More on particle MCMC and pseudomarginal methods

MVE187-MSA101 “Computational methods for Bayesian statistics”, 2021-2022

Umberto Picchini
@uPicchini

Chalmers University of Technology and University of Gothenburg
Sweden

Goals for today:

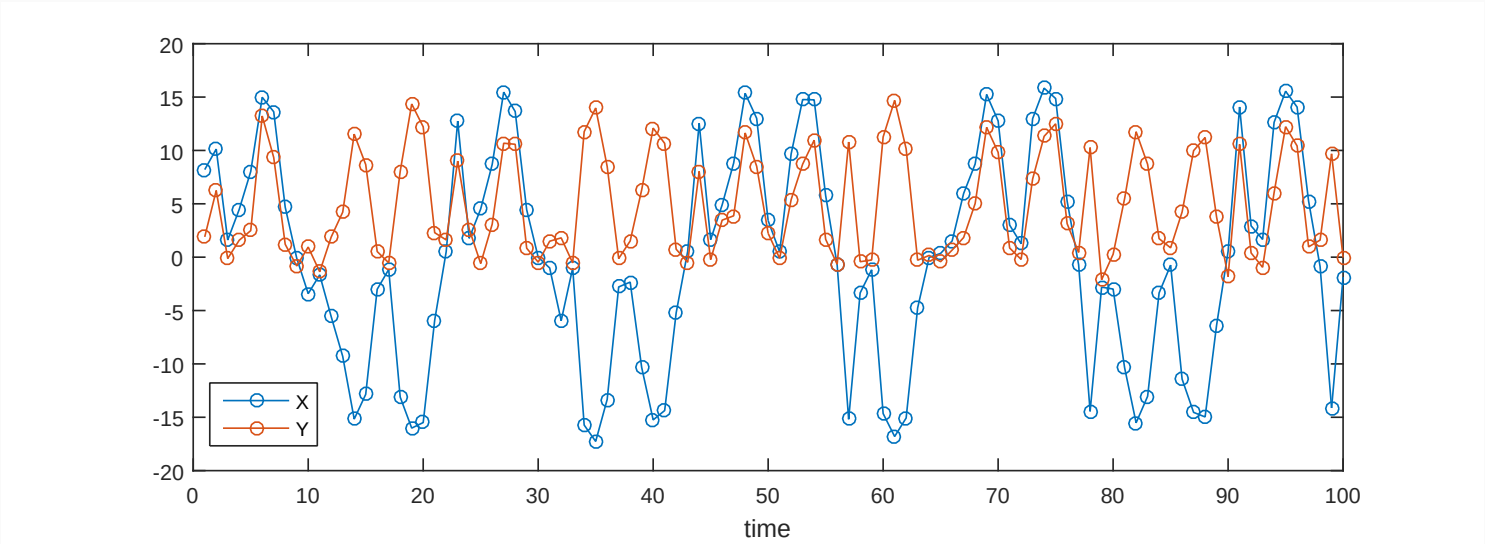
- Recap of the MCMC sampling from $\hat{\pi}(\theta|y_{1:T})$ with embedded SMC approximation for $p(y_{1:T}|\theta)$;
- example of Bayesian inference for a state-space model driven by an SDE;
- related tuning problems;
- show that what we do is *exact-approximate* inference: the "pseudo-marginal" MCMC approach.

Some recap from the previous lecture

An example we considered was that of:

$$\begin{cases} x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{(1+x_{t-1}^2)} + 8\cos(1.2(t-1)) + v_t, \\ y_t = 0.05x_t^2 + e_t, \\ v_t \sim N(0,q) \\ e_t \sim N(0,r) \end{cases}$$

Data generated with $q = 0.1$ and $r = 1$.



Then we have shown the logic behind the construction of a *sequential* approximation to the likelihood
$p(y_{1:T}|\theta) = p(y_1|\theta) \cdot \prod_{t=2}^{T} p(y_t|y_{1:t-1};\theta)$.

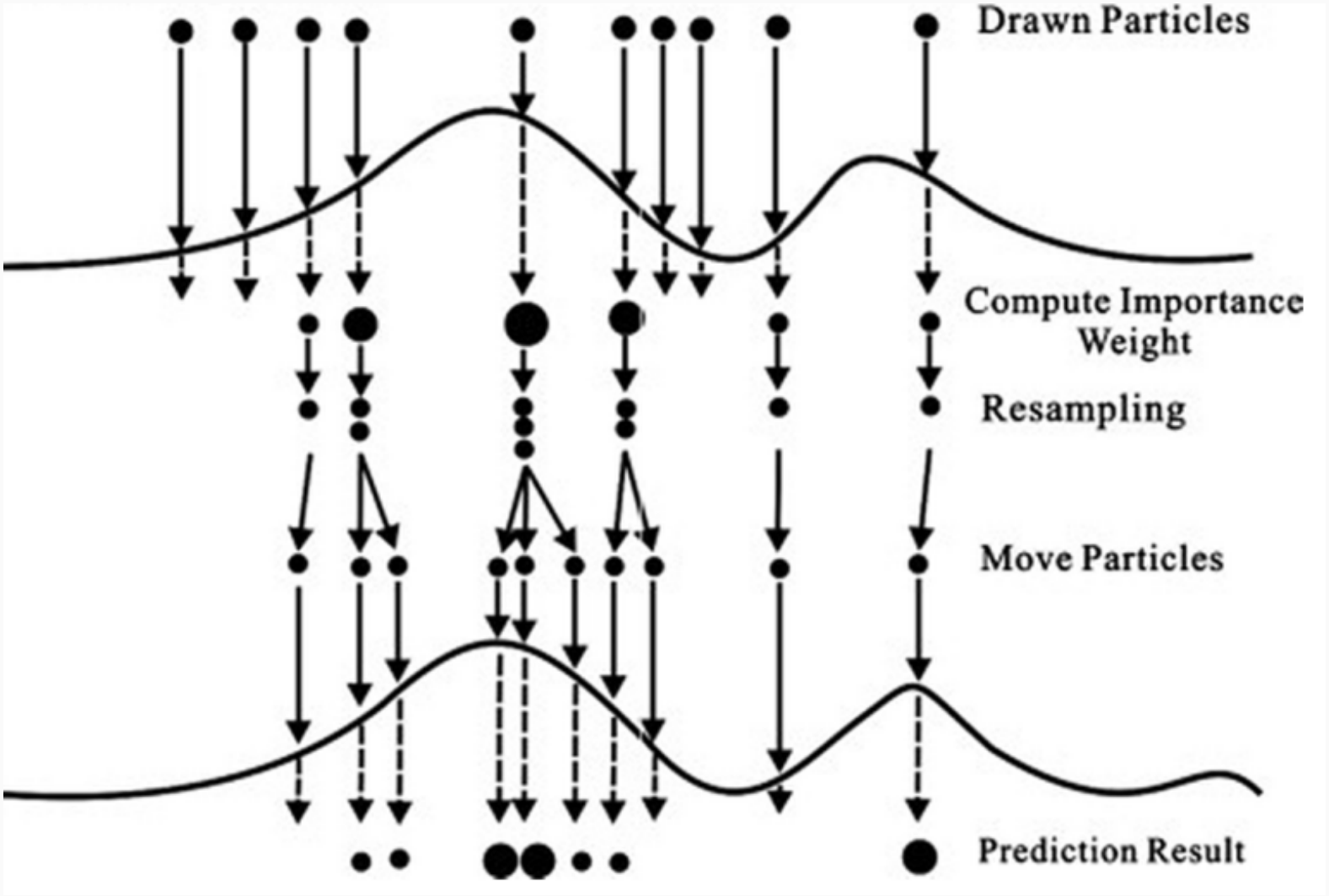
Basically via SMC we have $\hat{p}(y_t|y_{1:t-1};\theta) = \frac{1}{N}\sum_{i=1}^{N} w_t^i$

