Technical report

Interacting Particle Markov Chain Monte Carlo

Tom Rainforth*, Christian A. Naesseth*, Fredrik Lindsten, Brooks Paige, Jan-Willem van de Meent, Arnaud Doucet and Frank Wood

* equal contribution

• **Please cite this version:**

Tom Rainforth*, Christian A. Naesseth*, Fredrik Lindsten, Brooks Paige, Jan-Willem van de Meent, Arnaud Doucet and Frank Wood. *Interacting Particle Markov Chain Monte Carlo*. arXiv, 2016. (* equal contribution)

Submitted to the International Conference on Machine Learning.

Abstract

We introduce *interacting particle Markov chain Monte Carlo* (iPMCMC), a PMCMC method that introduces a coupling between multiple 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 total computational budget. An additional advantage of the iPMCMC method is that it is suitable for distributed and multi-core architectures.

Keywords: sequential Monte Carlo, Markov chain Monte Carlo, particle Markov chain Monte Carlo, parallelisation

Interacting Particle Markov Chain Monte Carlo

Tom Rainforth* TWGR@ROBOTS.OX.AC.UK
Department of Engineering Science, University of Oxford

Christian A. Naesseth* CHRISTIAN.A.NAESSETH@LIU.SE
Department of Electrical Engineering, Linköping University

Fredrik Lindsten FREDRIK.LINDSTEN@IT.UU.SE
Department of Information Technology, Uppsala University

Brooks Paige BROOKS@ROBOTS.OX.AC.UK
Department of Engineering Science, University of Oxford

Jan-Willem van de Meent JWVDM@ROBOTS.OX.AC.UK
Department of Engineering Science, University of Oxford

Arnaud Doucet DOUCET@STATS.OX.AC.UK
Department of Statistics, University of Oxford

Frank Wood FWOOD@ROBOTS.OX.AC.UK
Department of Engineering Science, University of Oxford

* equal contribution

Abstract

We introduce *interacting particle Markov chain Monte Carlo* (iPMCMC), a PMCMC method that introduces a coupling between multiple 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 total computational budget. An additional advantage of the iPMCMC method is that it is suitable for distributed and multi-core architectures.

Keywords: sequential Monte Carlo, Markov chain Monte Carlo, particle Markov chain Monte Carlo, parallelisation

1. Introduction

MCMC methods are a fundamental tool for generating samples from a posterior density in Bayesian data analysis (see e.g., Robert and 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).

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. The interaction between algorithms 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 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 is an instance of the α-SMC-based (Whiteley et al., 2016) PMCMC method introduced and studied by Huggins and Roy (2015). However, iPMCMC allows for an arbitrary number of CSMC and standard SMC algorithms with interaction. Our experimental evaluation shows that iPMCMC outperforms both independent PG samplers and a single PG sampler with equivalent computational budget.

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

$$x_t | x_{1:t-1} \sim f_t(x_t | x_{1:t-1}), \tag{1a}$$
$$y_t | x_{1:t} \sim g_t(y_t | x_{1:t}), \tag{1b}$$

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 target distributions.

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

Input: data $y_{1:T}$, number of particles N, proposals q_t
$x_1^i \sim q_1(x_1)$
$w_1^i = \frac{g_1(y_t|x_1^i)\mu(x_1^i)}{q_1(x_1^i)}$
for $t = 2$ **to** T **do**
  $a_{t-1}^i \sim \text{Discrete}\left(\left\{\bar{w}_{t-1}^j\right\}_{j=1}^N\right)$
  $x_t^i \sim q_t(x_t|x_{1:t-1}^{a_{t-1}^i})$
  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 $p(x_{1:t}|y_{1:t}) = p(y_{1:t})^{-1}p(x_{1:t}, y_{1:t}),\ t = 1, \ldots, T$. 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:

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

where $\bar{w}_{t-1}^j = w_{t-1}^j / \sum_\ell w_{t-1}^\ell$. 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 algorithm and theoretical justification in the next section.

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:

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

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 particle Gibbs (PG) algorithm was first introduced by Andrieu et al. (2010) to construct, using SMC methods, an efficient Gibbs sampler for latent variable models with static parameters. Since then the PG algorithm and the extension by Lindsten et al. (2014) has found numerous applications in machine learning. We will in this paper focus on the problem of sampling the latent variables, which in PG is done using a so-called conditional sequential Monte Carlo (CSMC) algorithm. The name conditional comes from that we deterministically set one particle trajectory with corresponding

Algorithm 2 Conditional sequential Monte Carlo

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

ancestor indices to the retained MCMC sample, denoted by $x'_{1:T}$, from the previous step. The practical modification to Algorithm 1 is minimal, we simply deterministically set the N:th ancestor $a_{t-1}^N = N$ and latent variable $x_t^N = x'_t$. See Algorithm 2 for a summary of the CSMC algorithm.

3. Interacting Particle Markov Chain Monte Carlo

Our main goal with the iPMCMC method is to increase the efficiency of PMCMC, in particular particle Gibbs. The basic PG typically suffers from the *path degeneracy* effect of SMC samplers, i.e. sample impoverishment due to frequent resampling, which leads to bad mixing of the Markov chain. Since we force one trajectory, the conditional part, to survive to the end it means that for early time steps we will almost always pick the corresponding sample from last iteration. 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. The iPMCMC can alleviate this issue by, from time to time, completely switch out a CSMC particle system with a completely independent SMC one, resulting in improved mixing of the Markov chain.

The iPMCMC 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 workers or nodes, and assign an index $m = 1, \ldots, M$. At every iteration of the MCMC algorithm, we have P of the M nodes running local CSMC algorithms, with the remaining $M - P$ nodes running independent SMC. The CSMC nodes are given an identifier $c_j \in \{1, \ldots, M\},\ j = 1, \ldots, P$ with $c_j \neq c_i,\ i \neq j$ and we write $c_{1:P} = \{c_1, \ldots, c_P\}$. Furthermore, let $\mathbf{x}_m^i = x_{1:T,m}^i$ be the internal particle trajectories of node m.

