Interacting Particle Markov Chain Monte Carlo

Tom Rainforth[1]* TWGR@ROBOTS.OX.AC.UK
Christian A. Naesseth[2]* CHRISTIAN.A.NAESSETH@LIU.SE
Fredrik Lindsten[3] FREDRIK.LINDSTEN@IT.UU.SE
Brooks Paige[1] BROOKS@ROBOTS.OX.AC.UK
Jan-Willem van de Meent[1] JWVDM@ROBOTS.OX.AC.UK
Arnaud Doucet[1] DOUCET@STATS.OX.AC.UK
Frank Wood[1] FWOOD@ROBOTS.OX.AC.UK

∗ equal contribution
[1] The University of Oxford, Oxford, United Kingdom
[2] Linköping University, Linköping, Sweden
[3] Uppsala University, Uppsala, Sweden

Abstract

We introduce *interacting particle Markov chain Monte Carlo* (iPMCMC), a PMCMC method based on an interacting pool of standard and conditional sequential Monte Carlo samplers. Like related methods, iPMCMC is a Markov chain Monte Carlo sampler on an extended space. We present empirical results that show significant improvements in mixing rates relative to both non-interacting PMCMC samplers, and a single PMCMC sampler with an equivalent memory and computational budget. An additional advantage of the iPMCMC method is that it is suitable for distributed and multi-core architectures.

1. Introduction

MCMC methods are a fundamental tool for generating samples from a posterior density in Bayesian data analysis (see e.g., Robert & Casella (2013)). Particle Markov chain Monte Carlo (PMCMC) methods, introduced by Andrieu et al. (2010), make use of sequential Monte Carlo (SMC) algorithms (Gordon et al., 1993; Doucet et al., 2001) to construct efficient proposals for the MCMC sampler.

One particularly widely used PMCMC algorithm is particle Gibbs (PG). The PG algorithm modifies the SMC step in the PMCMC algorithm to sample the latent variables conditioned on an existing particle trajectory, resulting in what is called a conditional sequential Monte Carlo (CSMC) step. The PG method was first introduced as an efficient Gibbs sampler for latent variable models with static parameters (Andrieu et al., 2010). Since then, the PG algorithm and the extension by Lindsten et al. (2014) have found numerous applications in e.g. Bayesian non-parametrics (Valera et al., 2015; Tripuraneni et al., 2015), probabilistic programming (Wood et al., 2014; van de Meent et al., 2015) and graphical models (Everitt, 2012; Naesseth et al., 2014; 2015).

A drawback of PG is that it can be particularly aversely effected by *path degeneracy* in the CSMC step. The forced survival of the conditional trajectory means that whenever resampling of the trajectories results in a common ancestor, this ancestor must correspond to the existing trajectory. Consequently, the mixing of the Markov chain for the early steps in the state sequence can become very slow, if insufficient particles are used to prevent degeneracy.

In this paper we propose the interacting particle Markov chain Monte Carlo (iPMCMC) sampler. In iPMCMC we run a pool of CSMC and unconditional SMC algorithms as parallel processes that we refer to as nodes. After each run of this pool, we apply successive Gibbs updates to the indexes of the CSMC nodes, such that the indices of the CSMC nodes changes. Hence, the nodes from which retained particles are sampled can change from one MCMC iteration to the next. This lets us trade off exploration (SMC) and exploitation (CSMC) to achieve improved mixing of the Markov chains. Crucially, the pool provides numerous candidate indices at each Gibbs update, giving a significantly higher probability that an entirely new retained particle will be "switched in" than in non-interacting alternatives.

This interaction requires only minimal communication; each node must report an estimate of the marginal likelihood and receive a new role (SMC or CSMC) for the next sweep. This

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

means that iPMCMC is embarrassingly parallel and can be run in a distributed manner on multiple computers.

We prove that iPMCMC is a partially collapsed Gibbs sampler on the extended space containing the particle sets for all nodes. In the special case where iPMCMC uses only *one* CSMC node, it can in fact be seen as a non-trivial and unstudied instance of the α-SMC-based (Whiteley et al., 2016) PMCMC method introduced by Huggins & Roy (2015). However, with iPMCMC we extend this further to allow for an arbitrary number of CSMC and standard SMC algorithms with interaction. Our experimental evaluation shows that iPMCMC outperforms both independent PG samplers as well as a single PG sampler with the same number of particles run longer to give a matching computational budget.

An implementation of iPMCMC is provided in the probabilistic programming system *Anglican*[1] (Wood et al., 2014), whilst illustrative MATLAB code, similar to that used for the experiments, is also provided[2].

2. Background

We start by briefly reviewing sequential Monte Carlo (Gordon et al., 1993; Doucet et al., 2001) and the particle Gibbs algorithm (Andrieu et al., 2010). Let us consider a non-Markovian latent variable model of the following form

$$x_t | x_{1:t-1} \sim f_t(x_t | x_{1:t-1}), \qquad \text{(1a)}$$

$$y_t | x_{1:t} \sim g_t(y_t | x_{1:t}), \qquad \text{(1b)}$$

where $x_t \in \mathsf{X}$ is the latent variable and $y_t \in \mathsf{Y}$ the observation at time step t, respectively, with transition densities f_t and observation densities g_t; x_1 is drawn from some initial distribution $\mu(\cdot)$. The method we propose is not restricted to the above model, it can in fact be applied to an arbitrary sequence of targets.

We are interested in calculating expectation values with respect to the posterior distribution $p(x_{1:T}|y_{1:T})$ on latent variables $x_{1:T} := (x_1, \dots, x_T)$ conditioned on observations $y_{1:T} := (y_1, \dots, y_T)$, which is proportional to the joint distribution $p(x_{1:T}, y_{1:T})$,

$$p(x_{1:T}|y_{1:T}) \propto \mu(x_1) \prod_{t=2}^{T} f_t(x_t|x_{1:t-1}) \prod_{t=1}^{T} g_t(y_t|x_{1:t}).$$

In general, computing the posterior $p(x_{1:T}|y_{1:T})$ is intractable and we have to resort to approximations. We will in this paper focus on, and extend, the family of particle Markov chain Monte Carlo algorithms originally proposed by Andrieu et al. (2010). The key idea in PMCMC is to use SMC to construct efficient proposals of the latent variables $x_{1:T}$ for an MCMC sampler.