Simplest case has $w_t^i = p(y_t|x_t^i)$ ("bootstrap filter").

The secret to SMC is

- "propagate particles x_t^i forward",
- then "weight" the particles,
- "resample particles". The last operation is essential to let unimportant particles die.

Propagation→weighting→resampling→propagation of resampled particles→weighting→etc



What do we need to run the bootstrap filter?

Two things only:

1. being able to *forward propagate the latent X process* (particles);
2. evaluate the observational density $p(y_t|x_t;\theta)$ at each triplet (y_t, x_t, θ).

Point 1 means (for the example): write a code that **at the current value of** $\theta^* = (q^*, r^*)$, and starting from x_0 it produces x_1, and once in x_1 it gives you x_2 etc until the final $t = T$.

$$x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{(1+x_{t-1}^2)} + 8\cos(1.2(t-1)) + N(0,q)$$

But at each time t we do the above starting from each **resampled** particle $\tilde{x}_{t-1}^i$ so we have

$$x_t^i = 0.5\tilde{x}_{t-1}^i + 25\frac{\tilde{x}_{t-1}^i}{(1+\tilde{x}_{t-1}^{i^2})} + 8\cos(1.2(t-1)) + N(0,q), \qquad i = 1,...,N$$

Pseudo-algorithm for the example

We are at a generic iteration of Metropolis-Hastings with proposal $\theta^* = (q^*, r^*)$ being evaluated (maybe will be accepted or rejected).

1 $t = 0$ (initialize) e.g. set $x_0^i = 0$, assign $\tilde{w}_0^i = 1/N$, $i = 1, ..., N$

2 here all particles have the same weight, call them $\tilde{x}_0^i$

3 propagate all particles forward towards $t = 1$

$$x_1^i = 0.5\tilde{x}_0^i + 25\frac{\tilde{x}_0^i}{(1+\tilde{x}_0^{i^2})} + 8\cos(1.2(0)) + N(0, q^*), \qquad i = 1, ..., N$$

4 weight the particles via

$$w_1^i \propto p(y_1|x_1^i) \equiv \mathcal{N}(0.05(x_1^i)^2, r^*), \quad i = 1, ..., N$$

5 Likelihood term: $\hat{p}(y_1|\theta^*) = \frac{1}{N}\sum_{i=1}^N w_1^i$

6 normalise weights as $\tilde{w}_1^i = w_1^i / \sum_{i=1}^N w_1^i$. We interpret each $\tilde{w}_1^i \in (0, 1)$ as a **probability to be resampled** for x_1^i

8 *propagate forward* towards t=2

$$x_2^i = 0.5\tilde{x}_1^i + 25\frac{\tilde{x}_1^i}{(1+\tilde{x}_1^{i^2})} + 8\cos(1.2(1)) + N(0, q^*), \qquad i = 1, ..., N$$

9 weight the particles via

$$w_2^i \propto p(y_2|x_2^i) \equiv \mathcal{N}(0.05(x_2^i)^2, r^*), \quad i = 1, ..., N$$

10 Likelihood term: $\hat{p}(y_2|y_1; \theta^*) = \frac{1}{N}\sum_{i=1}^N w_2^i$

11 normalise weights as $\tilde{w}_2^i = w_2^i / \sum_{i=1}^N w_2^i$.

12 from the set $\{x_2^i, \tilde{w}_2^i\}_{i=1}^N$, resample particles with replacement N times to obtain the set $\{\tilde{x}_2^i, i = 1, ..., N\}$.

13 etc. continue until you reach the last time-point $t = T$

Return the approximate likelihood for the current $\theta^* = (q^*, r^*)$

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

So we can now happily run our Metropolis-Hastings sampler to sample from the posterior:

1. current value is θ_r and current estimated likelihood is $\hat{p}(y_{1:T}|\theta_r)$;
2. propose a new $\theta^* \sim q(\theta^*|\theta_r)$, e.g. $\theta^* \sim N(\theta_r, \Sigma_\theta)$ for some covariance matrix Σ_θ.
3. run the bootstrap filter using θ^* and obtain $\hat{p}(y_{1:T}|\theta^*)$.
4. compute
$$A = \frac{\hat{p}(y_{1:T}|\theta^*)}{\hat{p}(y_{1:T}|\theta_r)} \times \frac{\pi(\theta^*)}{\pi(\theta_r)} \times \frac{q(\theta_r|\theta^*)}{q(\theta^*|\theta_r)}$$
Draw a uniform $u \sim U(0,1)$ and if $u < A$ **accept** θ^* and set $\theta_{r+1} := \theta^*$. Otherwise, **reject** θ^*, set $\theta_{r+1} := \theta_r$.
5. Set $r := r + 1$, go to 1 and repeat.

Accepted parameters are from

$$\hat{\pi}(\theta|y_{1:T}) \propto \hat{p}(y_{1:T}|\theta) \times \pi(\theta)$$

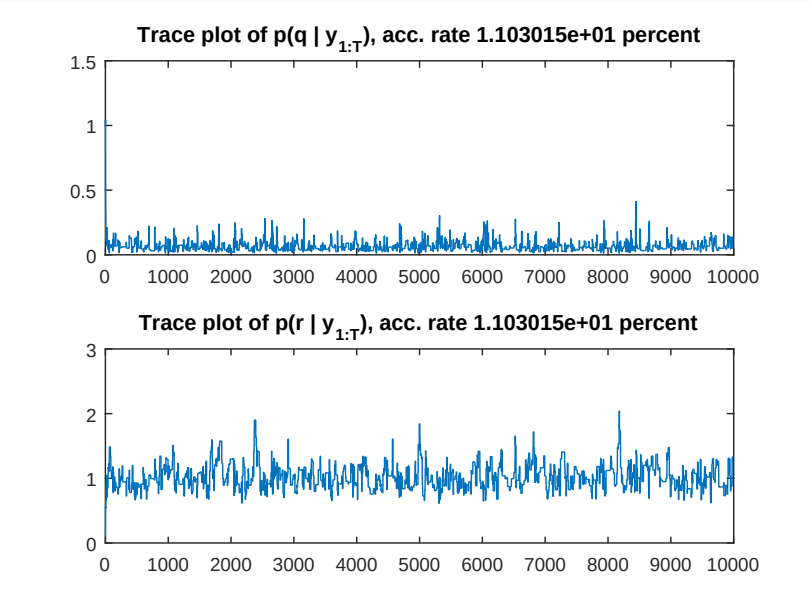
but later we prove that actually samples are from the **exact posterior**

