Particle Markov Chain Monte Carlo Methods

Bjarke Hautop

Student ID: 202005674

Master's Thesis in Statistics

Aarhus University
Supervisor: Jan Pedersen
March 4, 2025

Contents

1 Introduction

2 Monte Carlo Methods

In this chapter, we provide the theoretical foundation for Particle Markov chain Monte Carlo (PMCMC). We follow the work of Andrieu et al. [2010], Doucet and Johansen [2009], and Kroese et al. [2013].

Let μ be a σ-finite reference measure on the space $\mathcal{X}$ where each x_i takes values, and denote by $\mu^{\otimes n}$ the corresponding product measure on $\mathcal{X}^n$. Let $x_{1:n} = (x_1, x_2, \ldots, x_n)$ and suppose we have a density $\pi_n(x_{1:n})$ (with respect to $\mu^{\otimes n}$) given by

$$\pi_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n}, \tag{2.1}$$

where $\gamma_n(x_{1:n})$ is the unnormalized density and the normalizing constant is defined as

$$Z_n = \int_{\mathcal{X}^n} \gamma_n(x_{1:n})\, d\mu^{\otimes n}(x_{1:n}).$$

In many applications and for notational simplicity, we for the rest of this chapter let $\mathcal{X} \subseteq \mathbb{R}^d$ and μ be the Lebesgue measure, in which case Z_n is given by

$$Z_n = \int \gamma_n(x_{1:n})\, dx_{1:n}. \tag{2.2}$$

Let $X_{1:n} \sim \pi_n(x_{1:n})$ and suppose we generate N i.i.d. samples $x_{1:n}^{(1)}, x_{1:n}^{(2)}, \ldots, x_{1:n}^{(N)}$. We can then approximate $\pi_n(x_{1:n})$ by the empirical measure

$$\pi_n^{\mathrm{MC}}(x_{1:n}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_{1:n}^{(i)}}(x_{1:n}),$$

where $\delta_{X_{1:n}^{(i)}}(x_{1:n})$ denotes the Dirac measure centered at $X_{1:n}^{(i)}$. We can also approximate any marginal $\pi_n(x_k)$ as

$$\pi_n^{\mathrm{MC}}(x_k) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X_k^{(i)}}(x_k).$$

The expectation of any function $H_n : \mathcal{X}^n \to \mathbb{R}$ is given by

$$I_n(H_n) \coloneqq \mathbb{E}_{X_{1:n} \sim \pi_n}[H_n(X_{1:n})] = \int H_n(x_{1:n}) \pi_n(x_{1:n})\, dx_{1:n},$$

and by the law of large numbers (LLN) we can estimate it by

$$I_n^{\mathrm{MC}}(H_n) \coloneqq \int H_n(x_{1:n}) \pi_n^{\mathrm{MC}}(x_{1:n}) dx_{1:n} = \frac{1}{N} \sum_{i=1}^{N} H_n(x_{1:n}^{(i)}).$$

However, this requires that we can sample from $\pi_n(x_{1:n})$, which often is not the case when it is a complex high-dimensional distribution.

2.1 Importance Sampling

A way to solve this issue is to use importance sampling (IS). Here we introduce an importance density $q_n(x_{1:n})$ which we can sample from and such that

$$\pi_n(x_{1:n}) > 0 \implies q_n(x_{1:n}) > 0.$$

For the remainder of this chapter, we let

$$X_{1:n} \sim q_n(x_{1:n}).$$

Suppose we generate N i.i.d. samples $x_{1:n}^{(1)}, x_{1:n}^{(2)}, \ldots, x_{1:n}^{(N)}$. To correct for the fact that we sample from q_n we define the *unnormalized weight* function

$$w_n(x_{1:n}) \coloneqq \frac{\gamma_n(x_{1:n})}{q_n(x_{1:n})}$$

and define the *normalized weight* function

$$W_n^{(i)} \coloneqq \frac{w_n(X_{1:n}^{(i)})}{\sum_{j=1}^N w_n(X_{i:n}^{(j)})}.$$

From Equation (2.1) and Equation (2.2) we get

$$\pi_n(x_{1:n}) = \frac{w_n(x_{1:n}) q_n(x_{1:n})}{Z_n}, \tag{2.3}$$

and

$$Z_n = \int w_n(x_{1:n}) q_n(x_{1:n})\, dx_{1:n}. \tag{2.4}$$

We then define the IS estimators of respectively $\pi_n(x_{1:n})$ and Z_n as

$$\widehat{\pi}_n(x_{1:n}) = \sum_{i=1}^N W_n^{(i)} \delta_{X_{1:n}^{(i)}}(x_{1:n}), \tag{2.5}$$

$$\widehat{Z}_n = \frac{1}{N} \sum_{i=1}^N w_n(X_{1:n}^{(i)}). \tag{2.6}$$

Next, we will show what the relative variance of $\widehat{Z}_n$ is.

Theorem 2.1 (Relative variance of $\mathbb{V}(\widehat{Z}_n)$)**.** The relative variance of the IS estimate of the normalizing constant Z_n is given by

$$\frac{\mathbb{V}(\widehat{Z}_n)}{Z_n^2} = \frac{1}{N}\Big(\int \frac{\pi_n^2(x_{1:n})}{q_n(x_{1:n})}\, dx_{1:n} - 1 \Big).$$

Proof. The variance of $\widehat{Z}_n$ is

$$\begin{aligned}
\mathbb{V}(\widehat{Z}_n) &= \frac{1}{N}\,\mathbb{V}\big(w_n(X_{1:n})\big) \\
&= \frac{1}{N}\Big(\mathbb{E}[w_n^2(X_{1:n})] - \mathbb{E}[w_n(X_{1:n})]^2\Big) \\
&= \frac{1}{N}\Big(\int \frac{\gamma_n^2(x_{1:n})}{q_n^2(x_{1:n})} q_n(x_{1:n})\,dx_{1:n} - Z_n^2\Big) \\
&= \frac{1}{N}\Big(\int \frac{\gamma_n^2(x_{1:n})}{q_n(x_{1:n})}\,dx_{1:n} - Z_n^2\Big) \\
&= \frac{1}{N}\Big(\int \frac{Z_n^2\pi_n^2(x_{1:n})}{q_n(x_{1:n})}\,dx_{1:n} - Z_n^2\Big) \\
&= \frac{Z_n^2}{N}\Big(\int \frac{\pi_n^2(x_{1:n})}{q_n(x_{1:n})}\,dx_{1:n} - 1\Big).
\end{aligned}$$

Dividing by Z_n^2 gives the result. □

Furthermore, we can also estimate $I_n(H_n)$ by

$$I_n^{\mathrm{IS}}(H_n) \coloneqq \int H_n(x_{1:n})\widehat{\pi}(x_{1:n})\,dx_{1:n} = \sum_{i=1}^N W_n^{(i)} H_n(X_{1:n}^{(i)}) = \frac{\frac{1}{N}\sum_{i=1}^N w_n(X_{1:n}^{(i)})H_n(X_{1:n}^{(i)})}{\frac{1}{N}\sum_{i=1}^N w_n(X_{1:n}^{(i)})}.$$

Note, that the numerator is an unbiased estimate of $Z_n I_n(H_n)$, since

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^N w_n(X_{1:n}^{(i)})H_n(X_{1:n}^{(i)})\right] &= \mathbb{E}\left[w_n(X_{1:n})H_n(X_{1:n})\right] \\
&= \int w_n(x_{1:n})H_n(x_{1:n})q(x_{1:n})\,dx_{1:n} \\
&= \int H_n(x_{1:n})\gamma_n(x_{1:n})\,dx_{1:n} \\
&= Z_n I_n(H_n).
\end{aligned}$$

Similar calculations gives that the denominator is an unbiased estimate of Z_n. Thus, we have a ratio of unbiased estimates, which is not unbiased. However, it is still consistent, which follows by using the LLN and properties of a.s. convergence.

