Rheinische Friedrich-Wilhelms-Universität Bonn

BACHELORARBEIT

Particle Markov Chain Monte Carlo methods

Betreuer: JProf. Dr. Christian Pigorsch

Vorgelegt von

Stefan Harinko

Matrikelnummer: 2019081

Hermann-Wandersleb-Ring 6

53121 Bonn

Abgabetermin: 27.10.2010

Contents

1 Introduction

In economics one is often confronted with nonlinear and non-Gaussian relationships. Many economic time series exhibit strong patterns of nonlinear and non-Gaussian behavior. That's why e.g. stochastic volatility, regime switching and time-varying parameter models became so popular. Other applications can be found in various fields see e.g. [Doucet, de Freitas and Gordon (2001)] for specific algorithms. Most of the complex models lead to integrals that cannot be solved analytically. Methods to deal with high-dimensionality and complex dependence patterns are Markov Chain Monte Carlo (MCMC) and Sequential Monte Carlo (SMC) methods. Altough the convergence of MCMC algorithms is ensured under weak assumptions their speed to convergence is unreliable when using poor proposal distributions to explore the space.

[Andrieu, Doucet and Holenstein (2010)] used SMC to build proposal distributions for MCMC algorithms explaining it as approximations to standard and idealized MCMC methods which cannot be implemented in practice. One kind of SMC methods are particle filters therefore they call them Particle Markov Chain Monte Carlo methods (PMCMC).

The remaining part of this thesis is structured in the following way. The first part will explain the most basic (Sequential) Monte Carlo methods including the first practically usable SMC method named bootstrap [Gordon, Salmond and Smith (1993)] or particle filter which can be used in PMCMC methods. The next part introduces the well-known Metropolis-Hastings and Gibbs MCMC algorithms and their particle counterparts. The last section contains a simulation study applying PMCMC algorithms to often used nonlinear and non-Gaussian state space models.

2 Sequential Monte Carlo methods

As [Doucet and Johansen (2009)] point out particle methods for filtering have been the most common examples of SMC methods but are actually not the same thing because they include a broader range of alogrithms. In this thesis I generally won't make this distinction.

2.1 Problem statement: SMC for state space models

In econometrics state space or as a special case hidden Markov models are often used to describe dynamic systems. A state space model consists of two parts. One is characterizing

the hidden Markov process $\{X_n; n \geq 1\} \subset \mathcal{X}^{\mathbb{N}}$ with initial distribution $X_0 \sim \mu_\theta(\cdot)$ and transition probability density

$$X_{n+1}|(X_n = x) \sim f_\theta(\cdot|x) \tag{1}$$

with static parameters $\theta \in \Theta \subset \mathbb{R}^p$ which can be multidimensional. The second part is the observation process $\{Y_n; n \geq 1\} \subset \mathcal{Y}^{\mathbb{N}}$ with common marginal probability density (the observations are assumed conditionally independent given the state variables)

$$Y_{n+1}|(X_1, ..., X_n = x, ..., X_m) \sim g_\theta(\cdot|x). \tag{2}$$

The observations $y_{1:T} = \{y_1, ..., y_T\}$ are used to estimate the sequence of state variables $x_{0:T} = \{x_0, ..., x_T\}$ and the fixed parameters θ if unknown. For the SMC section the parameters are assumed known and estimation is considered later.

The uncertainty about the state variable is formulated as the posterior distribution $p_\theta(x_{0:n}|y_{1:n})$. The aim is to sequentially estimate the posterior and associated features of interest. Especially the marginal one-step-ahead predictive distribution $p_\theta(x_n|y_{1:n-1})$ and the filtering distribution $p_\theta(x_n|y_{1:T})$ which could be used recursively to estimate $p_\theta(x_n|y_{1:n})$ by applying Bayes' rule:

$$p_\theta(x_n|y_{1:n}) = \frac{p_\theta(y_n, x_n|y_{1:n-1})}{p_\theta(y_n|y_{1:n-1})} = \frac{p_\theta(y_n|x_n)p_\theta(x_n|y_{1:n-1})}{\int p_\theta(y_n|x_n)p_\theta(x_n|y_{1:n-1})dx_n}. \tag{3}$$

The first term in the numerator above is given by the model and could be seen as an update of the estimation when a new observation y_n arrives. The second term in the numerator could be estimated using last periods filtering distribution (starting with the known initial distribution of x_0) and the transition distribution given by the model $p_\theta(x_n|y_{1:n-1}) = \int p_\theta(x_n|x_{n-1})p_\theta(x_{n-1}|y_{1:n-1})dx_{n-1}$. The problem arises in most cases because the normalizing constant $p_\theta(y_n|y_{1:n-1}) = \int p_\theta(y_n|x_n)p_\theta(x_n|y_{1:n-1})dx_n$ cannot be calculated analytically.

In linear Gaussian models the Kalman filter can be used to solve the recursions. Or if the observation process is modelled as partially observed finite state space Markov chain an analytical solution is the Hidden Markov Model (HMM) filter. For more general models involving high-dimensionality, non-Gaussianity and nonlinearity various approximation methods haven been proposed. Hower the extended Kalman filter or Gaussian sum filter neglect certain important properties of the models and can lead to quite poor

results. Grid-based filters are applicable in low, but are not easy to implement in high dimensionality and suffer from computational complexity.

Sequential Monte Carlo (SMC) methods are a class of simulation based algorithms which can be flexible adapted to different problems. Given their sequential nature they can easily be implemented in a parallel environment to decrease their computing runtime by a great amount.

2.2 Perfect Monte Carlo sampling

Assuming we have N particles which are iid (independent and identically distributed) samples from $p(x_{0:n}|y_{1:t})$

$$\{x_{0:n}^{(i)}; i = 1, ..., N\}$$

we could use them to build an empirical distribution

$$\hat{p}_N(dx_{0:n}|y_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_{0:n}^{(i)}}(dx_{0:n})$$

where $\delta_{x_{0:n}^{(i)}}(dx_{0:n})$ is (delta-Dirac) mass located at particle $x_{0:n}^{(i)}$.

Estimates of various form can now be calculated using the N particles which form a discrete approximation of the continous distribution of $p(x_{0:n}|y_{1:n})$.

$$I_N(h_n) = \int h_n(x_{0:n})\hat{p}_N(dx_{0:n}|y_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} h_n(x_{0:n}^{(i)}).$$

If the posterior variance of $h_n(x_{0:n})$ satisfies $\sigma_{h_n}^2 < +\infty$ a CLT holds where the difference between estimated and true integral converges in distribution to a Normal distribution with zero mean.

$$\sqrt{N}(I_N(h_n) - I(h_n)) \overset{N\to\infty}{\longrightarrow} \mathcal{N}(0, \sigma_{h_n}^2)$$

Also a strong law of large numbers holds denoting almost sure convergence of the estimated integral to the true integral as $N \to \infty$

$$I_N(h_n) \underset{N\to\infty}{\overset{a.s.}{\longrightarrow}} I(h_n))$$

(as in [Doucet, de Freitas and Gordon (2001)],p.7).

The benefit from using Monte Carlo samples according to the target distribution is that the convergence rate is independent of the dimension of the integrand. But in the most interesting cases it is not feasible to sample from the posterior distribution $p(x_{0:n}|y_{1:t})$. For such complex problems Markov Chain Monte Carlo methods have been developed. But for recursive problems Sequential Monte Carlo methods (particle filters) are more suitable.

2.3 Particle filters

In this section I omitt the dependence on the fixed and known parameters θ. A Particle filter can recursively approximate the posterior $p(x_{0:n}|y_{1:n})$ after a new observation y_n is available. Also the filtering distribution $p(x_n|y_{1:n})$ can be used in a way that if you are only interested in the distribution of the current state x_n all approximation results before time $n-1$ can be dropped. That is why particle filters can be implemented fast and memory-friendly. Like in importance sampling samples are taken from an importance/proposal distribution $q_{0:n}(x_{0:n}|y_{1:n})$ which approximate the posterior distribution. To correct for the difference that they are from the "wrong" distribution importance weights are calculated.

$$w_n = \frac{p(x_{0:n}|y_{1:n})}{q_{0:n}(x_{0:n}|y_{1:n})} \tag{4}$$

Factorizing the numerator in (5) and the denominator in (6) and then plugging it in (4) leads to incremental weights (7) mostly using the Markovian model specification:

$$\frac{p(y_n|x_{0:n}, y_{1:n-1})p(x_{0:n}|y_{1:n-1})}{p(y_n|y_{1:n-1})} = \frac{p(y_n|x_{0:n}, y_{1:n-1})p(x_n|x_{0:n-1}, y_{1:n-1})p(x_{0:n-1}|y_{1:n-1})}{p(y_n|y_{1:n-1})} = \tag{5}$$

$$= \frac{p(y_n|x_n)p(x_n|x_{n-1})p(x_{0:n-1}|y_{1:n-1})}{p(y_n|y_{1:n-1})}$$

$$q_{0:n}(x_{0:n}|y_{1:n}) = q_n(x_n|x_{0:n-1}, y_{1:n})q_{0:n-1}(x_{0:n-1}|y_{1:n-1}) \tag{6}$$

$$w_n = \frac{p(y_n|x_n)p(x_n|x_{n-1})p(x_{0:n-1}|y_{1:n-1})}{p(y_n|y_{1:n-1})q_n(x_n|x_{0:n-1}, y_{1:n})q_{0:n-1}(x_{0:n-1}|y_{1:n-1})} \propto w_{n-1}\frac{p(y_n|x_n)p(x_n|x_{n-1})}{q_n(x_n|x_{0:n-1}, y_{1:n})}. \tag{7}$$

So at time n N samples are generated from the first part of the importance distribution (6)

$\{x_n^{(i)}\}_{i=1}^N$ which are added to the old trajectories $\{x_{0:n}^{(i)}\}_{i=1}^N = \{x_{0:n-1}^{(i)}, x_n^{(i)}\}_{i=1}^N$. At the end of every iteration the N samples or particles $\{x_{0:n}^{(i)}\}_{i=1}^N$ together with the current weights $\{w_n^{(i)}\}_{i=1}^N$ form a discrete approximation of the continous distribution.

When the denominator in (7) is reduced to $q_n(x_n|x_{n-1}, y_n)$ the incremental importance weights don't depend on the past observations $y_{1:n-1}$ and especially not on the past trajectories $\{x_{0:n-2}^{(i)}\}_{i=1}^N$

Introduced in [Gordon, Salmond and Smith (1993)] the *bootstrap filter* used the model given transition density as importance distribution $q_n(x_n|x_{n-1}, y_n) = p(x_n|x_{n-1})$. The weight update equation (7) is reduced to $w_n = w_{n-1}p(y_n|x_n)$. To address the weight degeneracy problem which means that over the recursion steps more probability mass will be located at few or only one particle Gordon et al. used multinomial resampling according to the weights w_n. After resampling the particles which added more to the discrete approximation are multiplied and the weights are evenly distributed afterwards as $w_n = \frac{1}{N}$. Altough the bootstrap filter is simple to implement it is not optimal in the sense that the proposal distribution doesnt't take in to account the current observation y_n.

The variance of the importance weights decrease the accuracy of the estimate in the following way. If the proposal density is close to the posterior density then:

$$\mathbb{E}_q\left(\frac{p(x_{0:n}|y_{1:n}}{q(x_{0:n}|y_{1:n}}\right) = 1$$

$$\mathbb{V}_q\left(\frac{p(x_{0:n}|y_{1:n}}{q(x_{0:n}|y_{1:n}}\right) = \left[\mathbb{E}_q\left(\frac{p(x_{0:n}|y_{1:n}}{q(x_{0:n}|y_{1:n}} - 1\right)^2\right] = 0.$$

So the weights should have a variance close to zero.

First improvements to the bootstrap filter include the Auxiliary Particle Filter [Pitt and Shephard (1999)] to use y_n in the proposal, resampling those past particles which are more likely under the observation y_n. Some MC variance in the weights results from the multinomial resampling step. Other resampling stratgies to reduce the MC variation include stratified resampling or residual resampling ([Kitagawa and Gersch (1996)], [Carpenter, Clifford and Fearnhead (1997)], [Higuchi (1997)],[Liu and Chen (1998)]). The stratified procedure e.g. divides the empirical cumulative density function of the weights into N parts and then draws one weight randomly in these sections.

In the original particle filter resampling is performed at every iteration. It could be better to resample only when a certain percentage of particles have very little weights

and don't add much to the estimation. The Effective Sample Size ESS (Liu 1996) $ESS = \frac{1}{\sum_{i=1}^{N}(\hat{w}_n^{(i)})^2}$ is one of the measures used. The interpretation based on the variability of the perfect weights is that if ESS=1 one particle has all the weight and if ESS=N the weights are equally distributed. So resample e.g. when ESS drops below 70% of the particles N.

Resampling is a key step to produce a good estimate of the filtering distribution $p(x_n|y_{1:n})$ but for the approximation of the joint distribution $p(x_{0:n}|y_{1:n})$ resampling means that fewer particles survive to represent earlier time steps. It hides the degeneracy problem to make the particle filter practical. As can be seen in the plot (1) using $N = 100$ and $N = 5000$ particles that the particle size should be great enough if the particle path from $1 : T$ should approximate $p(x_{1:T}|y_{1:T})$ and not only the last filter distribution.

