ON THE USE OF BACKWARD SIMULATION IN THE PARTICLE GIBBS SAMPLER

Fredrik Lindsten and Thomas B. Schön

Division of Automatic Control, Linköping University, SE-581 83 Linköping, Sweden.
{lindsten, schon}@isy.liu.se

ABSTRACT

The particle Gibbs (PG) sampler was introduced in [1] as a way to incorporate a particle filter (PF) in a Markov chain Monte Carlo (MCMC) sampler. The resulting method was shown to be an efficient tool for joint Bayesian parameter and state inference in nonlinear, non-Gaussian state-space models. However, the mixing of the PG kernel can be very poor when there is severe degeneracy in the PF. Hence, the success of the PG sampler heavily relies on the, often unrealistic, assumption that we can implement a PF without suffering from any considerate degeneracy. However, as pointed out by Whiteley [2] in the discussion following [1], the mixing can be improved by adding a backward simulation step to the PG sampler. Here, we investigate this further, derive an explicit PG sampler with backward simulation (denoted PG-BSi) and show that this indeed is a valid MCMC method. Furthermore, we show in a numerical example that backward simulation can lead to a considerable increase in performance over the standard PG sampler.

Index Terms— Particle Markov chain Monte Carlo, particle filter, particle Gibbs, backward simulation, Gibbs sampling.

1. INTRODUCTION

Consider a general, discrete-time state-space model with state-space X, parameterised by $\theta \in \Theta$,

$$x_{t+1} \sim f_\theta(x_{t+1} \mid x_t), \qquad y_t \sim g_\theta(y_t \mid x_t).$$

We take a Bayesian viewpoint and model θ as a random variable with prior density $p(\theta)$. Given observations $y_{1:T} \triangleq \{y_1, \dots, y_T\}$, we wish to identify the parameter θ as well as the system states $x_{1:T}$. That is, we seek the posterior density $p(\theta, x_{1:T} \mid y_{1:T})$.

The Gibbs sampler (see e.g. [3]) is an MCMC method which can be used to sample from some joint target density, when sampling from its conditionals is tractable. In an idealised Gibbs sampler, we would target the density $p(\theta, x_{1:T} \mid y_{1:T})$ by the following two-step sweep; **i)** Draw $\theta^\star \mid x_{1:T} \sim p(\theta \mid x_{1:T}, y_{1:T})$; **ii)** Draw $x^\star_{1:T} \mid \theta^\star \sim p_{\theta^\star}(x_{1:T} \mid y_{1:T})$. The first part of this sampling scheme is often straightforward (if we can use conjugate priors for the parameters) and is thus assumed to be tractable. However, the second part is often very difficult. The idea behind the PG sampler of [1] is to replace this step with a sample trajectory generated by a PF. In this work, we will see that it can be beneficial to complement the PF with a backward simulator when generating this sample trajectory. The reason is that it will result in a faster mixing Gibbs kernel. This idea was mentioned in the discussion by Whiteley [2] and also used in a special case of PG in [4]. In [5], backward simulation is used in the context of particle independent Metropolis-Hastings.

This work was supported by the project Calibrating Nonlinear Dynamical Models (Contract number: 621-2010-5876) and the Linneaus Center CADICS, both funded by the Swedish Research Council.

2. PARTICLE FILTER AND BACKWARD SIMULATOR

Before we go in to the details of the *particle Gibbs with backward simulation* (PG-BSi) method in Section 3, we review the PF and the backward simulator. Throughout this section we assume that the parameter θ is fixed and discuss how to approximately sample from the joint smoothing density $p_\theta(x_{1:T} \mid y_{1:T})$.