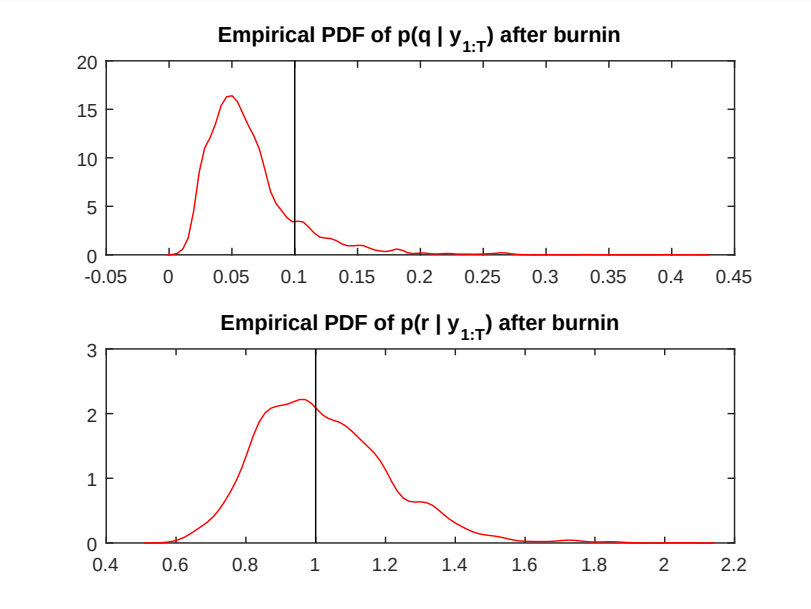
$$\pi(\theta|y_{1:T}) \propto p(y_{1:T}|\theta) \times \pi(\theta)$$

Then it was sort of informally implied that we just plug $\hat{p}(y_{1:T}|\theta)$ into Metropolis-Hastings and sample from

$$\hat{\pi}(\theta|y_{1:T}) \propto \hat{p}(y_{1:T}|\theta) \times \pi(\theta)$$

Where for our case study $\theta = (q, r)$.





But what are we really sampling from?

- the $\hat{p}(y_{1:T}|\theta) = \prod_{t=1}^{T}(\frac{1}{N}\sum_{i=1}^{N} w_t^i)$ is clearly a random variable.
- instead for standard Metropolis-Hastings $p(y_{1:T}|\theta)$ is a deterministic quantity.
- effectively, $\hat{p}(y_{1:T}|\theta) \equiv \int \hat{p}(y_{1:T}|\xi,\theta) \times p(\xi)d\xi$
- the $\xi \sim p(\xi)$ is a vector of random variates independent of θ.

Example:

$$\begin{cases} x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{(1+x_{t-1}^2)} + 8\cos(1.2(t-1)) + N(0,q), \\ y_t = 0.05x_t^2 + N(0,r), \end{cases}$$

or

$$\begin{cases} x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{(1+x_{t-1}^2)} + 8\cos(1.2(t-1)) + \sqrt{q}\cdot\xi_t^{(1)}, \quad \xi_t^{(1)} \sim N(0,1) \\ y_t = 0.05x_t^2 + \sqrt{r}\cdot\xi_t^{(2)}, \quad \xi_t^{(2)} \sim N(0,1) \end{cases}$$

So certainly

$$\xi = \left((\xi_1^{(1)}, \xi_1^{(2)}), ..., (\xi_T^{(1)}, \xi_T^{(2)}), ...?\right)$$

and what other random variates can we also include?

Resampling! Performing resampling involves the generation of pseudo-random variates so we finally have

$$\xi = \Big((\xi_1^{(1)}, \xi_1^{(2)}), ..., (\xi_T^{(1)}, \xi_T^{(2)}),$$
$$\text{plus ALL the variates produced during resampling}\Big)$$

The Metropolis-Hasting actually samples from an artificially extended posterior

$$\hat{\pi}(\theta, \xi | y_{1:T}) \propto \hat{p}(y_{1:T} | \xi, \theta) \cdot p(\xi) \cdot \pi(\theta)$$

1. current value is θ_r, propose a new $\theta^* \sim q(\theta^* | \theta_r)$, e.g. $\theta^* \sim N(\theta_r, \Sigma_\theta)$ for some covariance matrix Σ_θ.
2. Sample $\xi^* \sim p(\xi)$ (useful for propagation and resampling)
3. compute

$$A = \frac{\hat{p}(y_{1:T} | \xi^*, \theta^*)}{\hat{p}(y_{1:T} | \xi_r, \theta_r)} \times \frac{p(\xi^*)}{p(\xi_r)} \times \frac{\pi(\theta^*)}{\pi(\theta_r)} \times \frac{q(\theta_r | \theta^*)}{q(\theta^* | \theta_r)}$$

   Draw a uniform $u \sim U(0,1)$ and if $u < A$ **accept** (θ^*, ξ^*) and set $(\theta_{r+1}, \xi_{r+1}) := (\theta^*, \xi^*)$.
   Otherwise, **reject**, and set $(\theta_{r+1}, \xi_{r+1}) := (\theta_r, \xi_r)$.
4. Set $r := r + 1$, go to 1 and repeat.

Typically $\xi \sim N(0,1)$ or $\xi \sim U(0,1)$ and generally $p(\xi)$ independent of θ, so the ratio of the $p(\xi)$ cancels out.

So in practice just run Metropolis-Hastings as usual, and do not reestimate the likelihood at the denominator, just use the one you have previously accepted.

Incredible result

Quite astonishingly Andrieu and Roberts[1] proved that using an *unbiased and non negative* estimate of the likelihood function into the MCMC routine is *sufficient* to obtain exact Bayesian inference for θ!

That is using the Metropolis-Hastings acceptance probability

$$\min\left\{1, \frac{\hat{p}(y_{1:T}|\xi^*, \theta^*)}{\hat{p}(y_{1:T}|\xi, \theta)} \times \frac{\pi(\theta^*)}{\pi(\theta)} \times \frac{q(\theta|\theta^*)}{q(\theta^*|\theta)}\right\}$$

will return a Markov chain with stationary distribution $\pi(\theta|y_{1:T})$ *regardless the* ***finite*** *number N of particles used to approximate the likelihood!*.

The good news is that $\mathbb{E}_\xi(\hat{p}(y_{1:T}|\xi, \theta)) = p(y_{1:T}|\theta)$ with $\hat{p}(y_{1:T}|\xi, \theta)$ obtained via SMC.