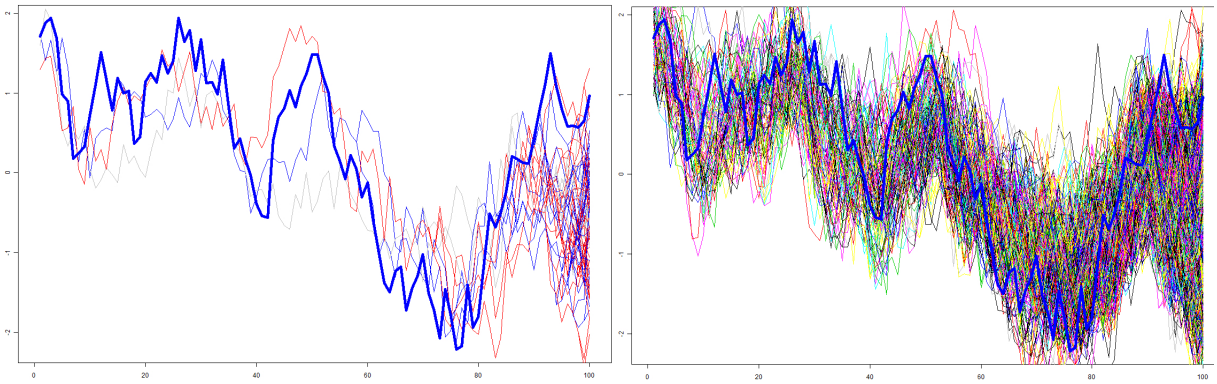

Figure 1: particle filter with N=5000 particles plotting surviving particle paths, same "color" means same ancestor at time 1

A Sample Importance Sampling Resampling/Bootstrap filter algorithm can be implemented in the following way:

- $t = 0$ for $i = 1, ..., N$ do
  sample $x_0^{(i)} \sim p(x_0)$

- for $t = 1, ..., T$ do
  sample for $i = 1, ..., N : x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)})$ compute importance weights $w_t^{(i)} = p(y_t|x_t^{(i)})$
  self-normalize weights: $\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^{N} w_t^{(i)}}$
  compute sequentially an estimate of $\hat{p}_\theta(y)$ from the unnormalized weights
  $\hat{p}_\theta(y_{1:t}) = \hat{p}_\theta(y_{1:t-1})(\frac{1}{N}\sum_{i=1}^{N} w_t^{(i)})$
  Resampling:
  for $i = 1, ..., N$ sample an index $j(i)$ such that the index is selected according to the discrete weight-distribution:

$P[j(i) = k] = \tilde{w}_t^{(k)}$

(resulting in a sample $x_t^{(i)} = \tilde{x}_t^{j(i)}$ where particles with high weights get duplicated more often which then have equal weights $w_t = \frac{1}{N}$

Implementation notes: to avoid numerical problems (underflows, precision) wherever very small probabilities are used a log transformation should be applied, e.g. in the step normalizing the weights or in MCMC algorithms were acceptance probabilities are calculated. A solution mentioned in [Cappe, Godsill and Moulines (2007)] is to subtract the largest log-weight from all log-weights to ensure that after normalizing the relatively larger weights can be calculated and too small values become zero due to underflow.

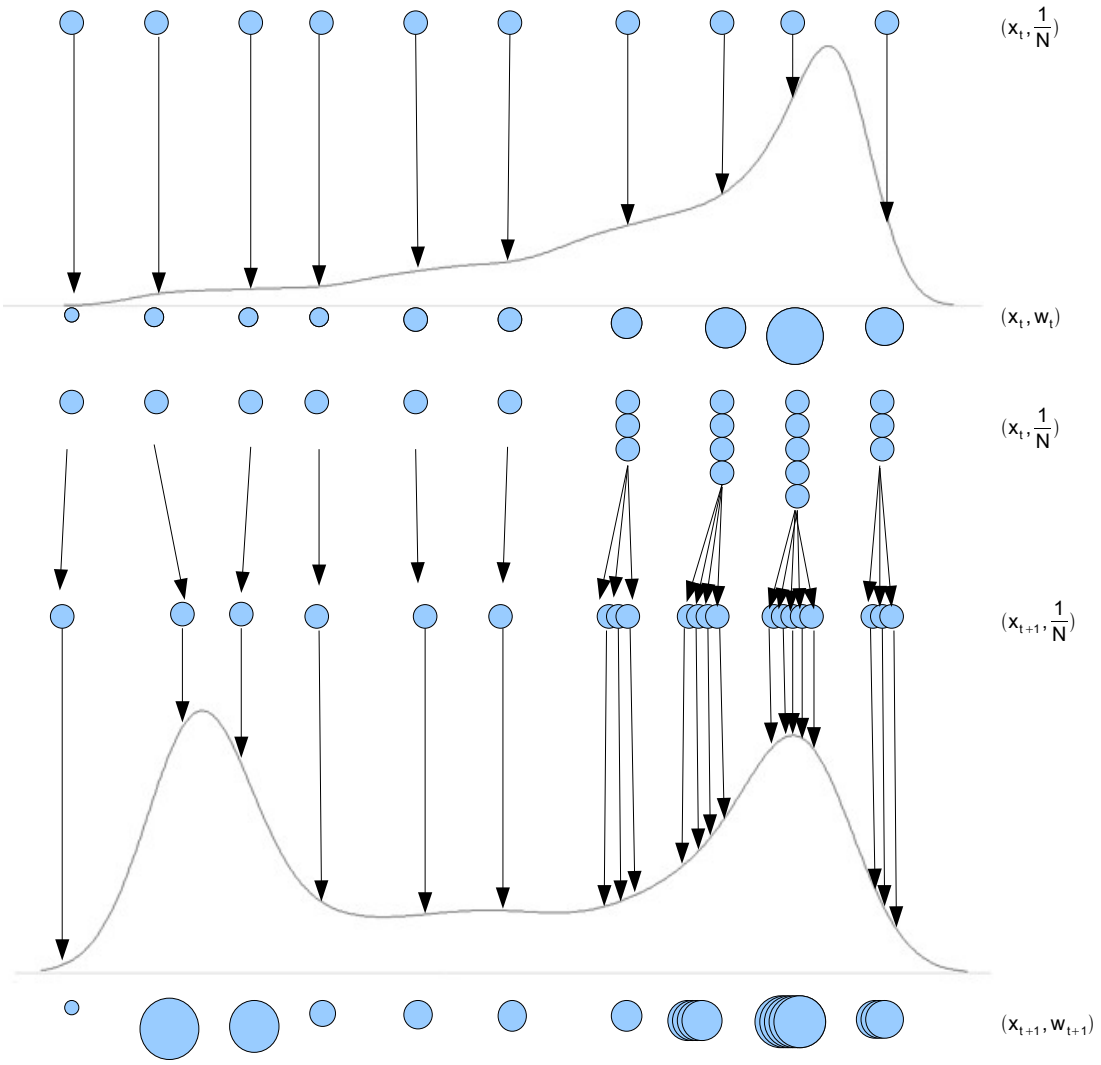

Figure 2: bootstrap filter: one transition and resampling step explained

For clarity explaining in figure 2 one step in a basic bootstrap filter from time t to t+1. Starting with an equally weighted sample from x_t (the particles in the first line of (2) don't represent all particles and are not necessarily equidistant). Then weights are

calculated according to the likelihood $p(y_t|x_t^{(i)})i \in \{1, ..., N\}$ of each particle to give an approximation of the filtering distribution $p(x_t|y_{1:t})$ as (x_t, w_t). After that the particles are duplicated according to their weights still providing the same approximation.

Then using the proposal distribution, in the boostrap filter case the model transition density $p(x_{t+1}|x_t)$ which approximates $p(x_{t+1}|y_{1:t})$. Using the newly available observation y_{t+1} the weights w_{t+1} are updated according to the likelihood $p(y_{t+1}|x_{t+1}^{(i)})$ to represent $p(x_{t+1}|y_{1:t+1})$. The Auxiliary Particle Filter from ([Pitt and Shephard (1999)]) would "preselect" those particles from $(x_t, \frac{1}{N})$ which are more likely under the new observation y_{t+1} to counter the depletion problem. In the graphic more particles from the top left would then be resampled in the next step to the bottom left.

Particle filters also give an estimate of the normalizing constant/marginal likelihood $p_\theta(y_{1:t})$. Most of the time approximating $p(y_{1:n}) = \int w_n q(x_{0:n}|y_{1:n}) dx_{0:n}$ by $\hat{p}(y_{1:n}) = \frac{1}{n}\sum_{i=1}^{N} w_n^{(i)}$ is impractical given that the sequential methods use resampling. But through the decomposition

$$p(y_{1:n}) = p(y_1) \prod_{n=2}^{T} p(y_n|y_{1:n-1})$$

with

$$p(y_n|y_{1:n-1}) = \int p(y_n|x_n) p(x_n|y_{1:n-1}) dx_n = \int p(y_n|x_{n-1}) p(x_{n-1}|y_{1:n-1}) dx_{n-1}$$

an approximation can either use the first or the last part, e.g.:

$$\hat{p}(y_n|y_{1:n-1}) = \sum_{i=1}^{N} p(y_n|x_{n-1}^{(i)}) \hat{w}_{n-1}^{(i)}$$

where $\hat{w}_{n-1}$ is the self-normalized weight.

2.4 Parameter estimation using SMC methods

Now considering the important problem of estimating the paramters θ in state space models. Almost all models contain unknown parameters which need to be estimated from the observations y_n. The distribution of interest is therefore $p(\theta, x_{1:t}|y_{1:t})$. Simple approaches to the problem would extend the state space by θ either by leaving $\theta_t = \theta$ fixed or adding small "roughening penalties" [Gordon, Salmond and Smith (1993)] $\theta_{t+1} = \theta_t + \zeta_{t+1}$ with $\zeta_{t+1} \sim \mathcal{N}(0, W_{t+1})$ with prespecified variance matrix W_{t+1}. In the first "solution" the parameters are drawn from a prior distribution and only the weights are

updated each time step and may therefore not represent the true posterior distribution. And the second one is in fact adding time-varying parameters and would probably result in a more diffuse posterior than the true one for the fixed parameters would look like.

One algorithm extending the auxiliary particle filter idea was proposed by [Liu and West (2001)] and addresses this problem in a better way by generating new parameters according to the changing target distribution. The Liu-West algorithm constructs an approximate target distribution which is continous both in x_t and θ. Sampling parameters every step from a continous proposal distribution. They use a Normal distribution instead of a point mass so that the mixture becomes a continous distribution. But using the following mixture $\pi(\theta) = \sum_{i=1}^{N} w_t^{(i)} \mathcal{N}(\theta^{(i)}, \Lambda)$ it would preserve the mean but result in $Var[\theta] = \Lambda + \Sigma > \Sigma$ (using variance decomposition and conditioning on an inidicator for the mixture component, where Σ is the variance matrix of the true parameters). Liu and West suggest to use paramters $a \in (0, 1)$ and $a^2 + h^2 = 1$ in the mean and variance part respectively to get $Var[\theta] = \Sigma$.

Particle filter estimates for the likelihood have been used in [Kim, Shephard and Chib (1998)], [Chib, Nardari and Shephard (2002)] for stochastic volatility models and in [An and Schorfheide (2007)] for dynamic stochastic general equilibrium models. Storvik (2002) proposed "particle learning" algorithms based on sufficient statistics. Similiarly the "exact particle filtering and particle learning" method from [Johannes and Polson (2006)] updates states, sufficient statistics and parameters recursively leading to promising results e.g. for stochastic volatlity models. Parameter weight degeneration can be observed in Particle Learning when the observation time n increases, so these methods are probably not suitable for the cases where new observations arrive sequentially.

3 Particle Markov Chain Monte Carlo

[Andrieu, Doucet and Holenstein (2010)] proposed using the output of SMC in MCMC algorithms which need samples from $p_\theta(x_{1:T}|y_{1:T})$. Therefore calling them approximations to 'idealized' MCMC algorithms. They showed that even using the basic bootstrap filter it is possible to target $p_\theta(x_{1:T}|y_{1:T})$ or $p(\theta, x_{1:T}|y_{1:T})$. And starting with fixed $N \geq 1$ particles the methods lead to convergent algorithms. So they provide a proven way for inference on parameters θ and states $x_{1:T}$. Despite the increased computationally effort associated with PMCMC methods they might lead to better inference than specialized

SMC algorithms and don't require much model specific user input.

3.1 Metropolis Hastings and Gibbs Sampler

Two (related) Markov Chain Monte Carlo methods are used in the Particle MCMC context. One method to sample from a distribution π is the Metropolis-Hastings algorithm. Starting with an initial value x at step i=0 new values are generated according to a proposal $x^* \sim q(\cdot|x_{i-1})$ the new value x^* is accepted with probability:

$$1 \wedge \frac{\pi(x^*)q(x_{i-1}|x^*)}{\pi(x_{i-1})q(x^*|x_{i-1})}.$$

If x_i^* is accepted it gets added to the chain if not the chain stays at the current value. Is the proposal distribution independent of the last value x_{i-1} we get a special case of the algorithm called Independent Metropolis-Hastings. In either case the values from every step are samples from the target distribution π. In practice a burn-in phase is used to get a starting sample.

The Gibbs-sampler can be used for the same problem if it's possible to generate samples from the full conditonals which are distributions to sample one component conditioned on the remaining ones. Suppose $\pi(x)$ is the target and can be decomposed into d components at time t $x^t = (x_1^t, ..., x_d^t)$ then in the next step $t+1$: x_i^{t+1} gets drawn from $\pi(x_i|x_1^{t+1}, ..., x_{i-1}^{t+1}, x_{i+1}^t, ..., x_d^t)$ for $i = 1, ..., d$. The method is narrowing down the dimension of the space compared to the normal Metropolis-Hastings algorithms. Using the local information of the conditionals the new proposed values are not so noisy and get accepted with probability 1. One special case is the collapsed Gibbs sampler mentioned in [Liu (2001)]. It "collapses" e.g. two random variables to one if it's possible to integrate one component out.

3.2 Particle Independent Metropolis Hastings

A standard independent MH algorithm targeting $p_\theta(x_{1:T}|y_{1:T})$ would propose new states independent of the current state. And accept new states * given old states with Metropolis-Hasting acceptance probability:

$$1 \wedge \frac{p_\theta(X_{1:T}^*|y_{1:T})q_\theta(X_{1:T}|y_{1:T})}{p_\theta(X_{1:T}|y_{1:T})q_\theta(X_{1:T}^*|y_{1:T})}.$$

Where $p_\theta(x_{1:T}|y_{1:T})$ as proposal would be optimal. Using the approximation given

by SMC a sample $X_{1:T}$ can be taken according to the particle weights at the end. But the marginal of $X^*_{1:T}$ is intractable. The distribution is given by $q_\theta(dx_{1:T}|y_{1:T}) = \mathbb{E}\{\hat{p}_\theta(dx_{1:T}|y_{1:T})\}$ where expectation is taken with respect to all random variables generated by the SMC estimation. The expectation defined on the larger space leads to the simple acceptance probability.

- Iteration i=0:
  run SMC with target $p_\theta(x_{1:T}|y_{1:T})$, sample $X_{1:T}$ and calculate $\hat{p}_\theta(y_{1:T})(0)$

- Iterations i=1:many
  run same SMC, sample $X^*_{1:T}$ calculate $\hat{p}_\theta(y_{1:T})(i)^*$
  accept new state and marginal likelihood with probability:

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

  If accepted set current state and current likelihood to proposed ones, if not stay at old values.