2.1. Auxiliary particle filter

The auxiliary particle filter (APF) [6] can be used to approximate (and approximately sample from) the joint smoothing distribution. We assume that the reader is familiar with the APF and keep the presentation quite brief, mainly just to introduce the required notation. For readers not familiar with the PF or the APF, we refer to [7, 6]. Let $\{x^m_{1:t-1}, w^m_{t-1}\}_{m=1}^N$ be a weighted particle system targeting $p_\theta(x_{1:t-1} \mid y_{1:t-1})$. That is, the particle system defines an empirical distribution,

$$\widehat{p}^N_\theta(dx_{1:t-1} \mid y_{1:t-1}) \triangleq \sum_{m=1}^N \frac{w^m_{t-1}}{\sum_l w^l_{t-1}} \delta_{x^m_{1:t-1}}(dx_{1:t-1}),$$

which approximates the target distribution. In the APF, we propagate this sample to time t by sampling from a proposal kernel,

$$M^\theta_t(i_t, x_t) \triangleq \frac{w^{i_t}_{t-1}\nu^{i_t}_{t-1}}{\sum_l w^l_{t-1}\nu^l_{t-1}} R^\theta_t(x_t \mid x^{i_t}_{t-1}).$$

Here, the variable i_t is the index to an "ancestor particle" $x^{i_t}_{t-1}$ and R^θ_t is a proposal kernel which proposes a new particle at time t given this ancestor. The factors $\nu^{i_t}_{t-1} = \nu^\theta_{t-1}(x^{i_t}_{t-1}, y_t)$, known as adjustment multiplier weights, are used in the APF to increase the probability of sampling ancestors that better can describe the current observation. If these adjustment multiplier weights are identically equal to 1, we recover the standard PF.

Once we have generated N new particles (and ancestor indices) from the kernel M^θ_t, the particles are assigned importance weights according to $w^m_t = W^\theta_t(x^m_t, x^{i^m_t}_{t-1})$ for $m = 1, \dots, N$, where the weight function is given by,

$$W^\theta_t(x_t, x_{t-1}) \triangleq \frac{g_\theta(y_t \mid x_t) f_\theta(x_t \mid x_{t-1})}{\nu^\theta_{t-1}(x_{t-1}, y_t) R^\theta_t(x_t \mid x_{t-1})}.$$

This results in a new weighted particle system $\{x^m_{1:t}, w^m_t\}_{m=1}^N$, targeting the joint smoothing density at time t.

The method is initialised by sampling from a proposal density $x^m_1 \sim R^\theta_1(x_1)$ and assigning weights $w^m_1 = W^\theta_1(x^m_1)$ where the weight function is given by $W^\theta_1(x_1) \triangleq g_\theta(y_1 \mid x_1)p_\theta(x_1)/R^\theta_1(x_1)$.

After a complete run of the APF we can sample a trajectory from the empirical joint smoothing distribution by choosing particle $x^m_{1:T}$

Algorithm 1 Backward simulator

```
1. Initialise: Set $j_T = m$ with probability $w_T^m / \sum_l w_T^l$.
2. For $t = T-1 : -1 : 1$ do:
    (a) Given $x_{t+1}^{j_{t+1}}$, compute the smoothing weights,
        $$w_{t|T}^m = \frac{w_t^m f_\theta(x_{t+1}^{j_{t+1}} \mid x_t^m)}{\sum_l w_t^l f_\theta(x_{t+1}^{j_{t+1}} \mid x_t^l)}, \qquad m = 1, \ldots, N.$$
    (b) Set $j_t = m$ with probability $w_{t|T}^m$.
```

with probability proportional to w_T^m. Note that this particle trajectory consists of the ancestral lineage of x_T^m. Hence, if this lineage is defined by the indices $b_{1:T}$ (with $b_T = m$) we obtain a sample

$$x_{1:T}^\star = x_{1:T}^m = \{x_t^{b_1}, \ldots, x_T^{b_T}\}. \tag{1}$$

2.2. Backward simulation

In the original PG sampler by [1], sample trajectories are generated as in (1). However, due to the degeneracy of the APF, the resulting Gibbs sampler can suffer from poor mixing. We will illustrate this in Section 4. In this work, we use an alternative way to generate an approximate sample from the joint smoothing density, known as backward simulation. It was introduced as a particle smoother in [8]. In [2], it was suggested as a possible way to increase the mixing rate of a PG kernel, an idea which we elaborate on in this work.

