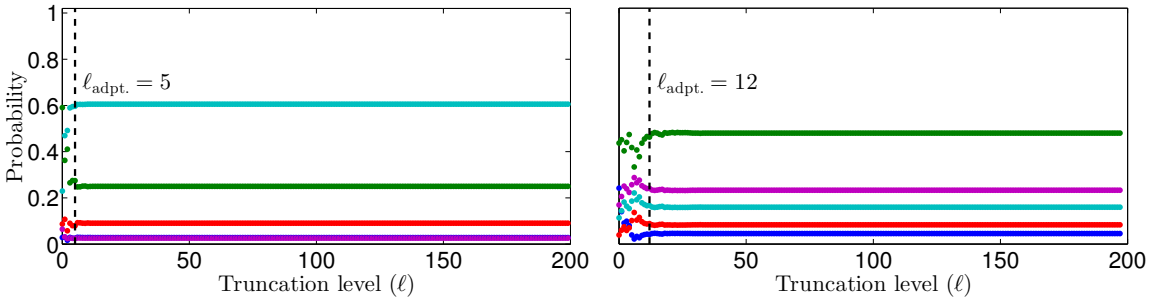

Figure 3: Probability under $\widehat{\rho}_\ell$ as a function of the truncation level ℓ for two different systems; one 5-dimensional (left) and one 20-dimensional (right). The $N = 5$ dotted lines correspond to $\widehat{\rho}_\ell(k)$ for $k \in \{1, \ldots, N\}$, respectively (N.B. two of the lines 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 $\upsilon = 0.1$ and $\tau = 10^{-2}$. See Section 7.2 for details on the experiments.

Using the approximation given by (23), the AS 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 adaptive strategy. Since the approximative weights (23) 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.

Let $\varepsilon_\ell = D_{\mathsf{TV}}(\widehat{\rho}_\ell, \widehat{\rho}_{\ell-1})$ be the total variation (TV) distance between the approximative AS distributions for two consecutive truncation levels. We then compute the exponentially decaying moving average of the sequence ε_ℓ, with forgetting factor $\upsilon \in [0, 1]$, 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 υ and τ. However, these parameters are much easier to reason about—a small value for υ 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 υ 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., 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 (23) can be used in a few different ways. First, as discussed above, we can simply replace ρ with $\widehat{\rho}_\ell$ 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 $\widehat{\rho}_\ell$ as an efficient proposal distribution for the MH algorithm suggested in Section 6.1, leading to an $O(NT\ell + T^2)$ complexity. 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 (23) when evaluating the acceptance probability (21). 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 stochastic volatility SSM and investigate the improvement in mixing offered by AS when PGAS is compared with PG. We do not consider PGBS in this example since, as we show in Proposition 2 in Appendix A, PGAS and PGBS are probabilistically equivalent in this scenario. The conditions of Proposition 2 imply that the weight function in the PF is independent of the ancestor indices. When applied to non-Markovian models, however, Proposition 2 does not apply, since the weight function then 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 state transition kernel is not available.

7.1 Stochastic Volatility Model with Leverage

Stochastic volatility (SV) models are commonly used to model the variation (or volatility) of a financial asset; see, e.g., Kim et al. (1998); Shephard (2005). Let r_t be the price of an asset at time t and let $y_t = \log(r_t/r_{t-1})$ denote the so-called *log-returns*. A typical SV model is then given by the SSM:

$$\begin{aligned} x_{t+1} &= \mu(1-\varphi) + \varphi x_t + \sigma v_t, \\ y_t &= \exp(-\tfrac{1}{2}x_t)e_t, \end{aligned} \qquad \begin{pmatrix} v_t \\ e_t \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right). \tag{24}$$

The correlation between the process noise and the observation noise allows for a leverage effect by letting the price of the asset influence the future volatility. Assuming stationarity, the distribution of the initial state is given by $\mu_\theta(x_1) = \mathcal{N}(x_1; \mu, \sigma^2/(1-\varphi^2))$. The unknown parameters of the model are $\theta = (\mu, \varphi, \sigma^2, \rho)$.

This system is used as a proof of concept, primarily to illustrate the superior mixing of PGAS when compared to PG. However, we also compare PGAS with the particle marginal MH (PMMH) algorithm by Andrieu et al. (2010), which has previously been used to calibrate SV models on the form (24) (Hallgren and Koski, 2014; Pitt et al., 2010). We analyze the Standard and Poor's (S&P) 500 data from 3/April/2006 to 31/March/2014, consisting

