Augmentation Schemes for Particle MCMC

Paul Fearnhead[1*], Loukia Meligkotsidou[2].

1. Department of Mathematics and Statistics, Lancaster University.
2. Department of Mathematics, University of Athens.

** Correspondence should be addressed to Paul Fearnhead. (e-mail: p.fearnhead@lancaster.ac.uk).*

Abstract

Particle MCMC involves using a particle filter within an MCMC algorithm. For inference of a model which involves an unobserved stochastic process, the standard implementation uses the particle filter to propose new values for the stochastic process, and MCMC moves to propose new values for the parameters. We show how particle MCMC can be generalised beyond this. Our key idea is to introduce new latent variables. We then use the MCMC moves to update the latent variables, and the particle filter to propose new values for the parameters and stochastic process given the latent variables. A generic way of defining these latent variables is to model them as pseudo-observations of the parameters or of the stochastic process. By choosing the amount of information these latent variables have about the parameters and the stochastic process we can often improve the mixing of the particle MCMC algorithm by trading off the Monte Carlo error of the particle filter and the mixing of the MCMC moves. We show that using pseudo-observations within particle MCMC can improve its efficiency in certain scenarios: dealing with initialisation problems of the particle filter; speeding up the mixing of particle Gibbs when there is strong dependence between the parameters and the stochastic process; and enabling further MCMC steps to be used within the particle filter.

This is the author-accepted version. The final publication is available at link.springer.com

Keywords: Dirichlet process mixture models, Particle Gibbs, Sequential Monte Carlo, State-space models, Stochastic volatility.

1 Introduction

Particle MCMC (Andrieu *et al.*, 2010) is a recent extension of MCMC. It is most naturally applied to inference for models, such as state-space models, where there is an unobserved stochastic process. Standard MCMC algorithms, such as Gibbs samplers, can often struggle with such models due to strong dependence between the unobserved process and the parameters (see e.g. Pitt and Shephard, 1999; Fearnhead, 2011). Alternative Monte Carlo methods, called particle filters, can be more efficient for inference about the unobserved process given known parameter values, but struggle when dealing with unknown parameters. The idea of particle MCMC is to embed a particle filter within an MCMC algorithm. The particle filter will then update the unobserved process given a specific value for the parameters, and MCMC moves will be used to update the parameter values. Particle MCMC has already been applied widely: in areas such as econometrics (Pitt *et al.*, 2012), inference for epidemics (Rasmussen *et al.*, 2011), and probabilistic programming (Wood *et al.*, 2014). For recent results that demonstrate the good theoretical properties of particle MCMC see Chopin and Singh (2013) and Del Moral *et al.* (2014).

The standard implementation of particle MCMC is to use an MCMC move to update the parameters and a particle filter to update the unobserved stochastic process (though see Murray *et al.*, 2012; Wood *et al.*, 2014, for alternatives). However, this may be inefficient, due to a large Monte Carlo error in the particle filter, or due to slow mixing of the MCMC moves. The idea of this paper is to consider generalisations of this standard implementation, which can lead to more efficient particle MCMC algorithms.

In particular, we suggest a data augmentation approach, where we introduce new latent variables into the model. We then implement particle MCMC on the joint posterior distribution of the parameters, unobserved stochastic process and latent variables. We use MCMC to update the latent variables and a particle filter to update the parameters and the stochastic process. The intuition behind this approach is that the latent variables can be viewed as containing information about the parameters and the stochastic process. The more information they contain, the lower the Monte Carlo error of the particle filter. However, the more information they contain, the stronger the dependencies in the posterior distribution, and hence the poorer the MCMC moves will mix. Thus, by carefully choosing our latent variables, we are able to appropriately trade-off the error in the particle filter against the mixing of the MCMC, so as to improve the efficiency of the particle MCMC algorithm.

The substantial interest in particle MCMC algorithms have led to a number of recent methodological developments. Examples include the use of backward simulation within particle Gibbs (Lindsten *et al.*, 2014), Rao-Blackwellised versions (Olsson and Ryden, 2011), algorithms the interleave different particle MCMC moves (Mendes *et al.*, 2014) and the use of gradient and Hessian information with the MCMC update (Dahlin *et al.*,

2014; Nemeth *et al.*, 2014). The re-parameterisation ideas presented in this paper could be employed together with many of these more advanced particle MCMC algorithms.

The ideas in this paper bear some similarity with the marginal augmentation approaches for improving the Gibbs sampler (e.g. van Dyk and Meng, 2001). In both cases, adding a latent variable to the model, and implementing the MCMC algorithm for this expanded model, can improve mixing. Our way of introducing the latent variables, and the way they are used are completely different though. However, both approaches can improve mixing for the same reason: introducing the latent variables reduces the correlation between variables updated at different stages of the Gibbs, or particle Gibbs, sampler.

In the next section we introduce particle filters and particle MCMC. Then in Section 3 we introduce our data augmentation approach. A key part of this is constructing a generic way of defining the latent variables so that the resulting particle MCMC algorithm is easy to implement. This we do by defining the latent variables to be observations of the parameters or of the stochastic process. By defining the likelihood for these observations to be conjugate to the prior for the parameters or the stochastic process we are able to analytically calculate quantities needed to implement the resultant particle MCMC algorithm. Furthermore, the accuracy of the pseudo-observations can be varied to allow them to contain more or less information. In Section 4 we investigate the efficiency of the new particle MCMC algorithms. We focus on three scenarios where we believe the data augmentation approach may be particularly useful. These are to improve the mixing of the particle Gibbs algorithm when there are strong dependencies between parameters and the unobserved stochastic process; to enable MCMC to be used within the particle filter; and to deal with diffuse initial distributions for the stochastic process. The paper ends with a discussion.

2 Particle MCMC

2.1 State-Space Models

For concreteness we consider application of particle MCMC to a state-space model, though both particle MCMC and the ideas we develop in this paper can be applied more generally. Throughout we will use $p(\cdot)$ and $p(\cdot|\cdot)$ to denote general marginal and conditional probabililty density functions, with the arguments making it clear which distributions these relate to.

Our state-space model will be parameterised by θ, and we introduce a prior distribution for this parameter, $p(\theta)$. We then have a latent discrete-time Markov process, $X_{1:T} = (X_1, \ldots, X_T)$. We do not observe the state process directly. Instead we take partial observations at each time-point, $y_{1:T} = (y_1, \ldots, y_T)$. We assume that the observation at any t just depends on the state process through its value at that time, x_t. Our interest

is in calculating, or approximating, the posterior for the parameters and states:

$$\begin{aligned} p(x_{1:T}, \theta | y_{1:T}) &\propto p(\theta) p(x_{1:T}|\theta) p(y_{1:T}|x_{1:T}, \theta) \\ &= p(\theta) \left[p(x_1|\theta) \prod_{t=2}^{T} p(x_t|x_{t-1}, \theta) \right] \left[\prod_{t=1}^{T} p(y_t|x_t, \theta) \right]. \end{aligned} \tag{1}$$

We frequently use the notation of an extended state vector $\mathcal{X}_t = (x_{1:t}, \theta)$, which consists of the full path of the state process to time t, and the value of the parameter. Thus $\mathcal{X}_T$ consists of the full state-process and the parameter, and we are interested in calculating or approximating $p(\mathcal{X}_T|y_{1:T})$.

2.2 Particle Filters

Particle filters are Monte Carlo algorithms that can be used to approximate posterior distributions for state-space models, such as (1). For reasons that will be apparent later, we will consider the generalisation where we condition on $\mathcal{Z}$, a function of $\mathcal{X}_T$. Thus the particle filter will target $p(\mathcal{X}_T|\mathcal{Z}, y_{1:T}) = p(x_{1:T}, \theta|\mathcal{Z}, y_{1:T})$. A simple particle filter algorithm is given in Algorithm 1.

Algorithm 1 Particle Filter Algorithm

Input:
A value of $\mathcal{Z}$.
The number of particle, N.

for $i = 1, \ldots, N$ **do**
  Sample $\mathcal{X}_1^{(i)}$ independently from $p(\mathcal{X}_1|\mathcal{Z})$.
  Calculate weights $w_1^{(i)} = p(y_1|\mathcal{X}_1^{(i)}, \mathcal{Z})$.
end for
Set $\hat{p}(y_1|\mathcal{Z}) = \frac{1}{N} \sum_{i=1}^{N} w_1^{(i)}$.
for $t = 2, \ldots, T$ **do**
  for $i = 1, \ldots, N$ **do**
    Sample j from $\{1, \ldots, N\}$ with probabilities proportional to $\{w_{t-1}^{(1)}, \ldots, w_{t-1}^{(N)}\}$.
    Sample $\mathcal{X}_t^{(i)}$ from $p(\mathcal{X}_t|\mathcal{X}_{t-1}^{(j)}, \mathcal{Z})$.
    Calculate weights $w_t^{(i)} = p(y_t|\mathcal{X}_t^{(i)}, \mathcal{Z})$.
  end for
  Set $\hat{p}(y_{1:t}|\mathcal{Z}) = \hat{p}(y_{1:t-1}|\mathcal{Z}) \left(\frac{1}{N} \sum_{i=1}^{N} w_t^{(i)} \right)$.
end for
Sample j from $\{1, \ldots, N\}$ with probabilities proportional to $\{w_T^{(1)}, \ldots, w_T^{(N)}\}$

Output: A value of the extended state, $\mathcal{X}_T^{(j)}$, and an estimate of the marginal likelihood $\hat{p}(y_{1:T}|\mathcal{Z})$.