[1] Andrieu and Roberts (2009), Annals of Statistics, 37(2) 697–725.

The previous result is, in my opinion, one of the most important statistical results of the last 30 years.

In fact, it offers an "exact-approximate" approach, where because of computing limitations we can only produce $N < \infty$ particles, while still be reassured to obtain *exact* (Bayesian) inference under minor assumptions.

But let's give a rapid (technically informal) look at why it works.

Key result: unbiasedness (del Moral 2004[2])
We have that

$$\mathbb{E}_{\xi}(\hat{p}(y_{1:T}|\xi,\theta)) = \int \hat{p}(y_{1:T}|\theta,\xi)p(\xi)d\xi = p(y_{1:T}|\theta)$$

with $\xi \sim p(\xi)$ vector of *all* random variates generated during SMC (both to *propagate forward* the state and to perform particles resampling).

---

[2]Easier to look at Pitt, Silva, Giordani, Kohn. J. Econometrics 171, 2012 or page 87 of Naesseth's PhD thesis 2018.

To prove the exactness of the approach we look at the (easier and less general) argument in sec. 2.2 of Pitt, Silva, Giordani, Kohn. J. Econometrics 171, 2012 or my even more introductive blog post.

To simplify the notation take $y := y_{1:T}$.

- $\hat{\pi}(\theta, \xi|y)$ approximate joint posterior of (θ, ξ) obtained via SMC

$$\hat{\pi}(\theta, \xi|y) = \frac{\hat{p}(y|\theta, \xi)p(\xi)\pi(\theta)}{p(y)}$$

(notice ξ and θ are assumed a-priori independent)

Notice we put $p(y)$ not $\hat{p}(y)$ at the denominator: this follows from the unbiasedeness assumption as we obtain $\int\int \hat{p}(y|\theta, \xi)p(\xi)\pi(\theta)d\xi d\theta = \int \pi(\theta)\{\int \hat{p}(y|\theta, \xi)p(\xi)d\xi\}d\theta = \int \pi(\theta)p(y|\theta)d\theta = p(y)$.

The exact (unavailable) posterior of θ is

$$\pi(\theta|y) = \frac{p(y|\theta)\pi(\theta)}{p(y)}$$

therefore the marginal likelihood (evidence) is

$$p(y) = \frac{p(y|\theta)\pi(\theta)}{\pi(\theta|y)}$$

and

$$\begin{aligned}\hat{\pi}(\theta, \xi|y) &= \frac{\hat{p}(y|\theta, \xi)p(\xi)\pi(\theta)}{p(y)} \\ &= \frac{\pi(\theta|y)\hat{p}(y|\theta, \xi)p(\xi)\cancel{\pi(\theta)}}{p(y|\theta)\cancel{\pi(\theta)}}\end{aligned}$$

Now, we know that applying an MCMC targeting $\hat{\pi}(\theta, \xi|y)$ then discarding the output pertaining to ξ corresponds to *integrate-out* ξ from the posterior

$$\int \hat{\pi}(\theta, \xi|y)d\xi = \frac{\pi(\theta|y)}{p(y|\theta)} \underbrace{\int \hat{p}(y|\theta, \xi)p(\xi)d\xi}_{\mathbb{E}(\hat{p}(y|\theta))=p(y|\theta)} = \pi(\theta|y)$$

We are thus performing a *pseudo-marginal* approach: "marginal" because we disregard ξ; *pseudo* because we use $\hat{p}(\cdot)$ not $p(\cdot)$.

Therefore we proved that, using MCMC on an (artificially) augmented posterior, then discard from the output all the random variates ξ created during SMC, returns samples from the **true** posterior. Exact Bayes!

Notice that discarding the ξ is something that we naturally do in Metropolis-Hastings hence nothing strange is happening here. The ξ are just instrumental, uninteresting, variates independent of θ and independent of $\{X_t\}$.

Actually...

When I wrote that we obtain samples from the true posterior, that's true.

However it is not quite like having exact samples as when we use conjugacy properties, or when the posterior is analytically available.

When we run a small number of iterations, not an infinite number, it can still happen that we struggle to explore the whole posterior surface thoroughly.

So the pseudomarginal method is not a silver-bullet. It comes with the usual problems of MCMC (eg difficulty with exploring multimodal surfaces).

Example: SSM with latent SDE

$$dx_t = f(x_t;\theta)dt + g(x_t;\theta)dB_t, \quad dB_t \sim_{iid} N(0, dt)$$
$$y_t = x_t + e_t, \quad e_t \sim_{iid} N(\cdot,\cdot)$$

We consider an Ornstein-Uhlenbeck (OU) process for the latent dynamics:

$$dx_t = -\beta(x_t - \alpha)dt + \sigma \cdot dB_t,$$
$$y_t = x_t + e_t, \quad e_t \sim_{iid} N(0, 0.316^2)$$

where

- $\alpha \in \mathbb{R}$ is the *stationary mean* of the process;
- $\beta > 0$ is the growth rate;
- $\sigma > 0$ diffusion coefficient (intensity of the intrinsic noise).

OU has known (Gaussian) transition densities, however for our purposes it is more useful to write *how* we simulate a path exactly:

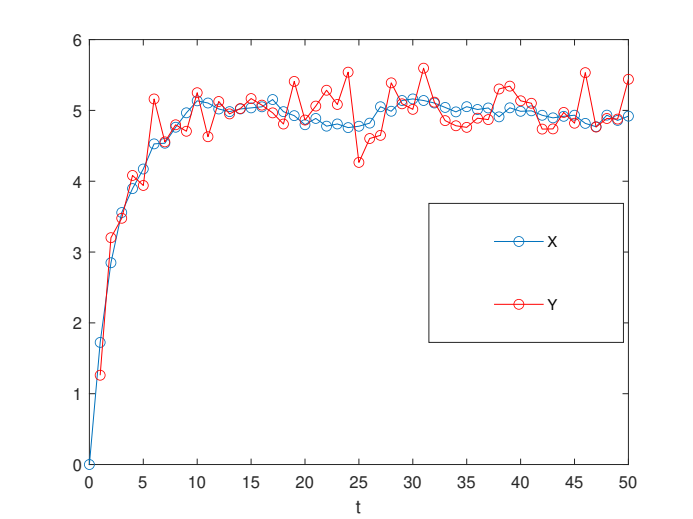
$$x_{t+\Delta} = \alpha + (x_t - \alpha)e^{-\beta\Delta} + \sqrt{\frac{\sigma^2}{2\beta}(1 - \exp(-2\beta\Delta))} \times \xi_{t+\Delta}$$

with $\xi_t \sim N(0,1)$ iid.