A natural choice for an importance density $q_n(x_{1:n})$ is one that minimizes the variance of $\widehat{Z}_n$. As shown in Theorem 2.2, this minimum variance is achieved when

$$q_n(x_{1:n}) = \pi_n(x_{1:n}).$$

However, we cannot select this, as this was the reason we used IS in the first place. Nonetheless, this result indicates that the importance density should closely resemble the target density.

We could now sample from $\pi_n(x_{1:n})$ using the above method. However, to generate a sequence of samples for each n, we would have that each step would grow linearly in n, as generating samples from $\pi_{n+1}(x_{1:n+1})$ depends on the previous samples up to time n. This makes such an algorithm unfeasible in practice.

Theorem 2.2 (Minimum Variance of IS)**.** The variance $\mathbb{V}[\widehat{Z}_n]$ is minimized if

$$q_n(x_{1:n}) = \pi_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n}.$$

The proof is done by using Lagrange multiplier, and is inspired by Shimao [2018].

Proof. By independence we have,

$$\mathbb{V}[\widehat{Z}_n] = \frac{1}{N}\,\mathbb{V}\left[w_n(X_{1:n})\right].$$

We then have

$$\mathbb{V}[w_n(X_{1:n})] = \mathbb{E}\left[\frac{\gamma_n(X_{1:n})^2}{q_n(X_{1:n})^2}\right] - Z_n^2 = \int \frac{\gamma_n(x_{1:n})^2}{q_n(x_{1:n})}\,dx_{1:n} - Z_n^2.$$

Minimizing $\mathbb{V}[\widehat{Z}_n]$ is thus equivalent to minimizing

$$J(q_n) = \int \frac{\gamma_n(x_{1:n})^2}{q_n(x_{1:n})}\,dx_{1:n},$$

subject to the constraint

$$\int q_n(x_{1:n})\,dx_{1:n} = 1.$$

We now introduce a Lagrange multiplier λ and form the Lagrangian

$$L(q_n, \lambda) = \int \frac{\gamma_n(x_{1:n})^2}{q_n(x_{1:n})}\,dx_{1:n} + \lambda\left(\int q_n(x_{1:n})\,dx_{1:n} - 1\right).$$

Taking the functional derivative with respect to $q_n(x_{1:n})$ and using chain rule we have

$$\frac{\delta L}{\delta q_n(x_{1:n})} = -\frac{\gamma_n(x_{1:n})^2}{q_n(x_{1:n})^2} + \lambda = 0.$$

Solving for $q_n(x_{1:n})$ yields

$$q_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{\sqrt{\lambda}}.$$

Enforcing the normalization condition we get

$$\int \frac{\gamma_n(x_{1:n})}{\sqrt{\lambda}}\,dx_{1:n} = 1 \quad \Longrightarrow \quad \frac{Z_n}{\sqrt{\lambda}} = 1,$$

so that $\sqrt{\lambda} = Z_n$. Therefore,

$$q_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n} = \pi_n(x_{1:n}).$$

This completes the proof. □

2.2 Sequential Importance Sampling

Sequential importance sampling (SIS) builds upon the basic idea of IS by exploiting a Markov structure of the importance distribution. Instead of sampling the full trajectory at once, SIS extends the particle trajectories sequentially, updating the importance weights recursively. This leads to significant computational savings.

We let our importance distribution have a Markov structure, that is

$$q_n(x_{1:n}) = q_1(x_1)q_2(x_2|x_1)\dots q_n(x_k|x_{1:k-1}),$$

and each element in the set $\{x_{1:t}\}$ we will refer to as a particle. For $n \geq 2$ define the *incremental importance weight* as

$$\alpha_n(x_{1:n}) \coloneqq \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1})q_n(x_n|x_{1:n-1})}.$$

The unnormalized weights can then be written recursively as

$$\begin{aligned} w_n(x_{1:n}) &= \frac{\gamma_n(x_{1:n})}{q_n(x_{1:n})} \\ &= \frac{\gamma_{n-1}(x_{1:n-1})}{q_{n-1}(x_{1:n-1})} \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1})q_n(x_n|x_{1:n-1})} \\ &= w_{n-1}(x_{1:n-1}) \cdot \alpha_n(x_{1:n}) \\ &= w_1(x_1) \prod_{k=2}^{n} \alpha_k(x_{1:k}). \end{aligned} \tag{2.7}$$

Summarizing, the SIS method is described in Algorithm 2.3, which has a time complexity of $O(NT)$, since the algorithm consists of two nested loops, one over the number of particles N, and one over the number of time steps T. Within the two nested loops, each operation is $O(1)$.

Algorithm 2.3 Sequential Importance Sampling (SIS)

1: Generate particles $x_1^{(i)}$ from $q_1(x_1)$ for $i = 1, \dots, N$
2: **for** each time step $n = 2, \dots, T$ **do**
3: **for** each particle $i = 1, \dots, N$ **do**
4: Generate particles $x_n^{(i)}$ from $q_n(x_n|x_{1:n-1}^{(i)})$
5: Compute the incremental importance weight:

$$\alpha_n^{(i)} = \frac{\gamma_n(x_{1:n}^{(i)})}{\gamma_{n-1}(x_{1:n-1}^{(i)})q_n(x_n^{(i)}|x_{1:n-1}^{(i)})}$$

6: Update the particle weight: $w_n^{(i)} = w_{n-1}^{(i)} \cdot \alpha_n^{(i)}$
7: **end for**
8: Normalize the weights: $W_n^{(i)} \leftarrow \frac{w_n^{(i)}}{\sum_{i=1}^{N} w_n^{(i)}}$
9: **end for**

We note, that this algorithm has time complexity $O(NT)$.

However, this method has a severe drawback, that the relative estimated variance $\mathbb{V}(\widehat{Z}_n)/Z_n^2$ increases exponentially in n even in simple examples. Example 2.4 illustrates this issue.

Example 2.4. Consider the case where $\mathcal{X} = \mathbb{R}$ and let the density $\pi_n(x_{1:n})$ be given by

$$\pi_n(x_{1:n}) = \prod_{k=1}^{n} \pi_n(x_k) = \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{x_k^2}{2}\Big).$$

Thus, the unnormalized density $\gamma_n(x_{1:n})$ is given by

$$\gamma_n = \prod_{k=1}^{n} \exp\Big(-\frac{x_k^2}{2}\Big),$$

and the normalizing constant Z_n is

$$Z_n = (2\pi)^{n/2}.$$

Ignoring that we could easily sample from π_n, we select the importance distribution q_n to sample from as

$$q_n(x_{1:n}) = \prod_{k=1}^{n} \pi_n(x_k) = \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{x_k^2}{2\sigma^2}\Big)$$

and we are interested in estimating Z_n. Recall from Theorem 2.1 that the relative variance of $\widehat{Z}_n$ is given by

$$\frac{\mathbb{V}(\widehat{Z}_n)}{Z_n^2} = \frac{1}{N}\Big(\int \frac{\pi_n^2(x_{1:n})}{q_n(x_{1:n})}\, dx_{1:n} - 1\Big).$$

In our case this factors over the coordinates and we have

$$\begin{aligned}
\int \frac{\pi_n(x_{1:n})^2}{q_n(x_{1:n})}\, dx_{1:n} &= \prod_{k=1}^{n} \int \frac{1/(2\pi)\exp(-x^2)}{1/\sqrt{2\pi\sigma^2}\exp(-x^2/(2\sigma^2))}\, dx_{1:k} \\
&= \left[\frac{\sqrt{2\pi\sigma^2}}{2\pi} \int \exp\Big(-x^2 + \frac{x^2}{2\sigma^2}\Big)\, dx\right]^n \\
&= \left[\frac{\sqrt{2\pi\sigma^2}}{2\pi} \int \exp\Big(-\Big(1 - \frac{1}{\sigma^2}\Big)x^2\Big)\, dx\right]^n.
\end{aligned}$$

