Particle Markov Chain Monte Carlo

Ella Kaye Nathan Cunningham Kaspar Märtens

January 14, 2016

Abstract

Particle Markov chain Monte Carlo (PMCMC) is a recent development in the field of MCMC methods, and is an important advancement. We explore the features of the algorithm and construct an implementation in a combination of R and C++. We examine an application of our code on a commonly used non-linear state space model, finding that our implementation estimates reasonably well the unknown parameters in the model.

1 Introduction

We present an introduction to particle Markov chain Monte Carlo (PMCMC) methods, drawing heavily on Andrieu, Doucet, and Holenstein (2010). PMCMC methods are a combination of standard Markov Chain Monte Carlo (MCMC) methods and sequential Monte Carlo (SMC) methods which aim to overcome issues arising in the application of standard MCMC to complex, high-dimensional data. The standard MCMC case, where we are targeting a distribution π, requires us to specify a suitable proposal distribution which is both easy to sample from and adequately reflects the characteristics of π. This can become increasingly difficult as the dimensionality or dependence structure of π increases.

SMC partially solves this problem by focusing the simulation on simpler subcomponents of π; in the case of a state-space model aiming to sample from $p(x_{1:t}|y_{1:t})$ SMC will break the problem down into first approximating $p(x_1|y_1)$ and $p(y_1)$ and then approximating $p(x_{1:2}|y_{1:2})$ and $p(y_{1:2})$ and so on with these 'particles' evolving according to importance sampling and resampling steps. PMCMC methods make use of SMC algorithms to provide a sample approximately distributed according to $p(x_{1:t}|y_{1:t})$ which is then accepted according to a modification of the Metropolis-Hastings step.

While useful in other areas, PMCMC methods are particularly suited for inference in state space models where we wish to estimate the posterior probability of a Markov model given some partial observations.

In Section 2 we present an overview of state-space models and the SMC algorithm. In Section 3 we turn to PMCMC methods. We implement and explore algorithms for SMC and the PMCMC methods known as particle independent Metropolis-Hastings (PIMH) and particle marginal Metropolis-Hastings (PMMH). To test our implementations, we reproduce an example from the original paper, applying these methods to a non-linear state space model, as discussed in Section 4.

2 Background

2.1 State Space Models

In a state space model (SMM), there is a latent process, $\{X_t; t \geq 1\} \subset \mathcal{X}^{\mathbb{N}}$ which is uniquely defined by its initial density, $X_1 \sim \mu_\theta(\cdot)$ and the transition probability density

$$X_{t+1}|(X_t = x) \sim f_\theta(\cdot|x),$$

for some given parameter $\theta \in \Theta$, possibly multidimensional. Additionally, there is an observed process, $\{Y_t; t \geq 1\} \subset \mathcal{Y}^{\mathbb{N}}$, through which we may learn about $\{X_t\}$, given that at each time point, t, the

observation y_t is generated depending on x_t. Moreover, the observations are conditionally independent given $\{X_t\}$, so we have, for $1 \leq t \leq m$

$$Y_t|(X_1, \ldots, X_t = x, \ldots, X_m) \sim g_\theta(\cdot|x),$$

where $g_\theta(y|x)$ is their marginal probability density. Throughout the report, we will use notation $x_{i:j}$ for the vector $(x_i, \ldots, x_j)$.

2.2 Sequential Monte Carlo

As Andrieu et al. (2010) explain, in the SSM context, SMC methods are a class of algorithms to approximate sequentially the sequence of posterior densities $\{p_\theta(x_{1:t}|y_{1:t}); t \geq 1\}$ as well as the sequence of marginal likelihoods $p_\theta(y_{1:t}; t \geq 1)$ for a given $\theta \in \Theta$. It does this by first generating N particles, $\{X_1^k\}_{k=1}^N$ from an initial proposal distribution and assigning them importance weights $\{W_1^k\}_{k=1}^N$, then, at each time step, propagating $\{X_{1:t}^k\}$ forward according to weights $\{W_{1:t}^k\}$, then updating the weights.

In its simplest form, SMC improves on sequential importance sampling by adding a resampling step. The procedure is outlined in Algorithm 1. Here $q_\theta(\cdot|\cdot)$ are the proposal densities for the importance sampling step. Note that, for notational simplicity, wherever the index k is used, we mean 'for all $k \in \{1, \ldots, N\}$'.