Algorithm 1 Sequential Monte Carlo (all for $i = 1, \dots, N$)

```
Input: data $y_{1:T}$, number of particles $N$, proposals $q_t$
$x_1^i \sim q_1(x_1)$
$w_1^i = \frac{g_1(y_1|x_1^i)\mu(x_1^i)}{q_1(x_1^i)}$
for $t = 2$ to $T$ do
    $a_{t-1}^i \sim \text{Discrete}\left(\{\bar{w}_{t-1}^\ell\}_{\ell=1}^N\right)$
    $x_t^i \sim q_t(x_t|x_{1:t-1}^{a_{t-1}^i})$
    Set $x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i)$
    $w_t^i = \frac{g_t(y_t|x_{1:t}^i) f_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}{q_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}$
end for
```

2.1. Sequential Monte Carlo

The SMC method is a widely used technique for approximating a sequence of target distributions: in our case $p(x_{1:t}|y_{1:t}) = p(y_{1:t})^{-1} p(x_{1:t}, y_{1:t}),\ t = 1, \dots, T$. At each time step t we generate a *particle system* $\{(x_{1:t}^i, w_t^i)\}_{i=1}^N$ which provides a weighted approximation to $p(x_{1:t}|y_{1:t})$. Given such a weighted particle system at time $t-1$, this is propagated forward in time to t by first drawing an ancestor variable a_{t-1}^i for each particle from its corresponding distribution:

$$\mathbb{P}(a_{t-1}^i = \ell) = \bar{w}_{t-1}^\ell. \qquad \ell = 1, \dots, N, \qquad (2)$$

where $\bar{w}_{t-1}^\ell = w_{t-1}^\ell / \sum_i w_{t-1}^i$. This is commonly known as the resampling step in the literature. We introduce the ancestor variables $\{a_{t-1}^i\}_{i=1}^N$ explicitly to simplify the exposition of the theoretical justification given in Section 3.1.

We continue by simulating from some given proposal density $x_t^i \sim q_t(x_t|x_{1:t-1}^{a_{t-1}^i})$ and re-weight the system of particles as follows:

$$w_t^i = \frac{g_t(y_t|x_{1:t}^i) f_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}{q_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}, \qquad (3)$$

where $x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i)$. This results in a new particle system $\{(x_{1:t}^i, w_t^i)\}_{i=1}^N$ that approximates $p(x_{1:t}|y_{1:t})$. A summary is given in Algorithm 1.

2.2. Particle Gibbs

The PG algorithm (Andrieu et al., 2010) is a Gibbs sampler on the extended space composed of all random variables generated at one iteration, which still retains the original target distribution as a marginal. Though PG allows for inference over both latent variables and static parameters, we will in this paper focus on sampling of the former. The core idea of PG is to iteratively run *conditional* sequential Monte Carlo (CSMC) sweeps as shown in Algorithm 2,

[1] http://www.robots.ox.ac.uk/~fwood/anglican

[2] https://bitbucket.org/twgr/ipmcmc

Algorithm 2 Conditional sequential Monte Carlo

```
Input: data $y_{1:T}$, number of particles $N$, proposals $q_t$, conditional trajectory $x'_{1:T}$
$x_1^i \sim q_1(x_1),\ i = 1, \dots, N-1$ and set $x_1^N = x'_1$
$w_1^i = \frac{g_1(y_1|x_1^i)\mu(x_1^i)}{q_1(x_1^i)},\ i = 1, \dots, N$
for $t = 2$ to $T$ do
    $a_{t-1}^i \sim \text{Discrete}\left(\{\bar{w}_{t-1}^{\ell}\}_{\ell=1}^{N}\right),\ i = 1, \dots, N-1$
    $x_t^i \sim q_t(x_t|x_{1:t-1}^{a_{t-1}^i}),\ i = 1, \dots, N-1$
    Set $a_{t-1}^N = N$ and $x_t^N = x'_t$
    Set $x_{1:t}^i = (x_{1:t-1}^{a_{t-1}^i}, x_t^i),\ i = 1, \dots, N$
    $w_t^i = \frac{g_t(y_t|x_{1:t}^i) f_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})}{q_t(x_t^i|x_{1:t-1}^{a_{t-1}^i})},\ i = 1, \dots, N$
end for
```

whereby each conditional trajectory is sampled from the surviving trajectories of the previous sweep. This *retained particle* index, b, is sampled with probability proportional to the final particle weights $\bar{w}_T^i$.

3. Interacting Particle Markov Chain Monte Carlo

The main goal of iPMCMC is to increase the efficiency of PMCMC, in particular particle Gibbs. The basic PG algorithm is particularly susceptible to the *path degeneracy* effect of SMC samplers, i.e. sample impoverishment due to frequent resampling. Whenever the ancestral lineage collapses at the early stages of the state sequence, the common ancestor is, by construction, guaranteed to be equal to the retained particle. This results in high correlation between the samples, and poor mixing of the Markov chain. To counteract this we might need a very high number of particles to get good mixing for all latent variables $x_{1:T}$, which can be infeasible due to e.g. limited available memory. iPMCMC can alleviate this issue by, from time to time, switching out a CSMC particle system with a completely independent SMC one, resulting in improved mixing.

iPMCMC, summarized in Algorithm 3, consists of M interacting separate CSMC and SMC algorithms, exchanging only very limited information at each iteration to draw new MCMC samples. We will refer to these internal CSMC and SMC algorithms as nodes, and assign an index $m = 1, \dots, M$. At every iteration, we have P nodes running local CSMC algorithms, with the remaining $M - P$ nodes running independent SMC. The CSMC nodes are given an identifier $c_j \in \{1, \dots, M\},\ j = 1, \dots, P$ with $c_j \neq c_k,\ k \neq j$ and we write $c_{1:P} = \{c_1, \dots, c_P\}$. Let $\mathbf{x}_m^i = x_{1:T,m}^i$ be the internal particle trajectories of node m.