Simulation setup

$$dx_t = -\beta(x_t - \alpha)dt + \sigma \cdot dB_t,$$
$$y_t = x_t + e_t, \quad e_t \sim_{iid} N(0, 0.316^2)$$

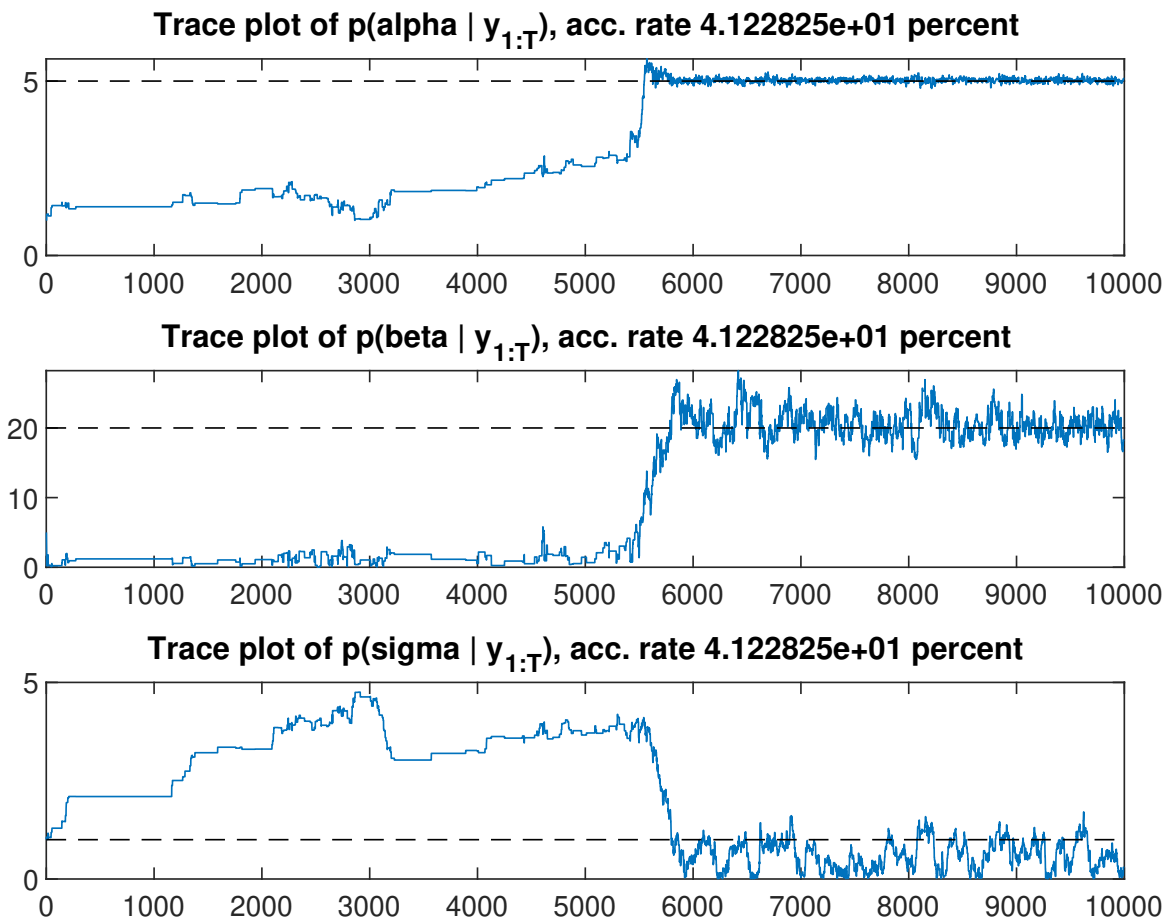
- $T = 50$ observations at equispaced integer times $t = 1, 2, ..., T$, so $\Delta = 1$.
- ground-truth parameters: $\alpha = 5$, $\beta = 20$, $\sigma = 1$.



Inference setup

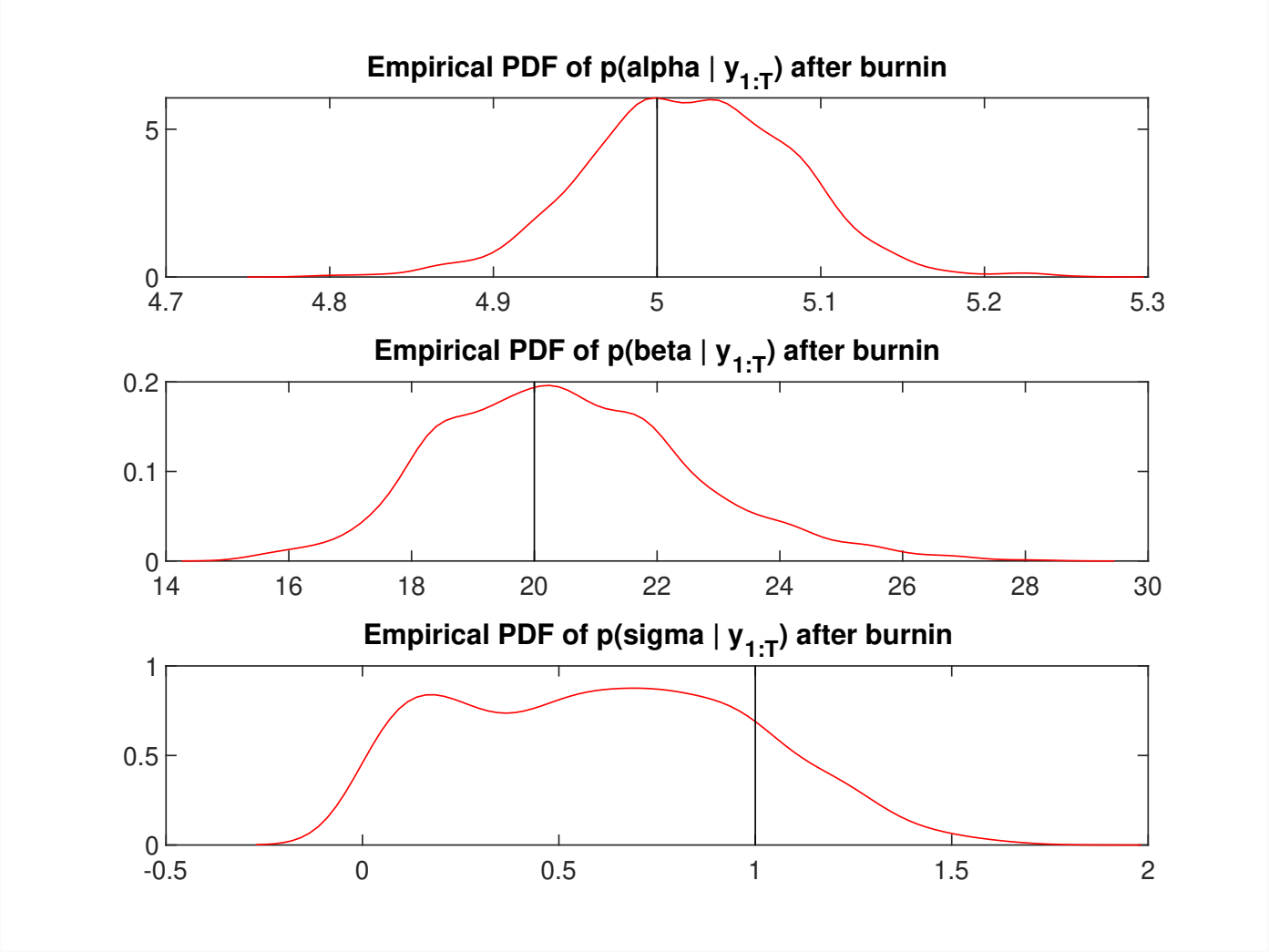
For SMC we use the bootstrap filter with $N = 50$ particles.