Consider a factorisation of the joint smoothing density as $p_\theta(x_{1:T} \mid y_{1:T}) = p_\theta(x_T \mid y_{1:T}) \prod_{t=1}^{T-1} p_\theta(x_t \mid x_{t+1}, y_{1:t})$. Here, the *backward kernel* density can be written as,

$$p_\theta(x_t \mid x_{t+1}, y_{1:t}) = \frac{f_\theta(x_{t+1} \mid x_t) p_\theta(x_t \mid y_{1:t})}{p_\theta(x_{t+1} \mid y_{1:t})}.$$

We note that the backward kernel depends on the filtering density $p_\theta(x_t \mid y_{1:t})$. The key enabler of the backward simulator is that this density can be readily approximated by an (A)PF, without suffering from degeneracy. By using the APF to approximate the filtering density, we obtain an approximation of the backward kernel,

$$\widehat{p}_\theta^N(dx_t \mid x_{t+1}, y_{1:t}) \triangleq \sum_{m=1}^N \frac{w_t^m f_\theta(x_{t+1} \mid x_t^m)}{\sum_l w_t^l f_\theta(x_{t+1} \mid x_t^l)} \delta_{x_t^m}(dx_t).$$

Based on this approximation, we may sample a particle trajectory backward in time, approximately distributed according to the joint smoothing density. The procedure is given in Algorithm 1. The algorithm generates a set of indices $j_{1:T}$ defining a backward trajectory according to

$$x_{1:T}^\star = \{x_t^{j_1}, \ldots, x_T^{j_T}\}. \tag{2}$$

Note that, since we only need to generate a single backward trajectory, the computational cost of the backward simulator is $O(N)$, i.e. of the same order as the APF. Hence, the computational complexity of the PG sampler is more or less unaffected by the introduction of the backward simulation step. Still, it is possible to reduce the complexity of the backward simulator even further by making use of rejection sampling [9].

3. PARTICLE GIBBS W. BACKWARD SIMULATION

Recall the idealised Gibbs sampler, briefly outlined in Section 1. The two steps of the method which are performed at each iteration are,

$$\theta^\star \mid x_{1:T} \sim p(\theta \mid x_{1:T}, y_{1:T}), \tag{3a}$$
$$x_{1:T}^\star \mid \theta^\star \sim p_{\theta^\star}(x_{1:T} \mid y_{1:T}). \tag{3b}$$

As previously mentioned, the idea behind the PG sampler by [1] is, instead of sampling according to (3b), to run an APF to generate a sample trajectory $x_{1:T}^\star$ as in (1). In the PG-BSi sampler proposed in this work, we instead generate a sample trajectory by backward simulation, as in (2). However, to simply replace step (3b) of the idealised Gibbs sampler with such an approximate sampling procedure will not result in a valid approach. In [1] this is solved by replacing the PF with a so called *conditional SMC sampler* and it is shown that the resulting PG sampler is a valid MCMC method. In this section, we make a similar derivation for the PG-BSi sampler.

The key in deriving the PG-BSi sampler is to consider a Markov chain in which the state consists of all the random variables generated by the particle filter and the backward simulator, i.e. the complete set of particles, particle indices and backward trajectory indices. For this cause, let us start by determining the density of all the random variables generated by the APF. Let $\mathbf{x}_t = \{x_t^1, \ldots, x_t^N\}$ and $\mathbf{i}_t = \{i_t^1, \ldots, i_t^N\}$ be the particles and ancestor indices, respectively, generated by the APF at time t. Note that i_t^m is the index of the ancestor of particle x_t^m. For a fixed θ the APF then samples from,

$$\psi^\theta(\mathbf{x}_{1:T}, \mathbf{i}_{2:T}) = \prod_{l=1}^N R_1^\theta(x_1^l) \prod_{t=2}^T \prod_{m=1}^N M_t^\theta(i_t^m, x_t^m).$$

Note that the APF is simply a method which generates a single sample from ψ, on the extended space $\mathsf{X}^{NT} \times \{1, \ldots, N\}^{N(T-1)}$.