[Andrieu, Doucet and Holenstein (2010)] proofed that the PIMH update leaves the target density invariant and the sampler is ergodic. So the generated Markov chain will eventually produce simulations from $p_\theta(x_{1:T}|y_{1:T})$ In particular for integrable functions h the law of large numbers can be applied (in MCMC often called *Ergodic theorem*).

$$\frac{1}{T}\sum_{t=1}^{T} h(X^*_{1:T}) \longrightarrow \mathbb{E}_{p(x_{1:T}|y_{1:T})}[h(X_{1:T})]$$

The proof of invariance uses a target and proposal density defined on the extended space including all random variables. Then showing that the ratio of target/proposal leads to the ratio of estimated normalizing constant to true normalizing constant which in the SSM case is $\frac{\hat{p}_\theta(y_{1:T})}{p_\theta(y_{1:T})}$ justifying the acceptance probability in the alogithm above. Ergodicity follows from the proof of invariance and results for IMH methods. Note that the acceptance probability increases with increasing number of particles because the SMC estimates in numerator and denominator ($\hat{p}_\theta(y_{1:T})$) converge to 1 if they are correct estimates of the true distribution.

3.3 Particle Marginal Metropolis Hastings

Similiar to regular marginal Metropolis Hastings an update ratio is defined which can be approximated by the output of a SMC algorithm. The algorithm should produce joint updates of θ and $x_{1:T}$ from $p_\theta(\theta, x_{1:T}|y_{1:T}) = p(\theta|y_{1:T})p_\theta(x_{1:T}|y_{1:T})$.

- Iteration i=0:
  set $\theta(0)$ arbitrarily and run SMC with target $p_{\theta(0)}(x_{1:T}|y_{1:T})$, sample $X_{1:T}(0)$ from SMC with marginal likelihood $\hat{p}_{\theta(0)}(y_{1:T})$

- Iteration i=1:many
  sample $\theta^* \sim q\{\cdot|\theta(i-1)\}$
  run SMC with target $p_{\theta^*}(x_{1:T}|y_{1:T})$, sample $X^*_{1:T}(i)$ and calculate $\hat{p}_{\theta^*}(y_{1:T})$
  with probability
  $$1 \wedge \frac{\hat{p}_{\theta^*}(y_{1:T})p(\theta^*)q\{\theta(i-1)|\theta^*\}}{\hat{p}_{\theta(i-1)}(y_{1:T})p(\theta)q\{\theta^*|\theta(i-1)\}}.$$
  If accepted set current state, parameters and current likelihood to proposed ones, if not stay at old values.

The update leaves the target distribution invariant and the acceptance probability converges to the same as in an idealized algorithm as the number of particles $\to \infty$. As before the proof requires the formulation of the proposal and target density $p(\theta, x_{1:T}|y_{1:T})$ at one iteration on an extended space including all variables involved in the SMC algorithm. Then as in the PIMH case show that the acceptance probability is correct. The proof of convergence is given in an even more general case in [Andrieu and Roberts (2009)]. The authors note that accepted and rejected particles can be used to estimate $\mathbb{E}h(\theta, x_{1:T})$. And it's obvious that getting samples $X^*_{1:T}$ can be delayed until a new chain move is accepted or even skipped if one is only interested in $p(\theta|y_{1:T})$.

3.4 Particle Gibbs sampler

A Gibbs MCMC algorithm which can iteratively sample from $p(\theta|x_{1:T}, y_{1:T})$ and $p_\theta(x_{1:T}|y_{1:T})$ to produce samples from $p(\theta, x_{1:T}|y_{1:T})$ without the need to design a proposal distribution for the parameters is not straightforwardly possbile as SMC samples from $\hat{p}_\theta(x_{1:T}|y_{1:T})$ won't leave $p(\theta, x_{1:T}|y_{1:T})$ invariant. Samples from the parameter space are taken given a path $X_{1:T}$. This path has to be fixed in the following SMC step. The conditional SMC

algorithm has to return an approximation where on every iteration all particles are resampled according to their weights including the fixed particle under the condition that the particles from the fixed path survive.

The conditional SMC update is initialized with a given path of particles with assoaciated lineage, $X_{1:T} = (X_1^{B_1}, X_2^{B_2}, ..., X_{T-1}^{B_{T-1}}, X_T^{B_T})$ where the lineage $B_{1:T}$ is used to get a list of positions a particle came from (see figure 3). $k \in \{1, ..., N\}$ denotes the particle number and $\mathcal{F}(\cdot|\mathbf{W_n})$ is a discrete probability distribution to sample particle numbers according to their normalized weights $\mathbf{W_n} = (W_n^1, ..., W_n^N)$ and gives ancestor indices A_n^k of the particles which get resampled for time $n+1$. The conditional SMC (on $B_1, ..., B_T$):

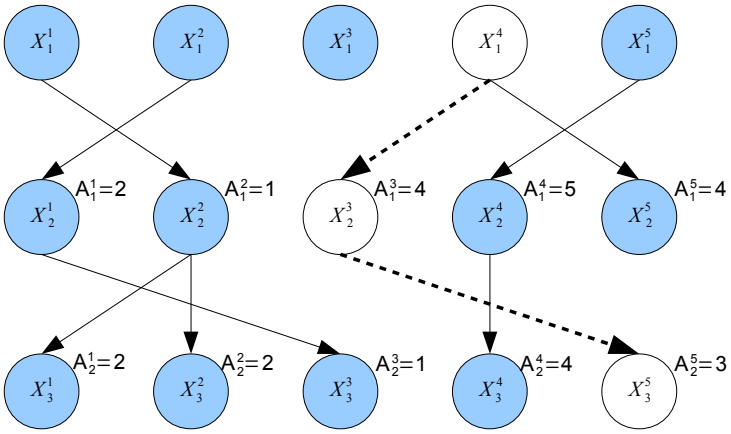

Figure 3: conditional SMC starting with dashed path

- iteration n=1 for $k \neq B_1$ sample $X_1^k \sim q(\cdot|y_1)$ calculate normalized weights $W_1^k \propto w_1(X_1^k)$

- iteration n=2:T
  for $k \neq B_n$ sample $A_{n-1}^k \sim \mathcal{F}(\cdot|\mathbf{W_{n-1}})$
  and for $k \neq B_n$ sample $X_n^k \sim q(\cdot|y_n, X_{n-1}^{A_{n-1}^k})$
  compute $W_n^k \propto w_n(X_{1:n}^k)$

The Particle Gibbs algorithm:

- initialise $\theta(0), X_{1:T}(0), B_{1:T}(0)$ arbitrarily

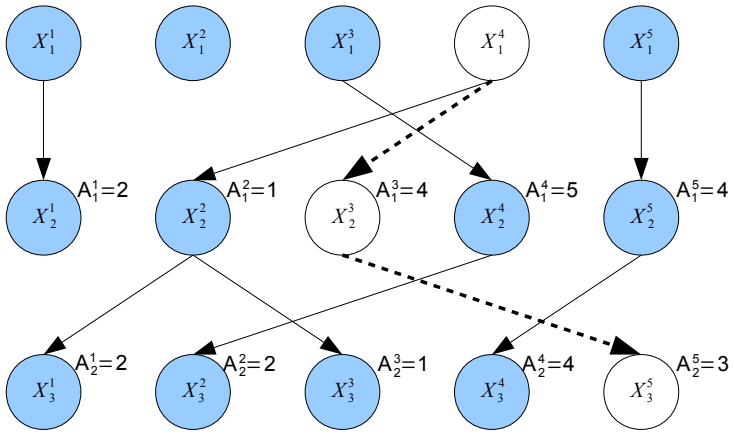

Figure 4: conditional SMC N-1 new paths conditioned on dashed path

- step(i): sample $\theta(i) \sim p\{\cdot|y_{1:T}, X_{1:T}(i-1)\}$
  run conditional SMC with target $p_{\theta(i)}(x_{1:T}|y_{1:T})$ conditioned on $X_{1:T}(i-1), B_{1:T}(i-1)$
  sample $X_{1:T}(i) \sim \hat{p}_{\theta(i)}(\cdot|y_{1:T})$ and remember lineage $B_{1:T}(i)$; i=i+1; goto step(i)

The Particle Gibbs sampler is ergodic and admits $p(\theta, x_{1:T}|y_{1:T})$ as invariant density using the results for the "idealized" Gibbs sampler. Note that the Particle Gibbs not only requires SMC conditioned on a certain path but also the same resampling times. So resampling can't be used dynamically depending on the effective sample size or other measures.

3.5 Proofs for Particle MCMC algorithms

The proofs for both Particle MH methods use analogies to standard MH algorithms and show that the stated acceptance ratios lead to the desired target distributions. The defined extended space relies on various random variables generated by the SMC algorithm. A joint density of $\bar{\mathbf{X}}_\mathbf{1}, ..., \bar{\mathbf{X}}_\mathbf{P}, \mathbf{A_1}, ...\mathbf{A_{P-1}}$ on $\mathcal{X}^{PN} \times \{1, ..., N\}^{(P-1)N}$ where N denotes the particle count and P the time. $\bar{\mathbf{X}}_\mathbf{n} = (X_n^1, ..., X_n^N) \in \mathcal{X}^N$ are the N particles at time $n = 1, ..., P$ and $\mathbf{A}_i$ their ancestors for time $i = 1, ..., P-1$. The joint density is then $\psi(\bar{\mathbf{x}}_\mathbf{1}, ..., \bar{\mathbf{x}}_\mathbf{n}, \mathbf{a_1}, ..., \mathbf{a_{P-1}})$ which is formulated as a product of the initial proposal distribution multiplied by a term expressing how following particles get propagated according to a resampling scheme which relies on the weights $w_1, ..., w_{n-1}$. The point is to get a proposal from the SMC algorithm for the Metropolis-Hastings update the following proposal

distribution can be defined.

$$q^N(k, \bar{\mathbf{x}}_\mathbf{1}, ..., \bar{\mathbf{x}}_\mathbf{P}, \mathbf{a_1}, ..., \mathbf{a_{P-1}}) : = w_P^k \psi(\bar{\mathbf{x}}_\mathbf{1}, ..., \bar{\mathbf{x}}_\mathbf{P}, \mathbf{a_1}, ..., \mathbf{a_{P-1}})$$

Where k is chosen from $\mathcal{F}(\cdot|\mathbf{W_P})$ and extends the space further by $\times 1, ..., N$. Using the proposal distribution q^N to define a target distribution $\tilde{\pi}^N$ which includes $\pi(x_{1:P})$ $(p(x_{1:P}|y_{1:P}))$ as a marginal leads to the ratio:

$$\frac{\hat{Z}}{Z} = \frac{\tilde{\pi}^N(k, \bar{\mathbf{x}}_\mathbf{1}, ..., \bar{\mathbf{x}}_\mathbf{P}, \mathbf{a_1}, ..., \mathbf{a_{P-1}})}{q^N(k, \bar{\mathbf{x}}_\mathbf{1}, ..., \bar{\mathbf{x}}_\mathbf{P}, \mathbf{a_1}, ..., \mathbf{a_{P-1}})}.$$

$\hat{Z}$ denotes the normalizing constant estimated by a particle filter and results from the weights in the ratio and the true constant Z from the true distribution π in $\tilde{\pi}^N$. Comparing the ratio with a standard independent MH algorithm suggests using $\hat{Z}$ to target $\pi(x_{1:P})$ $(p(x_{1:P}|y_{1:P}))$. The PMMH methods is basically the same as the PIMH case including θ and their proposal densities.

The proof for the Particle Gibbs sampler uses:

$$\tilde{\pi}^N(\theta, k, \bar{\mathbf{x}}_\mathbf{1}, ..., \bar{\mathbf{x}}_\mathbf{P}, \mathbf{a_1}, ..., \mathbf{a_{P-1}}) = \frac{1}{N^P} \pi(\theta, x_P^k) \prod_{i \neq i_n^k, n=1,...,P} (\quad SMC|lineage\ i) \tag{8}$$

where i^k is the ancestral lineage of x_p^k and $(SMC|lineage)$ stands for the densitity of the random variables produced by the SMC algorithm conditioned on a certain lineage.

$$\tilde{\pi}^N(\theta, \mathbf{x_P^k}, \mathbf{i^k}) = \frac{1}{N^P} \pi(\theta, x_P^k). \tag{9}$$

In (9) a given path is distributed according to the true distribution then the Particle Gibbs sampler consists of three steps:

- sample parameters conditioned on Path $\mathbf{x_p^k}$ and lineage $\mathbf{i^k}$ according to $\tilde{\pi}^N(\cdot|k, \mathbf{x_P^k}, \mathbf{i^k})$
- sample (new) paths and ancestral lineage according to $\tilde{\pi}^N(\cdot|\theta, k, \mathbf{x_P^k}, \mathbf{i^k})$
- sample k according to $\tilde{\pi}^N(\cdot|\theta, \bar{\mathbf{x}}_\mathbf{1}, ..., \bar{\mathbf{x}}_\mathbf{P}, \mathbf{a_1}, ..., \mathbf{a_{P-1}}) = \mathbf{w_P^k}$

Note that the SMC algorithm in the second step should produce new particle paths conditioned on the old path which from (9) should already be from the true distribution and the new path from the second part of (8). That part could be problematic in the process of finding new "good" paths when they share great portions of the old paths.

One part in Theorem 1 in the PMCMC paper [Andrieu, Doucet and Holenstein (2010)] concerns the SMC proposals total variance distance to the true distribution which can under certain assumption be limited by increasing the particle count.