Algorithm 1 Sequential Monte Carlo

At time $t = 1$:

(a) Sample $X_1^k \sim q_\theta(\cdot|y_1)$ and

(b) Compute and normalise the weights

$$w_1(X_1^k) := \frac{p_\theta(X_1^k, y_1)}{q_\theta(X_1^k|y_1)} = \frac{\mu_\theta(X_1^k) g_\theta(y_1|X_1^k)}{q_\theta(X_1^k|y_1)}, \tag{1}$$

$$W_1^k := \frac{w_1(X_1^k)}{\sum_{m=1}^N w_1(X_1^m)}.$$

At time $t = 2, \ldots, T$:

(a) Draw an 'ancestry vector' A_{t-1}^k from the categorical distribution, taking the vector $(W_{t-1}^1, \ldots, W_{t-1}^N)$ as weights

(b) Define $\bar{X}_{t-1}^k$ to be $X_{t-1}^{A_{t-1}^k}$, and let $\bar{X}_{t-1} = (\bar{X}_{t-1}^1, \ldots, \bar{X}_{t-1}^N)$

(c) Sample $X_t^k \sim q_\theta(\cdot|y_t, \bar{X}_{t-1})$ and set $X_{1:t}^k := (\bar{X}_{1:t-1}, X_t^k)$.

(d) Compute and normalise the weights

$$w_t(X_{1:t}^k) = \frac{f_\theta(X_t^k) g_\theta(y_t|\bar{X}_{t-1})}{q_\theta(X_t^k|y_t, \bar{X}_{t-1})}, \tag{2}$$

$$W_t^k := \frac{w_t(X_{1:t}^k)}{\sum_{m=1}^N w_t(X_{1:t}^m)}.$$

Essentially, steps (a) and (b) apply a resampling step on vector $(X_{t-1}^1, \ldots, X_{t-1}^N)$, denoting its resampled version by $(\bar{X}_{t-1}^1, \ldots, \bar{X}_{t-1}^N)$. To use this algorithm, we must know the data generation process (i.e. f_θ, g_θ), provide the observations $y_{1:T}$ and the proposal distribution q_θ. As a result, the algorithm provides an approximation of the joint posterior density at time T, $p_\theta(x_{1:t}|y_{1:t})$ given by

$$\hat{p}_\theta(\mathrm{d}x_{1:t}|y_{1:t}) := \sum_{k=1}^N W_T^k \delta_{X_{1:T}^k}(\mathrm{d}x_{1:T}), \tag{3}$$

as well as an estimate of the marginal likelihood $p_\theta(y_{1:T})$ given by

$$\hat{p}_\theta(y_{1:T}) := \hat{p}_\theta(y_1) \prod_{t=2}^T \hat{p}_\theta(y_t|y_{1:t-1}),$$

where

$$\hat{p}_\theta(y_t|y_{1:t-1}) = \frac{1}{N}\sum_{k=1}^{N} w_t(X_{1:t}^k)$$

is an estimate computed at time t of $p_\theta(y_t|y_{1:t-1})$. Note that sampling from the posterior distribution in Equation 3 is easy, given a realisation of the weights, as it is a discrete distribution with probability mass at points $X_{1:T}^k$. For further details on SMC, see Andrieu et al. (2010).

A default choice for the proposal distribution q_θ is to use μ at $t = 1$ and f for $t > 1$. This simplifies Equations 1 and 2 to $w_1(X_1^k) = g_\theta(y_1|X_1^k)$ and $w_t(X_{1:t}^k) = g_\theta(y_t|\bar{X}_{t-1})$. In our implementation of this algorithm, those were the proposals we adopted.

Figure 1 provides an illustration of SMC, where each of the black trajectories represents the 25-dimensional realisation of $X_{1:25}^k$. By assigning weights to each of these trajectories, we have obtained a weighted empirical measure in the 25-dimensional space, i.e. $\hat{p}_\theta(\mathrm{d}x_{1:t}|y_{1:t})$, which is the outcome of the SMC. While SMC has good performance in estimating the distribution of the last component, $p_\theta(x_t|y_{1:t})$ (the filtering distribution), there is a degeneracy problem when estimating the entire path, $p_\theta(x_{1:t}|y_{1:t})$ (the smoothing distribution). This occurs because of the resampling step. At every time step, some paths may die, thus the number of unique values representing each previous time point is monotonically decreasing.