The integral is finite if and only if $1 - 1/(2\sigma^2) > 0$, that is $\sigma^2 > 1/2$. In this case, it is equal to

$$\int_{-\infty}^{\infty} \exp\Big(-\Big(1 - \frac{1}{\sigma^2}\Big)x^2\Big)\, dx = \sqrt{\frac{\pi}{1 - 1/(2\sigma^2)}}.$$

Thus, for $\sigma^2 > 1/2$ we have

$$\begin{aligned}
\int \frac{\pi_n(x_{1:n})^2}{q_n(x_{1:n})}\, dx_{1:n} &= \left(\frac{\sqrt{2\pi\sigma^2}}{2\pi}\sqrt{\frac{\pi}{1 - 1/(2\sigma^2)}}\right)^n \\
&= \left(\sqrt{\frac{\sigma^4}{2\sigma^2 - 1}}\right)^n \\
&= \Big(\frac{\sigma^4}{2\sigma^2 - 1}\Big)^{n/2}.
\end{aligned}$$

Finally, we can conclude that for $\sigma^2 > 1/2$ the relative variance becomes $\mathbb{V}[\widehat{Z}_n] < \infty$ and

$$\frac{\mathbb{V}[\widehat{Z}_n]}{Z_n^2} = \frac{1}{N}\left[\Big(\frac{\sigma^4}{2\sigma^2 - 1}\Big)^{n/2} - 1\right].$$

Note that for any $1/2 < \sigma^2 \neq 1$ we have that $\sigma^4/(2\sigma^2 - 1) > 1$, and thus the variance increases exponentially with n.

For example, choosing $\sigma^2 = 1.2$ then we have a reasonably good importance distribution as $q_k(x_k) \approx \pi_n(x_k)$. However,

$$N\,\mathbb{V}[\widehat{Z}_n]/Z_n^2 \approx (1.103)^{n/2},$$

which for $n = 1000$ is roughly equal to $1.9 \cdot 10^{21}$. So we would need to use $N \approx 2 \cdot 10^{23}$ particles to obtain a relative variance of 0.01.

2.3 Resampling

Over time, many particles receive negligible weight, leading to a situation where only a few particles dominate the estimate. This phenomenon, known as weight degeneracy, can degrade the performance of the estimator (Doucet et al. [2000]). The idea for resampling is to get rid of particles with low weights with a high probability, so the focus is spent on high-probability regions instead of carrying forward particles with very low weights. This typically reduces the variance, see for instance Example 2.7.

The IS approximation $\widehat{\pi}_n(x_{1:n})$ of the target distribution $\pi_n(x_{1:n})$ is constructed using weighted samples drawn from $q_n(x_{1:n})$, meaning that these samples are not distributed according to $\pi_n(x_{1:n})$. To obtain a new set of samples that better approximates $\pi_n(x_{1:n})$, we employ resampling, where each particle $X_{1:n}^{(i)}$ is selected with probability proportional to its normalized weight $W_n^{(i)}$.

A simple resampling strategy is multinomial resampling, where the number of offspring $N_n^{(i)}$ assigned to each particle $X_{1:n}^{(i)}$ follows a multinomial distribution:

$$N_n^{(1:N)} = (N_n^{(1)}, \ldots, N_n^{(N)}) \sim \text{Multinomial}(N, W_n^{(1:N)}).$$

That is, each particle is independently resampled N times, with probabilities given by the normalized weights $W_n^{(i)}$.

After resampling, all particles are assigned equal weight, since the resampled particles are now an unweighted representation of the target distribution. Each selected particle appears with frequency $N_n^{(i)}$, and thus the empirical distribution assigns equal mass to all N retained particles. That is, the original weighted particle approximation of the target distribution is

$$\hat{p}(x) = \sum_{i=1}^{N} W_n^{(i)} \delta_{X_{1:n}^{(i)}}(x),$$

and after resampling, the new approximation is

$$\hat{p}^*(x) = \sum_{i=1}^{N} \frac{N_n^{(i)}}{N} \delta_{X_{1:n}^{(i)}}(x).$$

Taking expectations we obtain

$$\mathbb{E}\left[\hat{p}^*(x)\right] = \sum_{i=1}^{N} \mathbb{E}\left[\frac{N_n^{(i)}}{N}\right] \delta_{X_{1:n}^{(i)}}(x) = \sum_{i=1}^{N} W_n^{(i)} \delta_{X_{1:n}^{(i)}}(x) = \hat{p}(x),$$

showing that the resampled particle system should have equal weights to be unbiased.

Using the recursive structure of the unnormalized weights from Equation (2.7) a natural way to estimate Z_n is to define

$$\widetilde{Z}_1 \coloneqq \frac{1}{N} \sum_{i=1}^{N} w_1(x_1^i),$$

and for $n \geq 2$ estimate Z_n recursively by

$$\widetilde{Z}_n \coloneqq \widetilde{Z}_{n-1} \alpha_n^{\mathrm{MC}}, \tag{2.8}$$

where α_n^{MC} is the standard MC estimate of α_n.

A generic sequential importance sampling with resampling (SISR) algorithm is given in Algorithm 2.5. The time complexity of this algorithm is $O(NT)$, since it consists of two nested loops, one over the number of particles N, and one over the number of time steps T. Within the nested loops, each operation is $O(1)$. The resampling step, which involves drawing N indices according to the particle weights, is done within the T-loop and has a time complexity of $O(N)$. Therefore, the overall time complexity is $O(NT + NT) = O(NT)$.

Algorithm 2.5 Sequential Importance Sampling with Resampling (SISR)

1: Generate particles $x_1^{(i)}$ from $q_1(x_1)$ for $i = 1, \ldots, N$
2: **for** each time step $n = 2, \ldots, T$ **do**
3: **for** each particle $i = 1, \ldots, N$ **do**
4: Generate particles $x_n^{(i)}$ from $q_n(x_n | x_{1:n-1}^{(i)})$
5: Compute the incremental importance weight:

$$\alpha_n^{(i)} = \frac{\gamma_n(x_{1:n}^{(i)})}{\gamma_{n-1}(x_{1:n-1}^{(i)}) q_n(x_n^{(i)} | x_{1:n-1}^{(i)})}$$

6: Update the particle weight: $w_n^{(i)} = w_{n-1}^{(i)} \cdot \alpha_n^{(i)}$
7: **end for**
8: Normalize the weights: $W_n^{(i)} \leftarrow \frac{w_n^{(i)}}{\sum_{i=1}^{N} w_n^{(i)}}$
9: Draw N indices $\{a^{(i)}\}_{i=1}^{N}$ from $\{1, \ldots, N\}$ according to the probabilities $\{W_n^{(i)}\}$
10: Set $X_{1:n}^{(i)} \leftarrow X_{1:n}^{(a^{(i)})}$ for all i
11: Reset weights: $w_n^{(i)} = 1$
12: **end for**

While resampling effectively eliminates low-weight particles, it also causes many distinct trajectories to vanish over successive iterations. In effect, resampling resets the system by providing a reliable approximation of the current state's marginal distribution, albeit at the cost of losing detailed ancestral information. The deeper problem of weight degeneracy

is that trying to represent a high-dimensional distribution with a finite number of samples will inevitably fail.

An alternative would be to increase the number of particles at each iteration; however, this approach quickly becomes infeasible due to the exponential growth in the required number of particles (Doucet and Johansen [2009]).

Now we provide a result similar to Theorem 2.1 for the case with resampling.