4 Simulations

4.1 Nonlinear toy example

A often used [Gordon, Salmond and Smith (1993)], [Kitagawa (1996)] benchmark toy example is

$$X_n = \frac{X_{n-1}}{2} + 25\frac{X_{n-1}}{1 + X_{n-1}^2} + 8cos(1.2n) + V_n$$

$$Y_n = \frac{X_n^2}{20} + W_n$$

with $X_1 \overset{iid}{\sim} \mathcal{N}(0, 5)$, $V_n \overset{iid}{\sim} \mathcal{N}(0, \sigma_v^2)$, $W_n \overset{iid}{\sim} \mathcal{N}(0, \sigma_w^2)$. Where inference about the true state $X_{1:T}$ and/or parameters $\theta = (\sigma_v^2, \sigma_w^2)$ is complicated by the multimodal posterior density $p_\theta(x_{1:T}|y_{1:T})$. At first the Particle Independent Metropolis-Hastings method is used to target $p(x_{1:T}|y_{1:T})$ with known parameters. Then the distribution $p(\theta, x_{1:T}|y_{1:T})$ is targeted by the Particle Marginal Metropolis-Hastings and Particle Gibbs. All Particle MCMC methods use the bootstrap filter and resampling at every step. By simply running

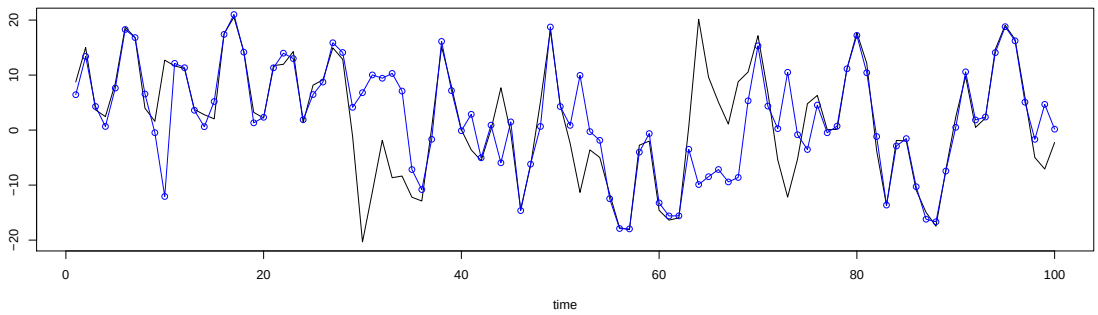

Figure 5: true states x and Bootstrap median estimates marked with points

the boostrap filter on the model with known parameters $\sigma_v^2 = 1, \sigma_w^2 = 10$ one can see in (5) that the median estimates (with points) for the filtering distribution $p(x_n|y_{1:T})$ are sometimes at the opposite side. This makes clear that the filter isn't perfectly estimating the true states even with known true parameters because at some time steps more than

50% of the particles are at the wrong side of the multimodal distribution meaning the sign of the state variable x_n wasn't correctly determined (using 2.5%, 50% and 97.5% quantiles like in ([Johannes and Polson (2006)])).

4.1.1 Particle Independent Metropolis Hastings

Two cases are simulated with $T = 100$, $\sigma_v^2 = 10$, $\sigma_w^2 = 1$ respectively $\sigma_v^2 = 10$, $\sigma_w^2 = 10$. Acceptance rates for the PIMH after 5000 iterations are printed in table 1. When σ_w^2 is lower (and the observations are more precise) than σ_v^2 the SMC algorithm samples particles from the prior using the high σ_v^2 which leads to samples further away from the likelihood resulting in a low acceptance rate. Clearly increasing the particle count leads to higher acceptance rates. This should be expected as the simulated estimates in the PIMH acceptance ratio converge to their true values and the ratio therefore to 1. Accounting for the computational complexity PIMH can be used where a relatively small particle size N results in a fast mixing Markov chain. So cases with low acceptance rates still can be useful if the MCMC iterations on the other hand can be increased. Considering that the PMMH ratio includes the same ratio as in the PIMH setup one could expect that the acceptance rates also depend on how "precise" or on how low the variance of the simulated loglikelihoods is.

σ_v^2, σ_w^2	N=100	N=200	N=500	N=1000	N=3000
10,1	0,05	0,23	0,41	0,55	0,75
10,10	0,33	0,49	0,66	0,75	0,86

Table 1: PIMH acceptance rates

Comparing for the model $\sigma_v^2 = 10, \sigma_w^2 = 1$ the estimated states for different particles counts in (6) with 200 and in (7) with 6000 particles. No significant improvement can be seen. As expected with increased N-particles the accepted states increased but the median, 2.5% and 97.5% quantiles didn't change much.

4.1.2 Particle Marginal Metropolis Hastings

For a PMMH run with 50.000 iterations and 6000 particles figure (8) shows traceplots, acfs and histograms for the parameters (σ_v^2, σ_w^2). Clearly the acf drops reasonably fast and the traceplots (with burn-in of 5000 iterations) doesn't show any sign of not mixing well. From the histograms one could conclude that the PMMH does estimate the posterior distribution of the lower right parameter for the observation quite well and is not

completely far away from the true value for the variance in the state process. But the prior in the PMMH ratio seems to have great effect. Choosing a more informative prior did result in more precise histograms but still a loose InverseGamma 0.01 prior was quite good.

4.1.3 Particle Gibbs

The implementation used as conditional SMC the basic bootstrap filter and parameter sampling as in a standard Gibbs sampler. For better results a PIMH step was used: the likelihood of two conditional SMC runs conditioned on the same path were computed and accepted with $1 \wedge \frac{p_\theta^{**}(y)}{p_\theta^{*}(y)}$. The result was slightly better with the MH step as can be seen in (9),(10). Parameters $(\sigma_v^2, \sigma_w^2) = (\tau^2, \sigma^2)$ with $\nu_1 = \nu_0 + n, n_1 = n_0 + n$ got sampled with inverse-gamma prior according to

$$(\tau^2|x_{0:n}) \sim InverseGamma(\nu_1/2, \nu_1\tau_1^2/2)$$

$$(\sigma^2|y_{1:n}, x_{1:n}) \sim InverseGamma(n_1/2, n_1\sigma_1^2/2)$$

$$Z = \begin{pmatrix} x_0 \cdots x_{n-1} \\ \frac{x_0}{1+x_0^2} \cdots \frac{x_{n-1}}{1+x_{n-1}^2} \\ cos(0) \cdots cos(n-1) \end{pmatrix}$$

$$\nu_1\tau_1^2 = \nu_0\tau_0^2 + (x - Z'(0.5\ 25\ 8)')'(x - Z'(0.5\ 25\ 8)')$$

$$n_1\sigma_1^2 = n_0\sigma_0^2 + \sum_{t=1}^{N}(y_t - \frac{x_t^2}{20})^2$$

The particle Gibbs depends on a good path used in the conditional SMC update. The simulation runs starting with wrong parameters needed appropriately more particles to increase the chance of finding a good path. Then new parameters are sampled nearly from correct states and also the subsequent conditional SMCs sample new almost correct states conditioned on the old almost correct states.

The effect of using to less particles can be seen in figures (11), (12) and (13). False parameters were inserted from iteration 2000 to 3000. The sampled parameters drop in steps towards the true parameters in the cases with 200 and 300 observations somehow fast but for 350 observations it took the 250 particles 72000 iterations to find a new particle path close to the true states (13). In this case it was sufficient to use more than 500 particles to find the new path reasonably fast, with 500 particles see (14) and even faster with N700 particles in (16). Using a MH step comparing the loglikelihood of

two conditional SMC calls with 500 particles did result in a faster drop but doubled the runtime (15).

Aside from the problem of determining the right particle size the Particle Gibbs algorithm in this model gives good results. And in contrast to the PMMH it is nearly independent of a defined prior, needs no RW variance adjustment for a good acceptance rate and gives parameter samples at every iteration. Comparing PG to PMMH using the cosinus toy model ($\sigma_v^2 = 10$, $\sigma_w^2 = 10$) with nearly the same computationally cost (the PG has to backtrack the particle paths so particle size in the PMMH is doubled), the figure for PG (17) shows better results than the figure for PMMH (18).

4.2 PMMH for basic Stochastic Volatility model

The SV model used is of the simple form:

$$y_t = e^{h_t/2}\varepsilon_t$$
$$h_t = \mu + \phi h_{t-1} + \tau\eta_t$$

with $\varepsilon_t, \eta_t \sim \mathcal{N}(0,1), \mathbb{E}(\varepsilon_t\eta_{t+h}) = 0$ for all h and $\mathbb{E}(\varepsilon_t\varepsilon_{t+l}) = \mathbb{E}(\eta_t\eta_{t+l}) = 0$ for all $l \neq 0$.

In the PMMH algorithm Random-Walk proposals for the parameters with small variances were used. In most simulation runs the particle size had to be increased with increased observations for the simulated loglikelihood's variance to decline (to increase the acceptance rate). Noticeable are the auto correlation functions which didn't decline to zero and even got negative in every simulation considered. The variables are proposed blockwise as proposing individually didn't make a remarkable difference and probably would just increase the MCMC iteration steps needed to get the same Markov chain quality. In the appendix graphics for two slightly different simulations are shown. In the first case (19), the particle size and iteration steps were appropriate and in the second case (20) the particle size wasn't linearly adjusted to the increased observations. Altough the traceplots and acfs look acceptable only increasing the MCMC iterations didn't give similiar good results.

The influence of different priors on σ^2 can be seen in figures (21)-(26) (prior for μ was $N(0,1)$, uniform on ϕ). Especially a InverseGamma prior with mean 0.36 had a very negative effect in a model with variance parameter 0.04 (23) but the prior which is very spiky at 0.04 is also not too bad for a model with parameter 0.36. Using the "loose"

InverseGamma 0.01 prior like in the PMCMC paper with tuned RW proposals for faster dropping acfs was the best choice for different parameter values.

5 Conclusion

The methods proposed in the paper "Particle Markov Chain methods" from Andrieu, Doucet and Holenstein (2010) are a novel way of combining SMC and MCMC methods. Altough others used similiar algorithms they provided a theoretically sound formulation and proofs. Despite being computationally very demanding PMCMC methods give a simple way for inference in highdimensional and non-Gaussian distributions. Complex examples e.g. in the PhD thesis by [Holenstein (2009)] suggest that these methods provide valid results in many cases. But even the authors used only 500 to 1000 observations in the Levy-driven SV model so parallel programming with CUDA as in [Lee, Yau, Giles, Doucet and Holmes (2009)] seems inevitable in even more complex problems given the speed-ups of more than hundred-folds for the SMC implementations on standard graphics hardware. PMCMC could then at least be used to compare results to other probably more specific solutions in a more reasonably time because in advance it is not clear in which cases and with which particle size and iteration count the PMCMC methods are better than other methods. First modifications already mentionend in the paper could be further examined, combining PG and PMMH steps, using better particle filters or how different resampling schemes in the SMC part effect the results especially for the Particle Gibbs sampler.

A Appendix

Simulation graphics and R code used to generate them including the simple bootstrap filter with calculation of $p_\theta(y)$, random walk PMMH, the condtional SMC and state-path-backtracking used in the Particle Gibbs sampler. See figures (27),(28) and (29). Fast demo for PG cosinus model: http://www.harinko.de/PGdemo.R and for PMMH SV model: http://www.harinko.de/SVOLfastdemo.R .

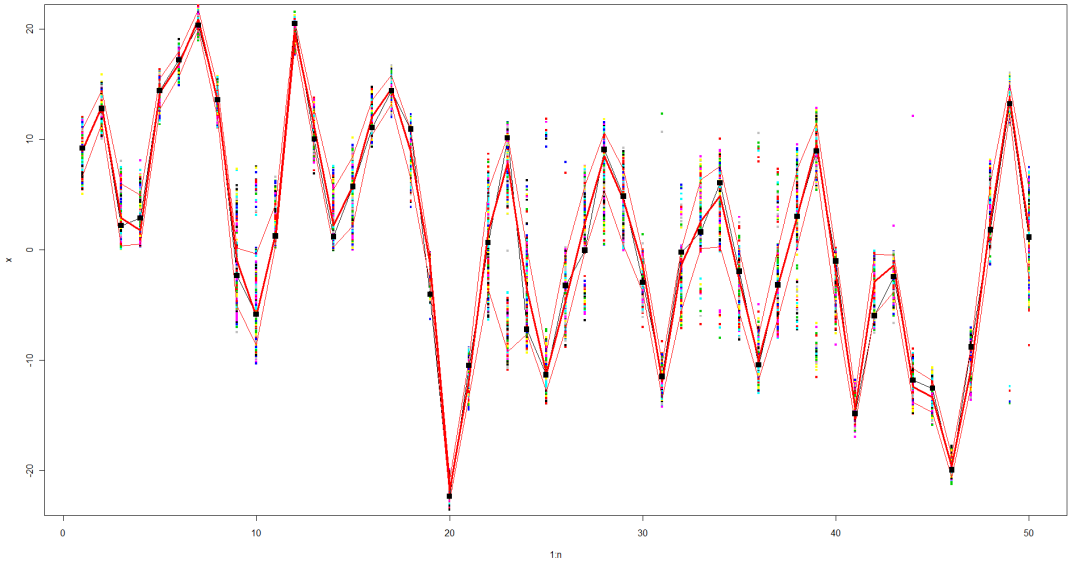

Figure 6: PIMH N=200 particles, n=50 observations, 10000 iterations

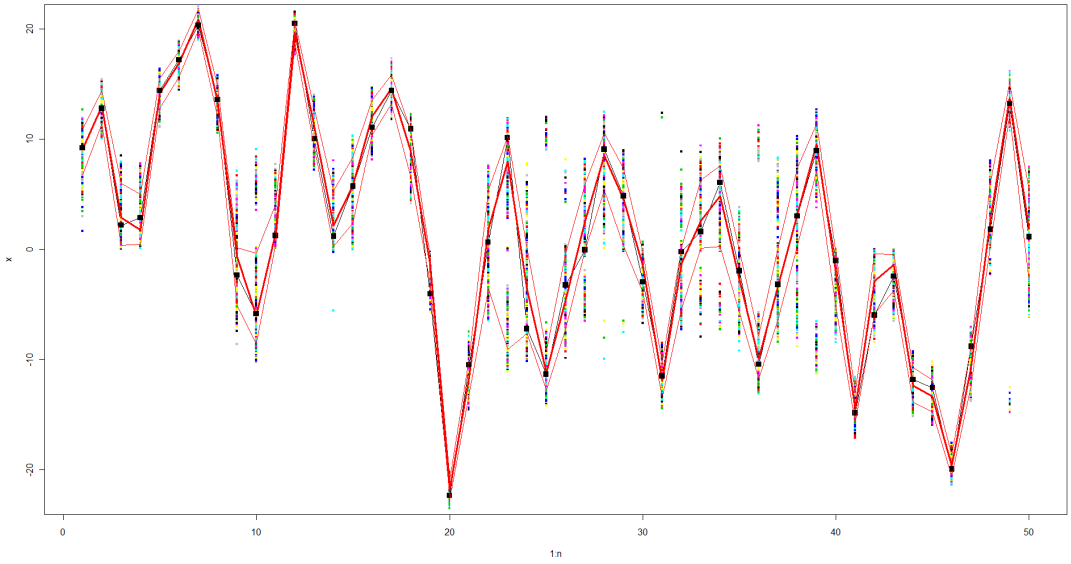

Figure 7: PIMH N=6000 particles, n=50 observations, 10000 iterations

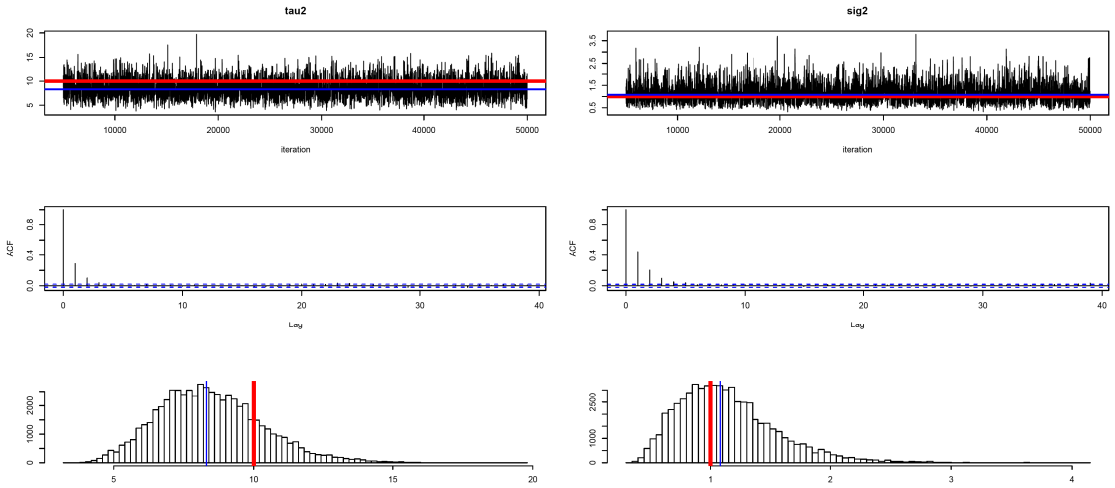

Figure 8: PMMH nonlinear model N=6000 particles 50000 PMMH iterations for model $\sigma_v^2 = 10$ $\sigma_w^2 = 1$