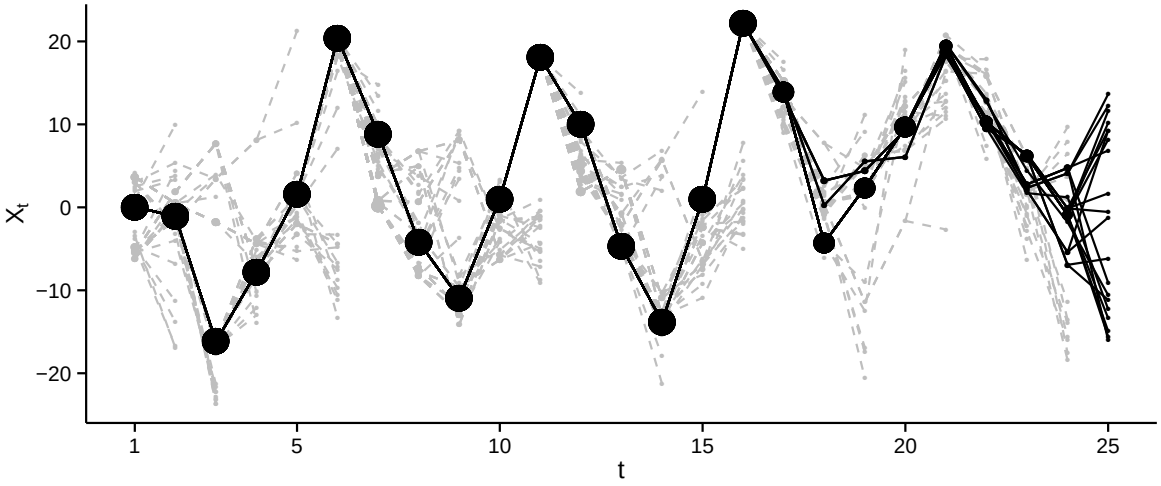

Figure 1: Illustration of the SMC to obtain $N = 20$ particles $X_{1:t}^1, \ldots, X_{1:t}^N$ for $1 \leq t \leq 25$. At each time point t, we assign weights to the previous particles $X_{1:(t-1)}^k$ and resample these according to their weights. Grey dashed trajectories denote particles which were not selected at some time t (these will not be used at any following step). Black trajectories show the final particles $X_{1:25}^k$, with point size denoting the number of unique X_t^k values. Note that up to time $t = 17$, all the previous particles $X_{1:17}^k$ are identical which is also known as the degeneracy problem.

3 Particle Markov chain Monte Carlo methods for state space models

Particle Markov chain Monte Carlo (PMCMC) methods are MCMC algorithms that target $p(\theta, x_{1:T}|y_{1:T})$, using the output of an SMC algorithm targeting $p(x_{1:T}|y_{1:T})$ using $N \geq 1$ particles as a proposal for a Metropolis-Hastings update.

In the remainder of this paper, we describe two PMCMC methods. We give the algorithms that we implemented, and refer the reader to Andrieu et al. (2010) for details and proofs. The first, particle independent Metropolis-Hastings (PIMH) sampler assumes that θ is known, and it is outlined in Algorithm 2.

Algorithm 2 Particle Independent Metropolis-Hastings

Step1: initialisation, $i = 0$:
(a) Run an SMC algorithm targeting $p_\theta(x_{1:T}|y_{1:T})$, outputting $\hat{p}_\theta(x_{1:T}|y_{1:T})$.
(b) Sample $X_{1:T}(0) \sim \hat{p}_\theta(\cdot|y_{1:T})$ and let $\hat{p}_\theta(y_{1:T})(0)$ denote the corresponding marginal likelihood estimate.