We initialize the method with P trajectories $\mathbf{x}'_{1:P}[0] = (\mathbf{x}'_1[0], \ldots, \mathbf{x}'_P[0])$ which act as the initial retained particles, where the index $[\cdot]$ denotes MCMC iteration. Then, at each MCMC iteration r, the workers $c_{1:P}$ run CSMC (Algorithm 2) with the previous MCMC sample $\mathbf{x}'_j[r-1]$ as the retained particle. The remaining $M - P$ workers run standard (unconditional) SMC, i.e. Algorithm 1. This step, by far the most computationally demanding, can be trivially parallelised over the M workers. However, it need not be parallelised to see convergence benefits and reduced memory footprints compared to competing PMCMC methods.

Algorithm 3 iPMCMC sampler

Input: number of workers M, conditional workers 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 worker by simulating c_j according to (4).
    Set new MCMC sample $\mathbf{x}'_j[r] = \mathbf{x}_{c_j}^{b_j}$ by simulating b_j according to (5).
  end for
end for

The next step involves exchange of information—we need access to the normalisation constant estimates from each node to sample the next MCMC output $\mathbf{x}'_j[r]$. The normalisation constant estimate for each node m is computed using the internal particle system as $\hat{Z}_{\pi_T,m} = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} w_{t,m}^i$. These estimates are then used to set new conditional workers by simulating from

$$\mathbb{P}(c_j = m | c_{1:P\backslash j}) = \hat{\tilde{Z}}^j_{\pi_T,m}, \quad j = 1, \ldots, P \tag{4}$$

with

$$\hat{\tilde{Z}}^j_{\pi_T,m} = \frac{\hat{Z}_{\pi_T,m} \mathbb{1}_{m \notin c_{1:P\backslash j}}}{\sum_{n=1}^{M} \hat{Z}_{\pi_T,n} \mathbb{1}_{n \notin c_{1:P\backslash j}}},$$
$$c_{1:P\backslash j} = \{c_1, \ldots, c_{j-1}, c_{j+1}, \ldots, c_P\}.$$

This ensures that all the conditional node indices $c_{1:P}$ will be distinct.

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

$$\mathbb{P}(b_j = i | c_j) = \bar{w}^i_{T,c_j},$$
$$\mathbf{x}'_j[r] = \mathbf{x}_{c_j}^{b_j}. \tag{5}$$

A summary of the iPMCMC algorithm can be found in Algorithm 3 and a high-level overview of the algorithm flow can be found in Figure 1.

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

$$\mathbb{E}[f(\mathbf{x})] \approx \frac{1}{RP} \sum_{r=1}^{R} \sum_{j=1}^{P} f(\mathbf{x}'_j[r]). \tag{6}$$

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

3.1 Theoretical Justification

In this section we will give some crucial results to justify the proposed iPMCMC sampler. This section is fairly brief and it is helpful to be familiar with the proof of PG in Andrieu et al. (2010). We

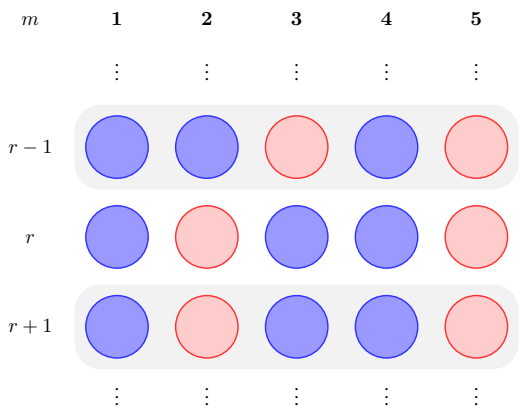


Figure 1: A high-level illustration of the iPMCMC algorithm flow for $M = 5$ and $P = 2$. CSMC nodes are *red* and SMC nodes are *blue*, which means that for example at iteration $r - 1$ we have $c_{1:P} = \{3, 5\}$.

start by defining some additional notation needed in the proof. 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 $\mathbf{b}_{c_j} = (b_{1,c_j}, \ldots, b_{T,c_j})$ and $b_{T,c_j} = b_j,\ b_{t,c_j} = a_{t,c_j}^{b_{t+1,c_j}}$. Note that $\mathbf{b}_{c_j}$ is thus defined recursively by tracing the ancestor dependence from particle b_j in node c_j, i.e. $\mathbf{x}_{c_j}^{b_j} = (x_{1,c_j}^{b_{1,c_j}}, \ldots, x_{T,c_j}^{b_{T,c_j}})$. This is in fact corresponds to the particle trajectory we condition on in the CSMC sampler of node c_j. Furthermore, let the posterior distribution of the latent variables be denoted by $\pi_T(\mathbf{x}) := p(x_{1:T}|y_{1:T})$. Finally we note that the SMC and CSMC algorithms induce the following distributions over the random variables generated by the procedures (respectively):

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

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

(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 and Park, 2008) for the target distribution*

$$\tilde{\pi}(\xi_{1:M}, c_{1:P}, b_{1:P}) = \frac{1}{N^{PT}\binom{M}{P}} \prod_{\substack{m=1 \\ m \notin c_{1:P}}}^{M} q_{SMC}(\xi_m) \cdot \prod_{j=1}^{P} \left[\pi_T\left(\mathbf{x}_{c_j}^{b_j}\right) \mathbb{1}_{c_j \notin c_{1:j-1}} q_{CSMC}\left(\xi_{c_j} \backslash \{\mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\} \mid \mathbf{x}_{c_j}^{b_j}, \mathbf{b}_{c_j}\right)\right]. \quad (7)$$

Proof See Appendix A. ■

Remark 2 *Note that 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* (7) *is given by*

$$\tilde{\pi}\left(\mathbf{x}_{c_{1:P}}^{b_{1:P}}, c_{1:P}, b_{1:P}\right) = \frac{\prod_{j=1}^{P} \pi_T\left(\mathbf{x}_{c_j}^{b_j}\right) \mathbb{1}_{c_j \notin c_{1:j-1}}}{N^{PT}\binom{M}{P}}. \quad (8)$$

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 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 and 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