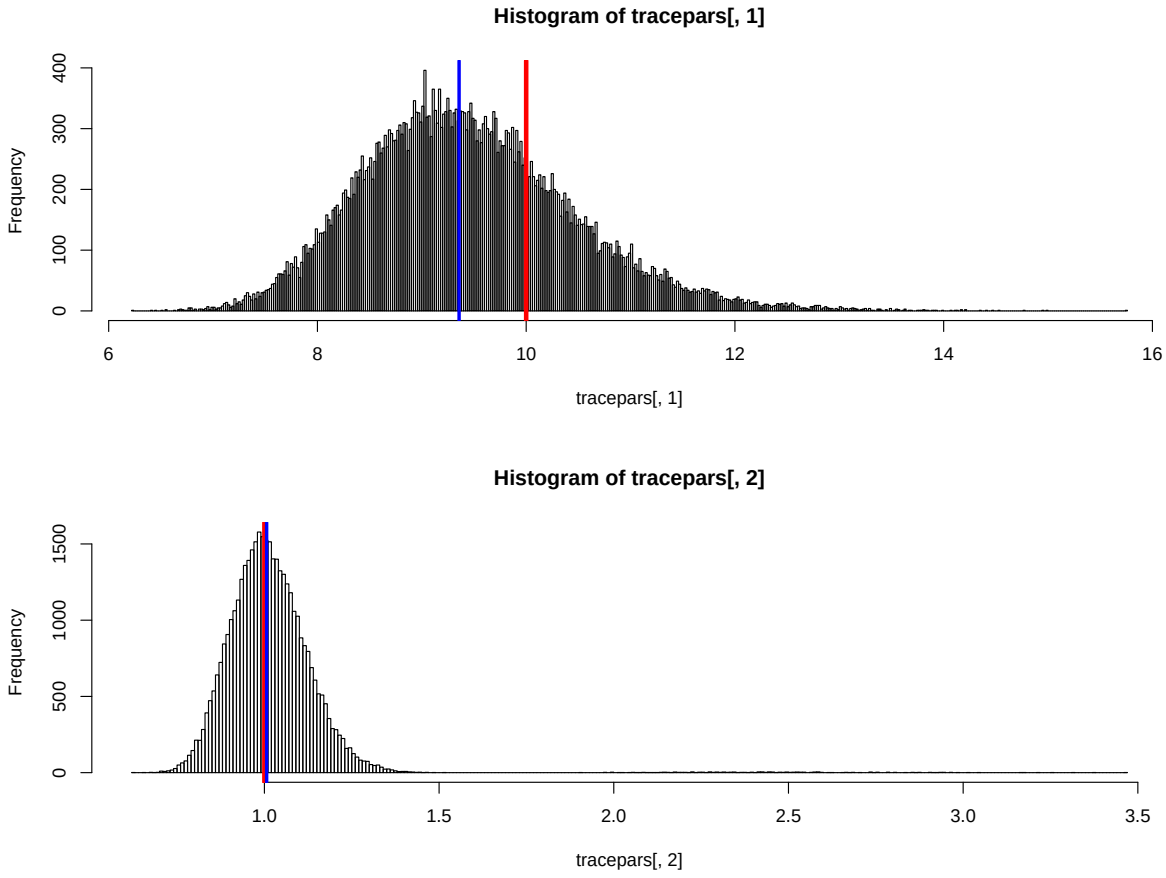

Figure 9: Particle Gibbs with MH step for nonlinear model, 40.000 iterations

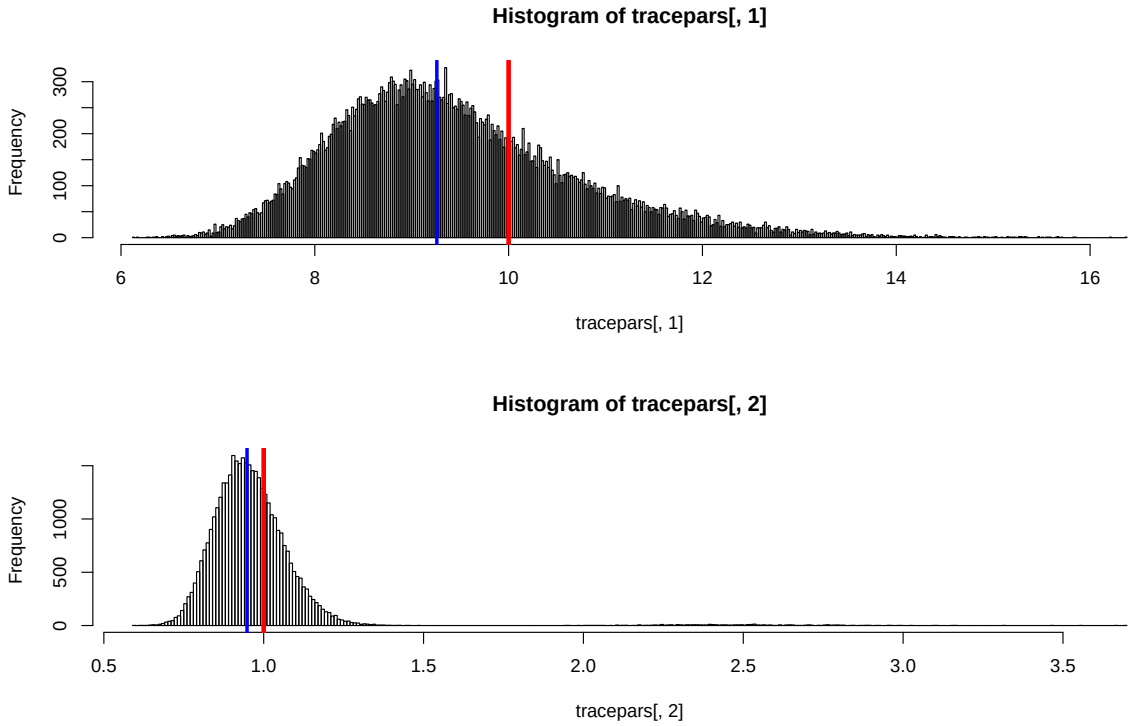

Figure 10: Particle Gibbs for nonlinear model, 40.000 iterations

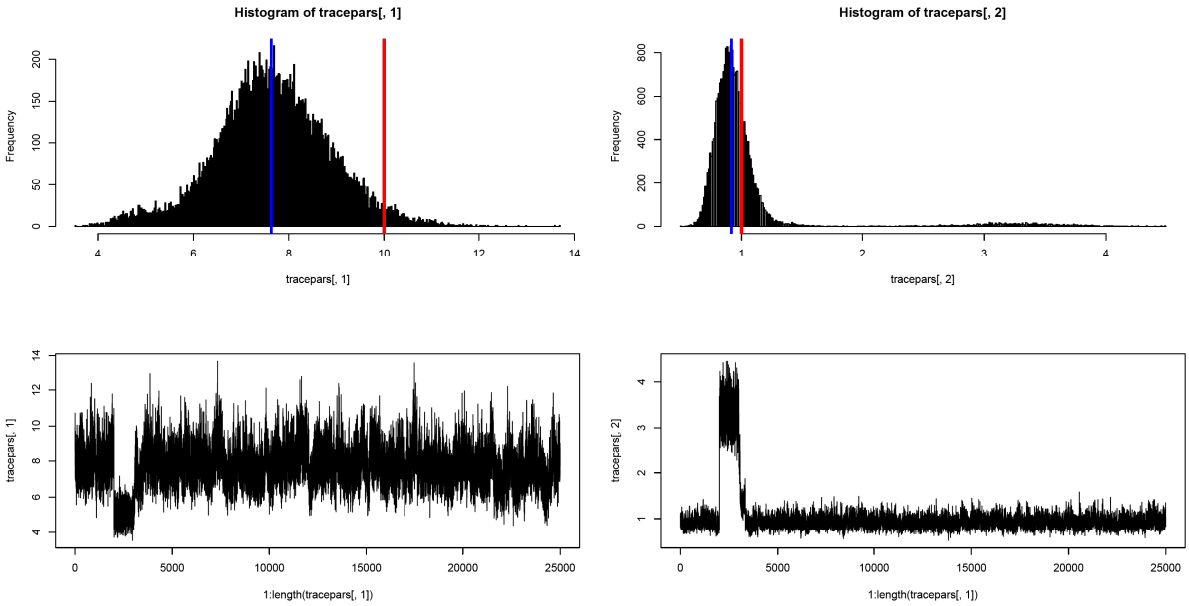

Figure 11: PG toy model N250 particles n250 observations

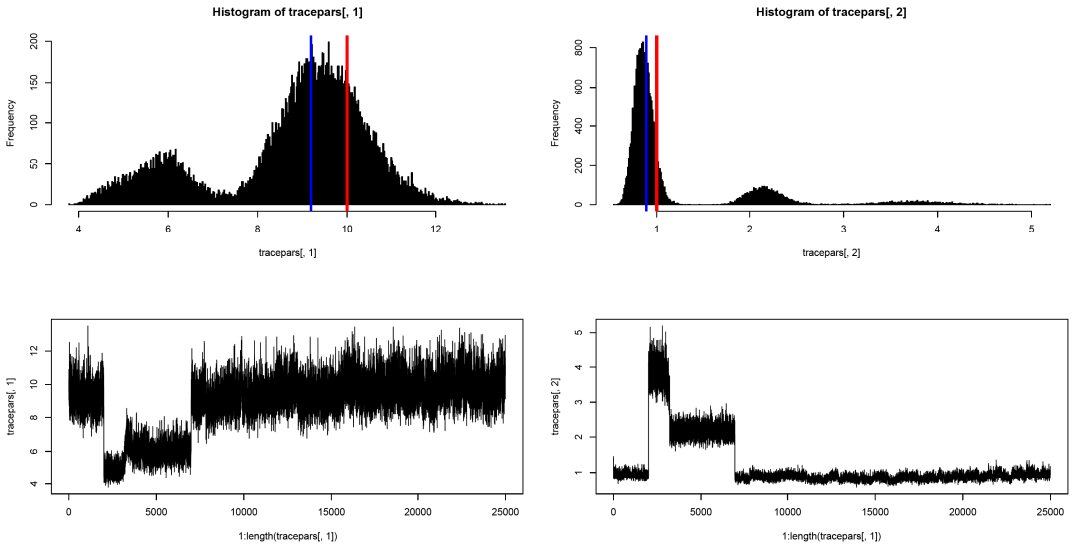

Figure 12: PG toy model N250 particles n300 observations

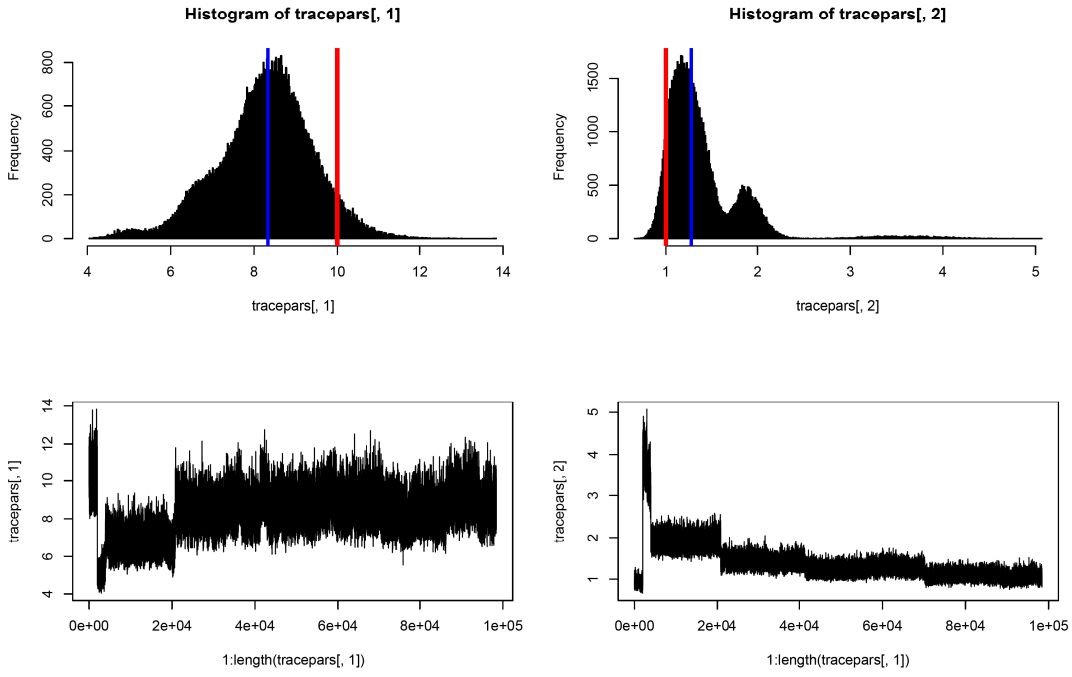

Figure 13: PG toy model N250 particles n350 observations 100000 mcm iterations

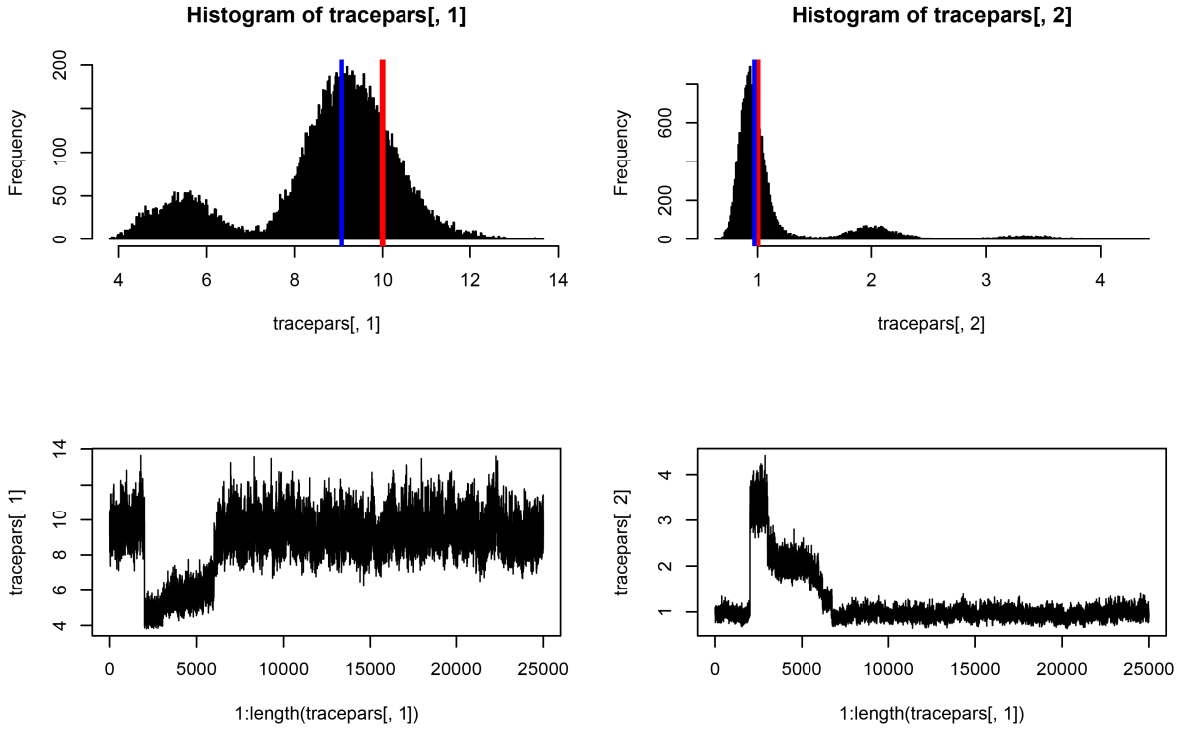

Figure 14: PG toy model N500 particles n350 observations

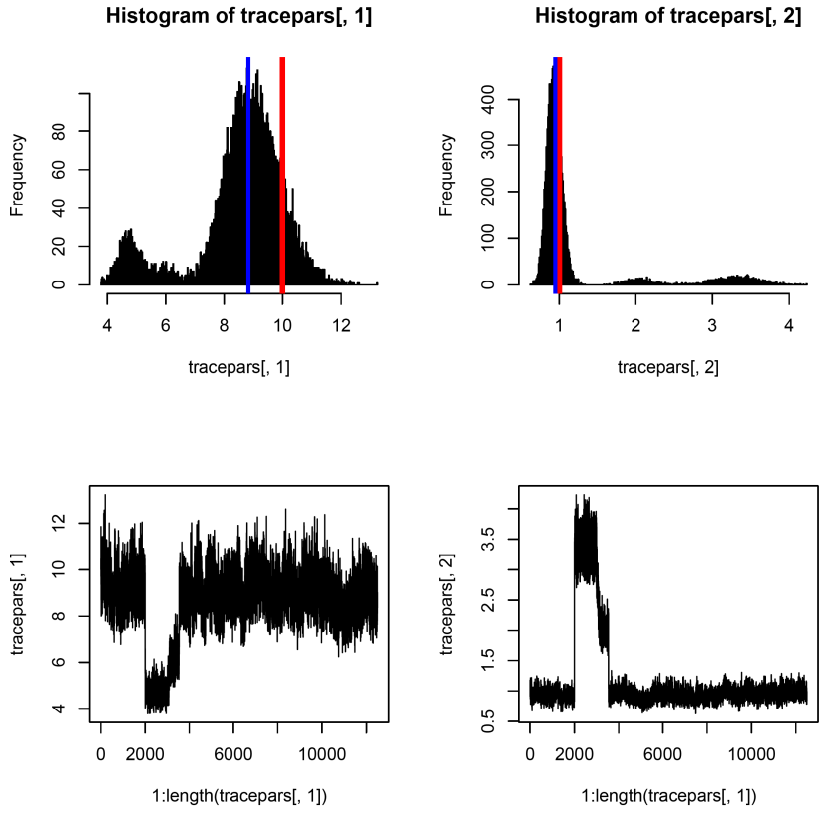

Figure 15: PG with MH toy model N500 particles n350 observations

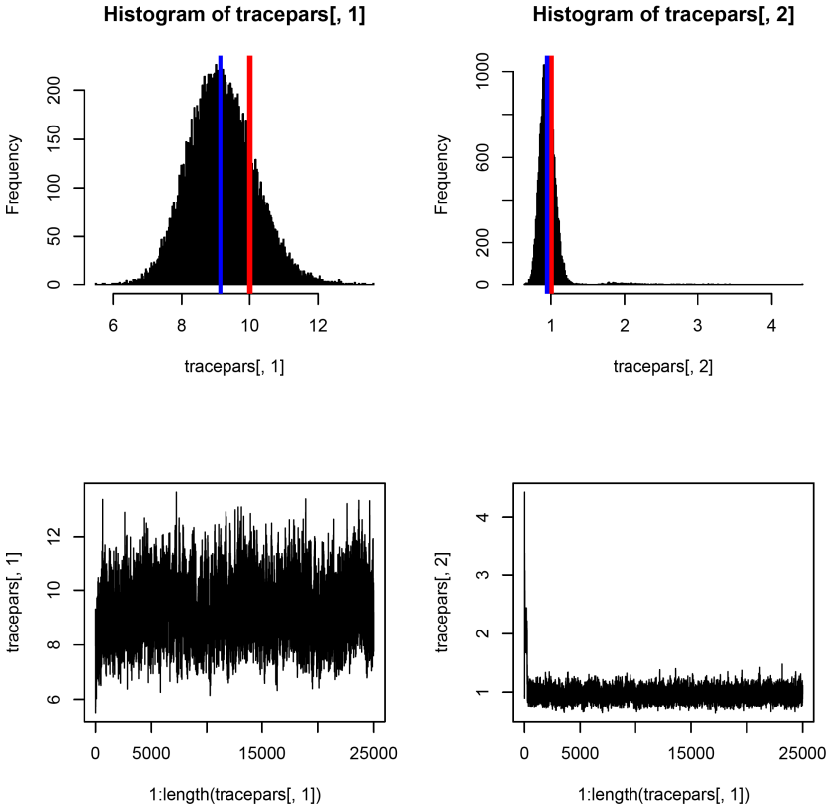

Figure 16: PG toy model N700 particles n350 observations

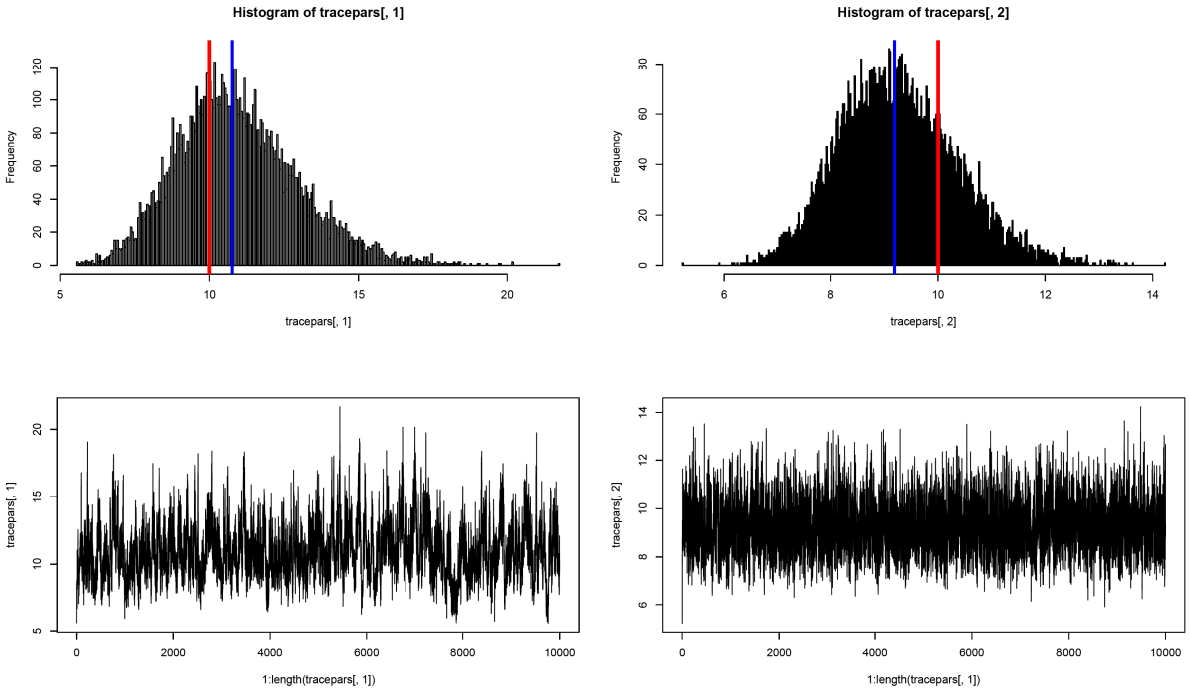

Figure 17: PG for toy model (10,10) 10000 iterations

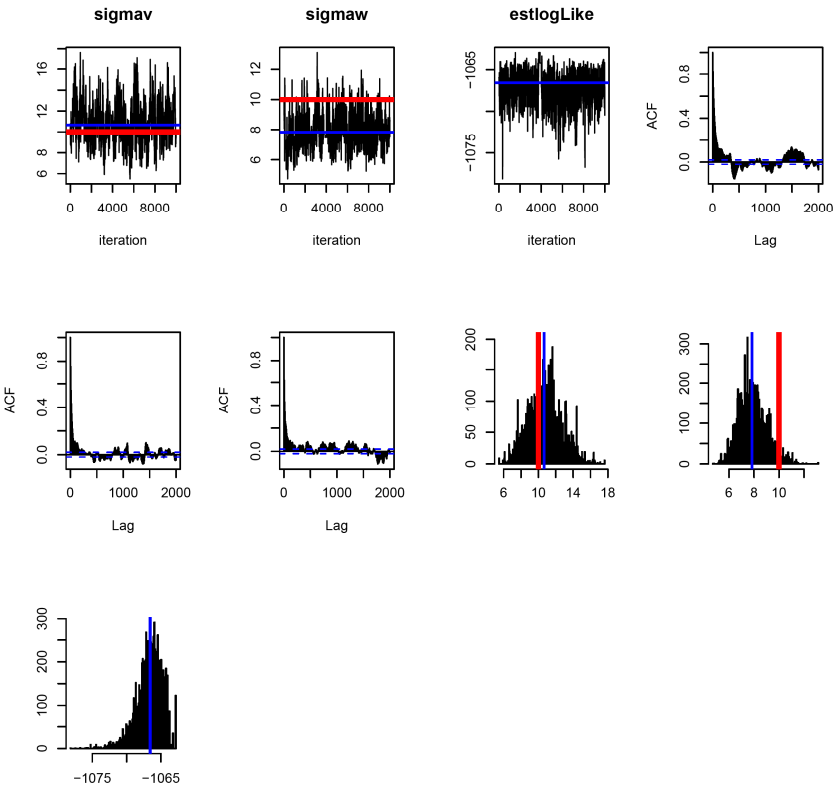

Figure 18: PMMH for toy model (10,10) 10000 iterations

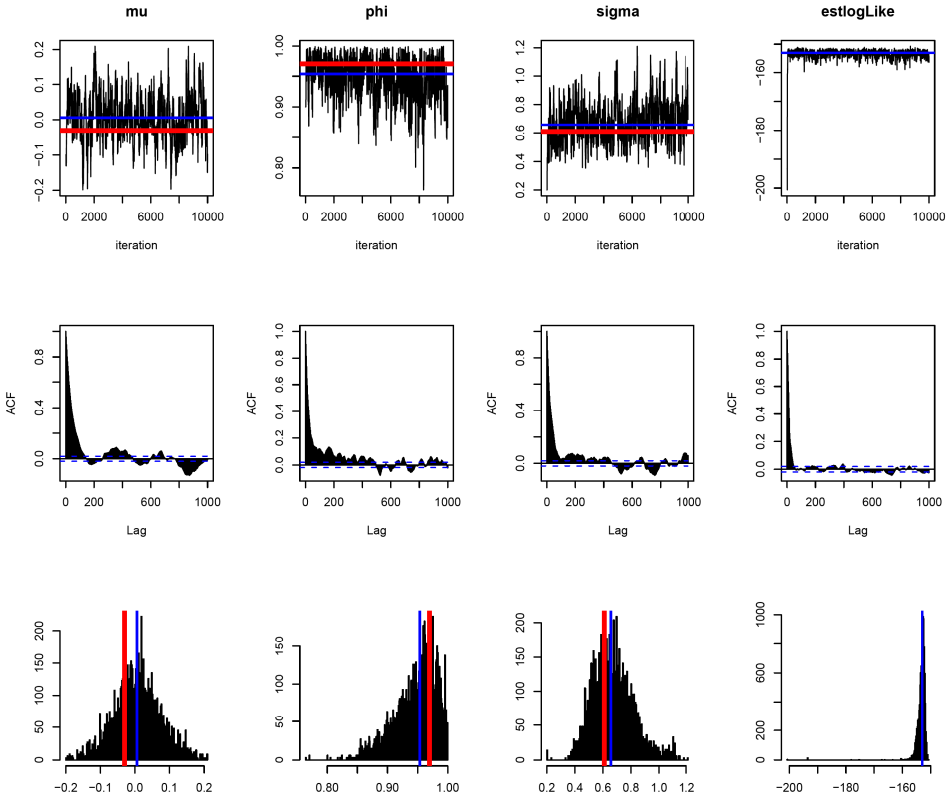

Figure 19: PMMH SV model N=1000 particles n=100 observations 10000 iterations

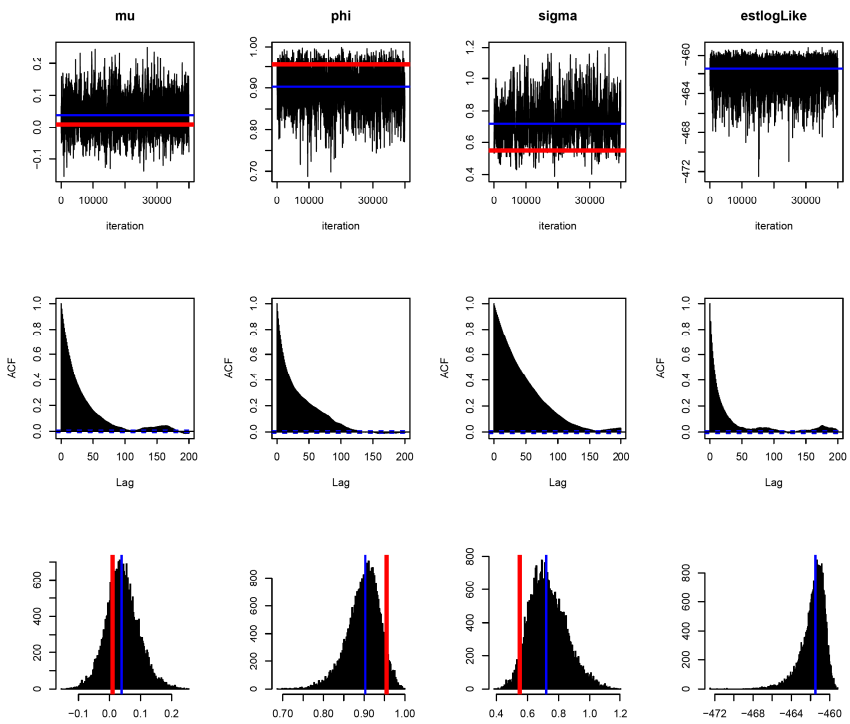

Figure 20: PMMH SV model N=1500 particles 250 observations 40000 mcmc iterations

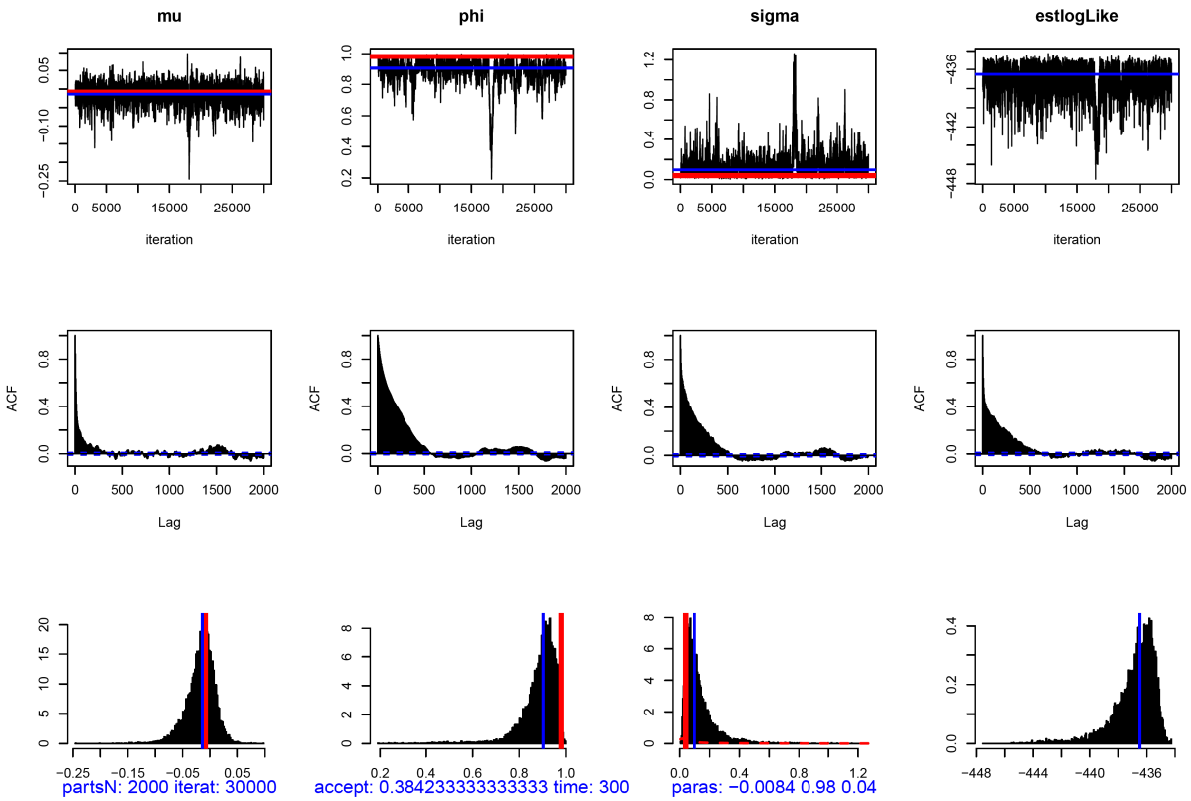

Figure 21: SVOL $\sigma^2 = 0.04$, loose prior

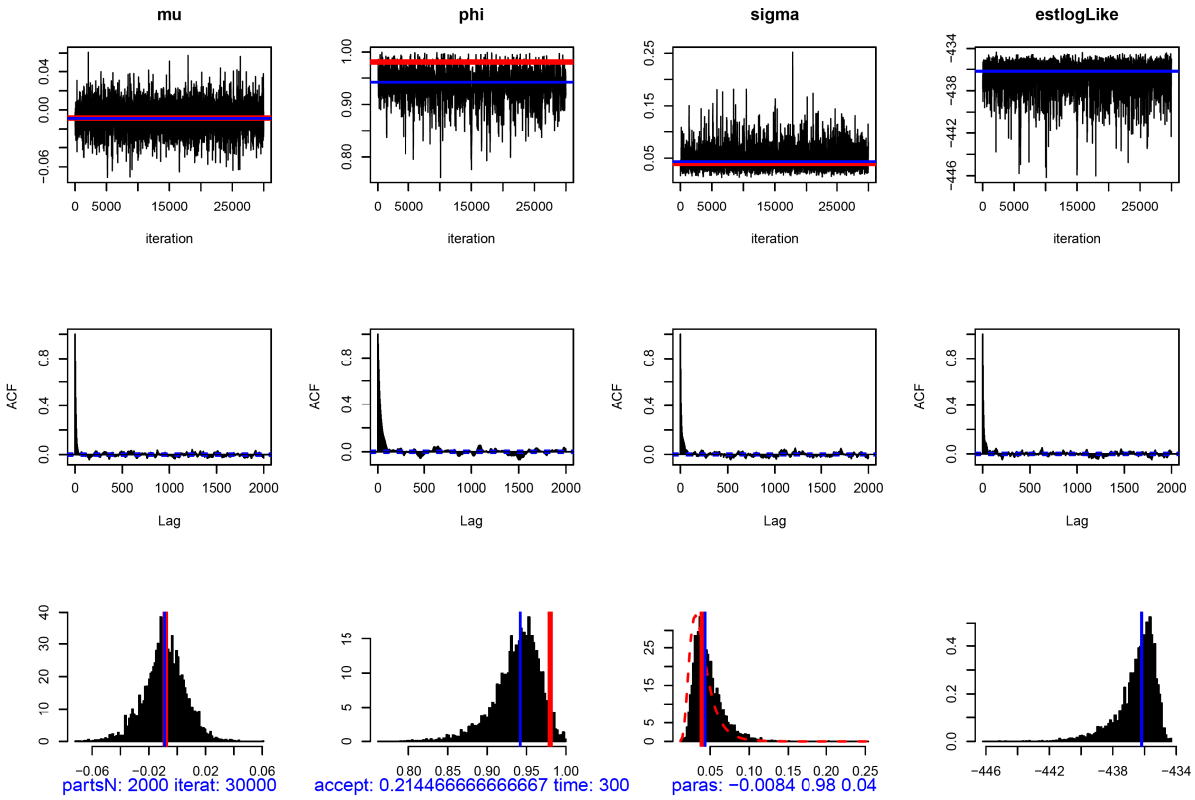

Figure 22: SVOL $\sigma^2 = 0.04$, mean 0.04 prior

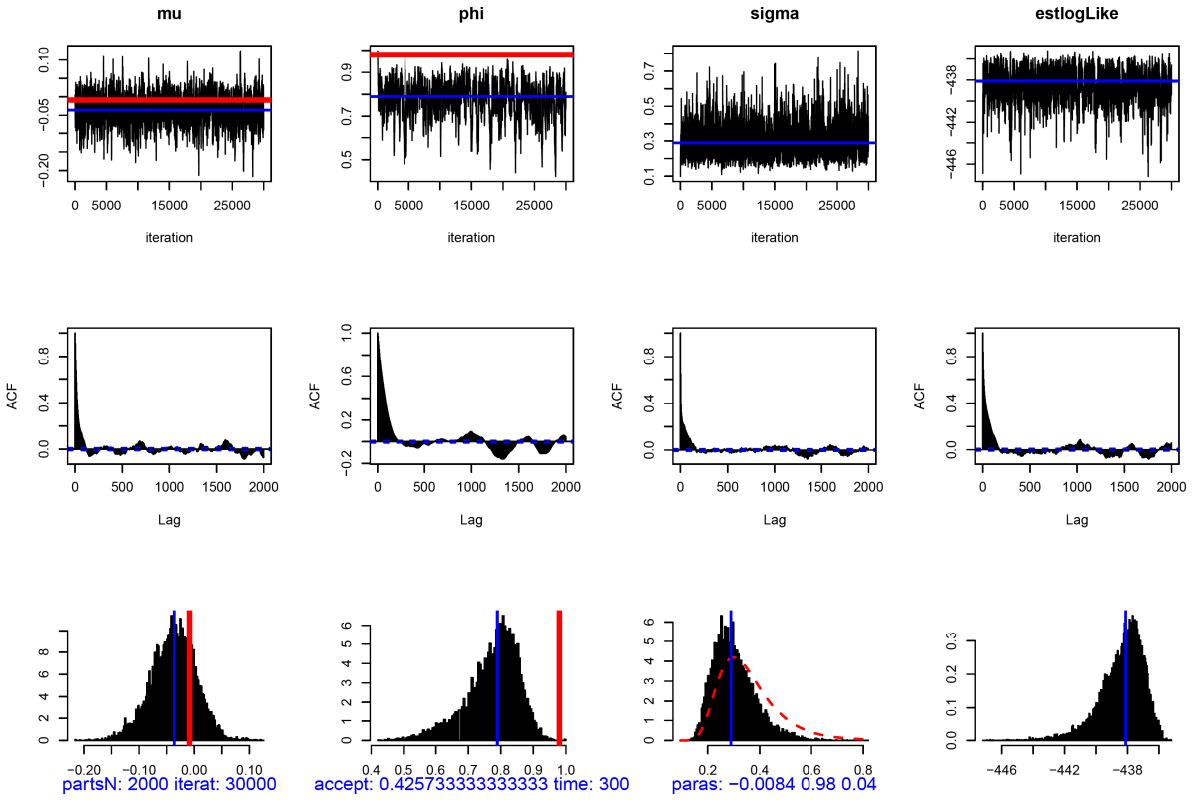

Figure 23: SVOL $\sigma^2 = 0.04$, mean 0.36 prior

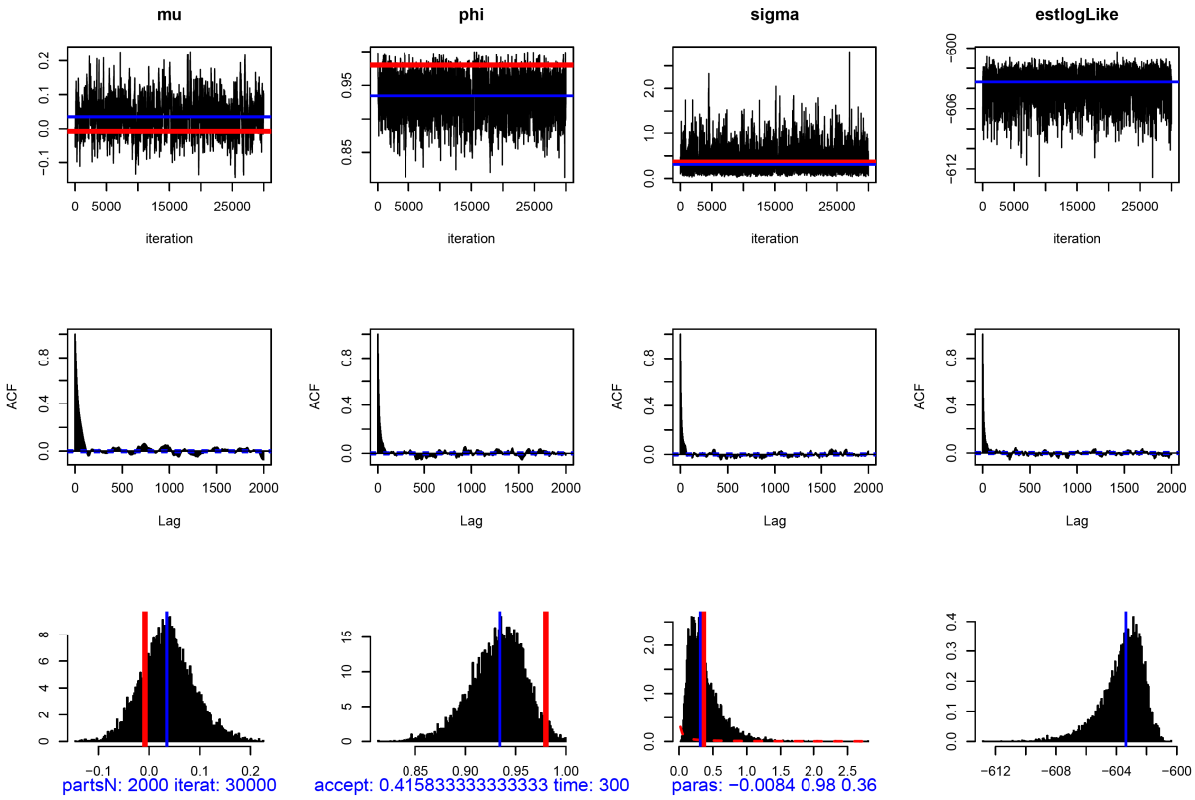

Figure 24: SVOL $\sigma^2 = 0.36$, loose prior

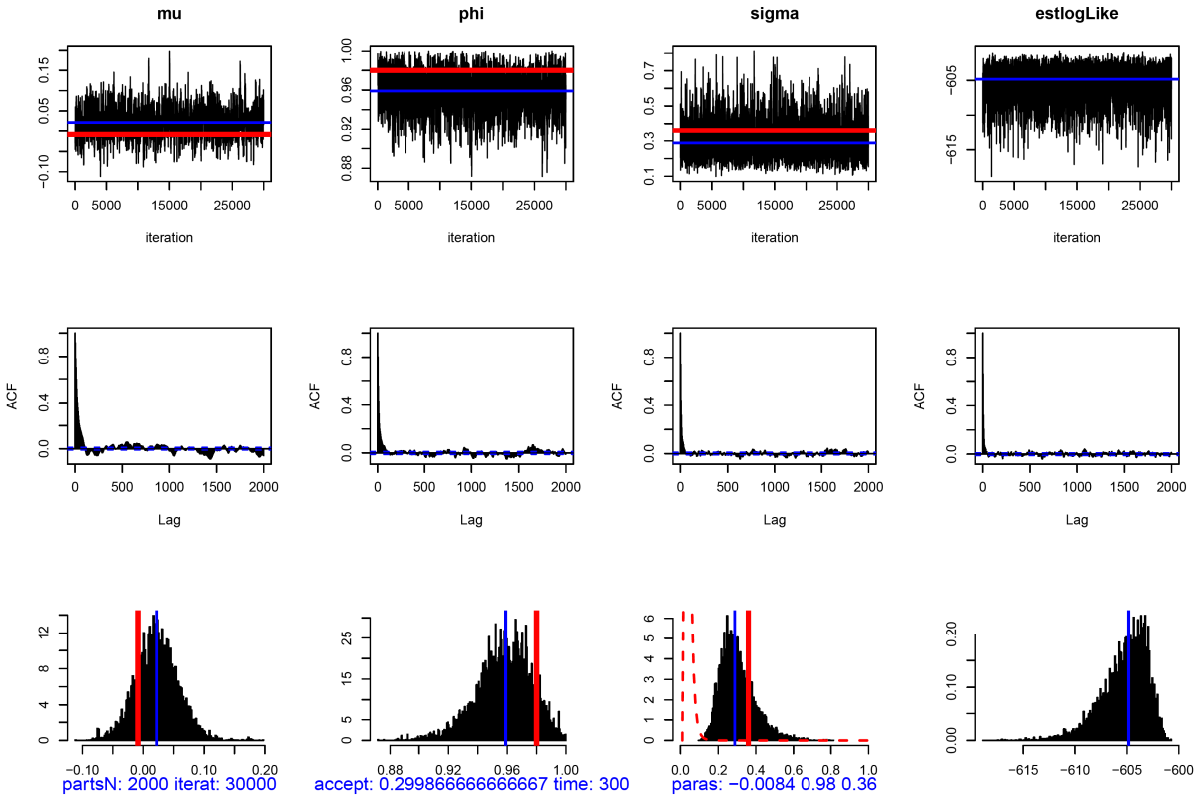

Figure 25: SVOL $\sigma^2 = 0.36$, mean 0.04 prior

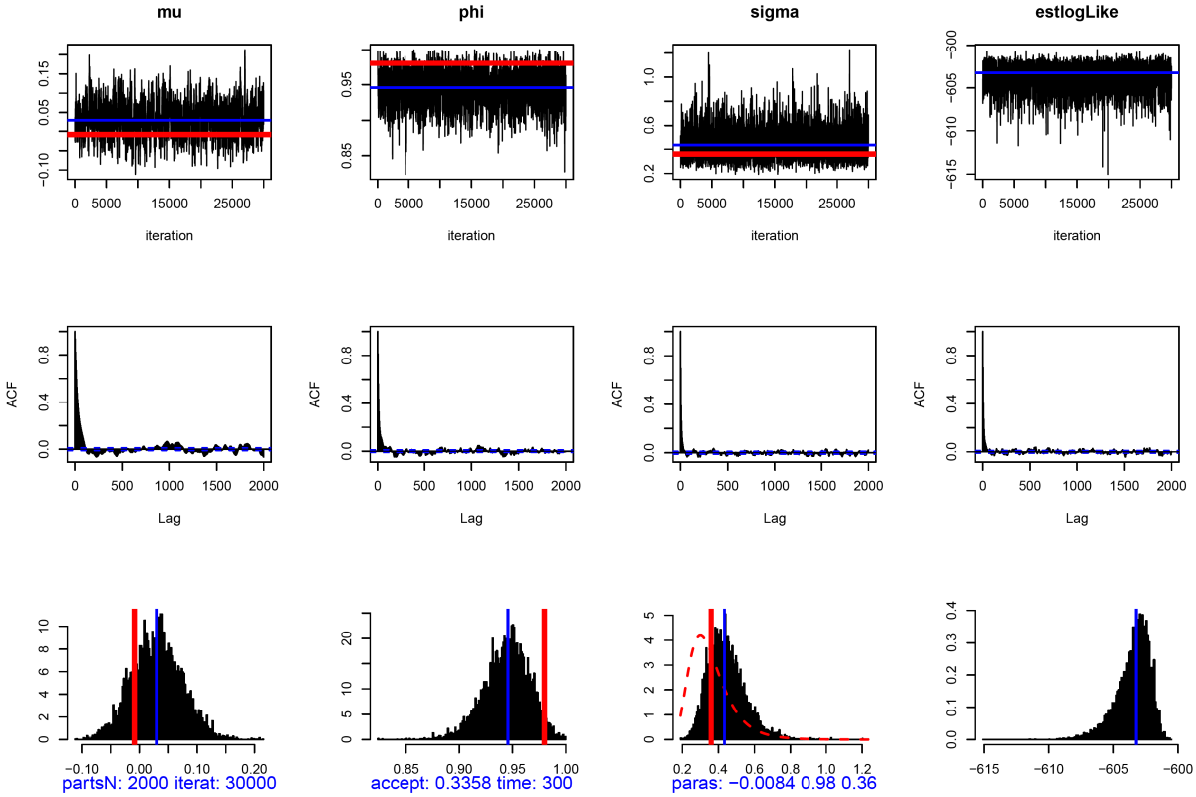

Figure 26: SVOL $\sigma^2 = 0.36$, mean 0.36 prior

```
rm(list=ls())
library(pscl)
# generate simple stochastic volatlity model
# y=exp(h_t/2)eps
# h_t=mu+phi*h_{t-1}+tau*eps2 with eps,eps2~N(0,1)
set.seed(45678)
n     = 50
alpha=-0.01
beta  =  0.96
tau=0.33
ptr=c(alpha,beta,tau,0)
y     = rep(0,n)
x     = rep(0,n)
x[1] = rnorm(alpha,tau/sqrt(1-beta^2) )