Priors: $\alpha \sim U(1, 10)$, $\beta \sim \text{InvGamma}(3, 50)$, $\sigma \sim \text{InvGamma}(3, 4)$.

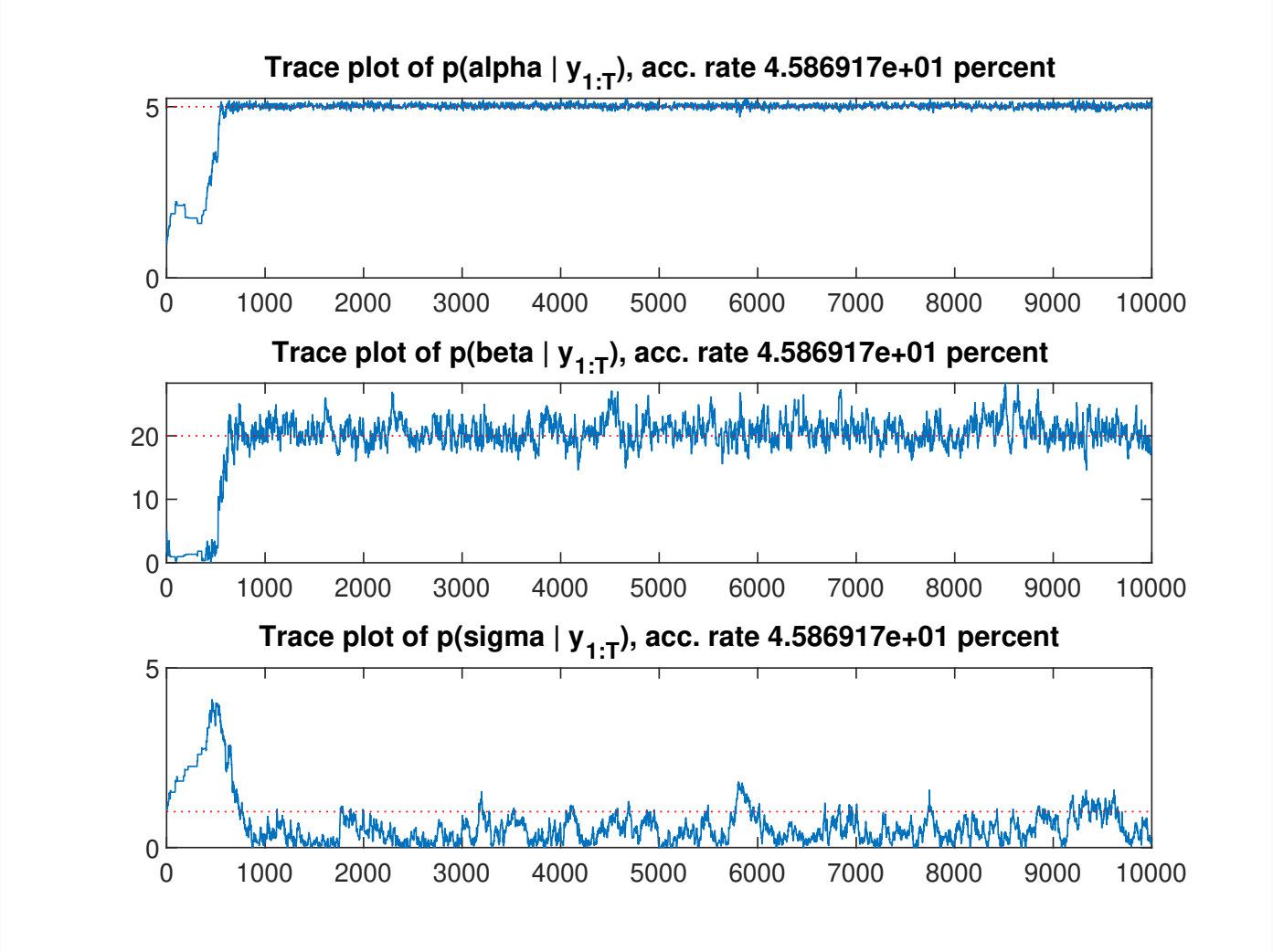


Marginal posteriors.

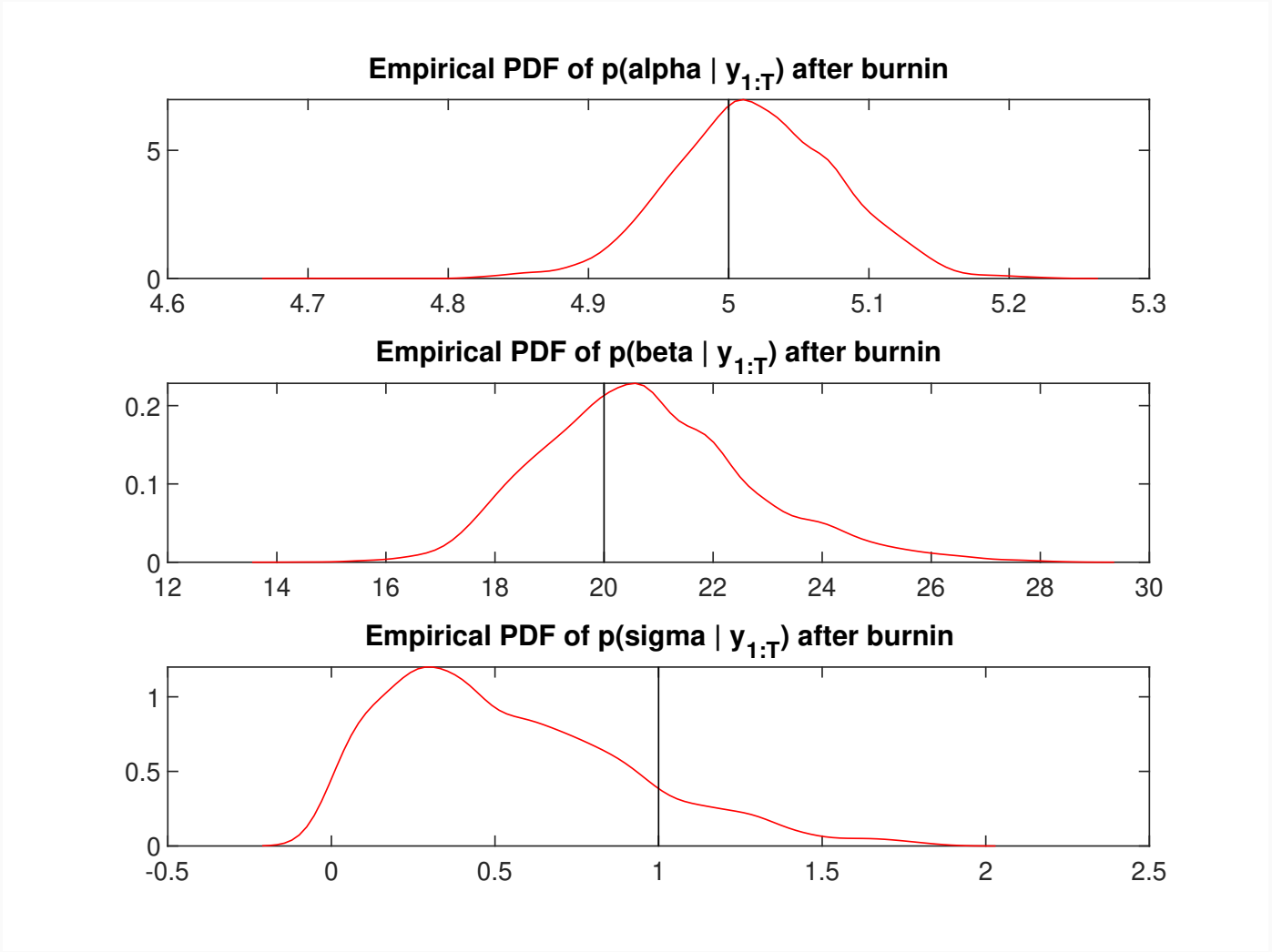
Priors: $\alpha \sim U(1, 10)$, $\beta \sim \text{InvGamma}(3, 50)$, $\sigma \sim \text{InvGamma}(3, 4)$.