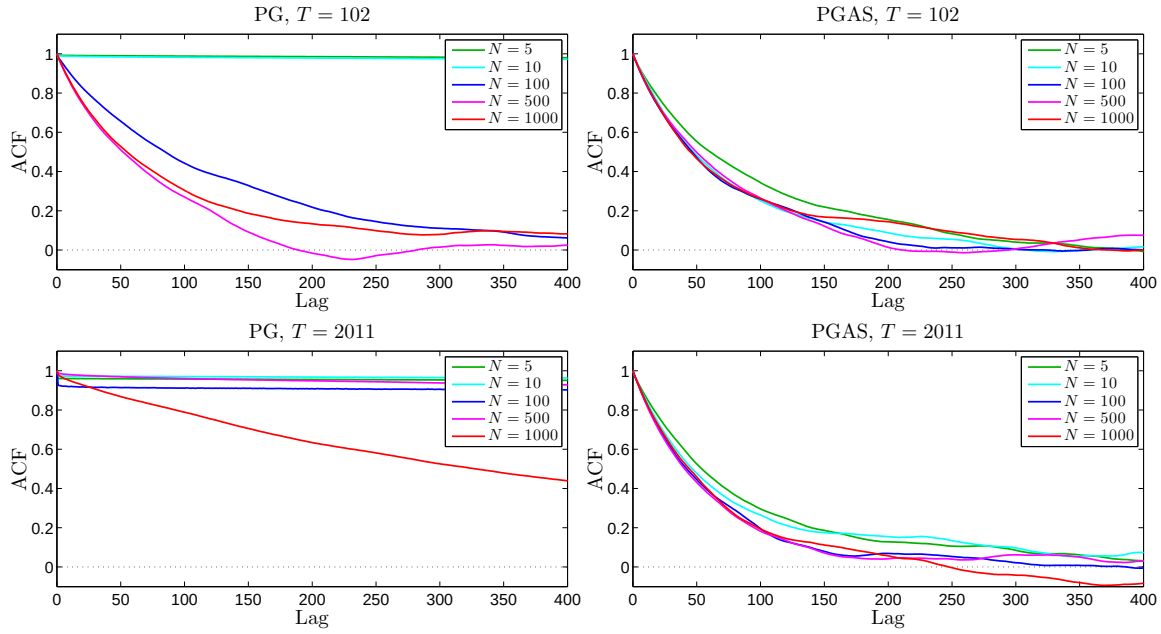

Figure 4: ACFs the parameter σ^2 for PG (left column) and for PGAS (right column) for the S&P 500 data consisting of $T = 102$ (top row) and $T = 2\,011$ (bottom row) observations, respectively. The results are reported for different number of particles N. (This figure is best viewed in color.)

of $T = 2\,011$ observations.[3] We consider the the PGAS sampler (Algorithm 4) as well as the PG and PMMH samplers by Andrieu et al. (2010), all for a range of different number of particles, $N \in \{5, 10, 100, 500, 1\,000\}$. All methods are simulated for 50 000 iterations, whereafter the first 10 000 samples are discarded as burn-in. For updating θ, PG and PGAS simulate the parameters one at a time from their respective conditionals, whereas PMMH uses a Gaussian random walk tuned according to an initial trial run. Additional details on the experiments are given in Appendix C.

To evaluate the mixing of the samplers, we compute the autocorrelation functions (ACFs) for the sequences $\theta[n] - \mathbb{E}[\theta \mid y_{1:T}]$.[4] We start by considering the simpler problem of analyzing a small subset of the data, consisting of $T = 102$ samples (1/November/2013–31/March/2014). The results for PG and PGAS for the parameter σ^2 are reported in the top row of Figure 4. Similar results hold for the other parameters as well. We see that the PG sampler requires a fairly large N to obtain good mixing and using $N \leq 10$ causes the sampler to get completely stuck. For PGAS, on the other hand, the ACF is much more robust to the choice of N. Indeed, we obtain comparable mixing rates for any number of particles $N \geq 5$. This suggests that the sampler in fact performs very closely to a fictive

3. The data was acquired from the Yahoo Finance web page `https://finance.yahoo.com/q/hp?s=%5EGSPC&a=03&b=3&c=2006&d=02&e=31&f=2014&g=d`
4. The "true" posterior mean is computed from a long (500 000 samples) run of PMMH.

	w/o sub-sampling		w sub-sampling	
	PGAS	PMMH	PGAS	PMMH
$N = 5$	111.7	$> 10^4$	24.6	3 923.3
$N = 10$	96.6	$> 10^4$	**20.7**	6 187.0
$N = 100$	71.3	4 796.1	21.1	1 146.8
$N = 500$	73.3	59.2	47.8	33.5
$N = 1\,000$	72.6	31.5	80.9	**31.5**

Table 1: Average inefficiencies for the SV model.

"ideal" Gibbs sampler, i.e., a sampler that simulates $x_{1:T}$ from the true joint smoothing distribution.

Next, we rerun the methods on the whole data set with $T = 2\,011$ observations. The results are shown in the bottom row of Figure 4. The effect can be seen even more clearly in this more challenging scenario. Again, we find PGAS to perform very closely to an ideal Gibbs sampler for any $N \geq 5$. In other words, for this model it is the mixing of the ideal Gibbs sampler, not the intrinsic particle approximation, that is the limitation of PGAS. The big difference in mixing between PG and PGAS 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.

We now turn to a comparison between PGAS and PMMH. However, in doing so it is important to realize that these two methods, while both being instances of PMCMC, have quite different properties. In particular, just as PGAS can be thought of as an approximation of an ideal Gibbs sampler, PMMH can be viewed as an approximation of an ideal marginal MH sampler. Consequently, their respective performances depend on the properties of these ideal samplers and the preference for one method over the other heavily depends on the specific problem under study. Nevertheless, we apply both methods to the S&P 500 data (with $T = 2\,011$). To evaluate the mixing, we compute[5] the *inefficiencies*:

$$\mathsf{IF} \triangleq 1 + 2\sum_{j=1}^{\infty} \mathsf{ACF}(j)$$

for the four parameters of the model, where $\mathsf{ACF}(j)$ is the ACF at lag j. The interpretation of the inefficiency is that we need $n \times \mathsf{IF}$ draws from the Markov chain to obtain the same precision as using n i.i.d. draws from the posterior. The average inefficiencies for the four parameters for PGAS and PMMH are reported in Table 1.

In the two columns to the left, the inefficiencies for PGAS and PMMH, respectively, are given without taking the computational cost of the algorithms into account. As above we find that PGAS is quite insensitive to the number of particles N, with only a minor increase in inefficiency as we reduce N from 1 000 to 5. PMMH requires a larger number of particles to mix well—this is in agreement with previous analyzes and empirical studies (Doucet et al., 2014; Andrieu et al., 2010). Using too few particles with PMMH causes the method to get stuck, hence the very large inefficiency values. For large N, however, PMMH

5. We use the *initial monotone sequence estimator* by Geyer (1992) to estimate the inefficiencies.

outperforms PGAS. The reason for this is that the limiting behavior (as we increase N) of PMMH is that of an ideal marginal MH sampler. For the model under study, it is apparent that this marginal MH sampler has better mixing properties than the ideal Gibbs sampler.