y[1] = rnorm(1,0,exp(x[1]/2))
for (t in 2:n){
  x[t] = rnorm(1,alpha+beta*(x[t-1]),tau)
  y[t]= rnorm(1,0,1)*exp(x[t]/2)
}

# particle count
N=1200
sisr=function(mu,phi,sigma){
x=rnorm(N,0,1.2)
ytot=0; meds=NULL
for(t in 1:n){
 # transition particles
 x1 = mu + phi*(x) + rnorm(N,0,sigma)
 # weights are the likelihood observing the proposed particles
 w  = dnorm(y[t],0,exp(x/2),log=F)
 # alway resample according to weights
 x  = sample(x1,size=N,replace=T,prob=w)
 # calculate simulated loglikelihood p(y)
 ytot=ytot+log(mean(w))
 }
ret=list(1,ytot)
return(ret)
}

# Random walk proposols for PMMH step
propose=function(a,b,c){
 # check after proposal if PHI is element of (-1,1)
 v1=0.04; v2=0.04; v3=0.07
c(rnorm(1,a,v1), rnorm(1,b,v2), exp(rnorm(1,log(c),v3)) )
}

# density of prior distribution
dprior=function(a,c){
# NormalInverseGamma, spiky prior
c( log(dnorm(a,0,2)*densigamma(c^2,3,1)) )
}

```

Figure 27: rPMMH part1

```
#initialise with true values for step 0
out=sisr(0.1,0.9,0.4);loglikold=out[[2]]
muold=0.1;phiold=0.9;sigmaold=0.4;tracepars=NULL;nsteps=1;k=1
niter=50000
# for iterations 1:50000 PMMH
for ( i in 1:niter ){
if(i == k*100){
 cat(k*100,Sys.time(),nsteps,"\n");k=k+1
}
 parsnew=propose(muold,phiold,sigmaold)
 if ( (parsnew[[2]] < -1) || (parsnew[[2]] > 1) ){
 phinew=phiold
 }
 else { phinew=parsnew[[2]];  }
 munew=parsnew[[1]]; sigmanew=parsnew[[3]]
# new SMC using new parameters
 out=sisr(munew,phinew,sigmanew)
 logliknew=out[[2]]
 num=logliknew + dprior(munew,sigmanew) + log(sigmanew/sigmaold)
 den=loglikold + dprior(muold,sigmaold)
 if ( log(runif(1)) < (num-den) ){
  #accepted
  nsteps=nsteps+1
  tracepars=rbind(tracepars,c(munew,phinew,sigmanew,logliknew))
  muold=munew; phiold=phinew; sigmaold=sigmanew
  loglikold=logliknew
 }else{
 tracepars=rbind(tracepars,c(muold,phiold,sigmaold,loglikold))
 }
}
```

Figure 28: rPMMH part2

```
1
2  # conditional SMC using XVECT as frozen path
3  sisrCOND=function(mu,phi,sigma,XVECT){
4  x=rnorm(N,0,1.1); INDs=NULL; xs=NULL
5  for(t in 1:n){
6   x1 = mu + phi*(x) + rnorm(N,0,sigma)
7   x1[N]= XVECT[t]
8   w  = dnorm(y[t],0,exp(x1/2))