If we stopped this particle filter algorithm at the end of iteration t, we would have a set of values for the extended state, often called particles, each with an associated weight. These weighted particles give an approximation to $p(\mathcal{X}_t|y_{1:t}, \mathcal{Z})$. At iteration $t+1$ we propagate the particles and use importance sampling to create a set of weighted particles to approximate $p(\mathcal{X}_{t+1}|y_{1:t+1}, \mathcal{Z})$. This involves first generating new particles at time $t+1$ through (i) sampling particles from the approximation to $p(\mathcal{X}_t|y_{1:t}, \mathcal{Z})$; and (ii) propagating these particles by simulating values for X_{t+1} from the transition density of the state-process, $p(X_{t+1}|\mathcal{X}_t, \mathcal{Z})$. Secondly, each of these particles at $t+1$ is then given a weight proportional to the likelihood of the observation y_{t+1} for that particle value. See Doucet *et al.* (2000) and Fearnhead (2008) for more details. At the end of iteration T we can output a single value of $\mathcal{X}_T$, by sampling once from the particles at time T, with the probability of choosing a particle being proportional to its weight.

A by product of the importance sampling at iteration $t+1$ is that we get a Monte Carlo estimate of $p(y_{t+1}|y_{1:t}, \mathcal{Z})$, and the product of these for $t = 1, \ldots, T$ gives an unbiased estimate of the marginal likelihood $p(y_{1:T}|\mathcal{Z})$ (Del Moral, 2004, proposition 7.4.1). This unbiased estimate will be key to the implementation of particle MCMC.

2.3 Particle MCMC

The idea of particle MCMC is to use a particle filter within an MCMC algorithm. There are two generic implementations of particle MCMC: particle marginal Metropolis-Hastings (PMMH) and particle Gibbs.

Particle marginal Metropolis-Hastings Algorithm

First we describe the particle marginal Metropolis-Hastings (PMMH) sampler (Andrieu *et al.*, 2010, Section 2.4.2). This involves choosing $\mathcal{Z}$, an appropriate function of the extended state, $\mathcal{X}_T$. Our MCMC algorithm has a state that is $\{\mathcal{Z}, \mathcal{X}_T, \hat{p}(y_{1:T}|\mathcal{Z})\}$, a value for this function, a corresponding value for the extended state and an estimate for the marginal likelihood given the current value of $\mathcal{Z}$. We assume that $\mathcal{Z}$ has been chosen so that we can both implement the particle filter of Algorithm 1, and also that we can calculate the marginal distribution, $p(\mathcal{Z})$. A common choice is $\mathcal{Z} = \theta$, though see below for other possibilities.

Within each iteration of PMMH we first propose a new value for $\mathcal{Z}$, using a random walk proposal. Then we run a particle filter to both propose a new value of $\mathcal{X}_T$ and to calculate an estimate for the marginal likelihood. These new values are then accepted with a probability that depends on the ratio of the new and old estimates of the marginal likelihood. Full details are given in Algorithm 2.

One intuitive interpretation of this algorithm is that we are using a particle filter to sample a new value of $\mathcal{X}'_T$ from an approximation to $p(\mathcal{X}'_T|\mathcal{Z}', y_{1:T})$ within a standard

Algorithm 2 Particle marginal Metropolis-Hastings Algorithm (Andrieu *et al.*, 2010)

Input:

An initial value $\mathcal{Z}^{(0)}$.

A proposal distribution $q(\cdot|\cdot)$.

The number of particles, N, and the number of MCMC iterations, M.

1: Run Algorithm 1 with N particles, conditioning on $\mathcal{Z}^{(0)}$, to obtain $\mathcal{X}_T^{(0)}$ and $\hat{p}(y_{1:T}|\mathcal{Z}^{(0)})$.

2: **for** $i = 1, \ldots, M$ **do**

3: Sample $\mathcal{Z}'$ from $q(\mathcal{Z}|\mathcal{Z}^{(i-1)})$.

4: Run Algorithm 1 with N particles, conditioning on $\mathcal{Z}'$, to obtain $\mathcal{X}_T'$ and $\hat{p}(y_{1:T}|\mathcal{Z}')$.

5: With probability

$$\min\left\{1, \frac{q(\mathcal{Z}^{(i-1)}|\mathcal{Z}')\hat{p}(y_{1:T}|\mathcal{Z}')p(\mathcal{Z}')}{q(\mathcal{Z}'|\mathcal{Z}^{(i-1)})\hat{p}(y_{1:T}|\mathcal{Z}^{(i-1)})p(\mathcal{Z}^{(i-1)})}\right\}$$

set $\mathcal{X}_T^{(i)} = \mathcal{X}_T'$, $\mathcal{Z}^{(i)} = \mathcal{Z}'$ and $\hat{p}(y_{1:T}|\mathcal{Z}^{(i)}) = \hat{p}(y_{1:T}|\mathcal{Z}')$; otherwise set $\mathcal{X}_T^{(i)} = \mathcal{X}_T^{(i-1)}$, $\mathcal{Z}^{(i)} = \mathcal{Z}^{(i-1)}$ and $\hat{p}(y_{1:T}|\mathcal{Z}^{(i)}) = \hat{p}(y_{1:T}|\mathcal{Z}^{(i-1)})$

6: **end for**

Output: A sample of extended state vectors: $\{\mathcal{X}_T^{(i)}\}_{i=1}^M$.

MCMC algorithm. If we ignore the approximation, and denote the current state by $(\mathcal{X}_T, \mathcal{Z})$, then the acceptance probability of this MCMC algorithm would be

$$\min\left\{1, \frac{q(\mathcal{Z}|\mathcal{Z}')p(y_{1:T}|\mathcal{Z}')p(\mathcal{Z}')}{q(\mathcal{Z}'|\mathcal{Z})p(y_{1:T}|\mathcal{Z})p(\mathcal{Z})}\right\},$$

as the $p(\mathcal{X}_T'|\mathcal{Z}')$ terms cancel as they appear in both the target and the proposal. The actual acceptance probability we use just replaces the, unknown, marginal likelihoods with our estimates. The magic of particle MCMC is that despite these two approximations, both in the proposal distribution for $\mathcal{X}_T$ given $\mathcal{Z}$ and in the marginal likelihoods, the resulting MCMC algorithm has the correct stationary distribution.

Particle Gibbs

The alternative particle MCMC algorithm, particle Gibbs, aims to approximate a Gibbs sampler. A Gibbs sampler that targets $p(\mathcal{Z}, \mathcal{X}_T|y_{1:T})$ would involve iterating between (i) sampling a new value for $\mathcal{Z}$ from its full-conditional given the other components of $\mathcal{X}_T$; and (ii) sampling a new value for $\mathcal{X}_T$ from its full conditional given $\mathcal{Z}$, $p(\mathcal{X}_{1:T}|\mathcal{Z}, y_{1:T})$.

Implementing step (i) is normally straightforward. For example if $\mathcal{Z} = \theta$, this involves sampling new parameter values from their full conditional given the path of the state-process. For many models, for example where there is conjugacy between the prior for the parameter and the model for the state and observation process, this distribution can

be calculated analytically.

The difficulty, however, comes with implementing step (ii). The idea of particle Gibbs is to use a particle filter to approximate this step. Denote the current value for $\mathcal{X}_{1:T}$ by $\mathcal{X}^*_{1:T}$. Then, informally, this involves implementing a particle filter but conditioned on one of the particles at time T being $\mathcal{X}^*_{1:T}$. We then sample one of the particles at time T from this conditioned particle filter and update $\mathcal{X}_{1:T}$ to the value of this particle. This simulation step is called a conditional particle filter, or a conditional SMC sampler. For full details of this see Andrieu *et al.* (2010).

2.3.1 Implementation

Whilst both particle MCMC algorithms have the correct stationary distribution regardless of the accuracy of the particle filter, the accuracy does affect the mixing properties. More accurate estimates of the marginal likelihood will lead to more efficient algorithms (Andrieu and Roberts, 2009). In implementing particle MCMC, as well as choosing details of the proposal distribution for $\mathcal{Z}$, we need also to choose the number of particles to use in the particle filter. Theory guiding these choices for PMMH is given in Pitt *et al.* (2012), Doucet *et al.* (2012) and Sherlock *et al.* (2015).

The standard implementation of particle MCMC will have $\mathcal{Z} = \theta$. However, our description is aimed to stress that particle MCMC is more general than this. It involves using MCMC proposals to update part of the extended state, and then a particle filter to update the rest. There is flexibility in choosing which part is updated by the MCMC move and which by the particle filter within the particle MCMC algorithm. For example, in order to deal with a diffuse initial distribution for the state-process, Murray *et al.* (2012) choose $\mathcal{Z} = (\theta, X_1)$, so that MCMC is used to update both the parameters and the initial value of the state-process. Alternatively, Wood *et al.* (2014) choose $\mathcal{Z} = \emptyset$, so that both the parameters and the path of the state are updated using the particle filter.