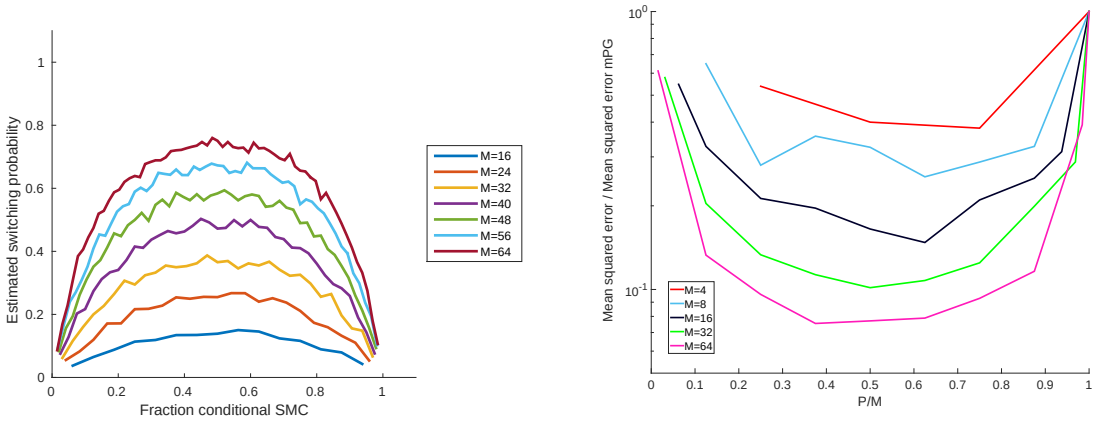
At each run of iPMCMC we generate MN full particle trajectories. Using only P of these as in (6) might seem a bit wasteful. We can however make use of all particles to estimate expectations of interest by, for each MCMC iteration r, averaging over the sampled conditional node indices $c_{1:P}$ and corresponding particle indices $b_{1:P}$. We can do this by replacing $f(\mathbf{x}'_j[r])$ in (6) by

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

calculated for each iteration r from the workers' internal particle systems. 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.

3.3 Choosing P

Before jumping into the full details of our experimentation, we consider the choice of P. Intuitively we can think of the independent SMC's as particularly useful if they are selected as the next



(a) (b)

Figure 2: Results on choosing P. *Left:* Estimation of switching probability for various settings M with $\sigma = 4$. *Right:* 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 (10) after 1000 MCMC iterations. 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).

conditional node. The probability of the event that at least one conditional node switches with an unconditional, is given by

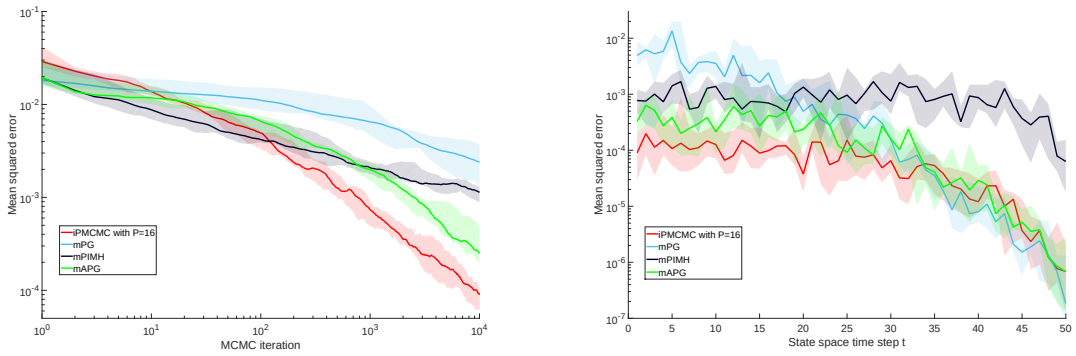
$$\mathbb{P}(\{\text{switch}\}) = 1 - \mathbb{E}\Big[\prod_{j=1}^{P} \frac{\hat{Z}_{\pi_T,c_j}}{\hat{Z}_{\pi_T,c_j} + \sum_{m \notin c_{1:P}}^{M} \hat{Z}_{\pi_T,m}}\Big]. \tag{9}$$

Now, 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. That means we have, with σ^2 being the variance of the (C)SMC estimate, $\log\left(Z_{\pi_T}^{-1}\hat{Z}_{\pi_T,c_j}\right) \sim \mathcal{N}(\frac{\sigma^2}{2}, \sigma^2)$ and $\log\left(Z_{\pi_T}^{-1}\hat{Z}_{\pi_T,m}\right) \sim \mathcal{N}(-\frac{\sigma^2}{2}, \sigma^2)$, $m \notin c_{1:P}$ at stationarity. Under this assumption we can accurately estimate the probability (17) for different choices of P an example of which is shown in Figure 2a along with additional analysis in the appendix. 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 2b for performance on the space model introduced in (10). 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.

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 the applicability of iPMCMC goes far beyond this and can be applied to arbitrary graphical models. We will focus our



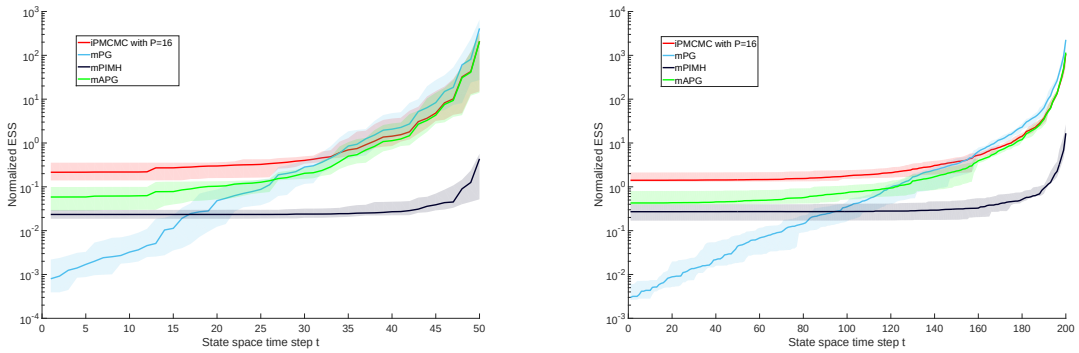
(a) Convergence in mean for full sequence

(b) Final error in mean for latent marginals

Figure 3: 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 the LGSSM given in (10) 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).