9   xIND  = sample(1:N,size=N-1,replace=T,prob=w)
10  xIND[N]=N
11  x=x1[xIND]
12  INDs=rbind(INDs,xIND)
13  xs=rbind(xs,x1)
14  }
15 ret=list(1,INDs,xs,1,w)
16 return(ret)
17 }
18
19 # get new state PATH from output=sisrCOND(mu,sigma,XVECT)
20 # return vector XV containing X_{1:T}
21 xvector=NULL
22 # select "start-index" according to weights at last stage
23 index=sample(1:N,size=1,replace=T,prob=output[[5]])
24 for(t in n:2){
25     # current particles at time t
26     xs_current=out[[3]][t,]
27     # current index vector
28     INDS_current=out[[2]][(t-1),]
29     # xs predecessor
30     xvector=rbind(xvector,xs_current[index])
31     # new index to look in the index-vector in the next iteration t-1
32     index=INDS_current[index]
33 }
34 xvector=rbind(xvector,out[[3]][1,index])
35 XV=rev(xvector)
36
```

Figure 29: conditional SMC with backtracking

References

[An and Schorfheide (2007)] *Bayesian analysis of DSGE models*. Econometric Reviews 26(2-4), 113-172.

[Andrieu, Doucet and Holenstein (2010)] *Particle Markov Chain Monte Carlo methods*. J. Royal Statistical Soc. B. 72, Part3, pp-289-342.

[Andrieu and Roberts (2009)] *The pseudo-marginal approach for efficient computation*. Ann. Statist., 37, 697-725.

[Cappe, Godsill and Moulines (2007)] *An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo*. Proceedings of the IEEE, Volume 95(5), 899-924.

[Carpenter, Clifford and Fearnhead (1997)] *An improved particle filter for nonlinear problems*. Technical Report, University of Oxford, Dept. of Statistics.

[Chib, Nardari and Shephard (2002)] *Markov chain Monte Carlo methods for stochastic volatility models*. Journal of Econometrics 108(2), 281-316.

[Doucet, de Freitas and Gordon (2001)] *Sequential Monte Carlo Methods in Practice*. Springer 2001.

[Doucet, Godsill and Andrieu (2000)] *On sequential Monte Carlo sampling methods for Bayesian filtering*. Statistics and Computing 10(3), 197-208 (2000).

[Doucet and Johansen (2009)] *A Tutorial on Particle Filtering and Smoothing: fifteen years later*. In Handbook of Nonlinear Filtering (eds. Crisan and Rozovsky), Cambridge University Press.