We obtain the similar inference with $N = 500$ (instead of N=50) but faster convergence:



With N=500:


Empirical PDF of p(alpha | $y_{1:T}$) after burnin
5
0
4.6
4.7
4.8
4.9
5
5.1
5.2
5.3
Empirical PDF of p(beta | $y_{1:T}$) after burnin
0.2
0.1
0
12
14
16
18
20
22
24
26
28
30
Empirical PDF of p(sigma | $y_{1:T}$) after burnin
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5

The effect of selecting N

What's the effect of choosing a small or a large number of particles N?

Thinks about it: the estimated likelihood used a stochastic procedure, it's not a deterministic approximation.

$\hat{p}(y_{1:T}|\theta)$ from either importance sampling or SMC is a random variable (variability induced by Monte Carlo).

The smaller the N the larger the variance of $\hat{p}(y_{1:T}|\theta)$.

The larger N the more precise the approximation (ie the smaller the variance is).

Here I fix the values of α and β to their true values, and instead consider the equispaced grid for $\sigma \in (0.1, 0.2, ..., 10)$.

I estimate the likelihood via bootstrap filter for each σ value in the grid, and repeat the estimation independently for 50 times.

I used N=10 particles. Below are loglikelihood values for the procedure. Recall the true $\sigma = 1$.

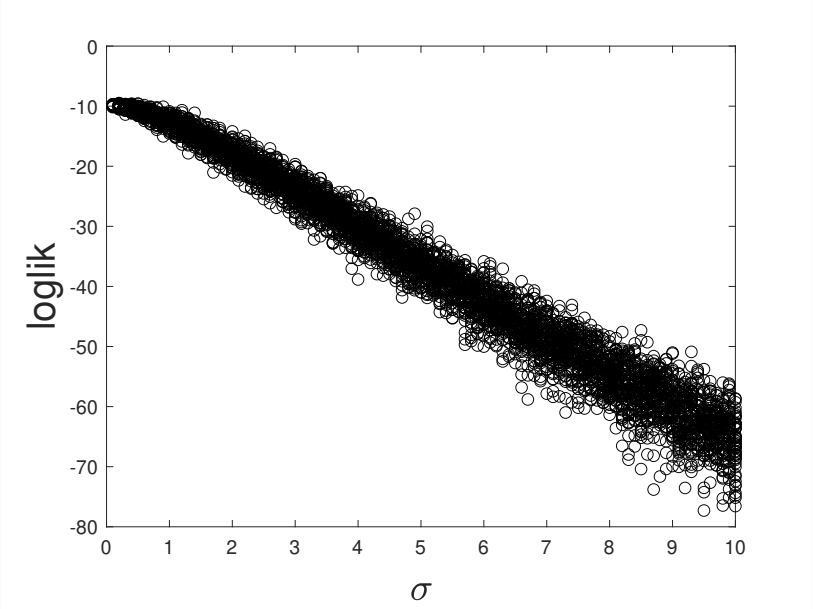


Figure 1: N=10

Same but with N=100.