To demonstrate this flexibility, and discuss its impact on the performance of the particle MCMC algorithm, we will consider a simple example.

2.4 Example of Particle MCMC for linear-Gaussian Model

We consider investigating the efficiency of particle MCMC for a simple linear-Gaussian model where we can calculate the posterior exactly. The model has a one-dimensional state process, defined by

$$X_1 = \sigma_1 \epsilon_1^{(X)}; \quad X_t = \gamma X_{t-1} + \sigma_X \epsilon_t^{(X)}, \text{ for } t = 2, \ldots, T,$$

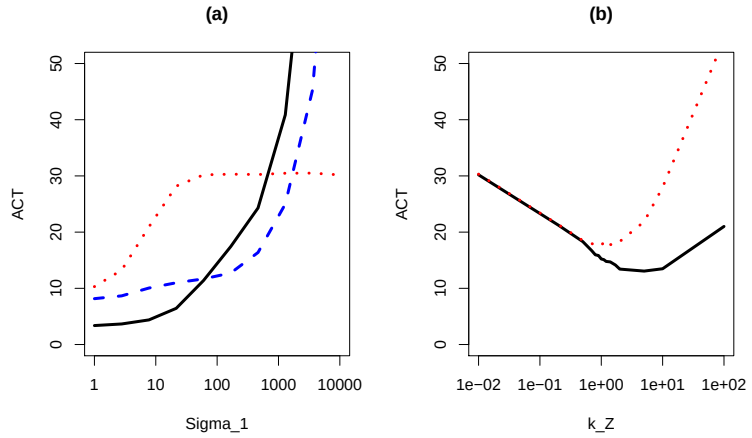

Figure 1: Autocorrelation Times (ACT) for Particle MCMC runs of the Linear-Gaussian Model. (a) ACT for three particle MCMC algorithms as we vary σ_1: $\mathcal{Z} = \emptyset$ (black), $\mathcal{Z} = \{\theta\}$ (blue dashed), $\mathcal{Z} = \{\theta, X_1\}$ (red dotted). (b) ACT for particle MCMC with pseudo-observations, $\mathcal{Z} = (Z_x, Z_\theta)$ as we vary the variance of the noise in the definition of pseudo-observations; the variance of the noise for Z_x and Z_θ is k_Z times the marginal posterior variance for X_1 and θ respectively. ACT is shown for X_1 (black) and Z_x (red dotted).

where $\epsilon_t^{(X)}$ are independent standard normal random variables. For $t = 1, \ldots, T$ we have observations

$$Y_t = \theta + X_t + \sigma_Y \epsilon_t^{(Y)},$$

where $\epsilon_t^{(Y)}$ are independent standard normal random variables. We assume that σ_1^2, σ_X^2, σ_Y^2 and γ are known, and thus the only unknown parameter is θ. Finally, we assume a normal prior for θ with mean 0 and variance σ_θ^2.

We simulated data for 100 times steps, with $\gamma = 0.99$, $\sigma_Y = 20$ and σ_X chosen so that that X_t process will have variance of 1 at stationarity. Our interest was in seeing how particle MCMC performs in situations where there is substantial uncertainty in X_1 and θ. Here we present results with $\sigma_\theta = 100$ as we vary σ_1. We implemented particle MCMC with $\mathcal{Z} = \emptyset$, $\mathcal{Z} = \{\theta\}$ and $\mathcal{Z} = \{\theta, X_1\}$. For the latter two implementations we used a random walk update for θ and X_1 with the variance set to the posterior variance; with independent random walk updates for θ and X_1 when $\mathcal{Z} = \{\theta, X_1\}$.

To evalulate performance we ran each particle MCMC algorithm using 100 particles and $250,000$ iterations. We removed the first quarter of iterations as burn-in, and calculated autocorrelation times for estimating θ. These are shown in Figure 1(a).

The results show the trade-off in the choice of $\mathcal{Z}$. Including more information in $\mathcal{Z}$ leads

to smaller proposed moves, with the proposed new values for θ and/or X_1 depending on their current values. However, more information in $\mathcal{Z}$, comes at the advantage of smaller Monte Carlo error in the estimate of the likelihood from the particle filter. This reduction in Monte Carlo error becomes increasingly important as the prior variance for X_1 increases. So for smaller values of σ_1 the best algorithm has $\mathcal{Z} = \emptyset$, whereas when we increase σ_1 first the choice of $\mathcal{Z} = \{\theta\}$ then the choice of $\mathcal{Z} = \{\theta, X_1\}$ performs better.

3 Augmentation Schemes for Particle MCMC

The example at the end of the previous section shows that the choice of which part of the extended state is updated by the particle filter, and which by a standard MCMC move, can have a sizeable impact on the performance of particle MCMC. Furthermore the default option for state-space models of updating parameters by MCMC and the state-process by a particle filter, is not always optimal.

The potential within this choice can be greatly enhanced by augmenting the original model. We will introduce an extra latent variable, Z, drawn from some distribution conditional on $\mathcal{X}_T$. This will introduce a new posterior distribution

$$p(\mathcal{X}_T, z|y_{1:T}) = p(\mathcal{X}_T|y_{1:T})p(z|\mathcal{X}_T), \tag{2}$$

where $p(\mathcal{X}_T|y_{1:T})$ is defined by (1) as before.

For any choice of $p(z|\mathcal{X}_T)$, if we marginalise Z out of (2) we get (1). Our approach will be to implement a particle MCMC algorithm for sampling from (2). This will give us samples $\{\mathcal{X}_T^{(i)}, z^{(i)}\}_{i=1}^M$ from (2), with the $\{\mathcal{X}_T^{(i)}\}_{i=1}^M$ from (1) as required. In implementing the particle MCMC algorithm, we will choose $\mathcal{Z} = z$. That is, we update the latent variable, Z, using the MCMC move, and we use a particle filter to update $\mathcal{X}_T$ conditional on Z and $y_{1:T}$.

Whilst, in theory, we have a completely free choice over the distribution of the new latent variable, Z, in practice we need to be able to easily implement the resulting particle MCMC algorithm. In practice this will mean that we need to be able to easily simulate from $p(\theta|z)$, $p(x_1|z,\theta)$ and, for $t = 2, \ldots, T$, $p(x_t|\mathcal{X}_{t-1}, z)$. For PMMH we will also need to be able to calculate the acceptance probability of the algorithm, which involves the term

$$p(z) = \int p(z|\mathcal{X}_T)p(\mathcal{X}_T)\mathrm{d}\mathcal{X}_T = \int p(z|\theta, x_{1:T})p(\theta)p(x_1)\prod_{t=2}^{T} p(x_t|x_{t-1})\mathrm{d}\theta\mathrm{d}x_{1:T}.$$

Thus we are restricted to cases where these conditional and marginal distributions can be calculated. We investigate possible generic choices in the next section.

3.1 Generic Augmentation Schemes: Pseudo-Observations

In choosing an appropriate latent variable Z we need to first consider the ease with which we can implement the resulting particle MCMC algorithm. A generic approach is to model Z as an observation of either θ or x_1 or both. As Z is a latent variable we have added to the model, we call these pseudo-observations.

By the Markov property of the state-process, if Z only depends on θ and/or x_1 then we have $p(x_t|\mathcal{X}_{t-1}, z) = p(x_t|\mathcal{X}_{t-1})$. Thus to be able to implement the particle filters we only need to choose our model for the pseudo-observation so that we can simulate from $p(\theta|z)$ and $p(x_1|z, \theta)$. To enable this we can let each component of Z be an independent pseudo-observation of a component of θ or x_1, with the likelihood for the pseudo-observation chosen so that the prior for the relevant component of θ or x_1 is conjugate to this likelihood. Conjugacy will ensure that not only can we simulate from the necessary conditional distributions but we can also calculate $p(z)$ as required to implement the particle MCMC algorithms. Constructing such models for the pseudo-observations is possible for many state-space models of interest. In some applications other choices for Z may be necessary or advisable: see Section 4.2 for an example.

To make these ideas concrete consider the linear Gaussian model of Section 2.4. We can choose $Z = (Z_\theta, Z_x)$ where Z_θ is a pseudo-observation of θ and Z_x is one of X_1. As we have both a Gaussian prior for θ and a Gaussian initial distribution for X_1, in each case a conjugate likelihood model arise from observations with additive Gaussian error. So for example we could choose

$$Z_\theta|\theta \sim \mathrm{N}(\theta, \tau^2). \tag{3}$$

This would give a marginal distribution of $Z_\theta \sim \mathrm{N}(0, \tau^2 + \sigma_\theta^2)$ and a conditional distribution of

$$\theta|z_\theta \sim \mathrm{N}\left(\frac{z_\theta\sigma_\theta^2}{\tau^2 + \sigma_\theta^2}, \frac{\tau^2\sigma_\theta^2}{\tau^2 + \sigma_\theta^2}\right).$$

Consider the case where we let Z depend only on θ. In specifying $p(z|\theta)$ we will have a choice as to how informative Z is about θ – for example the choice of τ in (3) for the linear Gaussian model example. As such this gives a continuum between the implementations of particle MCMC in Section 2.3. In the limit as Z is increasingly informative about θ, we converge on an implementation of particle MCMC where we update θ using MCMC and $X_{1:T}$ using the particle filter. As Z becomes less informative, we would tend to an implementation of particle MCMC where both θ and $X_{1:T}$ are updated through the particle filter.