Suppose we have access to P trajectories $\mathbf{x}'_{1:P}[0] = (\mathbf{x}'_1[0], \dots, \mathbf{x}'_P[0])$ corresponding to the initial retained particles, where the index $[\cdot]$ denotes MCMC iteration. At each iteration r, the nodes $c_{1:P}$ run CSMC (Algorithm 2) with the previous MCMC sample $\mathbf{x}'_j[r-1]$ as the retained particle. The remaining $M - P$ nodes run standard (unconditional) SMC, i.e. Algorithm 1. Each node m returns an estimate of the marginal likelihood for the internal particle system defined as

$$\hat{Z}_m = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} w_{t,m}^i. \tag{4}$$

Algorithm 3 iPMCMC sampler

```
Input: number of nodes $M$, conditional nodes $P$ and MCMC steps $R$, initial $\mathbf{x}'_{1:P}[0]$
for $r = 1$ to $R$ do
    Workers $1 : M \backslash c_{1:P}$ run Algorithm 1 (SMC)
    Workers $c_{1:P}$ run Algorithm 2 (CSMC), conditional on $\mathbf{x}'_{1:P}[r-1]$ respectively.
    for $j = 1$ to $P$ do
        Select a new conditional node by simulating $c_j$ according to (5).
        Set new MCMC sample $\mathbf{x}'_j[r] = \mathbf{x}_{c_j}^{b_j}$ by simulating $b_j$ according to (7)
    end for
end for
```

The new conditional nodes are then set using a single loop $j = 1 : P$ of Gibbs updates, sampling new indices c_j where

$$\mathbb{P}(c_j = m | c_{1:P\backslash j}) = \hat{\zeta}_m^j \tag{5}$$

$$\text{and} \quad \hat{\zeta}_m^j = \frac{\hat{Z}_m \mathbb{1}_{m \notin c_{1:P\backslash j}}}{\sum_{n=1}^{M} \hat{Z}_n \mathbb{1}_{n \notin c_{1:P\backslash j}}}, \tag{6}$$

defining $c_{1:P\backslash j} = \{c_1, \dots, c_{j-1}, c_{j+1}, \dots, c_P\}$. We thus loop once through the conditional node indices and resample them from the union of the current node index and the unconditional node indices[3], in proportion to their marginal likelihood estimates. This is the key step that lets us switch completely the nodes from which the retained particles are drawn.

One MCMC iteration r is concluded by setting the new samples $\mathbf{x}'_{1:P}[r]$ by simulating from the corresponding conditional node's, c_j, internal particle system

$$\begin{aligned} \mathbb{P}(b_j = i | c_j) &= \bar{w}_{T,c_j}^i, \\ \mathbf{x}'_j[r] &= \mathbf{x}_{c_j}^{b_j}. \end{aligned} \tag{7}$$

The potential to pick from updated nodes c_j, having run independent SMC algorithms, decreases correlation and

[3] Unconditional node indices here refers to all $m \notin c_{1:P}$ at that point in the loop. It may thus include nodes who just ran a CSMC sweep, but have been "switched out" earlier in the loop.

improves mixing of the MCMC sampler. Furthermore, as each Gibbs update corresponds to a one-to-many comparison for maintaining the same conditional index, the probability of switching is much higher than in an analogous non-interacting system.

The theoretical justification for iPMCMC is independent of how the initial trajectories $\mathbf{x}'_{1:P}[0]$ are generated. One simple and effective method (that we use in our experiments) is to run standard SMC sweeps for the "conditional" nodes at the first iteration.

The iPMCMC samples $\mathbf{x}'_{1:P}[r]$ can be used to estimate expectations for test functions $f : \mathsf{X}^T \mapsto \mathbb{R}$ in the standard Monte Carlo sense, with

$$\mathbb{E}[f(\mathbf{x})] \approx \frac{1}{RP} \sum_{r=1}^{R} \sum_{j=1}^{P} f(\mathbf{x}'_j[r]). \tag{8}$$

However, we can improve upon this if we have access to all particles generated by the algorithm, see Section 3.2.

We note that iPMCMC is suited to distributed and multi-core architectures. In practise, the particle to be retained, should the node be a conditional node at the next iteration, can be sampled upfront and discarded if unused. Therefore, at each iteration, only a single particle trajectory and normalisation constant estimate need be communicated between the nodes, whilst the time taken for calculation of the updates of $c_{1:P}$ is negligible. Further, iPMCMC should be amenable to an asynchronous adaptation under the assumption of a random execution time, independent of $\mathbf{x}'_j[r-1]$ in Algorithm 3. We leave this asynchronous variant to future work.

3.1. Theoretical Justification

In this section we will give some crucial results to justify the proposed iPMCMC sampler. This section is due to space constraints fairly brief and it is helpful to be familiar with the proof of PG in Andrieu et al. (2010). We start by defining some additional notation. Let $\xi := \{x_t^i\}_{\substack{i=1:N \\ t=1:T}} \bigcup \{a_t^i\}_{\substack{i=1:N \\ t=1:T-1}}$ denote all generated particles and ancestor variables of a (C)SMC sampler. We write ξ_m when referring to the variables of the sampler local to node m. Let the conditional particle trajectory and corresponding ancestor variables for node c_j be denoted by $\{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\}$, with $\mathbf{b}_{c_j} = (b_{1,c_j}, \ldots, b_{T,c_j})$, $b_{T,c_j} = b_j$ and $b_{t,c_j} = a_{t,c_j}^{b_{t+1,c_j}}$. Let the posterior distribution of the latent variables be denoted by $\pi_T(\mathbf{x}) := p(x_{1:T}|y_{1:T})$ with normalisation constant $Z := p(y_{1:T})$. Finally we note that the SMC and CSMC algorithms induce the respective distributions over the random variables generated by the procedures:

$$q_{\text{SMC}}(\xi) = \prod_{i=1}^{N} q_1(x_1^i) \cdot \prod_{t=2}^{T} \prod_{i=1}^{N} \left[\bar{w}_{t-1}^{a_{t-1}^i} q_t(x_t^i | x_{1:t-1}^{a_{t-1}^i}) \right],$$

$$q_{\text{CSMC}}\left(\xi \backslash \{\mathbf{x}', \mathbf{b}\} \mid \mathbf{x}', \mathbf{b}\right) = \prod_{\substack{i=1 \\ i \neq b_1}}^{N} q_1(x_1^i) \cdot \prod_{t=2}^{T} \prod_{\substack{i=1 \\ i \neq b_t}}^{N} \left[\bar{w}_{t-1}^{a_{t-1}^i} q_t(x_t^i | x_{1:t-1}^{a_{t-1}^i}) \right].$$

Note that running Algorithm 2 corresponds to simulating from q_{CSMC} using a fixed choice for the index variables $\mathbf{b} = (N \ldots, N)$. While these indices are used to facilitate the proof of validity of the proposed method, they have no practical relevance and can thus be set to arbitrary values, as is done in Algorithm 2, in a practical implementation.