Theorem 2.6 (Relative Asymptotic Variance with Resampling)**.** The relative asymptotic variance of the IS estimate of the normalizing constant Z_n with resampling at every time step is

$$\frac{\mathbb{V}(\widetilde{Z}_n)}{Z_n^2} = \frac{1}{N}\left[\left(\int \frac{\pi_1^2(x_1)}{q_1(x_1)}\,dx_1 - 1\right) + \sum_{k=2}^{n}\left(\int \frac{\pi_k^2(x_{1:k})}{\pi_{k-1}(x_{1:k-1})\,q_k(x_k|x_{1:k-1})}\,dx_{k-1:k} - 1\right)\right].$$

A proof is omitted but follows from the Feynman–Kac framework; see, e.g., Chapter 9 in Moral [2004] for a proof.

Notably, comparing Theorem 2.6 with Theorem 2.1 reveals that while resampling introduces additional variance, the resampling has a resetting property of the systems. Thus, we get that the associated errors accumulate linearly rather than multiplicatively. This becomes important when the dimension becomes large. We continue Example 2.4 using Theorem 2.6 to highlight this.

Example 2.7 (Example 2.4 continued)**.** Using Theorem 2.6 and using the derivation done in Example 2.4 we have that it is finite for $\sigma^2 > 1/2$ and the relative variance using resampling at every time step is approximately equal to

$$\begin{aligned}\frac{\mathbb{V}(\widetilde{Z}_n)}{Z_n^2} &\approx \frac{1}{N}\left[\left(\int \frac{\pi_1^2(x_1)}{q_1(x_1)}\,dx_1 - 1\right) + \sum_{k=2}^{n}\left(\int \frac{\pi_k^2(x_{1:k})}{\pi_{k-1}(x_{1:k-1})\,q_k(x_k|x_{1:k-1})}\,dx_{k-1:k} - 1\right)\right] \\ &= \frac{n}{N}\left[\left(\frac{\sigma^4}{2\sigma^2 - 1}^{1/2}\right) - 1\right]\end{aligned}$$

which is linear in n in contrast to the exponential growth of the IS estimate of the relative variance. If we again select $\sigma^2 = 1.2$ then to obtain a relative variance of 0.01 we only need $N \approx 10^4$ particles instead of the $N \approx 2 \cdot 10^{23}$ particles that were needed for the IS estimate to obtain the same precision. That is, we obtain an improvement by 19 orders of magnitude.

This setup favors SMC massively since the density $\pi_n(x_{1:n})$) factorizes. A more realistic example is given in Example 3.1.

Reducing variance of Resampling

While resampling helps mitigate weight degeneracy by focusing computational effort on high-probability regions, it introduces additional variance into the algorithm. We describe two distinct techniques that can reduce this extra variance, thereby improving the overall efficiency of the SISR algorithm. Notably, these approaches are complementary and can be applied simultaneously.

We can reduce the variance introduced during resampling by changing the sampling scheme. Several methods exists, but we will focus on stratified resampling, a method

often used in survey sampling Kiderlen [2022]. In stratified resampling, the interval $[0, 1]$ is divided into N equal strata, and one uniform random number is drawn from each sub-interval. That is, for $i = 1, \ldots, N$, we draw

$$u_i \sim \text{Uniform}\left(\frac{i-1}{N}, \frac{i}{N}\right).$$

Each u_i is then used to select a particle based on the cumulative normalized weights. In Douc et al. [2005] it was shown that stratified resampling always gives lower variance than multinomial resampling.

Another way to reduce the variance of the SISR algorithm is to only do the resampling step when we have many particles with low weights. We will call this sequential Monte Carlo with adaptive resampling (SISAR). A common metric used to decide when to trigger a resampling step is the *effective sample size* (ESS), first introduced by Liu and Chen [1995]. The ESS at time n is defined as

$$\text{ESS}_n \coloneqq \frac{1}{\sum_{i=1}^{N} \left(W_n^{(i)}\right)^2}.$$

ESS can take a value between 1 and N. When the ESS falls below a predetermined threshold, N_τ, often chosen as $N_\tau = N/2$, it is an indication that most of the weight is carried by only a few particles, and resampling is then warranted.

Thus, the SISAR Algorithm is just the SISR Algorithm described in algorithm 2.5 where the resampling is only done when the resampling condition is met. A generic algorithm incorporating adaptive resampling is described in Algorithm 2.8. The time complexity remains $O(NT)$.

Algorithm 2.8 Sequential Importance Sampling with Adaptive Resampling (SISAR)

1: Generate particles $x_1^{(i)}$ from $q_1(x_1)$ for $i = 1, \ldots, N$
2: **for** each time step $n = 2, \ldots, T$ **do**
3: **for** each particle $i = 1, \ldots, N$ **do**
4: Generate particles $x_n^{(i)}$ from $q_n(x_n | x_{1:n-1}^{(i)})$
5: Compute the incremental importance weight:

$$\alpha_n^{(i)} = \frac{\gamma_n(x_{1:n}^{(i)})}{\gamma_{n-1}(x_{1:n-1}^{(i)}) q_n(x_n^{(i)} | x_{1:n-1}^{(i)})}$$

6: Update the particle weight: $w_n^{(i)} = w_{n-1}^{(i)} \cdot \alpha_n^{(i)}$
7: **end for**
8: Normalize the weights: $W_n^{(i)} \leftarrow \frac{w_n^{(i)}}{\sum_{i=1}^{N} w_n^{(i)}}$
9: **if** Resampling condition is met **then**
10: Draw N indices $\{a^{(i)}\}_{i=1}^{N}$ from $\{1, \ldots, N\}$ according to the probabilities $\{W_n^{(i)}\}$
11: Set $x_{1:n}^{(i)} \leftarrow x_{1:n}^{(a^{(i)})}$ for all i
12: Reset weights: $w_n^{(i)} = 1$
13: **end if**
14: **end for**

Again, we can at any step n estimate Z_n by Equation (2.8).

3 State-Space Models

This chapter builds on the work of Doucet and Johansen [2009] and Kantas et al. [2015] ... cite

Suppose we have a hidden Markov model (HMM), also called a state-space model (SSM). Specifically, we consider a discrete-time Markov process $\{X_n; n \geq 1\}$, where each X_n takes values in $\mathcal{X}$. This process is characterized by an initial distribution

$$X_1 \sim \mu(x_1)$$

and the transition density

$$X_{n+1}|(X_n = x_n) \sim f_\theta(x_{n+1}|x_n) \tag{3.1}$$

for some parameter $\theta \in \Theta$. Our goal is to infer the latent states $\{X_n\}$ (or θ if it is unknown), given a sequence of noisy observations $\{Y_n; n \geq 1\}$, where each Y_n takes values in the space $\mathcal{Y}$. The observation Y_n is assumed to be conditionally independent of all other observations and states, given the latent state X_n, and is characterized by

$$Y_n|X_n = x_n \sim g_\theta(y_n|x_n). \tag{3.2}$$

Let $y_{1:T} = (y_1, \ldots, y_T)$ denote the sequence of observations up to time $T \geq 1$. For the rest of this chapter let $\mathcal{X} \subseteq \mathbb{R}^{d_x}$, $\mathcal{Y} \subseteq \mathbb{R}^{d_y}$ and let the densities be with respect to the corresponding Lebesgue measure.

The likelihood function $p_\theta(y_{1:n})$ is given by

$$p_\theta(y_{1:n}) = \int p_\theta(x_{1:n}, y_{1:n})\, dx_{1:n},$$

where $p_\theta(x_{1:n}, y_{1:n})$ is the joint density of $(X_{1:n}, Y_{1:n})$ which by Equations (3.1)-(3.2) is

$$p_\theta(x_{1:n}, y_{1:n}) = \mu_\theta(x_1) \prod_{k=1}^{n} f_\theta(x_k|x_{k-1}) \prod_{k=1}^{n} g_\theta(y_k|x_k).$$

Here, the product over the transition densities $f_\theta(x_k|x_{k-1})$ captures the temporal evolution of the latent states, while the product over the likelihoods $g_\theta(y_k|x_k)$ incorporates the information provided by the observations.