Step 2: for iteration $i \geq 1$:
(a) Run an SMC algorithm targeting $p_\theta(x_{1:T}|y_{1:T})$, outputting $\hat{p}_\theta(x_{1:T}|y_{1:T})$.
(b) Sample $X^*_{1:T} \sim \hat{p}_\theta(\cdot|y_{1:T})$ and let $\hat{p}_\theta(y_{1:T})^*$ denote the corresponding marginal likelihood estimate.
(c) With probability

$$1 \wedge \frac{\hat{p}_\theta(y_{1:T})^*}{\hat{p}_\theta(y_{1:T})(i-1)}, \tag{4}$$

set $X_{1:T}(i) = X^*_{1:T}$ and $\hat{p}_\theta(y_{1:T})(i) = \hat{p}_\theta(y_{1:T})^*$;
otherwise set $X_{1:T}(i) = X_{1:T}(i-1)$ and $\hat{p}_\theta(y_{1:T})(i) = \hat{p}_\theta(y_{1:T})(i-1)$.

The second, the particle marginal Metropolis-Hastings, jointly updates θ and $x_{1:T}$. The algorithm is much like Algorithm 2, except that at Step 1, we first set $\theta(0)$ arbitrarily, then in step (a) run a SMC algorithm targeting $p_{\theta(0)}(x_{1:T}|y_{1:T})$. Then, at Step 2, for each iteration we first sample $\theta^* \sim q\{\cdot|\theta(i-1)\}$, then run an SMC algorithm targeting $p_{\theta^*}(x_{1:T}|y_{1:T})$. Let $p(\theta)$ be the prior density on θ. In Step 2(c), we replace Equation 4 with the acceptance probability

$$1 \wedge \frac{\hat{p}_{\theta^*}(y_{1:T})p(\theta^*)}{\hat{p}_{\theta(i-1)}(y_{1:T})p\{\theta(i-1)\}} \frac{q\{\theta(i-1)|\theta^*\}}{q\{\theta^*|\theta(i-1)\}},$$

and set $\theta(i)$, $X_{1:T}(i)$ and $\hat{p}_{\theta(i)}(y_{1:T})$ accordingly.

4 Application to a non-linear state space model

We have implemented the PIMH and PMMH samplers in R, whereas the SMC step was coded in C++ to improve the performance[1]. To demonstrate and test these algorithms, we consider the following non-linear SSM:

$$\begin{aligned} X_t &= \frac{X_{t-1}}{2} + 25\frac{X_{t-1}}{1+X_{t-1}^2} + 8\cos(1.2t) + V_n \\ Y_t &= \frac{X_t^2}{20} + W_n, \end{aligned} \tag{5}$$

where $X_1 \sim \mathcal{N}(0, 5)$, $V_n \overset{iid}{\sim} \mathcal{N}(0, \sigma_V^2)$ and $V_n \overset{iid}{\sim} \mathcal{N}(0, \sigma_W^2)$. This model was also used to assess the performance of the PMCMC methods in Andrieu et al. (2010), allowing us to sense-check our implementation against the original. The posterior density is known to be highly multimodal as there is uncertainty about the sign of X_n, since it is only observed through its square.

In most of the experiments, we simulated 100 observations, $y_{1:100}$, according to the above model, varying the parameter values σ_V and σ_W and the number of particles N used for approximation. For the experiment shown in Figure 3, discussed in Section 4.1, we also varied the number of observations.

4.1 PIMH sampler

As a first check, we ran our implementation of Algorithm 2 for 10000 MCMC iterations, using 200 particles, and compared the filtering means $p_\theta(x_t|y_{1:t})$ to the true values of x_t from the simulation. We considered two scenarios: with a larger amount of noise ($\sigma_V^2 = 10$ and $\sigma_W^2 = 10$), and with relatively little noise ($\sigma_V^2 = 0.01$ and $\sigma_W^2 = 1$). The results are shown in Figure 2. As we would expect, when the

[1] Available at `https://github.com/EllaKaye/PMCMC`

variances are small, the sampler recovers the true signal very well, and when the variances are larger, it still tracks well.