Finally, in the rightmost two columns of Table 1 we report the average inefficiencies for the two samplers when we have matched their computational costs. We use PMMH with $N = 1\,000$ as the base algorithm. We then match the computational times (as measured by the `tic/toc` commands in Matlab) of the algorithms and modify the inefficiencies accordingly. This corresponds to sub-sampled versions of the algorithms, such that each iteration of any method take the same amount of time as one iteration of PMMH with $N = 1\,000$. In this comparison, we obtain the best overall performance for PGAS with $N = 10$.

While the computational complexities of both algorithms scale like $O(N)$, it is worth to note that the computational overhead is quite substantial (we use a vectorized implementation of the particle filters in Matlab). In fact, when reducing N from 1 000 to 5, the reduction in computational time is closer to a factor 5 than to a factor 200. Of course, the computational overhead will be less noticeable in more difficult scenarios where it is required to use $N \gg 1\,000$ for PMMH to mix well. Nevertheless, this effect needs to be taken into account when comparing algorithms with very different computational properties, such as PGAS and PMMH, and increasingly so when considering implementations on parallel computer architectures. That is, matching the algorithms "particle by particle" would be overly favorable for PGAS. We discuss this further in Section 8.

7.2 Degenerate 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 particles.

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

$$\begin{pmatrix} x_{t+1} \\ z_{t+1} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_t \\ z_t \end{pmatrix} + \begin{pmatrix} v_t \\ 0 \end{pmatrix}, \qquad v_t \sim \mathcal{N}(0, Q), \tag{25a}$$

$$y_t = C \begin{pmatrix} x_t \\ z_t \end{pmatrix} + e_t, \qquad e_t \sim \mathcal{N}(0, R). \tag{25b}$$

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 (25) as a non-Markovian model with latent process given by $\{x_t\}_{t\geq 1}$.

As a first illustration, we simulate $T = 200$ samples from a four-dimensional, single output system with poles[6] at -0.65, -0.12, and $0.22 \pm 0.10i$. We let $\dim(x_t) = 1$ and

6. The poles of a linear system are given by the eigenvalues of the matrix A and they encode the frequency response of the system.

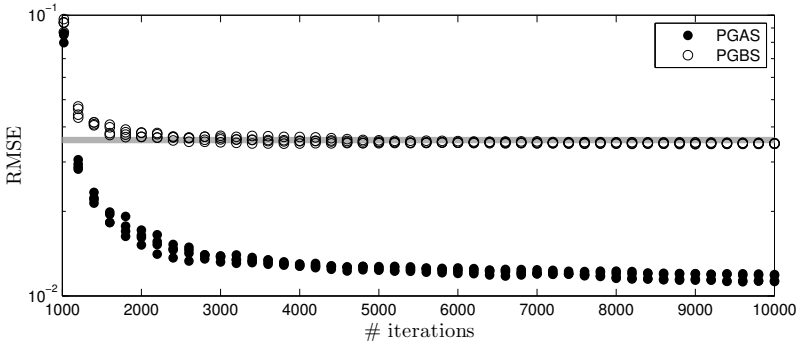

Figure 5: Running RMSEs for $x_{1:T}$ for five independent runs of PGAS (•) and PGBS (◦), respectively. The truncation level is set to $\ell = 1$. The thick gray line corresponds to a run of an untruncated FFBS particle smoother.

$Q = R = 0.1$. 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-simulation-based methods, such as PGAS and PGBS, as described in Section 6.2. 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 (23).

First, we consider the coarse approximation $\ell = 1$. We run PGAS and PGBS, both with $N = 5$ particles for 10 000 iterations (with the first 1 000 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, Bierman, 1973). 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 (Godsill et al., 2004), using $N = 10\,000$ particles and $M = 1\,000$ 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 AS 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 (25) of different orders (i.e., with different state dimensions d). 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 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

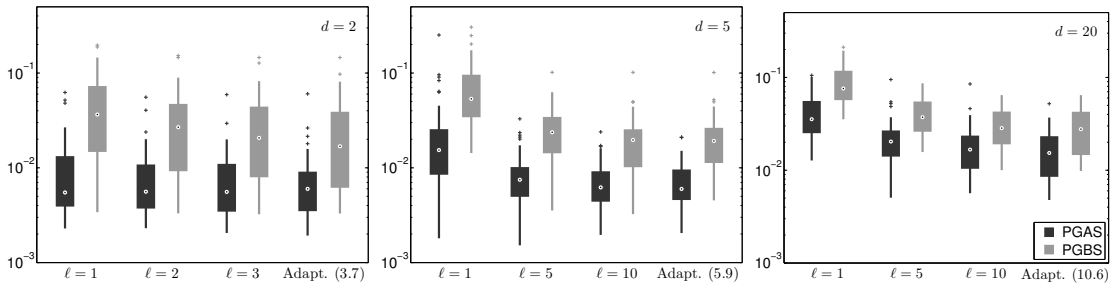

Figure 6: Box plots of the RMSE errors for PGAS (black) and PGBS (gray), for 150 random systems of different dimensions d (left, $d = 2$; middle, $d = 5$; right, $d = 20$). Different values for the truncation level ℓ are considered. The rightmost boxes correspond to an adaptive truncation and the values 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.

well as an adaptive level with $\upsilon = 0.1$ and $\tau = 10^{-2}$ (see Section 6.2). All other settings are as above.