However, the likelihood is often intractable, necessitating Monte Carlo methods. The literature splits this setup up into two problems, filtering and smoothing. We define them as follows:

- *Filtering:* At each time step n, the goal is to sequentially approximate: The joint conditional distribution of the latent states given the observations,

$$p_\theta(x_{1:n} \mid y_{1:n}),$$

and the marginal likelihood,

$$p_\theta(y_{1:n}).$$

So, at time $n = 1$, we approximate $p_\theta(x_1|y_1)$ and $p_\theta(y_1)$; at time $n = 2$, we approximate $p_\theta(x_{1:2}|y_{1:2})$ and $p_\theta(y_{1:2})$, and so on. This sequential framework aligns directly with the setup described in the previous chapter.

- *Smoothing:* The objective in smoothing is to estimate the latent states by using the entire sequence of observations $y_{1:T}$. In particular, approximating the joint conditional distribution

$$p_\theta(x_{1:n} \mid y_{1:T}), \quad n = 1, \ldots, T,$$

and the marginal conditional distributions

$$p_\theta(x_n \mid y_{1:T}), \quad n = 1, \ldots, T.$$

Because smoothing uses future observations, the resulting state estimates are generally more accurate and smoother than those obtained via filtering. This improvement is particularly beneficial when real-time estimation is not required.

That is, the difference between filtering and smoothing is whether the inference is carried out on-line (filtering) or off-line (smoothing). On-line inference is done sequentially when observations become available, and off-line inference uses a fixed number of observations.

For the remainder of this Chapter, we suppose that θ is known.

3.1 Filtering

We can write the posterior density $p_\theta(x_{1:n}|y_{1:n})$ and the likelihood $p_\theta(y_{1:n})$ recursively for $n \geq 2$

$$p_\theta(x_{1:n}|y_{1:n}) = p_\theta(x_{1:n-1}|y_{1:n-1}) \frac{f_\theta(x_n|x_{n-1}) g_\theta(y_n|x_n)}{p_\theta(y_n|y_{1:n-1})}$$

and

$$p_\theta(y_{1:n}) = p_\theta(y_{1:n-1}) p_\theta(y_n|y_{1:n-1})$$

where the predictive likelihood $p_\theta(y_n|y_{1:n-1})$ is given by

$$\begin{aligned} p_\theta(y_n \mid y_{1:n-1}) &= \int p_\theta(y_n, x_n \mid y_{1:n-1})\, dx_n \\ &= \int g_\theta(y_n \mid x_n)\, p_\theta(x_n \mid y_{1:n-1})\, dx_n \\ &= \int g_\theta(y_n \mid x_n) f_\theta(x_n \mid x_{n-1})\, p_\theta(x_{n-1} \mid y_{1:n-1})\, dx_{n-1:n}. \end{aligned}$$

Here we are in the setup discussed in Chapter 2, and we can for example use the SISAR algorithm. We will refer to these as particle filter algorithms.

Example 3.1 (SSM with known θ)**.** Consider the following non-linear Gaussian SSM model where $\theta = (\phi, \sigma_x, \sigma_y)$ is assumed to be known.

$$\begin{aligned} X_1 &\sim N(0,\, 1) \\ X_t &= \phi X_{t-1} + \sin(X_{t-1}) + \sigma_x V_t, \quad V_t \sim N(0,\, 1) \\ Y_t &= X_t + \sigma_y W_t, \quad W_t \sim N(0,\, 1). \end{aligned}$$

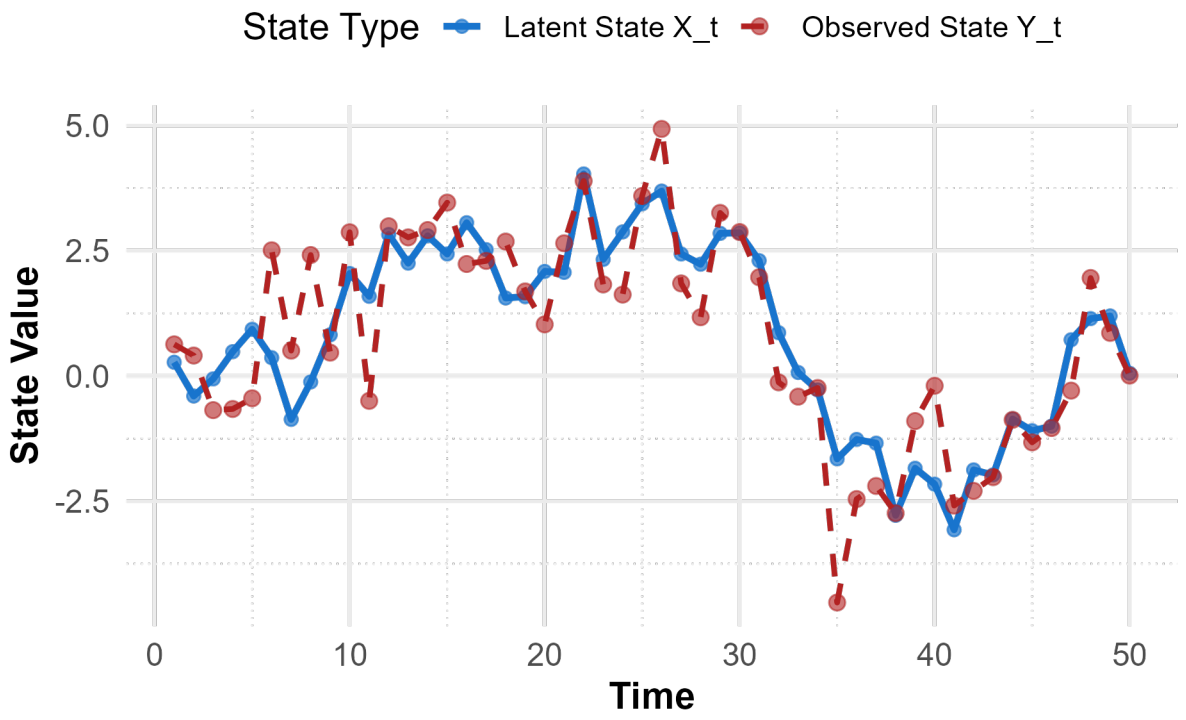

Figure 3.1: A simulated trajectory of the data generating process (DGP) in Example 3.1

Algorithm	SIS	SISR	SISAR
RMSE	1.08 (0.18)	0.75 (0.09)	0.75 (0.09)

Table 3.1: Comparison of RMSE performance for SIS, SISR, and SISAR algorithms based on 10,000 Monte Carlo replications for Example 3.1. The values are the estimated means (standard deviations).

We set $\phi = 0.7$, $\sigma_x = 1$, $\sigma_y = 1$, with a time horizon of $T = 50$, and $N = 1000$ particles. A single simulation of the SSM is visualized in Figure 3.1.

Our goal is to compare the performance of SIS, SISR, and SISAR for estimating the latent state X_t. Performance is evaluated using the root mean squared error (RMSE), with the estimates obtained from 10,000 Monte Carlo replications. For SISR and SISAR we use stratified sampling during the resampling step. The R code for this simulation is available at `https://github.com/BjarkeHautop/master-thesis/tree/main/R`.

The results are visualized in Figure 3.2 and in Table 3.1, expressed as mean (standard deviation). We see that SIS performs the worst, while SISR and SISAR almost gives the same estimates. However, SISAR is computationally more efficient since it does not resample at every time step. The poorer performance of SIS is due to weight degeneracy, where almost all the weight is carried by a few particles.