To gain some intuition about the effect of the choice of Z we implemented particle MCMC for the linear-Gaussian model with $Z = (Z_x, Z_\theta)$ chosen as above. We simulated data as described in Section 2.4, but with $\sigma_1 = \sigma_\theta = 1,000$. Our aim is to investigate how the performance of the new particle MCMC algorithm varies as we vary the variance

of the noise in the definition of Z_x and Z_θ. For this model we can calculate analytically the true posterior distribution for X_1 and θ, and we chose the variance of the pseudo observations to be proportional to the marginal posterior variances. So for a chosen k_Z we set

$$\text{Var}(Z_x|x_1) = k_Z\text{Var}(X_1|y_{1:100}), \quad \text{and} \quad \text{Var}(Z_\theta|\theta) = k_Z\text{Var}(\theta|y_{1:100}).$$

Figure 1 (b) shows the resulting auto-correlation times for X_1 and Z_X as we vary k_Z. Choosing $k_Z \approx 0$ gives auto-correlation times similar to running particle MCMC with $\mathcal{Z} = \{X_1, \theta\}$. As k_Z is increased the efficiency of the particle MCMC algorithm initially increases. The intuition is that there is an underlying ideal MCMC algorithm linked to particle MCMC, which is the MCMC algorithm we obtain if the SMC estimate of the marginal likelihood were exact. As k_Z increases we expect better mixing of this underlying MCMC algorithm as we are conditioning on less information (see also Section 3.2). However for very large k_Z values the efficiency of particle MCMC becomes poor. In this case it starts behaving like particle MCMC with $\mathcal{Z} = \emptyset$, for which the large Monte Carlo error in estimating the likelihood leads to poorer mixing. The best values of k_Z correspond to adding noise to the pseudo-observations which is similar in size to the marginal posterior variances of X_1 and θ, and we notice good performance for a relatively large range of k_Z values.

3.2 Pseudo Observations for Particle Gibbs

We can gain some understanding of the benefit of using pseudo observations within particle Gibbs, by considering the mixing properties of the idealised Gibbs sampler that particle Gibbs is approximating. Assume we introduce pseudo observations, Z, of the parameters. Particle Gibbs approximates a Gibbs sampler where we iterate between updating Z given $\mathcal{X}_T = (x_{1:T}, \theta)$ and $\mathcal{X}_T$ given Z. Andrieu *et al.* (2013) give results on the mixing of a Particle Gibbs algorithm in terms of the mixing of the underlying Gibbs sampler. Under certain regularity conditions, they show that spectral gap of a particle Gibbs algorithm is bounded by a constant times the spectral gap of the idealised Gibbs sampler it approximates. Furthermore, as the number of particles increases the lower bound on the spectral gap of particle Gibbs converges to the spectral gap of the idealised Gibbs sampler.

We can interpret these results informally, as saying that, if we use sufficiently many particles, we expect the mixing of particle Gibbs will be similar to that of the idealised Gibbs sampler. Standard results for the Gibbs sampler give the following results for the mixing of this idealised sampler.

Theorem 1. *Assume we have a Gibbs sampler that targets a joint distribution for* $(Z, \mathcal{X}_T)$*, where* $\mathcal{X}_T = (\theta, x_{1:T})$*, and which iterates between updates of* Z *given* $\mathcal{X}_T$ *and* $\mathcal{X}_T$ *given* Z*.*

(i) If Z is conditionally independent of $x_{1:T}$ given θ, and a is a vector, then the lag-1 correlation for $a^T Z$ is

$$\frac{a^T var(Z)a - a^T E\left[var(Z|\theta)\right] a}{a^T var(Z)a},$$

(ii) If $E(Z|\theta) = \theta$, and there exists a $\lambda > 0$ such that $Var(Z|\theta) - \lambda Var(\theta)$ is positive definite, then the lag-1 correlation of $a^T Z$ is bounded above by $1/(1+\lambda)$.

(iii) Finally, if $Z = \theta + \epsilon$ where ϵ is an independent copy of θ then the geometric rate of convergence of the algorithm is bounded above by $1/\sqrt{2}$.

Proof. For part (i) we use the result that from Liu *et al.* (1994) (see also Amit, 1991; Liu, 1994) that the lag-1 autocorrelation of $a^T Z$, for some fixed vector a is

$$1 - \frac{a^T \mathrm{E}\left[\mathrm{var}(Z|\mathcal{X}_T)\right] a}{a^T \mathrm{var}(Z)a}.$$

Now as $\mathrm{var}(Z) = \mathrm{var}(\mathrm{E}(Z|\theta)) + \mathrm{E}(\mathrm{var}(Z|\theta))$ we can re-write the result in (i) to give that the lag-1 correlation for $a^T Z$ is

$$\frac{a^T \mathrm{var}(\mathrm{E}(Z|\theta))a}{a^T \mathrm{var}(\mathrm{E}(Z|\theta))a + a^T \mathrm{E}(\mathrm{var}(Z|\theta))a}.$$

For part (ii), we have $\mathrm{E}(Z|\theta) = \theta$. So the lag-1 autocorrelation is

$$\frac{a^T \mathrm{var}(\theta)a}{a^T \mathrm{var}(\theta)a + a^T \mathrm{E}(\mathrm{var}(Z|\theta))a}.$$

This is a decreasing function of $a^T \mathrm{E}(\mathrm{var}(Z|\theta))a$. By the condition on $\mathrm{var}(Z|\theta)$, we have that $a^T \mathrm{E}(\mathrm{var}(Z|\theta)) > \lambda a^T \mathrm{var}(\theta)a$, which gives the required bound.

Finally part (iii) uses the fact that the geometric rate of convergence is the maximal correlation between $\mathcal{X}_T$ and Z (Liu *et al.*, 1994). The maximal correlation can then be bounded using standard results for the maximal correlation of partial sums of independent and indentically distributed random variables (Dembo *et al.*, 2001). □

These results have two important practical implications. Part (i) shows that by using pseudo-observations we can improve the mixing of the idealised Gibbs sampler, and that we would expect a greater improvement as we increase the variance of Z given θ. Furthermore, parts (ii) and (iii) show that in the case where Z is defined as θ plus noise, we can get lower bounds on the performance of the idealised Gibbs sampler through appropriate choice of the noise. This suggests that pseudo observations will be beneficial for particle Gibbs algorithms where there is substantial correlation between

θ and $x_{1:T}$, where the idealised Gibbs sampler relating to the standard particle Gibbs algorithm will mix very slowly. In such cases we would expect that the improvement in mixing of the idealised Gibbs sampler that we obtain by adding noise which is, say, similar in distribution to the posterior for θ will more than compensate the need for more particles to control the Monte Carlo variability of the conditional SMC sampler. Parts (ii) and (iii) also suggest that if we have Z such that $\mathrm{E}(Z|\theta) = \theta$, then we should choose the variance of Z to be of the order of the posterior variance of θ.

3.3 MCMC within PMCMC

One approach to improve the performance of a particle filter is to use MCMC moves within it (see Gilks and Berzuini, 2001). An example is to use a MCMC kernel to update particles prior to propagating them to the next time-step. This involves a simple adaptation of Algorithm 1. Assume that $K_{t-1}(\cdot|\cdot)$ is a Markov kernel that has $p(\mathcal{X}_{t-1}|y_{1:t-1}, \mathcal{Z})$ as its stationary distribution. Then we change step 9 of Algorithm 1 to:

9: Sample $\mathcal{X}^*_{t-1}$ from $K_{t-1}(\cdot|\mathcal{X}^{(j)}_{t-1})$, and $\mathcal{X}^{(i)}_t$ from $p(\mathcal{X}_t|\mathcal{X}^*_{t-1}, \mathcal{Z})$.

The use of such an MCMC step can be particularly helpful for updating parameters, as they help to ensure some diversity in the set of parameter values stored by the particles is maintained, and, as a consequence, can improve the accuracy of estimates of the marginal likelihood. Where possible, a common choice of kernel is to update just the parameters of the particle by sampling from the full conditional $p(\theta|x_{1:t-1}, y_{1:t-1})$. Often such updates can be implemented in a computationally efficient manner as the full conditional distribution just depends on the state-path through fixed-dimensional sufficient statistics (Storvik, 2002; Fearnhead, 2002). For recent examples of the benefits of using such MCMC moves see, for example, Carvalho *et al.* (2010a), Carvalho *et al.* (2010b) and Gramacy and Polson (2011).

For standard implementations of particle MCMC, where $\mathcal{Z} = \theta$, using MCMC to update the parameters within the particle filter is not possible. Whereas by introducing pseudo-observations for the parameters, Z, and then implementing particle MCMC with $\mathcal{Z} = Z$ we can use such MCMC moves within the particle filter, or conditional particle filter. This can be of particular benefit if we use information from all particles, rather than just a single one. Andrieu *et al.* (2010) suggest an approach for doing this using Rao-Blackwellisation idea. We consider an alternative approach in Section 4.1.