Again, we compute the posterior means of $x_{1:T}$ (discarding 1 000 samples) and RMSE values relative to 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 (Ginsberg et al., 2009). 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 (Keeling and Rohani, 2007; Rasmussen et al., 2011). The model, specified by a stochastic differential equation, is discretized according to the Euler-Maruyama method, yielding

$$S_{t+dt} = S_t + \mu\mathcal{P}dt - \mu S_t dt - (1 + Fv_t)\,\beta_t S_t \mathcal{P}^{-1} I_t dt, \tag{26a}$$

$$I_{t+dt} = I_t - (\gamma + \mu) I_t dt + (1 + Fv_t)\,\beta_t S_t \mathcal{P}^{-1} I_t dt, \tag{26b}$$

$$R_{t+dt} = R_t + \gamma I_t dt - \mu R_t dt, \tag{26c}$$

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 (Ginsberg et al., 2009). The idea is to use the frequency of influenza-related 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 Ginsberg et al. (2009), we use a linear relationship between the log-odds of the relative query counts and the log-odds of the proportion of infected individuals,

$$y_k \triangleq \operatorname{logit}(Q_k) = \rho \operatorname{logit}(\bar{I}_k/\mathcal{P}) + e_k, \qquad e_k \sim \mathcal{N}(0, \sigma^2), \tag{27}$$

where $\bar{I}_k$ is the mean value of I_t during the time interval $(\Delta(k-1), \Delta k]$ and $\operatorname{logit}(p) = \log(p/(1-p))$. As in Ginsberg et al. (2009) 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 overly large discretization errors. Instead, we sample the model (26) $m = 7$ times per week: $dt = \Delta/m$.

Rasmussen et al. (2011) use the PMMH sampler (Andrieu et al., 2010) 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 by Skvortsov and Ristic (2012). We investigate the possibility of using PGAS for joint state and parameter inference in the model (26)–(27). 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 (26)–(27) can be equivalently expressed as the non-Markovian latent variable model,

$$V_k \sim \mathcal{N}(0, I_m/\sqrt{dt}), \tag{28a}$$

$$y_k \sim g_\theta(y_k \mid V_{1:k}), \tag{28b}$$

for some likelihood function g_θ; see (29). 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 and the parameters (Golightly and Wilkinson, 2008; Papaspiliopoulos et al., 2003).

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 use an improper flat prior on $\mathbb{R}^6_+$ for θ. We generate eight years of data with weekly observations. The number of infected individuals I_t over this time period is shown in Figure 7. The first half of the data batch is used for estimation of the model parameters using PGAS. It is

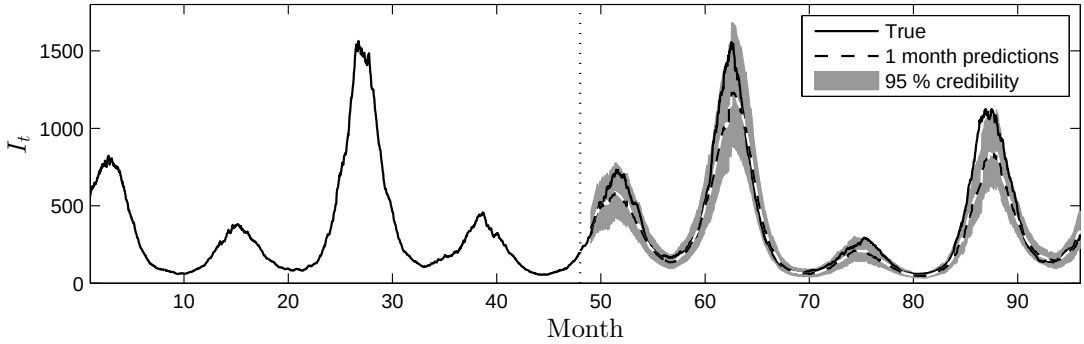

Figure 7: Disease activity (number of infected individuals I_t) over an eight year period. The first four years are used as estimation data, to find the unknown parameters of the model. For the consecutive four years, one-month-ahead predictions are computed using the estimated model.

worth pointing out that while the sampler effectively targets the collapsed model (28), it is most straightforwardly implemented using the original state variables from (26). 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 (26) which is used in the underlying particle filter. The innovation variables V_k need only be taken into account for the AS step. Let $V'_{1:T}$ be the reference innovation trajectory. To compute the AS weights (3) we need to evaluate the ratios,

$$\frac{p_\theta((V^i_{1:k-1}, V'_{k:T}), y_{1:T})}{p_\theta(V^i_{1:k-1}, y_{1:k-1})} \propto \prod_{\ell=k}^{T} g_\theta(y_\ell \mid V^i_{1:k-1}, V'_{k:\ell}).$$

Using (27), the observation likelihood can be written as

$$g_\theta(y_\ell \mid V^i_{1:k-1}, V'_{k:\ell}) = \mathcal{N}(y_\ell \mid \rho \operatorname{logit}(\bar{I}_\ell\{x^i_{k-1}, V'_{k:\ell}\}/\mathcal{P}), \sigma^2), \tag{29}$$

where $I_\ell\{x^i_{k-1}, V'_{k:\ell}\}$ is obtained by simulating the system (26) from time $\Delta(k-1)$ to time $\Delta\ell$, initialized at x^i_{k-1} and using the innovation sequence $V'_{k:\ell}$.

We run PGAS with $N = 10$ particles for 50 000 iterations (discarding the first 10 000). For sampling θ, we use MH steps with a Gaussian random walk proposal, tuned according to an initial trial run. The innovation variables $V_{1:T}$ are sampled from the PGAS kernel by Algorithm 2. 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. No truncation is used for the AS weights; instead we investigate the effect of using the strategy proposed in (20). That is, to reduce the computational cost we execute the AS step only with some probability η, otherwise we keep the current ancestry of the reference trajectory.