Now we are ready to state the main theoretical result.

Theorem 1. *The interacting particle Markov chain Monte Carlo sampler of Algorithm 3 is a partially collapsed Gibbs sampler (Van Dyk & Park, 2008) for the target distribution*

$$\tilde{\pi}(\xi_{1:M}, c_{1:P}, b_{1:P}) = \frac{1}{N^{PT}\binom{M}{P}} \prod_{\substack{m=1 \\ m \notin c_{1:P}}}^{M} q_{SMC}(\xi_m) \cdot \prod_{j=1}^{P} \pi_T\left(\mathbf{x}_{c_j}^{b_j}\right) \mathbb{1}_{c_j \notin c_{1:j-1}} \cdot \prod_{j=1}^{P} q_{CSMC}\left(\xi_{c_j} \backslash \{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\} \mid \mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\right). \tag{9}$$

Proof. See Appendix A at the end of the paper. □

Remark 1. The marginal distribution of $(\mathbf{x}_{c_{1:P}}^{b_{1:P}}, c_{1:P}, b_{1:P})$, with $\mathbf{x}_{c_{1:P}}^{b_{1:P}} = (\mathbf{x}_{c_1}^{b_1}, \ldots, \mathbf{x}_{c_P}^{b_P})$, under (9) is given by

$$\tilde{\pi}\left(\mathbf{x}_{c_{1:P}}^{b_{1:P}}, c_{1:P}, b_{1:P}\right) = \frac{\prod_{j=1}^{P} \pi_T\left(\mathbf{x}_{c_j}^{b_j}\right) \mathbb{1}_{c_j \notin c_{1:j-1}}}{N^{PT}\binom{M}{P}}. \tag{10}$$

This means that each trajectory $\mathbf{x}_{c_j}^{b_j}$ is marginally distributed according to the posterior distribution of interest, π_T. Indeed, the P retained trajectories of iPMCMC will in the limit $R \to \infty$ be independent draws from π_T.

Note that adding a backward or ancestor simulation step can drastically increase mixing when sampling the conditional trajectories $\mathbf{x}'_j[r]$ (Lindsten & Schön, 2013). In the iPMCMC sampler we can replace simulating from the final weights on line 7 by a backward simulation step. Another option for the CSMC nodes is to replace this step by internal ancestor sampling (Lindsten et al., 2014) steps and simulate from the final weights as normal.

3.2. Using All Particles

At each MCMC iteration r, we generate MN full particle trajectories. Using only P of these as in (8) might seem a bit wasteful. We can however make use of all particles to estimate expectations of interest by, for each Gibbs update j, averaging over the possible new values for the conditional node index c_j and corresponding particle index b_j. We can do this by replacing $f(\mathbf{x}'_j[r])$ in (8) by

$$\mathbb{E}_{c_j|c_{1:P\backslash j}}\left[\mathbb{E}_{b_{1:P}}\left[f(\mathbf{x}'_j[r])\right]\right] = \sum_{m=1}^{M} \hat{\zeta}_m^j \sum_{i=1}^{N} \bar{w}_{T,m}^i f(\mathbf{x}_m^i).$$

This procedure is referred to as a Rao-Blackwellization of a statistical estimator and is (in terms of variance) never worse than the original one. We highlight that each $\hat{\zeta}_m^j$, as defined in (6), depends on which indices are sampled earlier in the index reassignment loop. Further details, along with a derivation, are provided in the supplementary material.

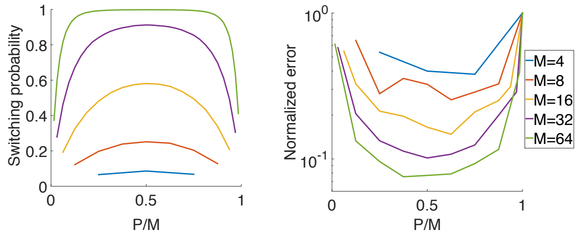
3.3. Choosing P

Before jumping into the full details of our experimentation, we quickly consider the choice of P. Intuitively we can think of the independent SMC's as particularly useful if they are selected as the next conditional node. The probability of the event that at least one conditional node switches with an unconditional, is given by

$$\mathbb{P}(\{\text{switch}\}) = 1 - \mathbb{E}\Big[\prod_{j=1}^{P} \frac{\hat{Z}_{c_j}}{\hat{Z}_{c_j} + \sum_{m \notin c_{1:P}}^{M} \hat{Z}_m}\Big]. \quad (11)$$

There exist theoretical and experimental results (Pitt et al., 2012; Bérard et al., 2014; Doucet et al., 2015) that show that the distributions of the normalisation constants are well-approximated by their log-Normal limiting distributions. Now, with σ^2 ($\propto \frac{1}{N}$) being the variance of the (C)SMC estimate, it means we have $\log\left(Z^{-1}\hat{Z}_{c_j}\right) \sim \mathcal{N}(\frac{\sigma^2}{2}, \sigma^2)$ and $\log\left(Z^{-1}\hat{Z}_m\right) \sim \mathcal{N}(-\frac{\sigma^2}{2}, \sigma^2)$, $m \notin c_{1:P}$ at stationarity, where Z is the true normalization constant. Under this assumption, we can accurately estimate the probability (11) for different choices of P an example of which is shown in Figure 1a along with additional analysis in the supplementary material. These provide strong empirical evidence that the switching probability is maximised for $P = M/2$.