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 appendix, none of which outperformed the methods considered here (with the exception of running serialized implementations with an increased number of particles at significant extra computational cost).

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 particle 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.



(a) LGSSM (b) NLSSM

Figure 4: Normalized effective sample size (NESS) for LGSSM (left) and NLSSM (right).

4.1 Linear Gaussian State Space Model

We first consider a linear Gaussian state space model (LGSSM) with 3 dimensional latent states x_t, 20 dimensional observations y_t and dynamics given by

$$x_1 \sim \mathcal{N}(\mu, V) \tag{10a}$$
$$x_t = \alpha x_{t-1} + \delta_{t-1} \qquad \delta_{t-1} \sim \mathcal{N}(0, \Omega) \tag{10b}$$
$$y_t = \beta x_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \Sigma). \tag{10c}$$

We set $\mu = [0, 1, 1]^T$, $V = 0.1\ \mathbf{I}$, $\Omega = \mathbf{I}_3$ 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 (10a)–(10c), each with a different emission matrix β generated by sampling each column independently from a symmetric Dirichlet distribution with concentration parameter 0.2.

Figure 3a 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 number 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 3b 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)

$$x_1 \sim \mathcal{N}\left(\mu, v^2\right) \tag{11a}$$

$$x_t = \frac{x_{t-1}}{2} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos\left(1.2t\right)\delta_{t-1} \tag{11b}$$

$$y_t = \frac{{y_t}^2}{20} + \varepsilon_t \tag{11c}$$

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 set of 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:

$$v_t^k(u_t^k) = \sum_{r=1}^{R}\sum_{m=1}^{M}\sum_{i=1}^{N} \bar{w}_{t,m}^{i,r}\eta_m^r \delta_{x_{t,m}^i[r]}(u_t^k) \tag{12}$$

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

$$\eta_m^r = \frac{1}{RP}\sum_{j=1}^{P}\frac{\hat{Z}_{\pi_T,m}^r \mathbb{1}_{m\notin c_{1:P\setminus j}^r}}{\sum_n \hat{Z}_{\pi_T,n}^r \mathbb{1}_{n\notin c_{1:P\setminus j}^r}} \tag{13}$$

as per the Rao-Blackwellized estimator described in 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

$$\text{ESS}_t = \left(\sum_{k=1}^{K}\left(v_t^k\right)^2\right)^{-1}. \tag{14}$$

Figure 4 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

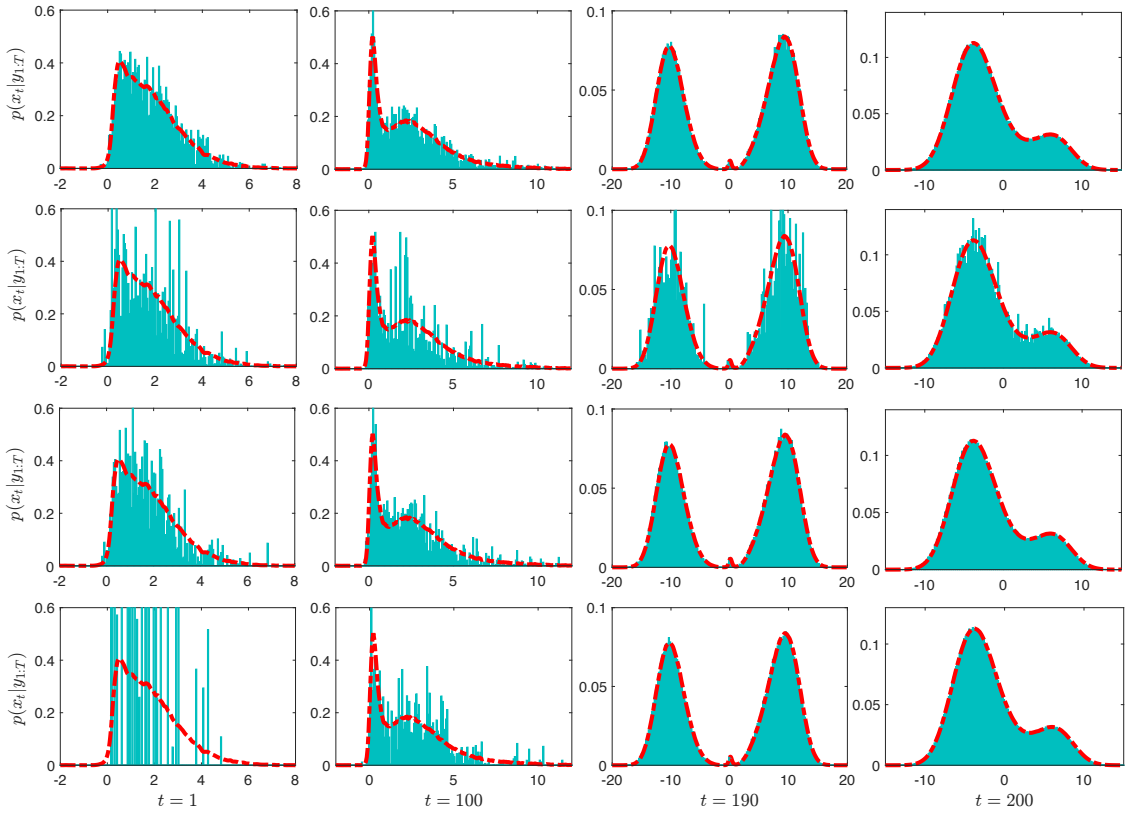


Figure 5: Histograms at $t = 1, 100, 190,$ and 200 for a single dataset generated from (11) with $T = 200$. Shown top to bottom is iPMCMC with $P = 16$, mPIMH, mAPG and mPG. Also show by the dashed red line is an approximate estimate of the ground truth, found by running a kernel density estimator on the combined samples from a small number of independent SMC sweeps, each with 10^7 particles.

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. The lack of Monte Carlo convergence rate appearing in Figure 3a also suggests this acceptance ratio is yet to converge, with the value in the ergodic regime likely to be even lower.