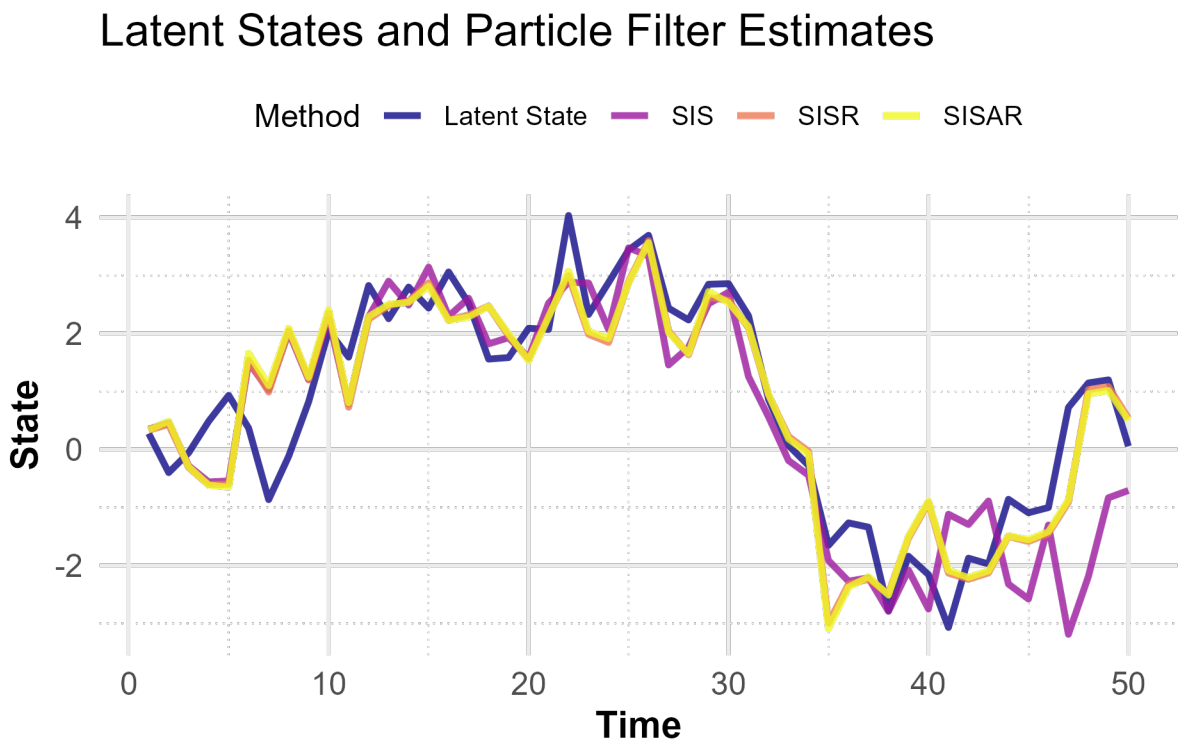

Figure 3.2: The particle filter estimates of the latent state of the DGP in Example 3.1. Note, that SISR and SISAR almost produces the same estimates.

3.2 Smoothing

Forward-Backward Recursions

By doing a decomposition of the joint distribution $p_\theta(x_{1:T}|y_{1:T})$ we see that conditional on $y_{1:T}$ that $\{X_n\}$ is an non-homogeneous Markov process

$$\begin{aligned} p_\theta(x_{1:T}|y_{1:T}) &= p_\theta(x_{1:T}|y_{1:T}) \prod_{n=1}^{T-1} p_\theta(x_n|x_{n+1}, y_{1:T}) \\ &= p_\theta(x_T|y_{1:T}) \prod_{n=1}^{T-1} p_\theta(x_n|x_{n+1}, y_{1:n}), \end{aligned} \tag{3.3}$$

where we in the 2nd equality used the Markov property. Using Bayes' Theorem, we can write the marginal $p_\theta(x_n|x_{n+1}, y_{1:n})$ as

$$p_\theta(x_n|x_{n+1}, y_{1:n}) = \frac{f_\theta(x_{n+1}|x_n) p_\theta(x_n|y_{1:n})}{p_\theta(x_{n+1}|y_{1:n})}. \tag{3.4}$$

By integrating out $(x_{1:n-1}, x_{n+1:T})$ in Equation (3.3) we have that the marginal distribution $p_\theta(x_n|y_{1:T})$ is given by

$$\begin{aligned} p_\theta(x_n|y_{1:T}) &= \int p_\theta(x_{1:T}|y_{1:T})\, dx_{1:n-1}\, dx_{n+1:T} \\ &= \int p_\theta(x_T|y_{1:T}) \prod_{k=1}^{T-1} p_\theta(x_k|x_{k+1}, y_{1:k})\, dx_{1:n-1}\, dx_{n+1:T} \\ &= \int p_\theta(x_T|y_{1:T}) \prod_{k=1}^{T-1} \frac{f(x_{k+1}|x_k)p_\theta(x_k|y_{1:k})}{p_\theta(x_{k+1}|y_{1:k})}\, dx_{1:n-1}\, dx_{n+1:T} \\ &= p_\theta(x_n|y_{1:n}) \int \frac{f(x_{n+1}|x_n)}{p_\theta(x_{n+1}|y_{1:n})} p_\theta(x_{n+1}|y_{1:T})\, dx_{n+1}. \end{aligned} \tag{3.5}$$

This is referred to as forward-backward smoothing, as a forward pass yields $\{p_\theta(x_n|y_{1:n})\}_{n=1}^T$, which can then be used in a backward pass to obtain $\{p_\theta(x_n|y_{1:T})\}_{n=1}^T$.

Forward-Filtering Backward-Sampling (FFBSm)

A common method for drawing samples from the joint smoothing distribution $p_\theta(x_{1:T}|y_{1:T})$ is the Forward-Filtering Backward-Sampling with marginalization (FFBSm) method. This method proceeds in two steps:

1. **Forward filtering:** The filtering distributions $\{p_\theta(x_n|y_{1:n})\}_{n=1}^T$ are approximated using a filtering method, such as those described in Chapter 2. In this step, particles are used to approximate the filtering distributions, with each particle $x_n^{(j)}$ associated with a weight $W_n^{(j)}$, where $W_n^{(j)}$ is the normalized weight of the particle $x_n^{(j)}$. The weight reflects how well the particle approximates the target distribution $p_\theta(x_n \mid y_{1:n})$. These weights are essential for guiding the importance of each particle in subsequent sampling steps.

2. **Backward sampling:**

   - First, we sample the final state X_T from the distribution $p_\theta(x_T \mid y_{1:T})$, which is proportional to the forward-filtered particles at time T. Since the forward filter gives a weighted sample $\{x_T^{(j)}\}$ we sample $X_T^{(i)}$ with probability $W_T^{(j)}$.
   - For each time step $n = T-1, T-2, \ldots, 1$, we sample the state X_n given the sampled state X_{n+1} and the observations $y_{1:n}$. The sampling procedure is based on the backward conditional distribution $p_\theta(x_n \mid X_{n+1}, y_{1:n})$, which is proportional to the joint likelihood of the transition from x_n to x_{n+1} and the forward filtering distribution. More formally, we have:

     $$p_\theta(x_n \mid X_{n+1}, y_{1:n}) = \frac{f_\theta(X_{n+1} \mid x_n) p_\theta(x_n \mid y_{1:n})}{p_\theta(X_{n+1} \mid y_{1:n})},$$

     where $f_\theta(X_{n+1} \mid x_n)$ is the state transition density, and $p_\theta(x_n \mid y_{1:n})$ is the forward filtering distribution for state x_n. To sample $X_n^{(i)}$, we first compute the backward weights $\tilde{w}_n^{(j)}$, which adjust the forward weights by the transition probability:

     $$\tilde{w}_n^{(j)} = W_n^{(j)} f_\theta(X_{n+1}^{(j)} \mid x_n^{(j)}).$$

- Finally, we normalize the backward weights:

$$\tilde{W}_n^{(j)} = \frac{\tilde{w}_n^{(j)}}{\sum_{j=1}^{N} \tilde{w}_n^{(j)}}.$$

Then, we sample an index k from $\{1, 2, \ldots, N\}$ with probability $\tilde{W}_n^{(k)}$, and set $X_n^{(i)} = x_n^{(k)}$. This step ensures that the sampled trajectory is consistent with the joint smoothing distribution $p_\theta(x_{1:T} \mid y_{1:T})$.