In practice we also see that best results are achieved when P makes up roughly half of the nodes, see Figure 1b for performance on the state space model introduced in (12). Note also that the accuracy seems to be fairly robust with respect to the choice of P. Based on these results, we set the value of $P = \frac{M}{2}$ for the rest of our experiments.



(a) Limiting log-Normal (b) Gaussian state space model

Figure 1. a) Estimation of switching probability for different choices of P and M assuming the log-Normal limiting distribution for $\hat{Z}_m$ with $\sigma = 3$. b) Median error in mean estimate for different choices of P and M over 10 different synthetic datasets of the linear Gaussian state space model given in (12) after 1000 MCMC iterations. Here errors are normalized by the error of a multi-start PG sampler which is a special case of iPMCMC for which $P = M$ (see Section 4).

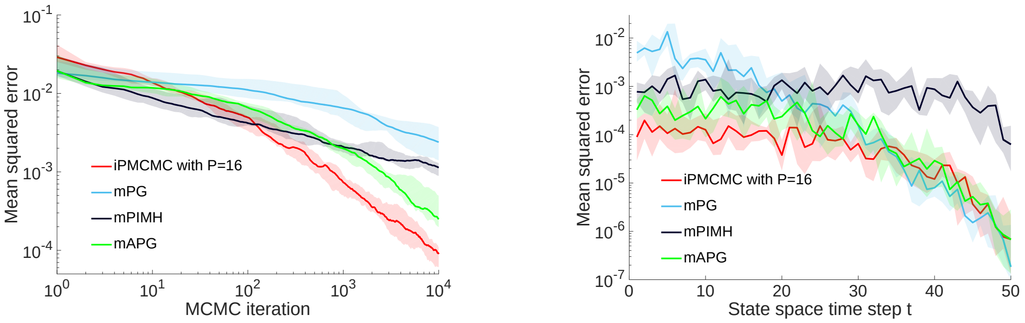
4. Experiments

To demonstrate the empirical performance of iPMCMC we report experiments on two state space models. Although both the models considered are Markovian, we emphasise that iPMCMC goes far beyond this and can be applied to arbitrary graphical models. We will focus our comparison on the trivially distributed alternatives, whereby M independent PMCMC samplers are run in parallel–these are PG, particle independent Metropolis-Hastings (PIMH) (Andrieu et al., 2010) and the alternate move PG sampler (APG) (Holenstein, 2009). Comparisons to other alternatives, including independent SMC, serialized implementations of PG and PIMH, and running a mixture of independent PG and PIMH samplers, are provided in the supplementary material. None outperformed the methods considered here, with the exception of running a serialized PG implementation with an increased number of particles, requiring significant additional memory ($O(MN)$ as opposed to $O(M + N)$).

In PIMH a new particle set is proposed at each MCMC step using an independent SMC sweep, which is then either accepted or rejected using the standard Metropolis-Hastings acceptance ratio. APG interleaves PG steps with PIMH steps in an attempt to overcome the issues caused by path degeneracy in PG. We refer to the trivially distributed versions of these algorithms as multi-start PG, PIMH and APG respectively (mPG, mPIMH and mAPG). We use Rao-Blackwellization, as described in 3.2, to average over all the generated particles for all methods, weighting the independent Markov chains equally for mPG, mPIMH and mAPG. We note that mPG is a special case of iPMCMC for which $P = M$. For simplicity, multinomial resampling was used in the experiments, with the prior transition distribution of the latent variables taken for the proposal. $M = 32$ nodes and $N = 100$ particles were used unless otherwise stated. Initialization of the retained particles for iPMCMC and mPG was done by using standard SMC sweeps.

4.1. Linear Gaussian State Space Model

We first consider a linear Gaussian state space model (LGSSM) with 3 dimensional latent states $x_{1:T}$, 20 dimen-



(a) Convergence in mean for full sequence

(b) Final error in mean for latent marginals

Figure 2. Mean squared error averaged over all dimensions and steps in the state sequence as a function of MCMC iterations (left) and mean squared error after 10^4 iterations averaged over dimensions as function of position in the state sequence (right) for (12) with 50 time sequences. The solid line shows the median error across the 10 tested synthetic datasets, while the shading shows the upper and lower quartiles. Ground truth was calculated using the Rauch–Tung–Striebel smoother algorithm (Rauch et al., 1965).

sional observations $y_{1:T}$ and dynamics given by

$$x_1 \sim \mathcal{N}(\mu, V) \tag{12a}$$
$$x_t = \alpha x_{t-1} + \delta_{t-1} \qquad \delta_{t-1} \sim \mathcal{N}(0, \Omega) \tag{12b}$$
$$y_t = \beta x_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \Sigma)\,. \tag{12c}$$

We set $\mu = [0, 1, 1]^T$, $V = 0.1$ $\mathbf{I}$, $\Omega = \mathbf{I}$ and $\Sigma = 0.1$ $\mathbf{I}$ where $\mathbf{I}$ represents the identity matrix. The constant transition matrix, α, corresponds to successively applying rotations of $\frac{7\pi}{10}$, $\frac{3\pi}{10}$ and $\frac{\pi}{20}$ about the first, second and third dimensions of x_{t-1} respectively followed by a scaling of 0.99 to ensure that the dynamics remain stable. A total of 10 different synthetic datasets of length $T = 50$ were generated by simulating from (12a)–(12c), each with a different emission matrix β generated by sampling each column independently from a symmetric Dirichlet distribution with concentration parameter 0.2.

Figure 2a shows convergence in the estimate of the latent variable means to the ground-truth solution for iPMCMC and the benchmark algorithms as a function of MCMC iterations. It shows that iPMCMC comfortably outperforms the alternatives from around 200 iterations onwards, with only iPMCMC and mAPG demonstrating behaviour consistent with the Monte Carlo convergence rate, suggesting that mPG and mPIMH are still far from the ergodic regime. Figure 2b shows the same errors after 10^4 MCMC iterations as a function of position in state sequence. This demonstrates that iPMCMC outperformed all the other algorithms for the early stages of the state sequence, for which mPG performed particularly poorly. Toward the end of state sequence, iPMCMC, mPG and mAPG all gave similar performance, whilst that of mPIMH was significantly worse.