By completing this with a backward simulation according to Algorithm 1, we generate the indices $j_{1:T}$ defining a backward trajectory. Let the joint density of $\{\mathbf{x}_{1:T}, \mathbf{i}_{2:T}, j_{1:T}\}$ be $\psi'^{,\theta}(\mathbf{x}_{1:T}, \mathbf{i}_{2:T}, j_{1:T})$. Then, the trajectory $x_{1:T}^{j_{1:T}} \triangleq \{x_1^{j_1}, \ldots, x_T^{j_T}\}$ will be distributed (jointly with the random indices $j_{1:T}$) according to a marginal of this density, $\psi'^{,\theta}(x_{1:T}^{j_{1:T}}, j_{1:T})$, which in general is not equal to the joint smoothing density. In other words, the trajectory $x_{1:T}^{j_{1:T}}$ is not distributed according to $p_\theta(x_{1:T} \mid y_{1:T})$ and to simply use this state trajectory in step (3b) of the idealised Gibbs sampler would not be a valid approach.

The key to circumventing this problem is to introduce a new, artificial target density, which should, **i)** admit the joint smoothing density as one of its marginals; **ii)** be "similar" to ψ^θ, to enable the application of an APF in the sampling procedure. Here, we define this density according to,

$$\phi(\theta, \mathbf{x}_{1:T}, \mathbf{i}_{2:T}, j_{1:T}) \triangleq \frac{p(\theta, x_{1:T}^{j_{1:T}} \mid y_{1:T})}{N^T}$$
$$\times \frac{\psi^\theta(\mathbf{x}_{1:T}, \mathbf{i}_{2:T})}{R_1^\theta(x_1^{j_1}) \prod_{t=2}^T M_t^\theta(i_t^{j_t}, x_t^{j_t})} \prod_{t=2}^T \frac{w_{t-1}^{i_t^{j_t}} f_\theta(x_t^{j_t} \mid x_{t-1}^{i_t^{j_t}})}{\sum_l w_{t-1}^l f_\theta(x_t^{j_t} \mid x_{t-1}^l)}.$$

This artificial target density differs from the one used in [1] in the last factor. One of the important properties of the density ϕ is formalised in the following proposition (all propositions are given without proofs, due to the lack of space).

Proposition 1. The marginal density of $\{\theta, x_{1:T}^{j_{1:T}}, j_{1:T}\}$ under ϕ is given by $\phi(\theta, x_{1:T}^{j_{1:T}}, j_{1:T}) = p(\theta, x_{1:T}^{j_{1:T}} \mid y_{1:T})/N^T$. Thus, with

Algorithm 2 CAPF – conditional APF (conditioned on $\{x'_{1:T}, j_{1:T}\}$)

```
1. Initialise:
   (a) Sample $x_1^m \sim R_1^\theta(x_1)$ for $m \neq j_1$ and set $x_1^{j_1} = x'_1$.
   (b) Set $w_1^m = W_1^\theta(x_1^m)$ for $m = 1, \ldots, N$.
2. For $t = 2, \ldots, T$ do:
   (a) Draw $\{i_t^m, x_t^m\} \sim M_t^\theta(i_t, x_t)$ for $m \neq j_t$ and set $x_t^{j_t} = x'_t$.
   (b) Draw $i_t^{j_t}$ according to,
       $$\mathrm{P}(i_t^{j_t} = m) = \frac{w_{t-1}^m f_\theta(x_t^{j_t} \mid x_{t-1}^m)}{\sum_l w_{t-1}^l f_\theta(x_t^{j_t} \mid x_{t-1}^l)}.$$
   (c) Set $w_t^m = W_t^\theta(x_t^m, x_{t-1}^{i_t^m})$ for $m = 1, \ldots, N$.
```

$\{\theta, \mathbf{x}_{1:T}, \mathbf{i}_{2:T}, j_{1:T}\}$ being a sample distributed according to ϕ, the density of $\{\theta, x_{1:T}^{j_{1:T}}\}$ is given by $p(\theta, x_{1:T}^{j_{1:T}} \mid y_{1:T})$. □