In Figure 8 we report the ACFs for the six parameters of the model, for η ranging from 0 to 1. As a comparison, we also provide the results for a run of the PMMH algorithm with $N = 1\,000$ particles and a random walk proposal distribution tuned according to an initial trial run. For most parameters, PMMH achieves better mixing than PGAS (however,

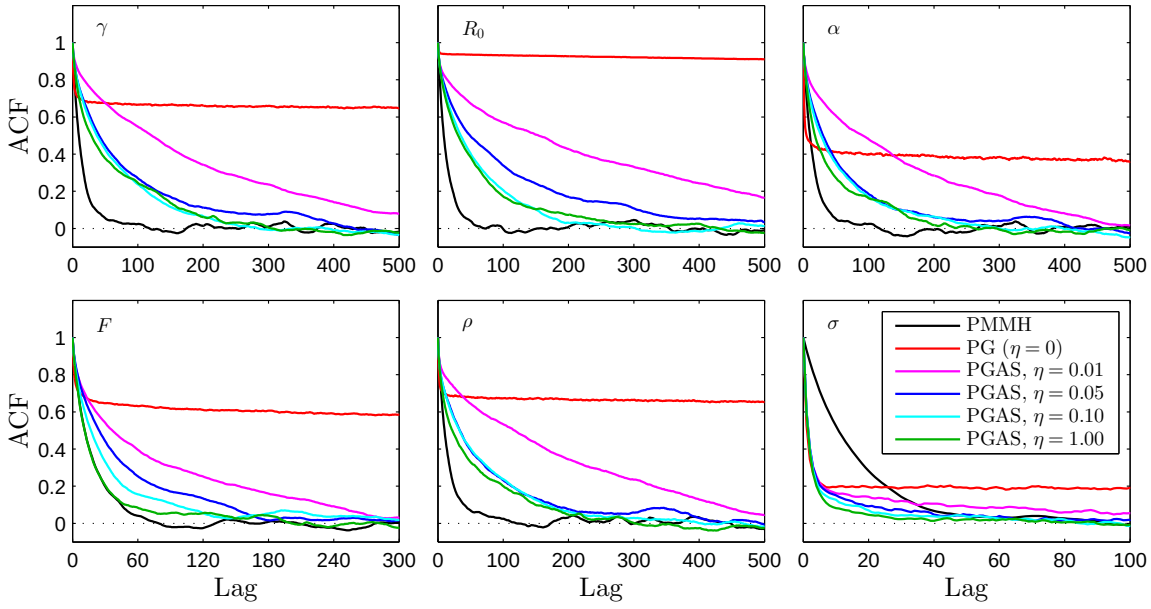

Figure 8: ACFs for PGAS with $N = 10$ and η ranging from 0 to 1. As comparison, we also show the ACF for PMMH with $N = 1\,000$. (This figure is best viewed in color.)

requiring a much larger N) which can be accredited to the fact that the ideal marginal MH sampler mixes better than the ideal Gibbs sampler.

Note that for $\eta = 0$, PGAS reduces to the standard PG algorithm. Since we use only $N = 10$ particles this sampler mixes very poorly, in agreement with our previous findings. However, interestingly, increasing the probability of ancestor sampling to as little as $\eta = 0.01$ results in a large improvement in mixing and with $\eta = 0.1$ we get results that are comparable to $\eta = 1$. This suggests that, in cases when the AS step is the computational bottleneck of the algorithm, it can indeed be a good idea to carry out this step only sporadically.

To further investigate the effect of η, we plot the update rates for the trajectory $V_{1:T}$ in Figure 9 (cf. Figure 1). As expected, the update rate deteriorates as we decrease η, but the relationship is clearly nonlinear. Specifically, for small values of η we get an average update rate which is larger than η; for instance $\eta = 0.1$ gives an average update rate of 0.31. The reason for this is that any ancestor index update will result in an update, not only for the corresponding latent variable, but also for a collection of neighboring latent variables. The number of variables that will be affected by changing one of the ancestor indices depends on how quickly the PF degenerates. Consequently, there is an inverse relationship between η and N; by increasing the number of particles we can get away with a smaller η and still obtain high update rates for the entire trajectory.

In Figure 10 we show histograms representing the estimated posterior parameter distributions, reported for PGAS with $\eta = 0.1$ and for PMMH. As can be seen, the true system parameters fall well within the credible regions. Finally, the identified model, based on PGAS with $\eta = 0.1$, is used to make one-month-ahead predictions of the disease activity for

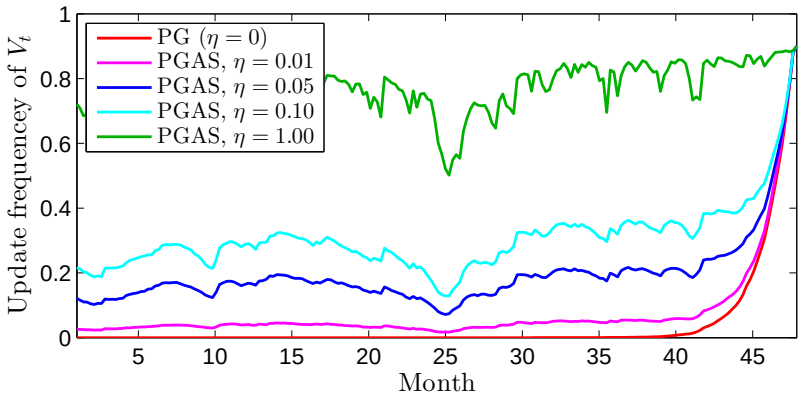

Figure 9: Update rates for $V_{1:T}$ for PGAS with $N = 10$ and with η ranging from 0 to 1. As a comparison, the PMMH sampler with $N = 1\,000$ particles attains an average acceptance probability of 0.19. (This figure is best viewed in color.)