4.2. Nonlinear State Space Model

We next consider the one dimensional nonlinear state space model (NLSSM) considered by, among others, Gordon et al. (1993); Andrieu et al. (2010)

$$x_1 \sim \mathcal{N}\left(\mu, v^2\right) \tag{13a}$$
$$x_t = \frac{x_{t-1}}{2} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t)\,\delta_{t-1} \tag{13b}$$
$$y_t = \frac{{x_t}^2}{20} + \varepsilon_t \tag{13c}$$

where $\delta_{t-1} \sim \mathcal{N}\left(0, \omega^2\right)$ and $\varepsilon_t \sim \mathcal{N}\left(0, \sigma^2\right)$. We set the parameters as $\mu = 0$, $v = \sqrt{5}$, $\omega = \sqrt{10}$ and $\sigma = \sqrt{10}$. Unlike the LGSSM, this model does not have an analytic solution and therefore one must resort to approximate inference methods. Further, the multi-modal nature of the latent space makes full posterior inference over $x_{1:T}$ challenging for long state sequences.

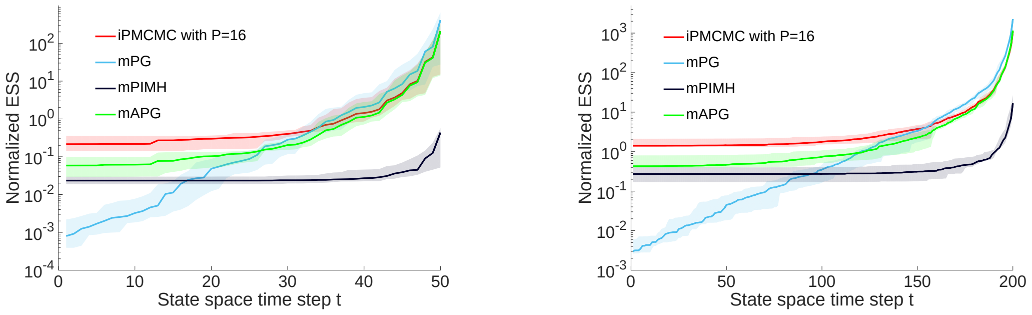
To examine the relative mixing of iPMCMC we calculate an effective sample size (ESS) for different steps in the state sequence. In order to calculate the ESS, we condensed identical samples as done in for example (van de Meent et al., 2015). Let

$$u_t^k \in \{x_{t,m}^i[r]\}_{m=1:M}^{i=1:N, r=1:R}, \quad \forall k \in 1 \ldots K,\ t \in 1 \ldots T$$

denote the unique samples of x_t generated by all the nodes and sweeps of particular algorithm after R iterations, where K is the total number of unique samples generated. The weight assigned to these unique samples, v_t^k, is given by the combined weights of all particles for which x_t takes the value u_t^k:

$$v_t^k = \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{i=1}^{N} \bar{w}_{t,m}^{i,r} \eta_m^r \delta_{x_{t,m}^i[r]}(u_t^k) \tag{14}$$

where $\delta_{x_{t,m}^i[r]}(u_t^k)$ is the Kronecker delta function and η_m^r is a node weight. For iPMCMC the node weight is given by as per the Rao-Blackwellized estimator described in



(a) LGSSM

(b) NLSSM

Figure 3. Normalized effective sample size (NESS) for LGSSM (left) and NLSSM (right).

Section 3.2. For mPG and mPIMH, η_m^r is simply $\frac{1}{RM}$, as samples from the different nodes are weighted equally in the absence of interaction. Finally we define the effective sample size as $\text{ESS}_t = \left(\sum_{k=1}^{K}\left(v_t^k\right)^2\right)^{-1}$.

Figure 3 shows the ESS for the LGSSM and NLSSM as a function of position in the state sequence. For this, we omit the samples generated by the initialization step as this SMC sweep is common to all the tested algorithms. We further normalize by the number of MCMC iterations so as to give an idea of the rate at which unique samples are generated. These show that for both models the ESS of iPMCMC, mPG and mAPG is similar towards the end of the space sequence, but that iPMCMC outperforms all the other methods at the early stages. The ESS of mPG was particularly poor at early iterations. PIMH performed poorly throughout, reflecting the very low observed acceptance ratio of around 7.3% on average.

It should be noted that the ESS is not a direct measure of performance for these models. For example, the equal weighting of nodes is likely to make the ESS artificially high for mPG, mPIMH and mAPG, when compared with methods such as iPMCMC that assign a weighting to the nodes at each iteration. To acknowledge this, we also plot histograms for the marginal distributions of a number of different position in the state sequence as shown in Figure 4. These confirm that iPMCMC and mPG have similar performance at the latter state sequence steps, whilst iPMCMC is superior at the earlier stages, with mPG producing almost no more new samples than those from the initialization sweep due to the degeneracy. The performance of PIMH was consistently worse than iPMCMC throughout the state sequence, with even the final step exhibiting noticeable noise.

5. Discussion and Future Work

The iPMCMC sampler overcomes degeneracy issues in PG by allowing the newly sampled particles from SMC nodes to replace the retained particles in CSMC nodes. Our experimental results demonstrate that, for the models considered, this switching in rate is far higher than the rate at which PG generates fully independent samples. Moreover, the results in Figure 1b suggest that the degree of improvement over an mPG sampler with the same total number of nodes increases with the total number of nodes in the pool.

The mAPG sampler performs an accept reject step that compares the marginal likelihood estimate of a single CSMC sweep to that of a single SMC sweep. In the iPMCMC sampler the CSMC estimate of the marginal likelihood is compared to a population sample of SMC estimates, resulting in a higher probability that at least one of the SMC nodes will become a CSMC node.

Since the original PMCMC paper in 2010 there have been several papers studying (Chopin & Singh, 2015; Lindsten et al., 2015) and improving upon the basic PG algorithm. Key contributions to combat the path degeneracy effect are backward simulation (Whiteley et al., 2010; Lindsten & Schön, 2013) and ancestor sampling (Lindsten et al., 2014). These can also be used to improve the iPMCMC method ever further.