In summary, the forward weights $W_n^{(j)}$ capture the likelihood of each particle given the observations up to time n, while the backward weights $\tilde{w}_n^{(j)}$ adjust for the transition probabilities between states. The pseudocode for the FFBSm algorithm is provided in Algorithm 3.2. The time complexity of the FFBSm algorithm is $O(N^2T)$. This time complexity arises the backward sampling step, which involves three nested loops. Hence, the total time complexity is $O(N^2T)$). This is significantly slower than the algorithms in the previous chapter that had a complexity of $O(NT)$.

Algorithm 3.2 Forward-Filtering Backward-Sampling (FFBSm)

Input: Observations $y_{1:T}$, number of particles N, initial distribution $\mu(x_1)$, state transition density $f_\theta(x_{n+1} \mid x_n)$, observation density $g_\theta(y_n \mid x_n)$
Forward Filtering:
Use a forward filtering algorithm (described in Chapter 2) to obtain, for each $n = 1, \ldots, T$, the particle approximation $\{x_n^{(j)}, W_n^{(j)}\}_{j=1}^N$ of $p_\theta(x_n \mid y_{1:n})$, where $W_n^{(j)}$ are the normalized weights
for $i = 1, \ldots, N$ **do**
    Backward Sampling:
    Sample index k from $\{1, \ldots, N\}$ with probability $W_T^{(k)}$
    Set $X_T^{(i)} = x_T^{(k)}$
    for $n = T-1, T-2, \ldots, 1$ **do**
        for $j = 1, \ldots, N$ **do**
            Compute backward weights:

$$\tilde{w}_n^{(j)} = W_n^{(j)} \, f_\theta\Big(X_{n+1}^{(i)} \mid x_n^{(j)}\Big)$$

        end for
        Normalize: $\tilde{W}_n^{(j)} \leftarrow \frac{\tilde{w}_n^{(j)}}{\sum_{j=1}^{N} \tilde{w}_n^{(j)}}$
        Sample index k from $\{1, \ldots, N\}$ with probability $\tilde{W}_n^{(k)}$
        Set $X_n^{(i)} = x_n^{(k)}$
    end for
end for
Output: N sampled trajectories $\{X_{1:T}^{(i)}\}_{i=1}^N$

Another approach for smoothing is the Generalised Two-Filter Formula, as discussed in BRESLER [1986].

Example 3.3 (Example 3.1 continued)**.** We continue Example 3.1 with the same model and setup, now performing smoothing using the FFBSm algorithm. The forward pass is carried

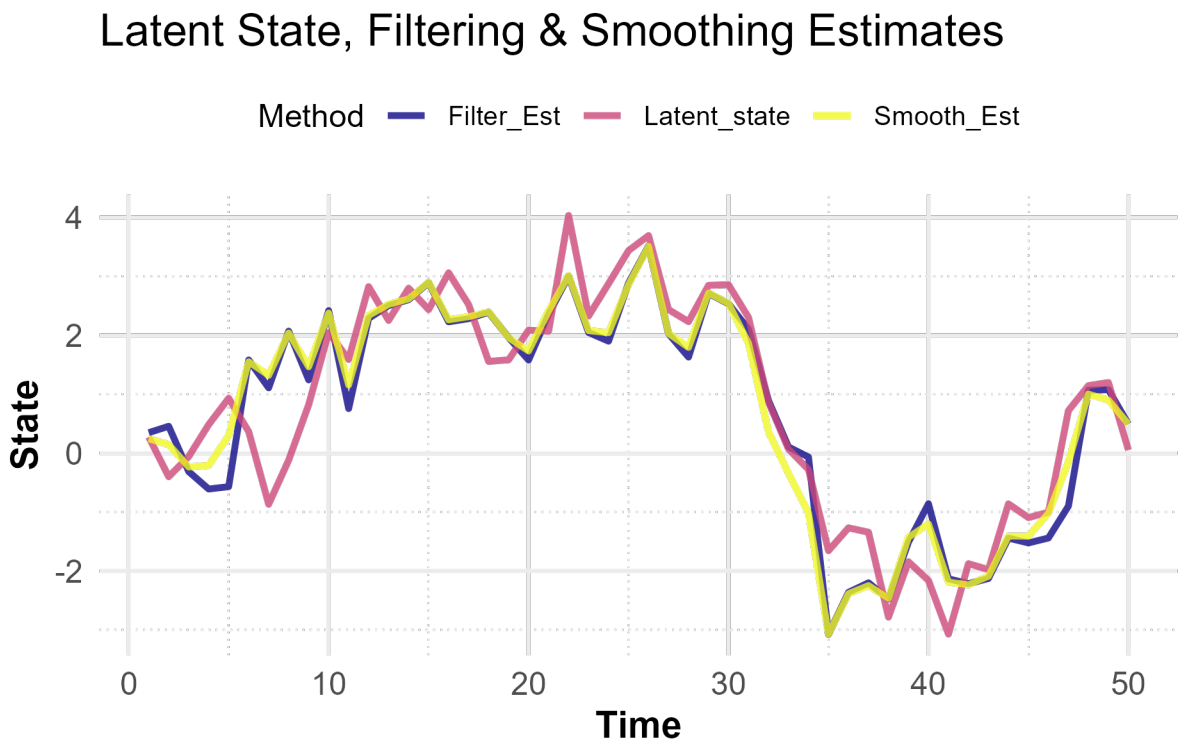

Figure 3.3: A simulated trajectory of the SSM in Example 3.3 alongside the filtering and smoothing estimates.

Algorithm	SIS	SISR	SISAR	FFBSm
RMSE	1.08 (0.18)	0.75 (0.09)	0.75 (0.09)	0.69 (0.08)

Table 3.2: Comparison of RMSE performance for SIS, SISR, SISAR, and FFBSm algorithms based on 10,000 Monte Carlo replications for Example 3.3. The values are the estimated means (standard deviations).

out using the SISAR algorithm with stratified resampling. Due to the computational cost of this approach, particularly the $O(N^2T)$ backward sampling step, the implementation was done in Julia for efficiency. The Julia code is available at `https://github.com/BjarkeHautop/master-thesis/tree/main/Julia`.

An example trajectory of the latent state X_t and the filtering estimate from the SISAR algorithm alongside the smoothed estimate of the FFBSm algorithm is shown in Figure 3.3. We estimate the RMSE and standard deviation based on 10,000 Monte Carlo iterations. Table 3.2 summarizes the results, including values from filtering algorithms for comparison.

4 Bayesian Inference

This chapter builds on the work of Dahlin and Schön [2019], Kantas et al. [2015] and ..., cite
We will purely focus on off-line methods, that is estimating $p(x_{0:T}, \theta | y_{0:T})$.

In many state-space models, both the parameters θ and the latent states $x_{1:T}$ are unknown and must be inferred simultaneously. In the previous chapter, we used the notation $p_\theta(\cdot)$ to indicate a density dependent on the parameter θ. In this chapter we will use the more explicit notation $p(\cdot|\theta)$, as it aligns more naturally with the Bayesian framework.