Next, we compared the acceptance rate of the PIMH update sampler for various combinations of number of particles and number of time points, first with $\sigma_V^2 = 10$ and $\sigma_W^2 = 10$ and then with $\sigma_V^2 = 10$ and $\sigma_W^2 = 1$. The results are shown in Figure 3, which recreates Figure 3 in Andrieu et al. (2010). The figure points to a need to determine a sensible trade-off between acceptance rate and number of particles, bearing in mind computational complexity, given that the expense of the SMC proposal increases roughly linearly with the number of particles. However, even with a small number of particles ($N \approx 500$) the acceptance rate is reasonably high.

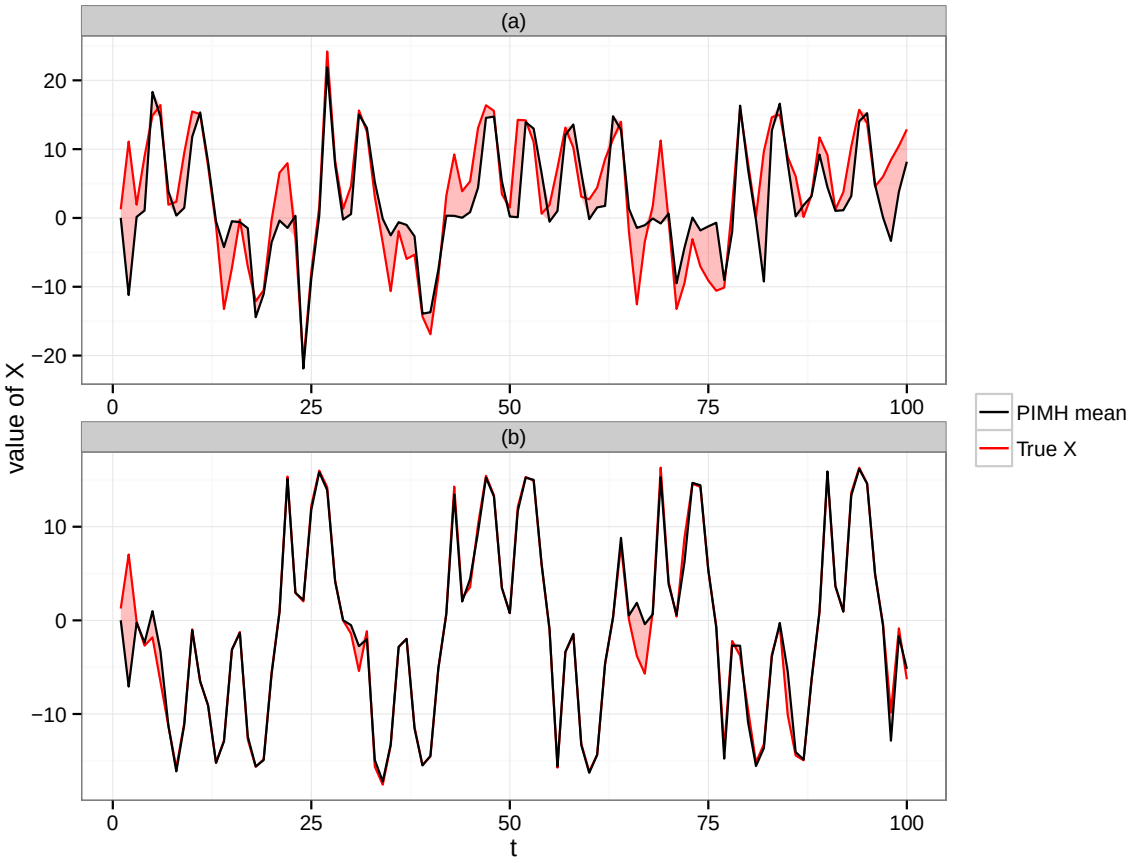

Figure 2: The discrepancy (red shaded area) between the true generated X_t values (red line) and its estimates obtained from the PIMH sampler (black line). We generated data according to model in Equation 5 under two scenarios: with parameter values (a) $\sigma_V^2 = 10$, $\sigma_W^2 = 10$, and (b) $\sigma_V^2 = 0.01$, $\sigma_W^2 = 1$. We ran the model with 200 particles and 10000 iterations. When the amount of noise decreases, the PIMH sampler uncovers the true signal near-perfectly, as expected.