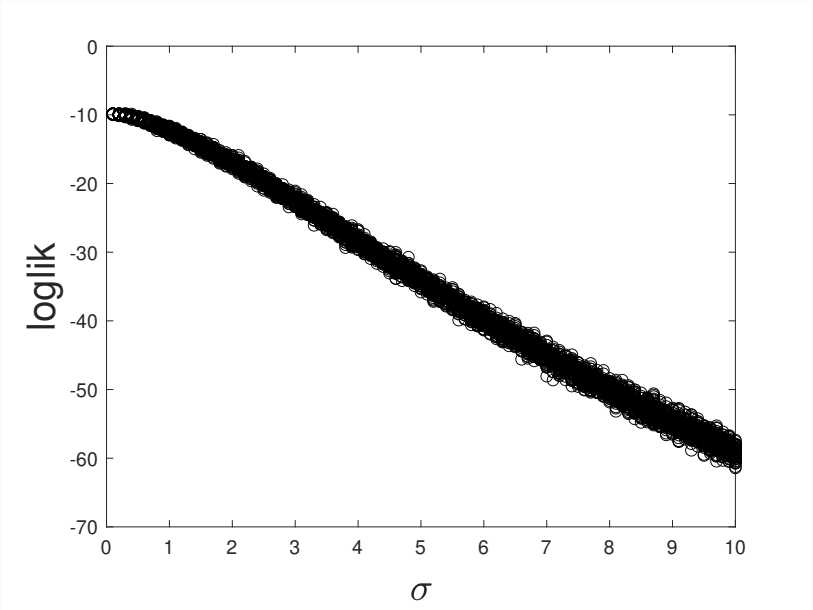
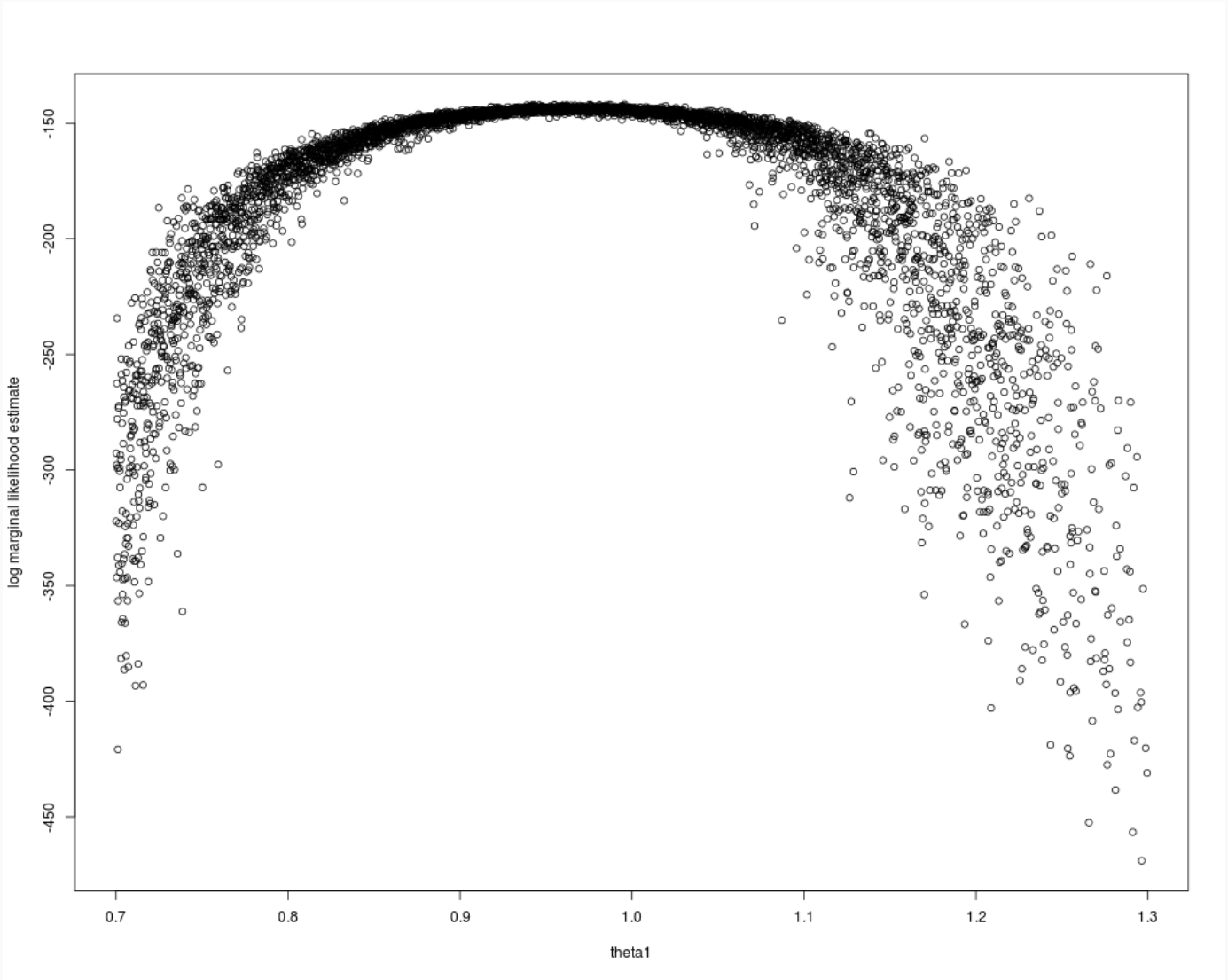


Figure 2: N=100

A more dramatic figure for a different model (see https://tinyurl.com/4h7k3utv)



The variability of the approximation depends also on θ, not only N.

If the current θ value is implausible, for the given data, the likelihood approximation gets noisy because many particles end-up in unimportant regions.

If the likelihood approximation via SMC gets very variable, it can occur that in Metropolis-Hastings (MH) **we occasionally accept an overestimated likelihood** $\rightarrow$ goes in the denominator of the MH ratio $\rightarrow$ difficult to accept further proposals $\rightarrow$ many rejections $\rightarrow$ chain slowly moving.

Easiest solution: increase the number of particles N but this will increase the computational effort.

Plug-and-play strategies

The use of the bootstrap filter into MCMC is sometimes denoted a "plug-and-play" strategy.

This means you can write a generic code that you can reuse for pretty much any state-space model without analytic calculations involved.

This is because you only need to define in your code how the states X advance/propagate from x_t to x_{t+1}.

And then we assume we can evaluate $p(y_t|x_t;\theta)$ pointwisely for any of its arguments.

That's it! You *plug* the model equations in your code, and you *play*.

Other plug-and-play methods

There exist several other plug-and-play methods. These are all examples of *simulation-based inference* methods, in that they are very generic and only require model simulations to get around the **intractability of the likelihood function**.

The R package pomp supports a number of plug-and-play methods:

- particle marginal methods
- iterated filtering
- approximate Bayesian computation (ABC)
- synthetic likelihoods
- ...and more.

Notice pseudomarginal methods, ABC and synthetic likelihoods are not only working with state-space models! They are very general methods.

Conclusions

- We have outlined a powerful methodology for **exact Bayesian inference**. Theoretically exact regardless the number of particles N.
- In practice, a too small N will have a negative impact on chain mixing $\rightarrow$ many rejections, *sticky chain*.
- the methodology is perfectly suited for state-space modelling. However it can deal with more general models.

Downsides when using bootstrap filter

- Recall, in general we wanted to propose particles $x_t^i \sim h(x_t^i | x_{0:t-1}^i, y_{1:t})$, i=1,...,N;
- the above (if implemented) allows particles at time t-1 to be able to "lookahead" to the next datapoint y_t;
- the bootstrap filter has $x_t^i \sim h(x_t | x_{0:t-1}^i, y_{1:t}) \equiv p(x_t | x_{t-1}^i)$. It is "myopic";
- if the dimension $\dim(x_t)$ increases, we might need a very very large N (computationally intensive);
- There are more intelligent SMC filters. Such as the "auxiliary particle filter" (Pitt& Shephard), or the use of "bridges" and "guided proposals" when discretizing an SDE numerically (lots of work by Schauer, and also Golightly-Wilkinson).

Important issues

How to tune the number of particles?

- Doucet, Pitt, and Kohn. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. arXiv:1210.1871 (2012).
- Pitt, dos Santos Silva, Giordani and Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. Journal of Econometrics 171, no. 2 (2012): 134-151.
- Sherlock, Thiery, Roberts and Rosenthal. On the efficiency of pseudo-marginal random walk Metropolis algorithms. arXiv:1309.7209 (2013).

More suggestions for further reading in my blog post:

https://tinyurl.com/4964pesp

Software

Some possibilities are:

- Birch
- LiBBi (C++ template library)
- Biips (C++ with interfaces to MATLAB/R)
- `demo("PMCMC")` in the R package `smfsb`.
- R package pomp
- accompanying MATLAB code for the book by S. Särkkä. Code available here.