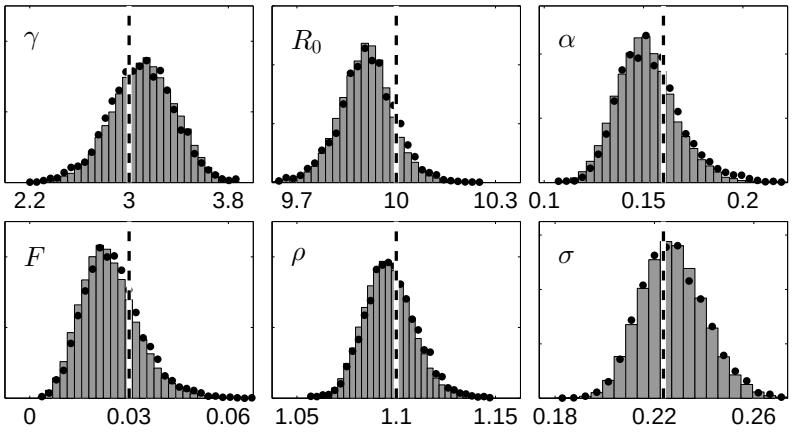

Figure 10: Posterior densities for the parameters of model (26)–(27) for PGAS; $N = 10$, $\eta = 0.1$ (gray bars) and for PMMH; $N = 1\,000$ (black dots). The true values are marked by vertical dashed lines.

the subsequent four years, as shown in Figure 7. 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 the possibility of using PGAS as a key component in different inference algorithms, enabling both Bayesian and frequentist parameter inference as well as state inference. However, PGAS is by no means limited to inference in state-space models. Indeed, we believe that the method can be 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 models PGAS is simpler to implement than PGBS as it requires less bookkeeping. It can also be more memory efficient; by using the techniques proposed by Jacob et al. (2013), it is possible to store the paths of the particle filter in PGAS with an expected memory cost bounded by $T + CN \log N$ for some constant C. This is in contrast with PGBS, which requires storage of all NT intermediate particles.

The aforementioned samplers—PG, PGAS, and PGBS—share the same interpretation of being PMCMC-versions of an ideal Gibbs sampler. A different type of PMCMC, however, is the PMMH sampler by Andrieu et al. (2010). To comprehensively compare PGAS with PMMH is nontrivial, since the two samplers have quite different properties. However, some of the most important differences are that, *(i)* in the limit $N \to \infty$, PMMH approaches a marginal sampler for θ, whereas PGAS approaches an ideal Gibbs sampler for θ and $x_{1:T}$, *(ii)* empirically, PGAS is more robust to small N/large T than PMMH, and *(iii)* PGAS defines a Markov kernel on the space of trajectories, which is not the case for PMMH, making it more suitable to use as a component in composite sampling schemes. Due to these differences—*(i)* being in favor for PMMH and *(ii)*–*(iii)* for PGAS—the preference for one sampler over the other depends heavily on the specific properties of the problem at hand.

Another important difference is that PMMH readily allows for parallelization over the particles. While this is of course possible also for PGAS, the fact that the sampler typically requires only a small number of particles limits the computational benefits of doing so. To enable PGAS to make better use of modern computational architectures, other approaches might therefore prove to be more fruitful. This includes, for instance, to couple PGAS with parallel MCMC methods (see, e.g., VanDerwerken and Schmidler 2013; Wilkinson 2005) or to use the PGAS Markov kernels together with SMC samplers (Del Moral et al., 2006)

instead of with classical MCMC. The practical usefulness of these approaches is a topic that requires further investigation.

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.

Acknowledgments

The authors would like to thank Sumeetpal S. Singh for pointing out that Metropolis-Hastings sampling can be useful for reducing the computational complexity of the AS step in the non-Markovian setting. This work was supported by: the project Probabilistic modelling of dynamical systems (Contract number: 621-2013-5524) funded by the Swedish Research Council, CADICS, a Linnaeus Center also funded by the Swedish Research Council, and the project Bayesian Tracking and Reasoning over Time (Reference: EP/K020153/1), funded by the EPSRC.

Appendix A. The Relationship between PGAS and PGBS for SSMs

As pointed out in Section 5, there is a close relationship between PGAS and PGBS, in particular when considering the special case of SSMs. PGBS is conceptually similar to PGAS, but it makes use of an explicit backward simulation pass; see Whiteley (2010); Whiteley et al. (2010) or Lindsten and Schön (2013, 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* AS (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,

$$\mathbb{P}(j_t = i \mid j_{t+1}) \propto w_t^i f_\theta(x_{t+1}^{j_{t+1}} \mid x_t^i), \tag{30}$$

and take $x^\star_{1:T} = x^{j_{1:T}}_{1:T}$ 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 $(\mathsf{X}^T, \mathcal{X}^T)$ defined by this procedure be denoted as $P^N_{\mathsf{BS},\theta}$. An interesting question to ask is whether or not the PGAS kernel P^N_θ and the PGBS kernel $P^N_{\mathsf{BS},\theta}$ are probabilistically equivalent. In the specific setting when both methods use the bootstrap proposal kernel in the internal particle filters, it turns out that this is indeed the case. We formalize this is Proposition 2 below. The analysis builds upon Olsson and Rydén (2011, Proposition 5), where the equivalence between a (standard) bootstrap PF and a backward simulator is established. Below, we adapt their argument to handle the case with conditioning on a reference trajectory and the AS step. For improved readability we provide the complete proof, though it should be noted that the main part is due to Olsson and Rydén (2011).