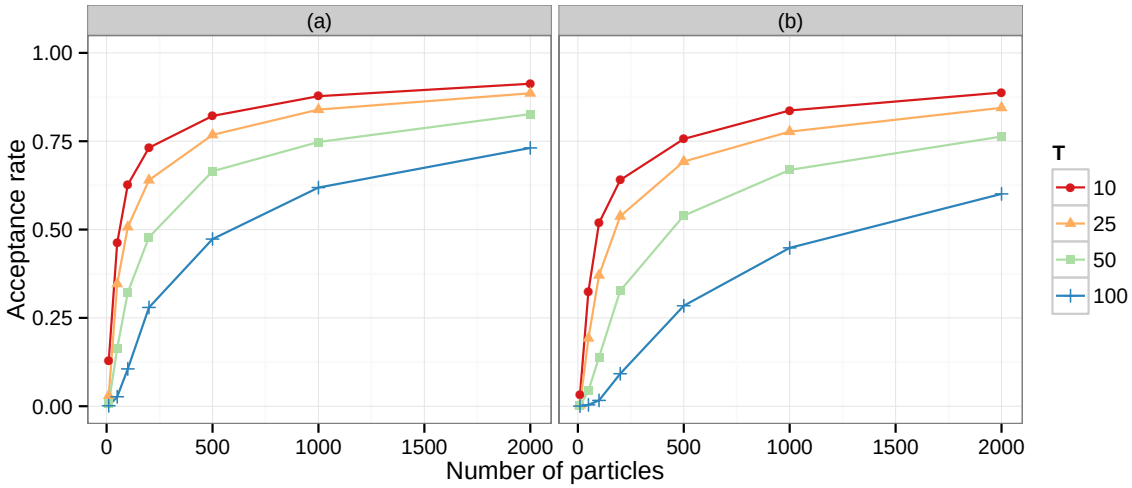

Figure 3: Acceptance rate for the PIMH sampler (y-axis) for various number of particles ($N \in \{10, 50, 100, 200, 500, 1000, 2000\}$, x-axis) and different number of observations ($T \in \{10, 25, 50, 100\}$, red, orange, green, blue). The data was generated according to the model in Equation 5, with parameter values (a) $\sigma_V^2 = 10$, $\sigma_W^2 = 10$, and (b) $\sigma_V^2 = 10$, $\sigma_W^2 = 1$.

4.2 PMMH sampler

The PIMH algorithm discussed previously assumes that the parameter θ is known. However, in a real life scenario, the parameter value is rarely known. The PMMH sampler accounts for this by assuming a prior distribution on θ, then updating its value at each MCMC iteration, generating a posterior distribution for θ.

For our model, we now assume that σ_V^2 and σ_W^2, are unknown, and place an uninformative inverse gamma prior distribution on each of them, $\mathcal{IG}(0.01, 0.01)$, and use the PMMH algorithm to estimate their posterior distributions.

Using the simulated data with $\sigma_V^2 = 10$ and $\sigma_W^2 = 1$ we ran the PMMH sampler with $N = 5000$ particles for 50000 observations with a burn-in of 10000 to estimate the unknown parameter, $\theta = (\sigma_V, \sigma_W)$. We use a normal random-walk proposal to propose new values of θ, with each of the components proposed independently with standard deviation of σ_V and σ_W 0.15 and 0.08 respectively. The PMMH sampler is initialised at $\theta_0 = (10, 10)$, which is significantly higher than the true values $\theta = (\sqrt{10}, 1)$. The resulting estimates are presented in Figure 4. The trace plots show the sampler is mixing well. However, the histograms show that the PMMH sampler appears to have over-estimated σ_v while under-estimating σ_w. This could be indicative of a problem in the underlying data, given that the simulated data contain just 100 observations.