4 Examples

4.1 Stochastic Volatility

A simple stochastic volatility model assumes a univariate state-process, defined as

$$X_1 = \sigma_0 \epsilon_1^{(X)}; \text{and } X_t = \gamma X_{t-1} + \sigma_X \epsilon_t^{(X)}, \text{ for } t = 2, \ldots, T,$$

where $\epsilon_t^{(X)}$ are independent standard normal random variables. For $t = 1, \ldots, T$ we have observations

$$Y_t = \sigma_Y \exp\{x_t\} \epsilon_t^{(Y)},$$

where $\epsilon_t^{(Y)}$ are independent standard normal random variables. Thus the state process governs the variance of the observations, with larger values of x_t meaning larger variability in the observation at time t.

We assume $\theta = (\gamma, \sigma_X, \sigma_Y)$ are unknown. We introduce independent priors, with γ having a normal distribution with mean μ_γ and variance σ_γ^2, but truncated to $(-1, 1)$; while $\beta_X = 1/\sigma_X^2$ and $\beta_Y = 1/\sigma_Y^2$ have gamma prior distributions with shape parameters a_X and a_Y respectively, and scale parameter b_X and b_Y respectively. We assume that $\sigma_0 = 1$.

We introduce a four-dimensional pseudo-observation $Z = (Z_X, Z_\gamma, Z_{\beta_X}, Z_{\beta_Y})$ where conditional on $(X_1, \gamma, \beta_X, \beta_Y)$

$$Z_X \sim \mathrm{N}(X_1, \tau_X^2), \quad Z_\gamma \sim \mathrm{N}(\gamma, \tau_\gamma^2),$$

$$Z_{\beta_X} \sim \text{gamma}(n_X, \beta_X), \text{and } Z_{\beta_Y} \sim \text{gamma}(n_Y, \beta_Y).$$

This choice for the pseudo-observations ensures that we can calculate the required marginal and conditional distributions, see Appendix A for details. To finalise the specification of these models we need to choose the values for τ_X, τ_γ, n_X and n_Y which determine how informative the pseudo-observations are.

Particle Gibbs

We first compare different implementations of the Particle Gibbs algorithm. Our focus here is to show that using pseudo-observations can improve mixing in scenarios where the idealised Gibbs sampler would mix poorly. This corresponds to situations where there is strong dependence in the state-process.

We simulated data with $T = 1,000$ observations, $\gamma = 0.99$, $\sigma_Y = 1$ and $\sigma_X = 1/(1 - 0.99^2)$, so that the stationary variance of the state process is 1. We present results for priors with $\mu_\gamma = 0.5$ and $\sigma_\gamma^2 = 0.5$; $a_X = 1$ and $b_X = 1/1000$; and $a_Y = 0.1$ and $b_Y = 0.1$. This corresponds to the true values of γ and β_X being in the tails of the prior, and a relatively uninformative prior for β_Y.

We implemented both the standard version of Particle Gibbs, with $\mathcal{Z} = \theta$, and Particle Gibbs with conditioning on the pseudo-observations, $\mathcal{Z} = Z$ defined above. Note that there exists a constant C such that $\mathrm{E}[\log(Z_{\beta_X} - C)] = \log(\beta_X)$, and $\mathrm{var}[\log(Z_{\beta_X})]$ is a constant that depends on n_X; and similarly for Z_{β_Y}. Thus Theorem 1 suggest we choose τ_X, τ_γ, n_X and n_Y so that the conditional variances of Z_X, Z_γ, $\log(Z_{\beta_X})$ and $\log(Z_{\beta_Y})$ are similar to the posterior variances for X_1, γ, $\log(\beta_X)$ and $\log(\beta_Y)$ respectively. We chose these tuning parameters so that the conditional variances were slightly smaller than the posterior variances we observed from a pilot run. For further comparison we show results for Particle Gibbs with no conditioning, $\mathcal{Z} = \emptyset$, again implemented with $N = 500$.

We ran the standard version with $N = 250$ particles, and the other two versions with $N = 500$. This was based on choosing N so that the estimate of the log-likelihood had a variance of around 1 (Pitt *et al.*, 2012). To compensate for the doubling of the computational cost of the conditional SMC sampler with the latter two versions, we ran the standard version of the Particle Gibbs for twice as many iterations. To ease comparison of results we then thinned the output by keeping the values of the chain on even iterations only.

Results are shown in Figure 2. The standard implementation performs badly here. This is because of strong dependencies between the parameters and the state-process that occurs for this model which means that the underlying Gibbs sampler mixes slowly. By conditioning on less information when running the conditional SMC sampler we reduce this dependence between the $\mathcal{X}_T$ and $\mathcal{Z}$ which improves mixing. However, choosing $\mathcal{Z} = \emptyset$ results in a substantial decrease in efficiency of the conditional SMC sampler. This is particularly pronounced due to the relatively uninformative priors we chose, and the fact that one of the parameter values was in the tail of the prior. If much more informative priors were chosen, using $\mathcal{Z} = \emptyset$ would give similar results to the use of pseudo-observations. Also this effect could be reduced slightly by increasing N further for this implementation of Particle Gibbs, but doing so will still lead to a less efficient sampler than using pseudo-observations.

PMMH with MCMC

We now compare PMMH on the stochastic volatility model. Our focus is purely on how using MCMC within the particle filter can help improve mixing over a standard PMMH algorithm. We simulated data with parameter values as above. To help reduce the computational cost involved in analysing this data, and hence implementing the simulation study, using PMMH we use more informative priors (which meant we could use fewer particles when running the particle filters), with $\mu_\gamma = 0.9$ and $\sigma^2_\gamma = 0.1$; $a_X = 1$ and $b_X = 1/100$; and $a_Y = 1$ and $b_Y = 1$.

We compared two implementations of PMMH, one with $\mathcal{Z} = \theta$ and one with $\mathcal{Z} = Z$. For the latter we were able to use MCMC within the particle filter to update the parameter

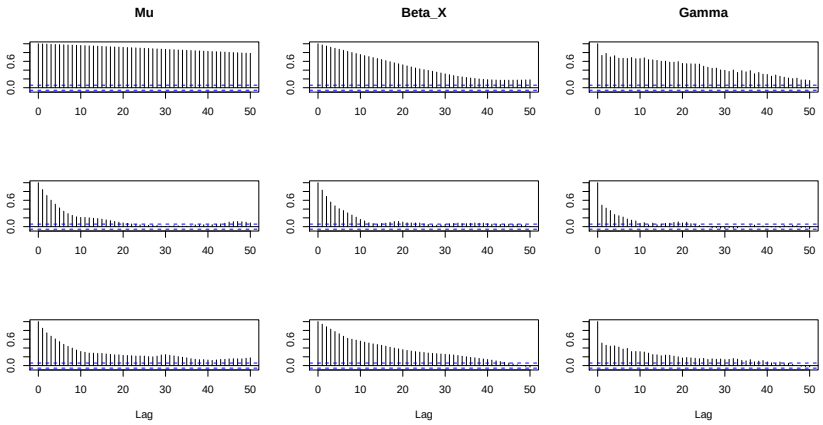

Figure 2: ACF plots for three runs of the Particle Gibbs conditioning on: $\mathcal{Z} = \theta$ (top row); $\mathcal{Z} = Z$ (middle row); and $\mathcal{Z} = \emptyset$ (bottom row). Each column corresponds to ACF for a different parameter: $\mu = \log(\beta_Y)$ (left column); β_X (middle column); and γ (right column).

values, using standard Particle Learning algorithms (Carvalho *et al.*, 2010a). Using the criteria of Pitt *et al.* (2012), we chose $N = 150$ and $N = 450$ particles respectively for these implementations. We used random walk proposals with the variances informed by a pilot run (Roberts and Rosenthal, 2001). Again we compensate for the slow running of the PMMH with pseudo-observations by running the other PMMH algorithm for three times as long, and thinning: keeping only every third value.

The main improvement in efficiency we observed with the second PMMH algorithm was through using the diversity in parameter values we obtain when using the Particle Learning Algorithm. Our approach for implementing this was to output a set of equally weighted particle values from the particle learning algorithm. We then make a decision as to whether to accept this set of particles, with the normal acceptance probability. Finally, we add an extra step to each iteration where we resample the state of the PMMH algorithm from the last stored set of particles. Full details are given in Algorithm 3 in Appendix B.