Proposition 2. *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. 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 (2) is given by $W_{\theta,t}(x_t) = g_\theta(y_t \mid x_t)$, i.e., it depends 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 $\{a^N_t\}_{t=2}^T$, meaning that the particle systems, excluding $\{a^N_t\}_{t=2}^T$, 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 $t = 1, \ldots, T$. Then,

$$P^N_\theta(x'_{1:T}, B) = \mathbb{E}\left[\prod_{t=1}^T \mathbb{1}_{B_t}(x_t^{b_t})\right], \qquad \text{and} \qquad P^N_{\mathsf{BS},\theta}(x'_{1:T}, B) = \mathbb{E}\left[\prod_{t=1}^T \mathbb{1}_{B_t}(x_t^{j_t})\right].$$

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 Olsson and Rydén (2011), we establish this result by induction. Hence, for $t < T$, assume that

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

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 $\{w^i_T\}_{i=1}^N$). Let

$$\begin{aligned}\Lambda_t(x_{t+1}^{j_{t+1}}, h) &\triangleq \mathbb{E}\left[h(x_t^{j_t}) \mid x_{t+1}^{j_{t+1}}\right] = \mathbb{E}\left[\mathbb{E}\left[h(x_t^{j_t}) \mid \mathbf{x}_t, x_{t+1}^{j_{t+1}}\right] \mid x_{t+1}^{j_{t+1}}\right] \\ &= \mathbb{E}\left[\sum_{i=1}^N h(x_t^i) \frac{w_t^i f_\theta(x_{t+1}^{j_{t+1}} \mid x_t^i)}{\sum_l w_t^l f_\theta(x_{t+1}^{j_{t+1}} \mid x_t^l)} \mid x_{t+1}^{j_{t+1}}\right],\end{aligned}$$

where we recall that $w_t^i = W_{\theta,t}(x_t^i)$ and where the last equality follows from (30). Consider,

$$\mathbb{E}\left[\prod_{s=t}^T h_s(x_s^{b_s})\right] = \mathbb{E}\left[\mathbb{E}\left[h_t(x_t^{b_t}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] \prod_{s=t+1}^T h_s(x_s^{b_s})\right]. \tag{31}$$

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

$$\mathbb{E}\left[h_t(x_t^{b_t}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] = \mathbb{E}\left[\mathbb{E}\left[h_t(x_t^{b_t}) \mid \mathbf{x}_t, x_{t+1}^{b_{t+1}}, b_{t+1}\right] \mid x_{t+1}^{b_{t+1}}, b_{t+1}\right]. \tag{32}$$

Recall that $b_t = a_{t+1}^{b_{t+1}}$. Consider first the case $b_{t+1} < N$. From (1), 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 AS procedure (Algorithm 2, line 8), the same expression holds also for $b_{t+1} = N$. We can thus write (32) as

$$\mathbb{E}\left[h_t(x_t^{b_t}) \mid x_{t+1:T}^{b_{t+1:T}}, b_{t+1:T}\right] = \mathbb{E}\left[\sum_{i=1}^N h_t(x_t^i) \frac{w_t^i f_\theta(x_{t+1}^{b_{t+1}} \mid x_t^i)}{\sum_l w_t^l f_\theta(x_{t+1}^{b_{t+1}} \mid x_t^l)} \mid x_{t+1}^{b_{t+1}}, b_{t+1}\right] = \Lambda_t(x_{t+1}^{b_{t+1}}, h_t),$$

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 (31) as

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

■

Appendix B. Proof of Proposition 1

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

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

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

$$\begin{aligned}\left(\sum_l w(l)\prod_{s=1}^\ell h_s(l)\right)\prod_{s=\ell+1}^M h_s(k) &\leq \sum_l \left(w(l)\prod_{s=1}^\ell h_s(l)\prod_{s=\ell+1}^M h_s(l)(1+\varepsilon_s)\right) \\ &= \left(\sum_l w(l)\prod_{s=1}^M h_s(l)\right)\prod_{s=\ell+1}^M (1+\varepsilon_s).\end{aligned}$$

It follows that the KL divergence is bounded according to,

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

■

Appendix C. Details on the Experiment in Section 7.1

The parameters of the SV model (24) are $\theta = (\mu, \varphi, \sigma^2, \rho)$. For μ and φ, we use the priors proposed by Kim et al. (1998) (who consider inference in an SV model without the

correlation parameter ρ), namely $\mu \sim \mathcal{N}(0, 10)$ and $\varphi = 2\varphi^\star - 1$ where $\varphi^\star$ is beta distributed; $\varphi^\star \sim \mathcal{B}(20, 1.5)$. Consequently, φ is supported on $(-1, 1)$ with a prior mean of 0.86. This choice is made to ensure stationarity and identifiability of the model. We also use the efficient rejection sampler proposed by Kim et al. (1998) to simulate φ from its posterior conditional distribution. For σ^2 and ρ, we note that the model (24) can be written as

$$\begin{aligned} x_{t+1} &= \mu(1-\varphi) + \varphi x_t + \sigma\rho y_t \exp(-\tfrac{1}{2}x_t) + \sigma\sqrt{1-\rho^2}v_t^\star, \\ y_t &= \exp(-\tfrac{1}{2}x_t)e_t, \end{aligned}$$

where $v_t^\star$ and e_t are mutually independent standard normal. To obtain an efficient updating formula for (σ^2, ρ), we assume a conjugate normal-inverse-gamma prior for the pair $(\vartheta, \varsigma^2) \triangleq (\sigma\rho, \sigma^2(1-\rho^2))$, with $\vartheta \mid \varsigma^2 \sim \mathcal{N}(0, \varsigma^2/0.05)$ and $\varsigma^2 \sim \mathcal{IG}(5/2, 0.05/2)$. We also investigated the possibility of letting σ^2 and ρ be *a priori* independent with an inverse gamma and a uniform prior, respectively, but we did not experience any notable differences in the posterior distributions.