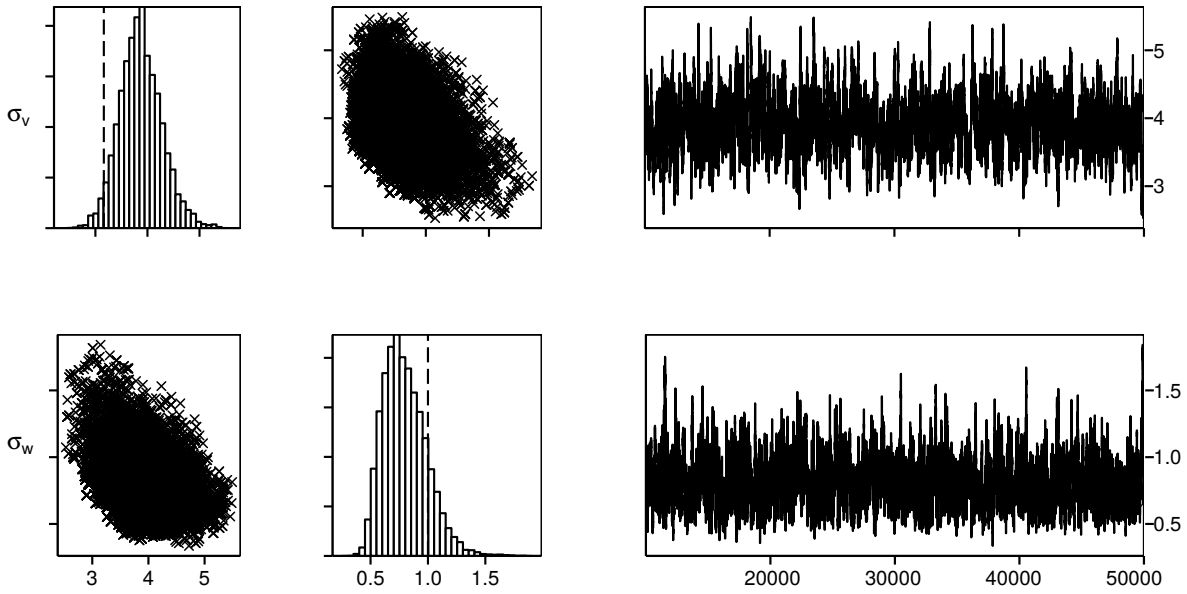

Figure 4: Histograms, scatter plots and trace plots of the simulated values of σ_V and σ_W over 50000 iterations after a burn-in of 10000 iterations. The true values are indicated by the dashed line on the histograms.

The ACFs for $\theta = (\sigma_V, \sigma_W)$ are presented in Figure 5 for 500 and 5000 particles. The ACF decreases more swiftly as the number of particles increases. This performance could be improved further with a thinning of the output, however this would come at considerable computational cost.

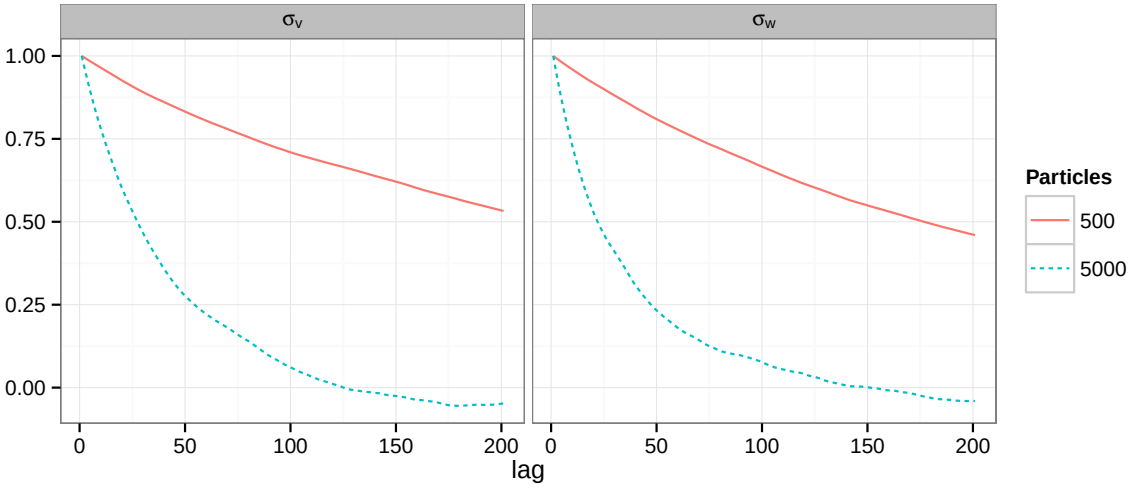

Figure 5: ACF of the unknown parameters σ_V and σ_W for 500 and 5000 particles.

References

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