Trace-plots from part of the PMMH run are shown in Figure 3. These highlight the main improvement that using particle learning within PMMH gives. Both runs of PMMH can have long periods were they reject the output of the particle filter. However, by utilising the diversity in the parameter values of the particles that are output when particle learning is used, the PMMH algorithm is still able to mix over different parameter values in that case. Calculations of effective sample sizes show that this leads to a roughly two to three-fold increase in effective sample sizes (for a given CPU cost) for estimating β_X

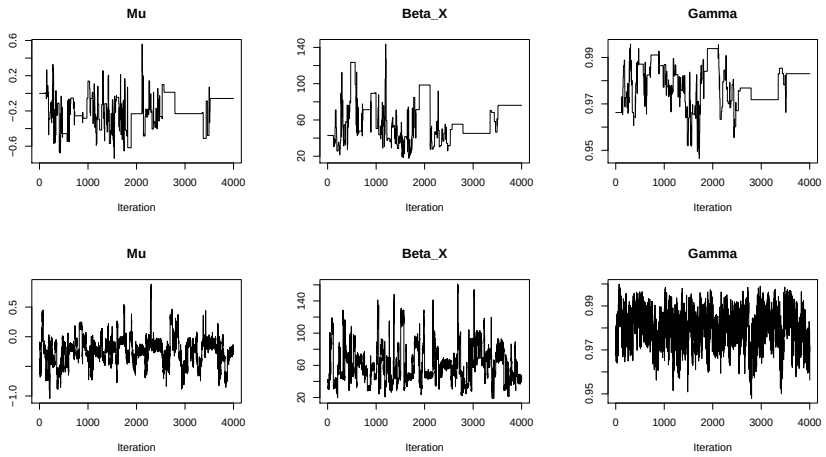

Figure 3: Trace plots for two runs of PMMH: $\mathcal{Z} = \theta$ (top row); and $\mathcal{Z} = Z$ (bottom row). Each column corresponds to a different parameter: $\mu = \log(\beta_Y)$ (left column); β_X (middle column); and γ (right column).

and γ. Note that we can use particle learning with particle MCMC if we choose $\mathcal{Z} = \emptyset$. The effectiveness of the resulting particle MCMC algorithm will depend crucially on how informative the priors are, and the degree of similarity between the prior and posterior. For uninformative priors, or priors which place little mass in areas of high posterior probability, using $\mathcal{Z} = \emptyset$ will be inefficient due to a large increase in the Monte Carlo error of the particle filter.

4.2 Dirichlet Process Mixture Models

We now consider inference for a mixture model used to infer population structure from population genetic data. Assume we have data from a set of diploid individuals, and this data consists of the genotype of each individual at a set of unlinked loci. Thus each locus will have a set of possible alleles (different genetic types), and the data for an individual at that locus will be which alleles are present on each of two copies of that individual's genome. We further assume that the individuals each come from one of an unknown number of populations. The frequency of each allele at each locus will vary across these populations. We wish to infer how many populations there are, and which individuals come from the same population.

This is an important problem in population genetics. We will consider a model based on that of Pritchard *et al.* (2000a). Though see Pritchard *et al.* (2000a), and Falush *et al.* (2003) for extensions of this model; Pritchard *et al.* (2000b) and Rosenberg *et al.* (2002) for example applications; and Price *et al.* (2006) and Patterson *et al.* (2006) for

alternative approaches to this problem.

Assume we have L loci. At locus l we have K_l alleles. The allele frequencies of these alleles in population j are given by $\mathbf{p}^{(j,l)} = (p_1^{(j,l)}, \ldots, p_{K_l}^{(j,l)})$. The genotype of individual i at locus l is $\mathbf{y}_{i,l} = (y_{i,l}^{(1)}, y_{i,l}^{(2)})$. Let x_i be a unobserved latent variable which defines the population that individual i is from. Then the conditional likelihood of $\mathbf{y}_i = (\mathbf{y}_{i,1}, \ldots, \mathbf{y}_{i,L})$ given $x_i = j$ is

$$p(\mathbf{y}_i | x_i = j) = \prod_{l=1}^{L} p_{y_{i,l}^{(1)}}^{(j,l)} p_{y_{i,l}^{(2)}}^{(j,l)}.$$

This model assumes the loci are unlinked and there is no admixture, hence conditional on x_i the data at each locus are independent.

We assume conjugate Dirichlet priors for the allele frequencies in each population. These priors are independent across both loci and population. For locus l the parameter vector of the Dirichlet prior is $(\lambda/K_l, \ldots, \lambda/K_l)$.

We use a mixture Dirichlet process (MDP) model (Ferguson, 1973) for the prior distribution of latent variables x_i. We will use the following recursive representation of the MDP model (Blackwell and MacQueen, 1973). Let $x_{1:i} = (x_1, \ldots, x_i)$ be the population of origin of the first i individuals, and define $m(x_{1:i})$ to be the number of populations present in $x_{1:i}$. We number these populations $1, \ldots, m(x_{1:i})$, and let $n_j(x_{1:i})$ be the number of these individuals assigned to population j. Then

$$p(x_{i+1} = j | x_{1:i}) = \begin{cases} n_j(x_{1:i})/(i+\alpha) & \text{if } j \leq m(x_{1:i}), \\ \alpha/(i+\alpha) & \text{if } j = m(x_{1:i}) + 1. \end{cases} \tag{4}$$

This model does not pre-specify the number of populations present in the data. Note that the actual labelling of populations under the MDP model is arbitrary, and the information in $x_{1:n}$ is essentially which subset of individuals belong to each of the populations. In our implementation the actual labels are defined by the order of the individuals in the data set. With population 1 being the population that the first individual belongs to, population 2 is the population that the first individual not in population 1 belongs to, and so on.

Inference for this model was considered in Fearnhead (2008) for the case where λ and α were known. Here we introduce hyperpriors for both these parameters, and perform inference using particle MCMC. We use independent gamma priors, with $\alpha \sim \text{Gamma}(5, 10)$ and $\lambda \sim \text{Gamma}(4, 1)$. If we condition on values for λ and α, then Fearnhead (2008) presents an efficient particle algorithm for this problem. This particle filter is based on ideas in Fearnhead and Clifford (2003) and Fearnhead (2004).

The particle filter of Fearnhead (2008) can struggle in applications where L is large, due to problems with initialisation. To show this we considered inference for $n = 80$ individuals at $L = 100$ loci, using a subset of data taken from Rosenberg *et al.* (2002).

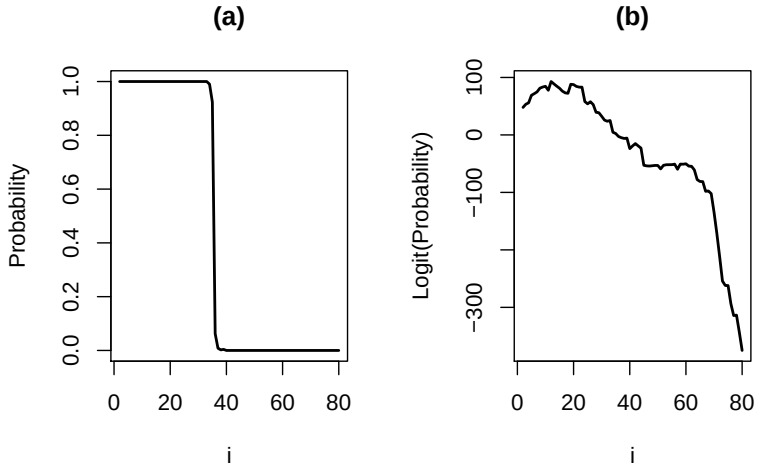

Figure 4: Plot of (a) $\Pr(X_1 = X_2|y_{1:i})$ and (b) $\text{logit}[\Pr(X_1 = X_2|y_{1:i})]$ for different sizes of data set i.

Figure 4 plots estimates of $\Pr(X_1 = X_2|y_{1:i})$ for increasing values of i. This shows how the posterior probability of the first two individuals being from the same population changes as we analyse data from more people. Initially this is close to 1, whereas once all data has been analysed the probability is essentially 0. This substantial change in probability causes problems in a particle filter, as all particles with $x_1 \neq x_2$ are likely to lost during resampling in the early iterations of the algorithm.

To overcome this problem of initialisation of the particle filter for this application we propose to introduce a pseudo observation, Z_x, that contains information about the populations of a random subset of the individuals. The distribution of Z_x given $x_{1:n}$ is obtained by (i) sampling the number of individuals in the subset, v say; (ii) choosing v individuals at random from the sample, $i_1, \ldots, i_v$; and (iii) letting $Z_x = \{(i_1, x_{i_1}), \ldots, (i_v, x_{i_v})\}$, the subset of individuals and their population labels.

As mentioned above, the actual values of the population labels is arbitrary, and Z_x just contains information about which of the individuals $i_1, \ldots, i_v$ belong to the same population. In practice, at each iteration we re-order the individuals in the sample so that individuals $i_1, \ldots, i_v$ become the first v individuals, and the order of the remaining individuals is chosen uniformly at random. The labels for the new first v individuals are changed to be consistent with our recursive represenation of the MDP model above.