[Gordon, Salmond and Smith (1993)] *Novel approach to nonlinear non-Gausian Bayesian state estimateion*. IEE Proc. F, 40, 107-113.

[Higuchi (1997)] *Monte Carlo filtering using the genetic algorithm operators*. Journal of Statistical Computation and Simulation 59, 1-23.

[Holenstein (2009)] *PhD thesis: Particle Markov Chain Monte Carlo* submitted for the degree of Doctor of Philosophy.

[Johannes and Polson (2006)] *Exact parameter learning and state filtering*, working paper.

[Kim, Shephard and Chib (1998)] *Stochastic volatility: likelihood inference and comparison with ARCH models*. The Review of Economic Studies 65(3), 361-393.

[Kitagawa (1996)] *Monte Carlo filter and smoother for non-Gaussian nonlinear dynamic systems*. J.Computnl. Graph. Statist.,5,1-25.

[Kitagawa and Gersch (1996)] *Smoothness Priors Analysis of Time Series*. Springer. Lecture Notes in Statistics, Vol.116.

[Lee, Yau, Giles, Doucet and Holmes (2009)] *On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods*. Submitted for publication.

[Liu (2001)] *Monte Carlo Strategies in Scientific Computing* Springer Verlag, New York, 1st edition, chapter 6.

[Liu and Chen (1998)] *Sequential Monte Carlo methods for dynamic systems*. Journal of the American Statistical Association 93, 1032-1044.

[Liu and West (2001)] *Combined parameter and state estimation in simulation-based filtering*. In Sequential Monte Carlo Methods in Practice p.197, Doucet, Freitas and Gordon. Springer 2001.

[Lopes] *Markov Chain Monte Carlo and SMC Methods in Stochastic Volatility Models* Universitiy of Chicago Booth School of Business, http://faculty.chicagobooth.edu/hedibert.lopes/teaching/UPCcourse/UPCcourse-handouts.pdf.

[Pitt and Shephard (1999)] *Filtering Via Simulation: Auxiliary Particle Filters*. Journal of the American Statistical Association, Vol. 94(446),590-591.

[Storvik (2002)] *Particle Filters for State-Space Models With the Presence of Unknown Static Parameters*. IEEE Tranactions on Signal Processing, 50(2), 281-289.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorstehende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Bonn, den