It should be noted that the ESS is not 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 5. These confirm that iPMCMC and mPG have similar performance at the latter state sequence steps, whilst iPMCMC is highly superior at the earlier stages, with mPG producing almost no unique particles, other than those from the initialization sweep. 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 PG algorithm is known to suffer from the degeneracy problem of SMC methods, leading to slow mixing and thus an inefficient MCMC sampler. The iPMCMC sampler overcomes these degeneracy issues by allowing the newly sampled particles from SMC nodes to replace the retained particles in CSMC nodes. Moreover, the results in Figure 2b suggest that the degree of improvement over the trivially parallelised PG sampler with the same total number of nodes in fact increases with the total number of nodes in the pool.

The mAPG sampler on the other hand performs an accept reject step that compares the marginal likelihood estimate of a single CSMC sweep to that of a single SMC sweep. This, as we saw in the experiments section, can lead to better mixing of the Markov chains. 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. This increased probability of switching leads to an even more efficient MCMC sampler that mixes significantly faster.

Since the original PMCMC paper in 2010 there have been several papers studying (Chopin and 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 and Schön, 2013) and ancestor sampling (Lindsten et al., 2014). These can also be used to improve the iPMCMC method ever further.

Finally, we remark that the Gibbs update in (4) of the conditional node indices requires no interaction between the CSMC nodes. This should make iPMCMC amenable to an asynchronous adaptation if we assume that we have a random execution time, independent of $\mathbf{x}'$, in Algorithm 2.

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). 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.

Appendix 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 and 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. We will show that the Gibbs sampler on the extended space, Υ with $\tilde{\pi}(\cdot)$ defined in (7), defined as follows

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

$$c_j \sim \tilde{\pi}(\ \cdot\ | \xi_{1:M}, c_{1:P\backslash j}),\ \ j = 1, \ldots, P, \tag{15b}$$

$$b_j \sim \tilde{\pi}(\ \cdot\ | \xi_{1:M}, c_{1:P}),\ \ j = 1, \ldots, P, \tag{15c}$$

is equivivalent to the iPMCMC method in Algorithm 3.

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

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

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 (7) which will be usefuly to prove correctness for the other two steps

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

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

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

From this it is easy to see that $\tilde{\pi}(c_j|\xi_{1:M}, c_{1:P\backslash j}) = \hat{\tilde{Z}}_{\pi_T,c_j}^{j}$, 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 (7), and we have shown that it is exactly equal to the iPMCMC sampler described in Algorithm 3. ■

Appendix B. Choosing P

For the purposes of this study we assume, without loss of generality, that the indices for the conditional nodes are always $c_{1:P} = \{1, \ldots, P\}$. Then we can show that the probability of the event that at least

one conditional nodes switches with an unconditional is given by

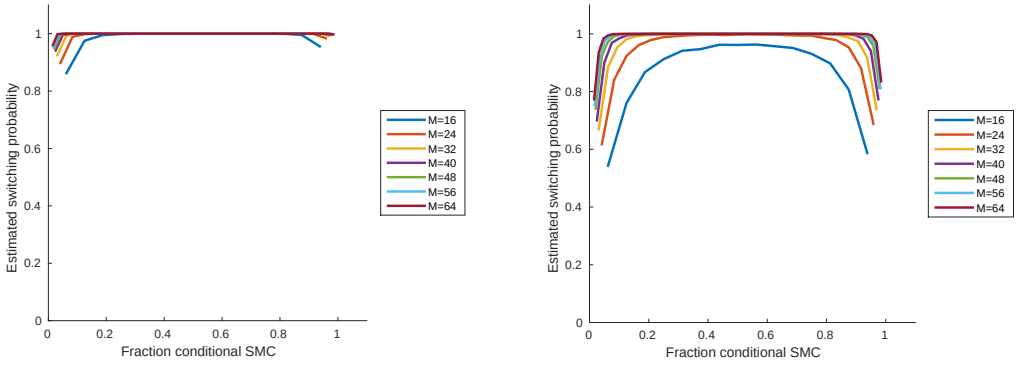
$$\mathbb{P}(\{\text{switch}\}) = 1 - \mathbb{E}\left[\prod_{j=1}^{P} \frac{\hat{Z}_{\pi_T,j}}{\hat{Z}_{\pi_T,j} + \sum_{m=P+1}^{M} \hat{Z}_{\pi_T,m}}\right]. \tag{17}$$

Now, there are some asymptotic (and experimental) results (Pitt et al., 2012; Bérard et al., 2014; Doucet et al., 2015) that indicate that a decent approximation for the distribution of the log of the normalisation constant estimates is Gaussian. This would mean the distributions of the conditional and unconditional normalisation constant estimates with variance σ^2 can be well-approximated as follows

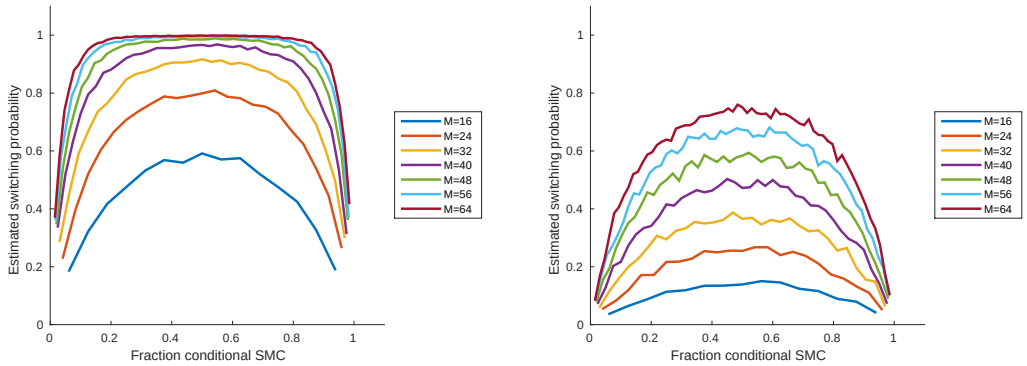
$$\log\left(\frac{\hat{Z}_{\pi_T,j}}{Z_{\pi_T}}\right) \sim \mathcal{N}(\frac{\sigma^2}{2}, \sigma^2), \quad j = 1, \ldots, P, \tag{18}$$

$$\log\left(\frac{\hat{Z}_{\pi_T,m}}{Z_{\pi_T}}\right) \sim \mathcal{N}(-\frac{\sigma^2}{2}, \sigma^2), \quad m = P+1, \ldots, M. \tag{19}$$