In implementing PMCMC we use $\mathcal{Z} = (\lambda, \alpha, Z_x)$. Our proposal distribution for Z_x is just its true conditional distribution given $x_{1:n}$. We can easily adapt the particle filter of Fearnhead (2008) to condition on $\mathcal{Z}$, by fixing the labels of the first v individuals in

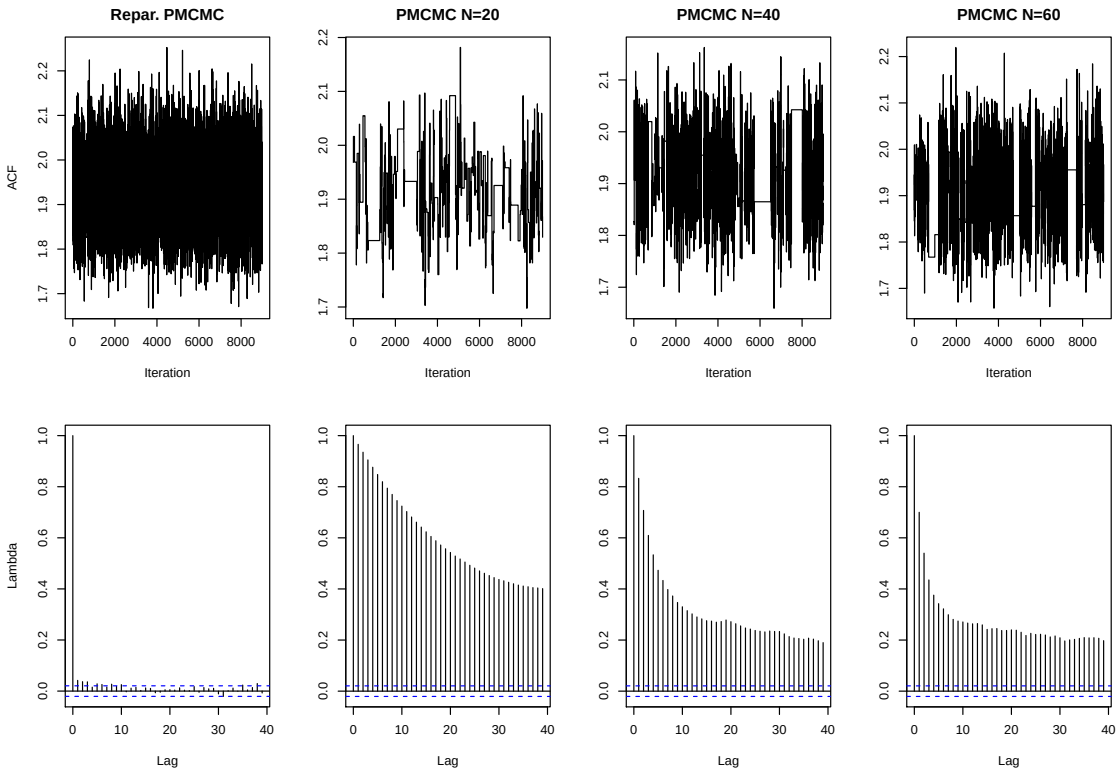

Figure 5: Trace plots (top) and acf plots (bottom) for λ for the reparameterised PMCMC with $Z = (\lambda, \alpha, Z_x)$ (left-hand column), and standard PMCMC with $Z = (\lambda, \alpha)$ (other columns). We ran the reparameterised PMCMC with $N = 20$ particles, and the standard PMCMC with $N = 20$, $N = 40$ and $N = 60$ particles. Results shown after removing the first 10^5 iterations as burn-in and thinning the remaining output by keeping only every 100th value.

the sample to those specified by Z_x. We use a random walk proposal for updating $\log \lambda$ and an independence proposal for α. Further details are given in Appendix C.

We compared the reparameterised PMCMC with this choice of Z with a standard PMCMC algorithm where $Z = (\lambda, \alpha)$. Our aim is purely to investigate the relative efficiency of the two implementations of PMCMC on this challenging problem. We ran each PMCMC algorithm for 10^6 iterations, storing only every 100th value. We implemented the new PMCMC algorithm using 20 particles for the particle filter, and with Z_x storing population information from an average of 5 individuals. We implemented the standard PMCMC algorithm with 20, 40 and 60 particles. Results, in terms of trace and acf plots for λ are shown in Figure 5.

We see that the reparameterised PMCMC algorithm has substantially better mixing than the standard PMCMC algorithm, even when the latter used 3 times as many particles, and hence would have three times the CPU cost per iteration. For all the standard PMCMC algorithms, the chain gets stuck for substantial periods of time. This is due to a large variance of the estimate of the likelihood. By running the particle filter conditional on Z_x we obtain a substantial reduction in the variance of our estimates of the likelihood, and hence avoid this problem.

Estimated auto-correlation times are 1.3 for the reparameterised PMCMC algorithm with 20 particles, and 105, 56 and 36 for the standard PMCMC with 20, 40 and 60 particles respectively. After taking account that the CPU cost of an iteration of PMCMC is proportional to the number of particles, this suggests the re-parameterised PMCMC is about 80 times more efficient than each of the standard PMCMC algorithms.

5 Discussion

We have introduced a way to generalise particle MCMC through data augmentation. The idea is to introduce new latent variables into the model, and then to implement particle MCMC where the MCMC moves update the latent variables, and the particle filter updates the rest of the variables in the model. By careful choice of the latent variables, we have shown this can lead to substantial gains in efficiency in situations where the standard particle MCMC algorithm performs poorly. For the Stochastic Volatility example of Section 4.1 we saw that it can help break down dependencies that make the particle Gibbs algorithm mix slowly, and can enable particle learning ideas to be used within the particle filter component of particle MCMC. It can also help for models where the particle filter struggles with initialisation, that is where at early time-steps the filter is likely to sample particles in areas that are inconsistent with the full data, as we saw in Section 4.2.

A key choice in implementing these ideas is choosing how informative the pseudo observations should be. We suggest choosing the variance of the pseudo observations to be of similar scale to the posterior variance, and in practice used a pilot run to estimate this. A better alternative could be to use adaptive MCMC methods (Andrieu and Thoms, 2008) to tune the variance of the pseudo observations. Note that while we have focussed on pseudo-observations of either the initial states of the process or of the parameters, the underlying idea is much more general. The only requirements on specifying the latent variable, Z, are that we need to be able to implement a particle filter conditional on Z, and to construct MCMC moves to update Z (see Section 3). Two possible extensions, each suggested by a reviewer, are using pseudo observations for all states (which can be implemented for if the state model is linear-Gaussian), and using the data augmentation ideas of Tanner and Wong (1987) in place of pseudo observations.

Acknowledgements: The first author was supported by the Engineering and Physical Sciences Research Council grant EP/K014463/1.

References

Amit, Y. (1991). On rates of convergence of stochastic relaxation for Gaussian and non-Gaussian distributions. *Journal of Multivariate Analysis* **38**(1), 82–99.

Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient computations. *Annals of Statistics* **37**, 697–725.

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing* **18**, 343–373.

Andrieu, C., Doucet, A. and Holenstein, R. (2010). Particle Markov chain Monte Carlo (with Discussion). *Journal of the Royal Statistical Society, Series B* **62**, 269–342.

Andrieu, C., Lee, A. and Vihola, M. (2013). Uniform Ergodicity of the Iterated Conditional SMC and Geometric Ergodicity of Particle Gibbs samplers. *ArXiv e-prints* , 1312.6432.

Blackwell, D. and MacQueen, J. B. (1973). Ferguson distributions via Polya urn schemes. *Annals of Statistics* **1**, 353–355.

Carvalho, C. M., Johannes, M. S., Lopes, H. F. and Polson, N. G. (2010a). Particle learning and smoothing. *Statistical Science* , 88–106.

Carvalho, C. M., Lopes, H. F., Polson, N. G. and Taddy, M. A. (2010b). Particle learning for general mixtures. *Bayesian Analysis* **5**, 709–740.

Chopin, N. and Singh, S. S. (2013). On the particle Gibbs sampler. *arXiv preprint arXiv:1304.1887* .

Dahlin, J., Lindsten, F. and Schön, T. B. (2014). Particle Metropolis–Hastings using gradient and Hessian information. *Statistics and Computing* , 1–12.

Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems With Applications*. Springer, New York.

Del Moral, P., Kohn, R. and Patras, F. (2014). On Feynman-Kac and particle Markov chain Monte Carlo models. *arXiv preprint arXiv:1404.5733* .

Dembo, A., Kagan, A., Shepp, L. A. *et al.* (2001). Remarks on the maximum correlation coefficient. *Bernoulli* **7**(2), 343–350.

Doucet, A., Godsill, S. J. and Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing* **10**, 197–208.

Doucet, A., Pitt, M. and Kohn, R. (2012). Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *ArXiv e-prints* .

van Dyk, D. A. and Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics* **10**(1), 1–50.

Falush, D., Stephens, M. and Pritchard, J. K. (2003). Inference of population structure using multilocus genotype data: Linked loci and correlated allele frequencies. *Genetics* **164**, 1567–1587.

Fearnhead, P. (2002). MCMC, sufficient statistics and particle filters. *Journal of Computational and Graphical Statistics* **11**, 848–862.

Fearnhead, P. (2004). Particle filters for mixture models with an unknown number of components. *Statistics and Computing* **14**, 11–21.

Fearnhead, P. (2008). Computational methods for complex stochastic systems: A review of some alternatives to MCMC. *Statistics and Computing* **18**, 151–171.

Fearnhead, P. (2011). MCMC for state-space models. In: *Handbook of Markov chain Monte Carlo* (eds. S. Brooks, A. Gelman, G. L. Jones and X. Meng), Chapman & Hall/CRC.

Fearnhead, P. and Clifford, P. (2003). Online inference for hidden Markov models. *Journal of the Royal Statistical Society, Series B* **65**, 887–899.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *Annals of Statistics* **1**, 209–230.

Gilks, W. R. and Berzuini, C. (2001). Following a moving target - Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society, Series B* **63**, 127–146.