This proposition has the following important implication; if we can construct an MCMC method (e.g. a Gibbs sampler) with stationary distribution ϕ, then the subchain given by the variables $\{\theta, x_{1:T}^{j_{1:T}}\}$ will have stationary distribution $p(\theta, x_{1:T} \mid y_{1:T})$. The construction of such an MCMC sampler is enabled by the second important property of our artificial density, namely that it is constructed in such a way that we are able to sample from the conditional $\phi(\mathbf{x}_{1:T}^{-j_{1:T}}, \mathbf{i}_{2:T} \mid \theta, x_{1:T}^{j_{1:T}}, j_{1:T})$. Here we have introduced the notation $\mathbf{x}_{1:T}^{-j_{1:T}} = \{\mathbf{x}_1^{-j_1}, \ldots, \mathbf{x}_T^{-j_T}\}$ and $\mathbf{x}_t^{-j_t} = \{x_t^1, \ldots, x_t^{(j_t)-1}, x_t^{(j_t)+1}, \ldots, x_t^N\}$. Sampling from this conditional can be done by running a so called conditional APF, given in Algorithm 2. The CAPF is similar to the conditional SMC introduced in [1], but with the addition of step 2(b). The reason for this difference lies in the interpretation of the indices $j_{1:T}$. Here they correspond to backward trajectory indices, whereas in the original PG sampler they are used to define an ancestral path.

As a final component of the PG-BSi sampler we need a way to generate a sample state trajectory. As previously pointed out, we aim to do this by running a backward simulator. The fact that the backward simulator indeed generates a sample from one of the conditionals of ϕ is assessed in the following proposition.

Proposition 2. The conditional $\phi(j_{1:T} \mid \theta, \mathbf{x}_{1:T}, \mathbf{i}_{2:T})$ under ϕ is given by,

$$\phi(j_{1:T} \mid \theta, \mathbf{x}_{1:T}, \mathbf{i}_{2:T}) = \frac{w_T^{j_T}}{\sum_l w_T^l} \prod_{t=2}^{T} \frac{w_{t-1}^{j_{t-1}} f_\theta(x_t^{j_t} \mid x_{t-1}^{j_{t-1}})}{\sum_l w_{t-1}^l f_\theta(x_t^{j_t} \mid x_{t-1}^l)}.$$

Consequently, the backward simulator given in Algorithm 1, conditioned on $\{\theta, \mathbf{x}_{1:T}, \mathbf{i}_{2:T}\}$, generates a sample from this conditional distribution. □

We are now ready to present the PG-BSi sampler, given in Algorithm 3. Based on the propositions and discussion above, the three steps of the method [2(a)–2(c)] can be interpreted as:

(a) Draw $\theta^\star \sim \phi(\theta \mid x_{1:T}^{j_{1:T}}, j_{1:T})$,

(b) Draw $\mathbf{x}_{1:T}^{\star, -j_{1:T}}, \mathbf{i}_{2:T}^\star \sim \phi(\mathbf{x}_{1:T}^{-j_{1:T}}, \mathbf{i}_{2:T} \mid \theta^\star, x_{1:T}^{j_{1:T}}, j_{1:T})$,

(c) Draw $j_{1:T}^\star \sim \phi(j_{1:T} \mid \theta^\star, \mathbf{x}_{1:T}^{\star, -j_{1:T}}, \mathbf{i}_{2:T}^\star, x_{1:T}^{j_{1:T}})$.

Hence, each step of Algorithm 3 is a standard Gibbs update, targeting the distribution ϕ, and will thus leave ϕ invariant (step (a) is

Algorithm 3 PG-BSi – particle Gibbs with backward simulation

```
1. Initialise: Set $\theta(0)$, $x_{1:T}(0)$ and $j_{1:T}(0)$ arbitrarily.
2. For $r \geq 1$, iterate:
   (a) Sample $\theta(r) \sim p(\theta \mid x_{1:T}(r-1), y_{1:T})$.
   (b) Run a CAPF targeting $p_{\theta(r)}(x_{1:T} \mid y_{1:T})$, conditioned on $\{x_{1:T}(r-1), j_{1:T}(r-1)\}$.
   (c) Run a backward simulator to generate $j_{1:T}(r)$. Set $x_{1:T}(r)$ to the corresponding particle trajectory.
```

known as a collapsed Gibbs update, see e.g. [3, Sec. 6.7]). It follows that ϕ is a stationary distribution of the PG-BSi sampler. To establish its validity as an MCMC method, the only thing that remains is to show that it is ergodic (and hence converges towards its stationary distribution). This property is established for the PG sampler in [1, Theorem 5] and the same result holds, with a similar argument, for the PG-BSi sampler. Hence, we shall not elaborate on this further, but simply conclude that the PG-BSi sampler indeed is a valid MCMC sampler.