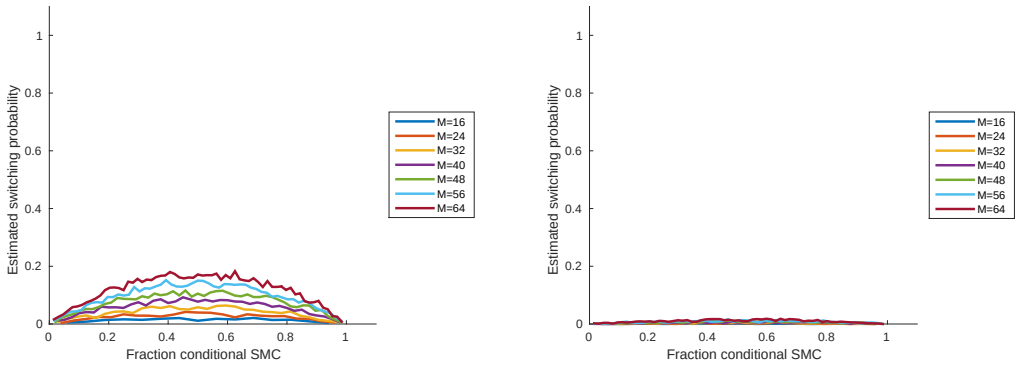
A straight-forward Monte Carlo estimation of the switching probability, i.e. $\mathbb{P}(\{\text{switch}\})$, can be seen in Figure 6 for various settings of σ and M. These results seem to indicate that letting $P \approx M/2$ maximises the probability of switching.



(a) $\sigma = 1$ (b) $\sigma = 2$



(c) $\sigma = 3$ (d) $\sigma = 4$

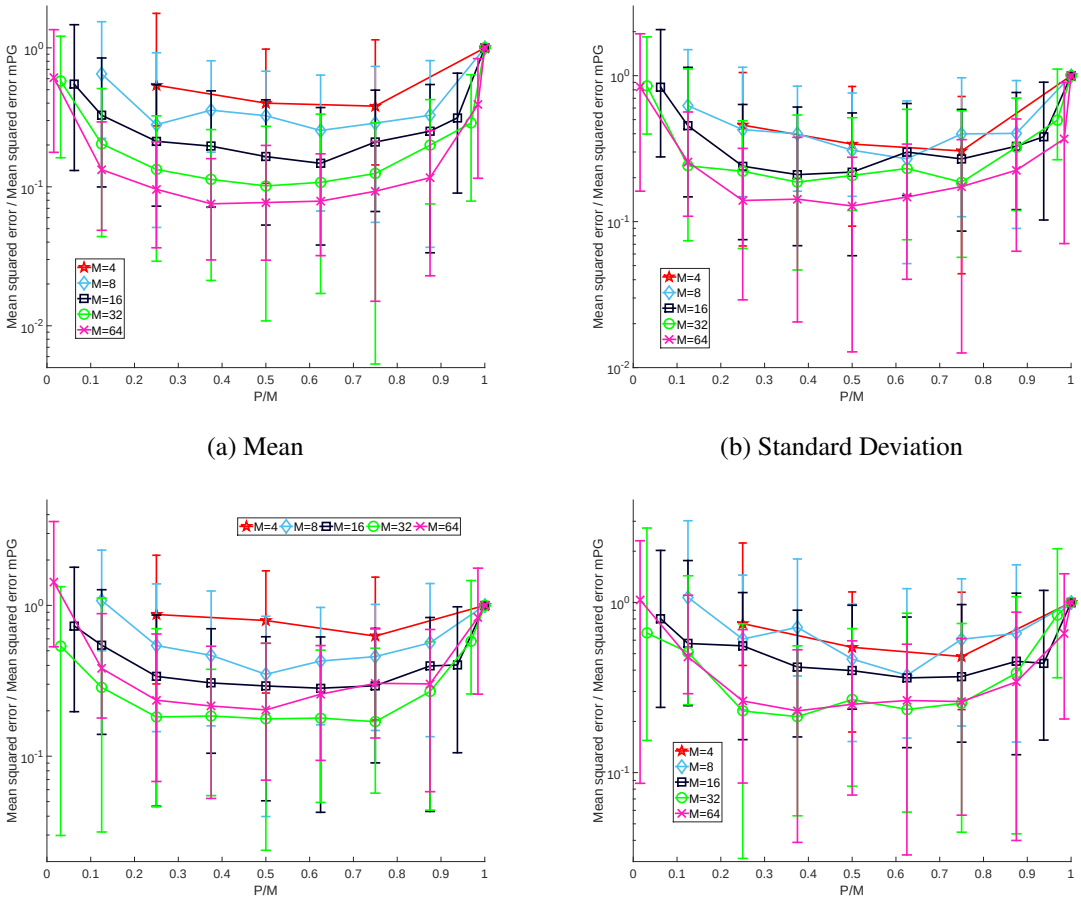


(e) $\sigma = 5$ (f) $\sigma = 6$

Figure 6: Estimation of switching probability for various settings of σ and M.

Appendix C. Additional Results Figures

In this section we provide additional results figures to complement those in the main body of the paper. These include convergence of additional metrics and comparisons to additional algorithms including Rao-Blackwellized independent SMC runs and serialized versions of PG and PIMH.