A. Proof of Theorem 1

The proof follows similar ideas as Andrieu et al. (2010). We prove that the interacting particle Markov chain Monte Carlo sampler is in fact a standard partially collapsed Gibbs sampler (Van Dyk & Park, 2008) on an extended space $\Upsilon := \mathsf{X}^{\otimes MTN} \times [N]^{\otimes M(T-1)N} \times [M]^{\otimes P} \times [N]^{\otimes P}$.

Proof. Assume the setup of Section 3. With $\tilde{\pi}(\cdot)$ with as per (9), we will show that the Gibbs sampler on the extended space, Υ, defined as follows

$$\begin{aligned}&\xi_{1:M}\backslash\{\mathbf{x}_{c_{1:P}}^{b_{1:P}}, \mathbf{b}_{c_{1:P}}\}\\ &\quad\sim \tilde{\pi}(\;\cdot\;|\mathbf{x}_{c_{1:P}}^{b_{1:P}}, \mathbf{b}_{c_{1:P}}, c_{1:P}, b_{1:P}),\end{aligned} \tag{15a}$$

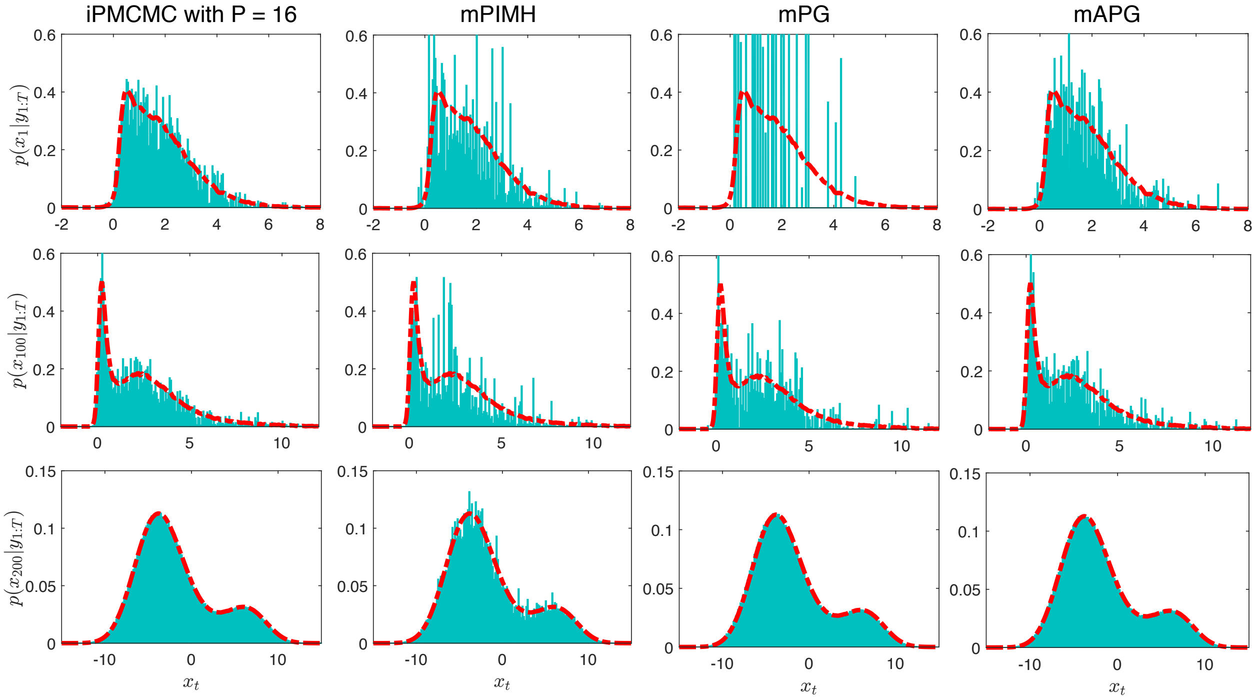


Figure 4. Histograms of generated samples at $t = 1, 100,$ and 200 for a single dataset generated from (13) with $T = 200$. Dashed red line shows an approximate estimate of the ground truth, found by running a kernel density estimator on the combined samples from a small number of independent SMC sweeps, each with 10^7 particles.

$$c_j \sim \tilde{\pi}(\ \cdot\ |\xi_{1:M}, c_{1:P\backslash j}), \quad j = 1, \dots, P, \qquad \text{(15b)}$$
$$b_j \sim \tilde{\pi}(\ \cdot\ |\xi_{1:M}, c_{1:P}), \quad j = 1, \dots, P, \qquad \text{(15c)}$$

is equivivalent to the iPMCMC method in Algorithm 3.

First, the initial step (15a) corresponds to sampling from

$$\tilde{\pi}(\xi_{1:M} \backslash \{\mathbf{x}_{c_{1:P}}^{b_{1:P}}, \mathbf{b}_{c_{1:P}}\} | \mathbf{x}_{c_{1:P}}^{b_{1:P}}, \mathbf{b}_{c_{1:P}}, c_{1:P}, b_{1:P}) = \prod_{\substack{m=1 \\ m \notin c_{1:P}}}^{M} q_{\text{SMC}}\left(\xi_m\right) \times \prod_{j=1}^{P} q_{\text{CSMC}}\left(\xi_{c_j} \backslash \{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\} \mid \mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}, c_j, b_j\right).$$

This, excluding the conditional trajectories, just corresponds to steps 3–4 in Algorithm 3, i.e. running P CSMC and $M - P$ SMC algorithms independently.