4. NUMERICAL ILLUSTRATION

We have argued that the PG-BSi sampler in Algorithm 3 should be preferable over the standard PG sampler in the sense that it produces a Gibbs sampler with better mixing properties. In Section 4.1 we illustrate this on a numerical example and show that PG-BSi can be far superior to PG for certain settings, and in Section 4.2 we discuss the reason for the big difference in performance between the algorithms.

4.1. Numerical example

We reuse an example from [1], used to illustrate the particle MCMC methods presented there. Consider the following state-space model,

$$x_{t+1} = 0.5x_t + 25\frac{x_t}{1+x_t^2} + 8\cos(1.2t) + v_t, \tag{4a}$$

$$y_t = 0.05x_t^2 + e_t, \tag{4b}$$

where $x_1 \sim \mathcal{N}(0, 5)$, $v_t \sim \mathcal{N}(0, \sigma_v^2)$ and $e_t \sim \mathcal{N}(0, \sigma_e^2)$; here $\mathcal{N}(0, \sigma^2)$ denotes a zero-mean Gaussian with variance σ^2. We set $\theta = \begin{pmatrix}\sigma_v^2 & \sigma_e^2\end{pmatrix}^T$.

We generate a set of observations $y_{1:500}$ according to (4) with $\sigma_v^2 = 10$ and $\sigma_e^2 = 1$. The parameter priors are modelled as inverse gamma distributed with hyperparameters $a = b = 0.01$. We then employ the PG sampler of [1] and the PG-BSi sampler of Algorithm 3 to estimate the posterior parameter distribution. Since the theoretical validity of the PG-BSi and the PG samplers can be assessed for any number of particles $N \geq 2$ (see [1, Theorem 5]), it is interesting to see the practical implications of using very few particles. Hence, we run the methods four times on the same data with $N = 5$, 20, 1000 and 5000 ($N = 5000$ as was used in [1]). The parameters are initialised at $\theta(0) = \begin{pmatrix}10 & 10\end{pmatrix}^\mathsf{T}$ in all experiments.

In Figure 1 we show the estimated posterior means of the standard deviations σ_v and σ_e (i.e. the square roots of the means of $\theta(1:r)$) as functions of the number of MCMC iterations r. The left column shows the results for the PG sampler. When we use $N = 5000$ particles, there is a rapid convergence indicating that the method mixes well, and quickly finds a region of high posterior probability. However, as we decrease the number of particles, the convergence gets much slower. Even for $N = 1000$, the method

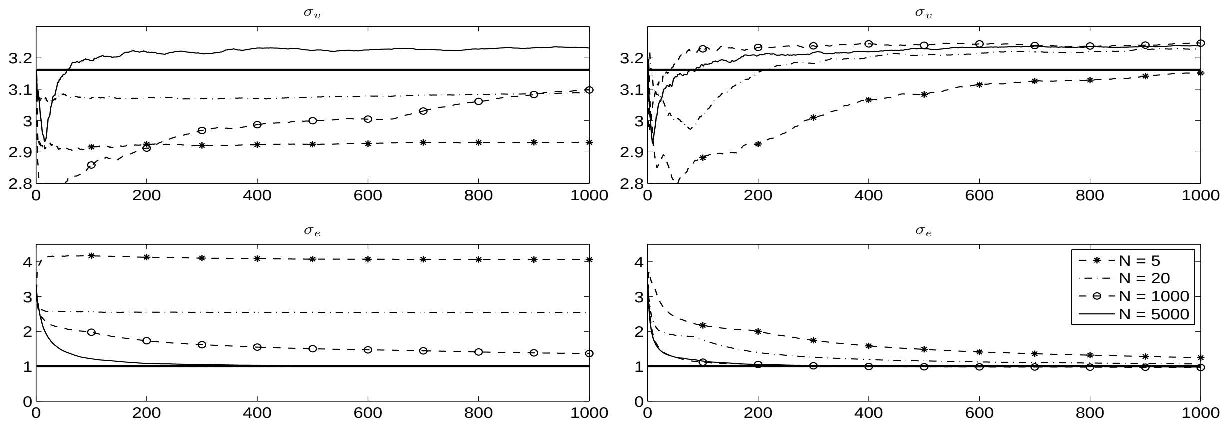

Fig. 1. Running posterior means of the standard deviations σ_v and σ_e for the PG sampler (left) and PG-BSi sampler (right) for different number of particles. The "true" values are shown as thick solid lines. Note that the estimate of σ_v do not converge to this "true" value, but this is not surprising since we consider just one data realisation. Hence, the posterior mean may very well differ from the "true" value.