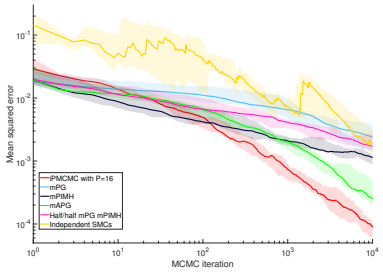
(a) Mean

(b) Standard Deviation

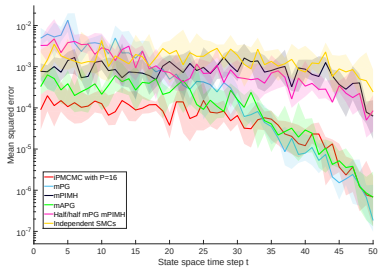
(c) Skewness

(d) Kurtosis

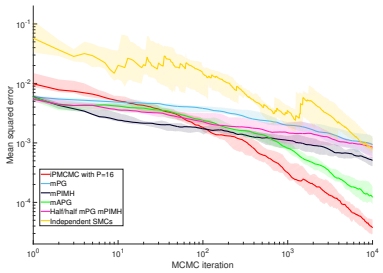
Figure 7: Median error in marginal moment estimates with different choices of P and M over 10 different synthetic datasets of the linear Gaussian state space model given in (10) after 1000 MCMC iterations. Errors are normalized by the error of a multi-start PG sampler which is a special case of iPMCMC for which $P = M$ (see Section 4). Error bars show the lower and upper quartiles for the errors. It can be seen that for all the moments then $P/M \approx 1/2$ give the best performance. For the mean and standard deviation estimates then the accuracy relative to the trivial distribution case $P = M$ shows a clear increase with M. This effect is also seen for the skewness and excess kurtosis estimates except for the distinction between the $M = 32$ and $M = 64$ cases. This could be because theses metrics have the same value under the prior and posterior.



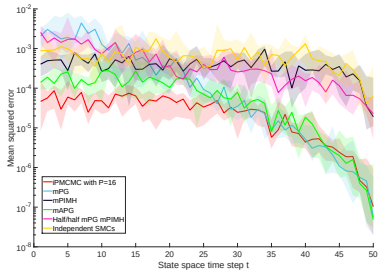
(a) Convergence in mean



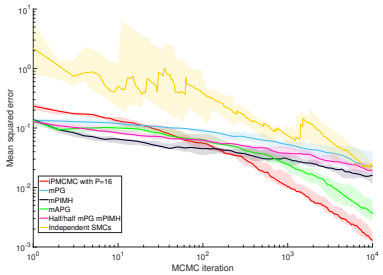
(b) Final error in mean



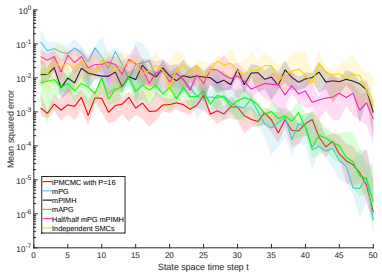
(c) Convergence in standard deviation



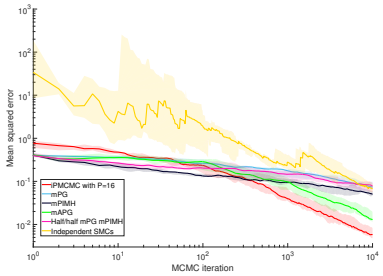
(d) Final error in standard deviation



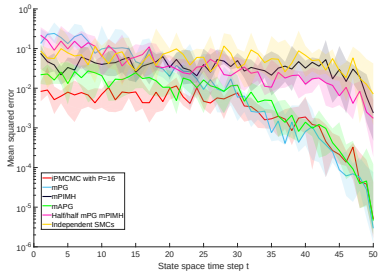
(e) Convergence in skewness



(f) Final error in skewness

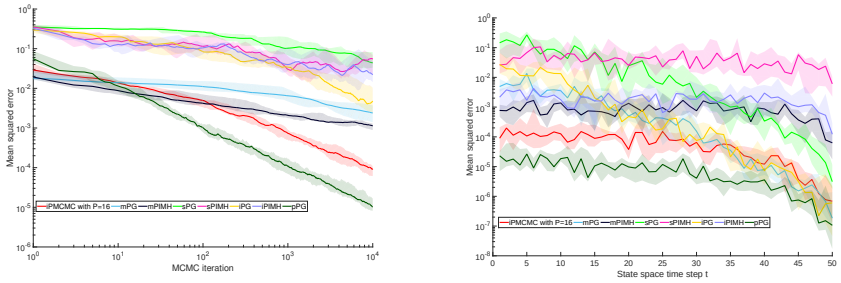


(g) Convergence in kurtosis



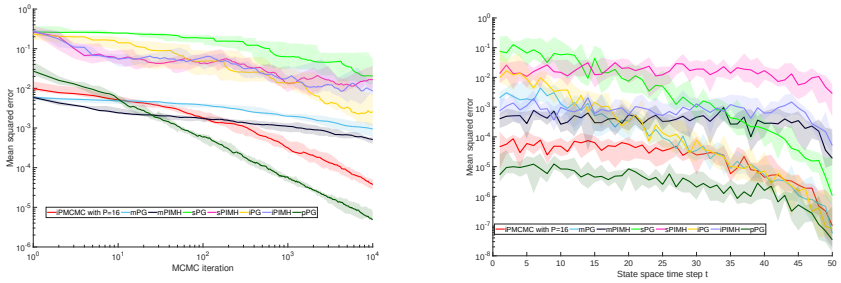
(h) Final error in kurtosis

Figure 8: Mean squared error in latent variable mean, standard deviation, skewness and kurtosis averaged of all dimensions of the LGSSM as a function of MCMC iteration (left) and position in the state sequence (right) for a selection of paraellelizable SMC and PMCMC methods. See Figure 3 for more details.