We continue with a reformulation of (9) which will be usefuly to prove correctness for the other two steps

$$\tilde{\pi}(\xi_{1:M}, c_{1:P}, b_{1:P}) = \frac{1}{\binom{M}{P}} \prod_{m=1}^{M} q_{\text{SMC}}\left(\xi_m\right) \times \prod_{j=1}^{P} \left[\mathbb{1}_{c_j \notin c_{1:j-1}} \bar{w}_{T,c_j}^{b_j} \pi_T\left(\mathbf{x}_{c_j}^{b_j}\right) \times \frac{q_{\text{CSMC}}\left(\xi_{c_j} \backslash \{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\} \mid \mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}, c_j, b_j\right)}{N^T \bar{w}_{T,c_j}^{b_j} q_{\text{SMC}}\left(\xi_{c_j}\right)} \right]$$
$$= \frac{1}{\binom{M}{P}} \prod_{m=1}^{M} q_{\text{SMC}}\left(\xi_m\right) \prod_{j=1}^{P} \frac{\hat{Z}_{c_j}}{Z} \mathbb{1}_{c_j \notin c_{1:j-1}} \bar{w}_{T,c_j}^{b_j}. \qquad (16)$$

Furthermore, we note that by marginalising (collapsing) the above reformulation, i.e. (16), over $b_{1:P}$ we get

$$\tilde{\pi}(\xi_{1:M}, c_{1:P}) = \frac{1}{\binom{M}{P}} \prod_{m=1}^{M} q_{\text{SMC}}\left(\xi_m\right) \prod_{j=1}^{P} \frac{\hat{Z}_{c_j}}{Z} \mathbb{1}_{c_j \notin c_{1:j-1}}.$$

From this it is easy to see that $\tilde{\pi}(c_j|\xi_{1:M}, c_{1:P\backslash j}) = \hat{\zeta}_{c_j}^{j}$, which corresponds to sampling the conditional node indices, i.e. step 6 in Algorithm 3. Finally, from (16) we can see that simulating $b_{1:P}$ can be done independently as follows

$$\tilde{\pi}(b_{1:P}|\xi_{1:M}, c_{1:P}) = \frac{\tilde{\pi}(b_{1:P}, \xi_{1:M}, c_{1:P})}{\tilde{\pi}(\xi_{1:M}, c_{1:P})} = \prod_{j=1}^{P} \bar{w}_{T,c_j}^{b_j}.$$

This corresponds to step 7 in the iPMCMC sampler, Algorithm 3. So the procedure defined by (15) is a partially collapsed Gibbs sampler, derived from (9), and we have shown that it is exactly equal to the iPMCMC sampler described in Algorithm 3. □

Acknowledgments

Tom Rainforth is supported by a BP industrial grant. Christian A. Naesseth is supported by CADICS, a Linnaeus Center, funded by the Swedish Research Council (VR). Fredrik Lindsten is supported by the project *Learning of complex dynamical systems* (Contract number: 637-2014-466) also funded by the Swedish Research Council. Frank Wood is supported under DARPA PPAML through the U.S. AFRL under Cooperative Agreement number FA8750-14-2-0006, Sub Award number 61160290-111668.

References

Andrieu, Christophe, Doucet, Arnaud, and Holenstein, Roman. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010. ISSN 1467-9868.

Bérard, Jean, Del Moral, Pierre, and Doucet, Arnaud. A lognormal central limit theorem for particle approximations of normalizing constants. *Electronic Journal of Probability*, 19(94):1–28, 2014.

Chopin, Nicolas and Singh, Sumeetpal S. On particle Gibbs sampling. *Bernoulli*, 21(3):1855–1883, 08 2015. doi: 10.3150/14-BEJ629.

Doucet, Arnaud, de Freitas, Nando, and Gordon, Neil. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2001.

Doucet, Arnaud, Pitt, Michael, Deligiannidis, George, and Kohn, Robert. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, pp. asu075, 2015.

Everitt, Richard G. Bayesian parameter estimation for latent Markov random fields and social networks. *Journal of Computational and Graphical Statistics*, 21(4):940–960, 2012.

Gordon, Neil J, Salmond, David J, and Smith, Adrian FM. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, 1993.

Holenstein, Roman. *Particle Markov chain Monte Carlo*. PhD thesis, The University Of British Columbia (Vancouver, 2009.

Huggins, Jonathan H. and Roy, Daniel M. Convergence of sequential Monte Carlo-based sampling methods. *ArXiv e-prints, arXiv:1503.00966v1*, March 2015.

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

Lindsten, Fredrik, Jordan, Michael I., and Schön, Thomas B. Particle Gibbs with ancestor sampling. *Journal of Machine Learning Research*, 15:2145–2184, june 2014.

Lindsten, Fredrik, Douc, Randal, and Moulines, Eric. Uniform ergodicity of the particle Gibbs sampler. *Scandinavian Journal of Statistics*, 42(3):775–797, 2015.

Naesseth, Christian A, Lindsten, Fredrik, and Schön, Thomas B. Sequential Monte Carlo for graphical models. In *Advances in Neural Information Processing Systems 27*, pp. 1862–1870. Curran Associates, Inc., 2014.

Naesseth, Christian A., Lindsten, Fredrik, and Schön, Thomas B. Nested sequential Monte Carlo methods. In *The 32nd International Conference on Machine Learning*, volume 37 of *JMLR W&CP*, pp. 1292–1301, Lille, France, jul 2015.

Pitt, Michael K, dos Santos Silva, Ralph, Giordani, Paolo, and Kohn, Robert. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.

Rauch, Herbert E, Striebel, CT, and Tung, F. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

Robert, Christian and Casella, George. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

Tripuraneni, Nilesh, Gu, Shixiang, Ge, Hong, and Ghahramani, Zoubin. Particle Gibbs for infinite hidden Markov Models. In *Advances in Neural Information Processing Systems 28*, pp. 2386–2394. Curran Associates, Inc., 2015.

Valera, Isabel, Francisco, Fran, Svensson, Lennart, and Perez-Cruz, Fernando. Infinite factorial dynamical model. In *Advances in Neural Information Processing Systems 28*, pp. 1657–1665. Curran Associates, Inc., 2015.

van de Meent, Jan-Willem, Yang, Hongseok, Mansinghka, Vikash, and Wood, Frank. Particle Gibbs with ancestor sampling for probabilistic programs. In *Proceedings of the 18th International conference on Artificial Intelligence and Statistics*, pp. 986–994, 2015.

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

Whiteley, Nick, Andrieu, Christophe, and Doucet, Arnaud. Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods. *ArXiv e-prints, arXiv:1011.2437*, 2010.

Whiteley, Nick, Lee, Anthony, and Heine, Kari. On the role of interaction in sequential Monte Carlo algorithms. *Bernoulli*, 22(1):494–529, 02 2016.

Wood, Frank, van de Meent, Jan Willem, and Mansinghka, Vikash. A new approach to probabilistic programming inference. In *Proceedings of the 17th International conference on Artificial Intelligence and Statistics*, pp. 2–46, 2014.