struggles and for $N = 20$ and $N = 5$ it does not seem to converge at all (in a reasonable amount of time). On the contrary, the PG-BSi sampler (right column) seems to be more or less unaffected by the large decrease in the number of particles. In fact, the PG-BSi sampler using just $N = 5$ performs equally well as the PG sampler using $N = 1000$ particles. It could be argued that one of the major strengths of particle MCMC methods is that they do not require an accurate approximation of the joint smoothing distribution at each iteration, but merely a single sample which is approximately distributed according to it. Hence, we do not need to waste computational resources on using too many particles, since we only extract a single particle trajectory anyway. However, for this to be true for the PG sampler, it seems crucial to complement it with a backward simulation pass.

4.2. Discussion

In the previous section we saw that the mixing of the PG sampler worsened significantly when we decreased the number of particles, whereas the PG-BSi sampler seemed to be more or less unaffected. This can be explained in terms of the well known degeneracy problem for particle filters. Let $x_{1:T}(r)$ be the state trajectory sampled at iteration r of the PG sampler. At iteration $r + 1$ we run a CAPF, conditioned on $x_{1:T}(r)$. Due to degeneracy, all particle trajectories generated by the CAPF will coincide for $t \ll T$. Now, since we condition on $x_{1:T}(r)$, the particle tree generated by the CAPF *must* contain $x_{1:T}(r)$. Hence, when we sample the trajectory $x_{1:T}(r + 1)$ at iteration $r+1$, this will to a large extent be identical $x_{1:T}(r)$. This results in a poor exploration of the state space, which in turn leads to a slowly mixing Gibbs kernel.

The PG-BSi method is able to circumvent this by exploring all possible particle combinations when generating a particle trajectory, and is thus not constrained to one of the ancestral paths. Due to this, the sample trajectory $x_{1:T}(r + 1)$ will, with high probability, be entirely different from $x_{1:T}(r)$. This leads to a much better exploration of the state space and thus a faster mixing Gibbs kernel.

5. CONCLUSIONS

We have investigated the possibility to apply a backward simulator to generate a sample trajectory in the particle Gibbs (PG) sampler. We have shown that this indeed leads to a valid MCMC method. To enable this we introduced a slight modification of the conditional particle filter, used in the original PG sampler. We have shown in a numerical example that the mixing of the Gibbs kernel can be increased substantially by using backward simulation. This is true especially when we use few particles and/or when the number of observations is large. As opposed to the original PG sampler (without backward simulation), the PG sampler with backward simulation is shown to be rather insensitive to a decreased number of particles. In a numerical example, for which the original PG sampler required around 1000 particles, the proposed method could manage with as few as 5 particles.

6. REFERENCES

[1] C. Andrieu, A. Doucet, and R. Holenstein, "Particle Markov chain Monte Carlo methods," *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 3, pp. 269–342, 2010.

[2] N. Whiteley, "Discussion on Particle Markov chain Monte Carlo methods," Journal of the Royal Statistical Society: Series B, 72, p 306–307, 2010.

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

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

[5] J. Olsson and T. Rydén, "Rao-Blackwellization of particle Markov chain Monte Carlo methods using forward filtering backward sampling," *IEEE Transactions on Signal Processing*, vol. 59, no. 10, pp. 4606–4619, 2011.

[6] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999.

[7] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[8] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 156–168, Mar. 2004.

[9] R. Douc, A. Garivier, E. Moulines, and J. Olsson, "Sequential Monte Carlo smoothing for general state space hidden Markov models," *Annals of Applied Probability*, vol. 21, no. 6, pp. 2109–2145, 2011.