In the experiments, all the samplers are initialized at $\theta[0] = (0, 0.975, 0.05, 0)$. For PMMH, we tune the covariance matrix of the random walk proposal distribution according to the posterior distribution obtained from an initial trial run, using PGAS with $N = 20$ for 10 000 iterations.

References

C. Andrieu, E. Moulines, and P. Priouret. Stability of stochastic approximation under verifiable conditions. *SIAM Journal on Control and Optimization*, 44(1):283–312, 2005.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):269–342, 2010.

A. Benveniste, M. Métivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, New York, USA, 1990.

G. J. Bierman. Fixed interval smoothing with discrete measurements. *International Journal of Control*, 18(1):65–75, 1973.

R. Chen and J. S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society: Series B*, 62(3):493–508, 2000.

N. Chopin and S. S. Singh. On particle Gibbs sampling. *Bernoulli*, 2014. Forthcoming.

P. de Jong and N. Shephard. The simulation smoother for time series models. *Biometrika*, 82(2):339–350, 1995.

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

B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94–128, 1999.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovskii, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

A. Doucet, M. K. Pitt, G. Deligiannidis, and R. Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. arXiv.org, arXiv:1210.1871v3, March 2014.

D. A. Van Dyk and T. Park. Partially collapsed Gibbs samplers: Theory and methods. *Journal of the American Statistical Association*, 103(482):790–796, 2008.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

P. Fearnhead, O. Papaspiliopoulos, and G. O. Roberts. Particle filters for partially observed diffusions. *Journal of the Royal Statistical Society: Series B*, 70(4):755–777, 2008.

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*. December 2013.

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*, 137(10):3068–3081, 2007.

C. J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

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*, 457:1012–1014, 2009.

S. J. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

A. Golightly and D. J. Wilkinson. Bayesian inference for nonlinear multivariate diffusion models observed with error. *Computational Statistics & Data Analysis*, 52(3):1674–1693, 2008.

A. Golightly and D. J. Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*, 1(6):807–820, 2011.

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*, 50(2):425–437, 2002.

J. Hallgren and T. Koski. Decomposition sampling applied to parallelization of Metropolis-Hastings. arXiv.org, arXiv:1402.2828, February 2014.

N. L. Hjort, C. Holmes, P. Müller, and S. G. Walker, editors. *Bayesian Nonparametrics*. Cambridge University Press, 2010.

P. E. Jacob, L. M. Murray, and S. Rubenthaler. Path storage in the particle filter. *Statistics and Computing*, 2013. doi: 10.1007/s11222-013-9445-x. (available online).

M.J. Keeling and P. Rohani. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press, 2007.

S. Kim, N. Shephard, and S. Chib. Stochastic volatility: Likelihood inference and comparison with ARCH models. *The Review of Economic Studies*, 65(3):361–393, 1998.

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.

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, March 2012.

F. Lindsten and T. B. Schön. Backward simulation methods for Monte Carlo statistical inference. *Foundations and Trends in Machine Learning*, 6(1):1–143, 2013.

F. Lindsten, M. I. Jordan, and T. B. Schön. Ancestor sampling for particle Gibbs. In P. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS) 25*, pages 2600–2608. 2012.

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.

J. S. Liu. Peskun's theorem and a modified discrete-state Gibbs sampler. *Biometrika*, 83 (3):681–682, 1996.

J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001.

G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Whiley Series in Probability and Statistics. John Wiley & Sons, New York, USA, second edition, 2008.

L. M. Murray, E. M. Jones, and J. Parslow. On disturbance state-space models and the particle marginal Metropolis-Hastings sampler. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):494–521, 2013.

J. Olsson and T. Rydén. Rao-Blackwellization of particle Markov chain Monte Carlo methods using forward filtering backward sampling. *IEEE Transactions on Signal Processing*, 59(10):4606–4619, 2011.

O. Papaspiliopoulos, G. O. Roberts, and M. Sköld. Non-centered parameterisations for hierarchical models and data augmentation. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, editors, *Bayesian Statistics 7*, pages 307–326. Oxford University Press, 2003.

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

M. K. Pitt, R. S. Silva, P. Giordani, and R. Kohn. Auxiliary particle filtering within adaptive Metropolis-Hastings sampling. arXiv.org, arXiv:1006.1914, June 2010.

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*, 171:134–151, 2012.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

D. A. Rasmussen, O. Ratmann, and K. Koelle. Inference for nonlinear epidemiological models using genealogies and time series. *PLoS Comput Biology*, 7(8), 2011.

B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter: particle filters for tracking applications*. Artech House, London, UK, 2004.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

N. Shephard, editor. *Stochastic Volatility: Selected Readings*. Oxford University Press, 2005.

A. Skvortsov and B. Ristic. Monitoring and prediction of an epidemic outbreak using syndromic observations. *Mathematical Biosciences*, 240(1):12–19, 2012.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

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.

D. N. VanDerwerken and S. C. Schmidler. Parallel Markov chain Monte Carlo. arXiv.org, arXiv:1312.7479, December 2013.

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*, 51:457–478, 2013.

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*, 85(411):699–704, 1990.

N. Whiteley. Discussion on Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):306–307, 2010.

N. Whiteley, C. Andrieu, and A. Doucet. Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods. Technical report, Bristol Statistics Research Report 10:04, 2010.

D. J. Wilkinson. Parallel Bayesian computation. In *Handbook of Parallel Computing and Statistics*. Chapman & Hall/CRC, 2005.