Gramacy, R. B. and Polson, N. G. (2011). Particle Learning of Gaussian Process Models for Sequential Design and Optimization. *Journal of Computational and Graphical Statistics* **20**, 102–118.

Lindsten, F., Jordan, M. I. and Schön, T. B. (2014). Particle Gibbs with ancestor sampling. *The Journal of Machine Learning Research* **15**(1), 2145–2184.

Liu, J. S. (1994). Fraction of missing information and convergence rate of data augmentation. In: *Computing Science and Statistics: Proc. 26th Symposium on the Interface*, Interface Foundation of North America, Fairfax Station, VA, 490–496.

Liu, J. S., Wong, W. H. and Kong, A. (1994). Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika* **81**(1), 27–40.

Mendes, E. F., Carter, C. K. and Kohn, R. (2014). On general sampling schemes for Particle Markov chain Monte Carlo methods. *arXiv preprint arXiv:1401.1667* .

Murray, L. M., Jones, E. M. and Parslow, J. (2012). On Disturbance State-Space Models and the Particle Marginal Metropolis-Hastings Sampler. *ArXiv e-prints* .

Nemeth, C., Sherlock, C. and Fearnhead, P. (2014). Particle Metropolis adjusted Langevin algorithms. *arXiv preprint arXiv:1412.7299* .

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

Patterson, N., Price, A. L. and Reich, D. (2006). Population structure and eigenanalysis. *PLoS genetics* **2**(12), e190.

Pitt, M. K. and Shephard, N. (1999). Analytic convergence rates, and parameterization issues for the Gibbs sampler applied to state space models. *Journal of Time Series Analysis* **20**, 63–85.

Pitt, M. K., dos Santos Silva, R., Giordani, P. and Kohn, R. (2012). On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics* **171**(134-151).

Price, A. L., Patterson, N. J., Plenge, R. M., Weinblatt, M. E., Shadick, N. A. and Reich, D. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *Nature genetics* **38**(8), 904–909.

Pritchard, J. K., Stephens, M. and Donnelly, P. (2000a). Inference of population structure using multilocus genotype data. *Genetics* **155**, 945–959.

Pritchard, J. K., Stephens, M., Rosenberg, N. A. and Donnelly, P. (2000b). Association mapping in structured populations. *American Journal of Human Genetics* **67**, 170–181.

Rasmussen, D. A., Ratmann, O. and Koelle, K. (2011). Inference for nonlinear epidemiological models using genealogies and time series. *PLoS Comput Biol* **7**(8), e1002136.

Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science* **16**, 351–367.

Rosenberg, N. A., Pritchard, J. K., Weber, J. L., Cann, H. M., Kidd, K. K., Zhivotovsky, L. A. and Feldman, M. W. (2002). Genetic structure of human populations. *Science* **298**, 2381–2385.

Sherlock, C., Thiery, A. H. and Roberts, G. O. (2015). On the efficiency of pseudo marginal random walk Metropolis algorithms. *Annals of Statistics* **43**(238–275).

Storvik, G. (2002). Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transaction on Signal Processing* **50**, 281–289.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association* **82**(398), 528–540.

Wood, F., vand de Meent, J. W. and Mansinghka, V. (2014). A new approach to probabilistic programming inference. In: *AISTATS*.

A Calculations for the Stochastic Volatility Model

First consider Z_{β_X}. Standard calculations give

$$\begin{aligned} p(\beta_X|z_{\beta_X}) &\propto p(\beta_X)p(z_{\beta_X}|\beta_X) \\ &\propto \beta_x^{a_x-1}\exp\{-b_x\beta_x\}\,(\beta_X^{n_x}\exp\{-\beta_x z_{\beta_X}\}) \end{aligned}$$

This gives that the conditional distribution of β_X given z_{β_X} is gamma with parameters $n_x + a_x$ and $b_x + z_{\beta_X}$. Furthermore the marginal distribution for Z_{β_X} is

$$\begin{aligned} p(z_{\beta_X}) &= \int p(\beta_X)p(z_{\beta_X}|\beta_X)\mathrm{d}\beta_X \\ &= \frac{b_x^{a_x} z_{\beta_x}^{n_x-1}}{\Gamma(a_x)\Gamma(n_x)} \int \beta_x^{a_x+n_x-1}\exp\{-(b_x + z_{\beta_X}\beta_x\}\mathrm{d}\beta_X \\ &= \left(\frac{\Gamma(a_x+n_x)b_x^{a_x}}{\Gamma(a_x)\Gamma(n_x)}\right)\left(\frac{z_{\beta_x}^{n_x-1}}{(z_{\beta_x}+b_x)^{n_x+a_x}}\right) \end{aligned}$$

The calculations for Z_{β_Y} are identical.

Calculations for Z_X and Z_γ are as for the linear Gaussian model (see Section 3).

B PMMH Algorithm with Particle Learning

Algorithm 3 Particle marginal Metropolis-Hastings Algorithm with Particle Learning

Input:

An initial value $\mathcal{Z}^{(0)}$.

A proposal distribution $q(\cdot|\cdot)$.

The number of particles, N, and the number of MCMC iterations, M.

1: Run a Particle Learning Algorithm with N particles, conditioning on $\mathcal{Z}^{(0)}$, to obtain a set of equally weighted particle $\{\mathcal{X}_T^{(0,j)}\}_{j=1}^N$ and $\hat{p}(y_{1:T}|\mathcal{Z}^{(0)})$.

2: Obtain $\mathcal{X}_T^0$ by sampling uniformaly at random from $\{\mathcal{X}_T^{(0,j)}\}_{j=1}^N$

3: **for** $i = 1, \ldots, M$ **do**

4: Sample $\mathcal{Z}'$ from $q(\mathcal{Z}|\mathcal{Z}^{(i-1)})$.

5: Run a Particle Learning Algorithm with N particles, conditioning on $\mathcal{Z}'$, to obtain a set of equally weighted particle $\{\mathcal{X}_T^{(*,j)}\}_{j=1}^N$ and $\hat{p}(y_{1:T}|\mathcal{Z}')$.

6: With probability

$$\min\left\{1, \frac{q(\mathcal{Z}^{(i-1)}|\mathcal{Z}')\hat{p}(y_{1:T}|\mathcal{Z}')p(\mathcal{Z}')}{q(\mathcal{Z}'|\mathcal{Z}^{(i-1)})\hat{p}(y_{1:T}|\mathcal{Z}^{(i-1)})p(\mathcal{Z}^{(i-1)})}\right\}$$

set $\{\mathcal{X}_T^{(i,j)}\}_{j=1}^N = \{\mathcal{X}_T^{(*,j)}\}_{j=1}^N$ and $\hat{p}(y_{1:T}|\mathcal{Z}^{(i)}) = \hat{p}(y_{1:T}|\mathcal{Z}')$; otherwise set $\{\mathcal{X}_T^{(i,j)}\}_{j=1}^N = \{\mathcal{X}_T^{(i-1,j)}\}_{j=1}^N$ and $\hat{p}(y_{1:T}|\mathcal{Z}^{(i)}) = \hat{p}(y_{1:T}|\mathcal{Z}^{(i-1)})$

7: Obtain $\mathcal{X}_T^i$ by sampling uniformaly at random from $\{\mathcal{X}_T^{(i,j)}\}_{j=1}^N$

8: **end for**

Output: A sample of extended state vectors: $\{\mathcal{X}_T^{(i)}\}_{i=1}^M$.

C Calculations for the Dirichlet Process Mixture Model

The conditional distribution of Z_x given $x_{1:n}$ can be split into (i) the marginal distribution for v, $p(v)$; (ii) the conditional distribution of the sampled individuals, $i_1, \ldots, i_v$, given v. Given $i_1, \ldots, i_v$, the clustering of these individuals is deterministic, being defined by the clustering $(x_{i_1}, \ldots, x_{i_v})$.

The marginal distribution of Z_x thus can be written as

$$p(Z_x) = p(v)p(i_1, \ldots, i_v|v)p(x_{i_1}, \ldots, x_{i_v}).$$

Where we that, due to uniform sampling of the individuals,

$$p(i_1, \ldots, i_v|v) = \left(\begin{array}{c} n \\ v \end{array} \right).$$

Finally $p(x_{i_1}, \ldots, x_{i_v})$ is given by the Dirichlet process prior. If we relabel the populations so that $x_{i_1} = 1$, population 2 is the population of the first individual in $i_1, \ldots, i_v$ that is not in population 1, and so on; then for $v > 1$,

$$p(x_{i_1}, \ldots, x_{i_v}) = \prod_{j=2}^{v} p(x_{i_j}|x_{i_1}, \ldots, x_{i_{j-1}}),$$

with $p(x_{i_j}|x_{i_1}, \ldots, x_{i_{j-1}})$ defined by (4).

Within the PMMH we use a proposal for Z_x given $X_{1:n}$ that is its full conditional

$$q(Z_x|x_{1:n}) = p(Z_x|x_{1:n}) = p(v)p(i_1, \ldots, i_v|v).$$

In practice we take the distribution of v to be a Poisson distribution with mean 5, truncated to take values less than n. (Similar results were observed as we varied both the distribution and the mean value.)