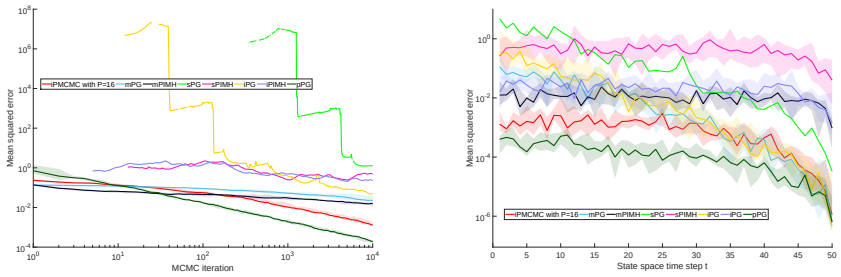
(a) Convergence in mean

(b) Final error in mean



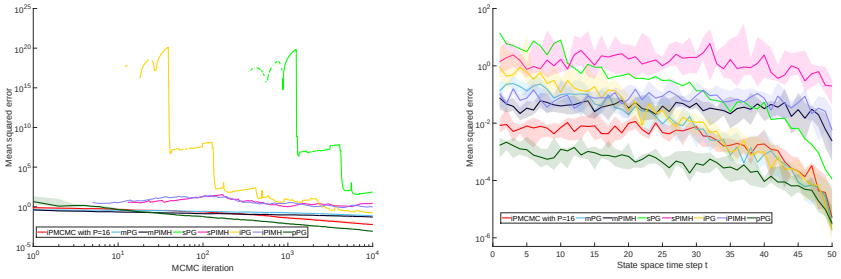
(c) Convergence in standard deviation

(d) Final error in standard deviation



(e) Convergence in skewness

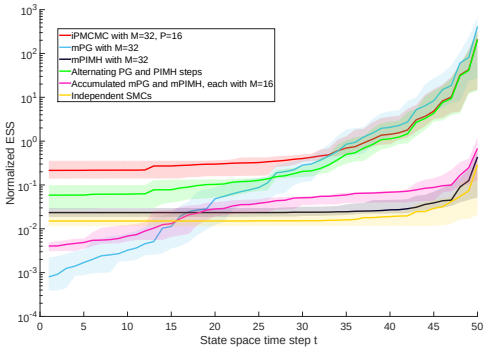
(f) Final error in skewness



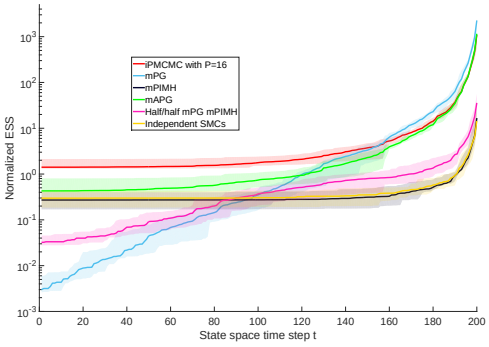
(g) Convergence in kurtosis

(h) Final error in kurtosis

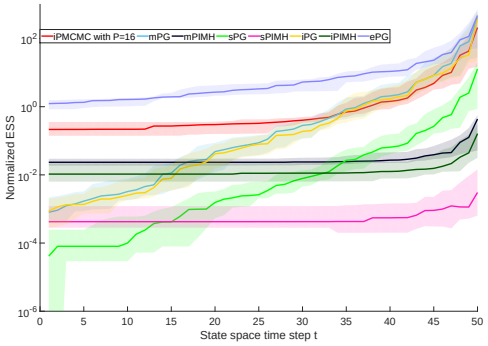
Figure 9: Mean squared error in latent variable mean, standard deviation, skewness and kurtosis averaged of all dimensions of the LGSSM as a function of MCMC iteration (left) and position in the state sequence (right) for iPMCMC, mPG, mPIMH and a number of serialized variants. Key for legends: sPG = single PG chain, sPIMH = single PIMH chain, iPG = single PG chain run 32 times longer, iPIMH = single PIMH chain run 32 times longer and pPG = single PG with 32 times more particles. For visualization purposes, the chains with extra iterations have had the number of MCMC iterations normalized by 32 such that the x axis values represent an equivalent total computational budget to the distributed variants.



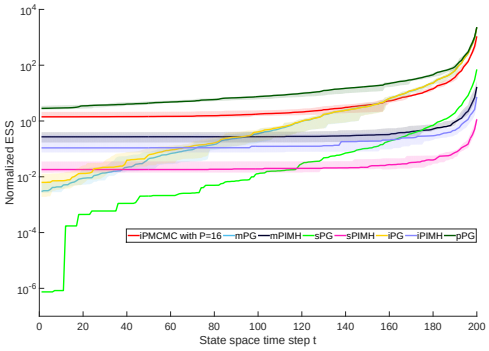
(a) ESS of distributed methods for LGSSM



(b) ESS of distributed methods for NLSSM



(c) ESS comparison to series equivalents for LGSSM



(d) ESS comparison to series equivalents for NLSSM

Figure 10: Normalized effective sample size for LGSSM (left) and NLSSM (right) for a number of distributed and series models. Key for legends as per Figure 9 with the x-axis values again scaled by 1/32 for the serialized variants run for additional MCMC iterations.

References

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

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

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

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

Arnaud Doucet, MK Pitt, George Deligiannidis, and Robert Kohn. Efficient implementation of markov chain monte carlo when using an unbiased likelihood estimator. *Biometrika*, page asu075, 2015.

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

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

Roman Holenstein. *Particle markov chain monte carlo*. PhD thesis, The University Of British Columbia (Vancouver, 2009.

J. H. Huggins and D. M. Roy. Convergence of Sequential Monte Carlo-based Sampling Methods. *ArXiv e-prints*, March 2015.

F. Lindsten, M. I. Jordan, and T. B. Schön. Particle Gibbs with Ancestor Sampling. *Journal of Machine Learning Research*, 15:2145–2184, june 2014.

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

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

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

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

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

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

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

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

Isabel Valera, Fran Francisco, Lennart Svensson, and Fernando Perez-Cruz. Infinite Factorial Dynamical Model. In *Advances in Neural Information Processing Systems 28*, pages 1657–1665. Curran Associates, Inc., 2015.

Jan-Willem van de Meent, Hongseok Yang, Vikash Mansinghka, and Frank Wood. Particle Gibbs with Ancestor Sampling for Probabilistic Programs. In *Proceedings of the 18th International conference on Artificial Intelligence and Statistics*, pages 986–994, 2015.

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

Nick Whiteley, Christophe Andrieu, and Arnaud Doucet. Efficient bayesian inference for switching state-space models using discrete particle markov chain monte carlo methods. *arXiv preprint arXiv:1011.2437*, 2010.

Nick Whiteley, Anthony Lee, and Kari Heine. On the role of interaction in sequential monte carlo algorithms. *Bernoulli*, 22(1):494–529, 02 2016. doi: 10.3150/14-BEJ666. URL `http://dx.doi.org/10.3150/14-BEJ666`.

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