In a Bayesian framework, a prior distribution $p(\theta)$ is assigned to the parameters, and inference is carried out on the joint posterior distribution given the observations $y_{1:T}$:

$$\begin{aligned} p(\theta, x_{1:T} \mid y_{1:T}) &= \frac{p(y_{1:T}|\theta, x_{1:T})\, p(\theta, x_{1:T})}{p(y_{1:T})} \\ &= \frac{p(y_{1:T}|\theta, x_{1:T})\, p(\theta)\, p(x_{1:T} \mid \theta)}{p(y_{1:T})} \\ &= \frac{p(y_{1:T}|\theta, x_{1:T})\, p(\theta)\, p(x_{1:T} \mid \theta)}{p(y_{1:T})} \\ &\propto p(y_{1:T}|\theta, x_{1:T})\, p(\theta)\, p(x_{1:T} \mid \theta), \end{aligned} \tag{4.1}$$

where

$$p(y_{1:T}) = \int \int p(y_{1:T}|\theta, x_{1:T}) p(\theta) p(x_{1:T}|\theta)\, dx_{1:T}\, d\theta$$

is dropped since it is just a normalizing constant. Using the Markov structure of the SSM we can simplify $p(x_{1:T}|\theta)$ as

$$p(x_{1:T}|\theta) = p(x_1|\theta) \prod_{t=2}^{T} p(x_t|x_{t-1}|\theta).$$

When the latent states are not of primary interest, they can be integrated out to obtain the marginal posterior for the parameters:

$$\begin{aligned} p(\theta \mid y_{1:T}) &= \int p(\theta, x_{1:T} \mid y_{1:T})\, dx_{1:T} \\ &\propto p(\theta)\, p(y_{1:T}|\theta), \end{aligned} \tag{4.2}$$

with the marginal likelihood defined as

$$p(y_{1:T}|\theta) = \int p(x_{1:T}, y_{1:T}|\theta)\, dx_{1:T}.$$

Since neither Equation (4.1) or (4.2) is generally tractable, Markov chain Monte Carlo (MCMC) methods are widely used to approximate the posterior distributions. However,

standard techniques such as one-variable-at-a-time Gibbs sampling tend to perform poorly for non-linear, non-Gaussian state-space models Kantas et al. [2015].

PMCMC algorithms overcome these challenges by leveraging particle methods to construct efficient high-dimensional proposal distributions, thereby enhancing the performance of MCMC in these complex settings.

Particle Marginal Metropolis-Hastings (PMMH)

A Marginal Metropolis–Hastings (MMH) sampler would sample from the joint posterior $p(x_{1:T}, \theta | y_{1:T})$ by using the proposal density

$$q\big((x'_{1:T}, \theta') | x_{1:T}, \theta\big) = q(\theta'|\theta) p_{\theta'}(x'_{1:T} | y'_{1:T}), \tag{4.3}$$

where $q(\theta'|\theta)$ is a proposal density to obtain a candidate parameter θ' from θ, a candidate latent trajectory $x'_{1:T}$ from $x_{1:T}$, and an observation trajectory $y'_{1:T}$ from $y_{1:T}$. The acceptance probability is given by

$$1 \wedge \frac{p_{\theta'}(y_{1:T}) p(\theta') q(\theta|\theta')}{p_{\theta}(y_{1:T}) p(\theta) q(\theta'|\theta)}. \tag{4.4}$$

However, we can neither sample exactly from $p_{\theta'}(x'_{1:T}|y'_{1:T})$ nor compute the likelihood terms $p_{\theta'}(y_{1:T})$ and $p_{\theta}(y_{1:T})$ from the acceptance probability an implementation of MMH is impossible.

A Particle Marginal Metropolis-Hastings (PMMH) sampler is an approximation of the MMH sampler where particle methods are used to approximate these intractable terms. The pseudocode for the algorithm is given in Algorithm 4.1. It was shown in Andrieu et al. [2010] that using an unbiased estimator of $\widehat{p}_{\theta}(y_{1:T})$ produced by the particle filter is sufficient to guarantee that the Markov chain has the correct stationary distribution for any number of particles N.

The proposal density $q(\theta'|\theta)$ is often chosen as a random walk, that is θ'

$$q(\theta'|\theta) \sim N(\theta, \hat{\mathcal{P}}),$$

where $\hat{\mathcal{P}}$ is the estimated posterior covariance based on a pilot run (Dahlin and Schön [2019]).

Algorithm 4.1 Particle Marginal Metropolis-Hastings (PMMH)

1: **Input:** Observation sequence $y_{1:T}$; number of MCMC iterations M; number of particles N; prior density $p(\theta)$; proposal density $q(\theta' \mid \theta)$; and initial parameter value $\theta^{(0)}$.

2: **Initialization:** Set $m \leftarrow 0$. Run a particle filter (as explained in Chapter 2) with parameter $\theta^{(0)}$ to obtain a likelihood estimate $\widehat{p}_{\theta^{(0)}}(y_{1:T})$ and a latent state sample $x_{1:T}^{(0)}$.

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

4: Propose a new parameter: $\theta' \sim q(\theta' \mid \theta^{(m-1)})$.

5: Run a particle filter with θ' to obtain the likelihood estimate $\widehat{p}_{\theta'}(y_{1:T})$ and a latent state sample $x'_{1:T}$.

6: Compute the acceptance probability

$$\alpha = \min\left\{1,\ \frac{p(\theta')\,\widehat{p}_{\theta'}(y_{1:T})\,q(\theta^{(m-1)} \mid \theta')}{p(\theta^{(m-1)})\,\widehat{p}_{\theta^{(m-1)}}(y_{1:T})\,q(\theta' \mid \theta^{(m-1)})}\right\}.$$

7: With probability α, set

$$\theta^{(m)} \leftarrow \theta', \quad x_{1:T}^{(m)} \leftarrow x'_{1:T}, \quad \widehat{p}_{\theta^{(m)}}(y_{1:T}) \leftarrow \widehat{p}_{\theta'}(y_{1:T}).$$

8: Otherwise, set

$$\theta^{(m)} \leftarrow \theta^{(m-1)}, \quad x_{1:T}^{(m)} \leftarrow x_{1:T}^{(m-1)}, \quad \widehat{p}_{\theta^{(m)}}(y_{1:T}) \leftarrow \widehat{p}_{\theta^{(m-1)}}(y_{1:T}).$$

9: **end for**

10: **Return:** The chain $\{\theta^{(m)}, x_{1:T}^{(m)}\}_{m=0}^{M}$.

Recall, that the particle filter algorithms discussed in Chapter 2 had time complexity $O(NT)$, and thus this algorithm has time complexity $O(MNT)$. Thus, we need to decide on how to allocate computational resources for N and M.

The parameter N directly impacts the variance of the likelihood estimator $\widehat{p}_\theta(y_{1:T})$. Increasing N reduces this variance, leading to a more stable acceptance probability. In practice, N is typically determined via pilot runs that monitor the variance of the log-likelihood estimate, with the common guideline being to choose N so that this variance is approximately around 1-1.7 when evaluated at the posterior mean of θ (Pitt et al. [2012] and Dahlin and Schön [2019]).

The number of MCMC iterations M determines the overall length of the Markov chain and hence the quality of the posterior approximation. A larger M facilitates a more thorough exploration of the posterior distribution, allowing for greater precision in the resulting parameter estimates after discarding an appropriate burn-in period. In practice, we often run several independent chains and use convergence diagnostics such as the potential scale reduction statistic, $\hat{R}$, and the effective sample size of each parameter, see also Appendix A.

Example 4.2 (SSM with unknown θ)**.** We consider the same model from Example 3.1

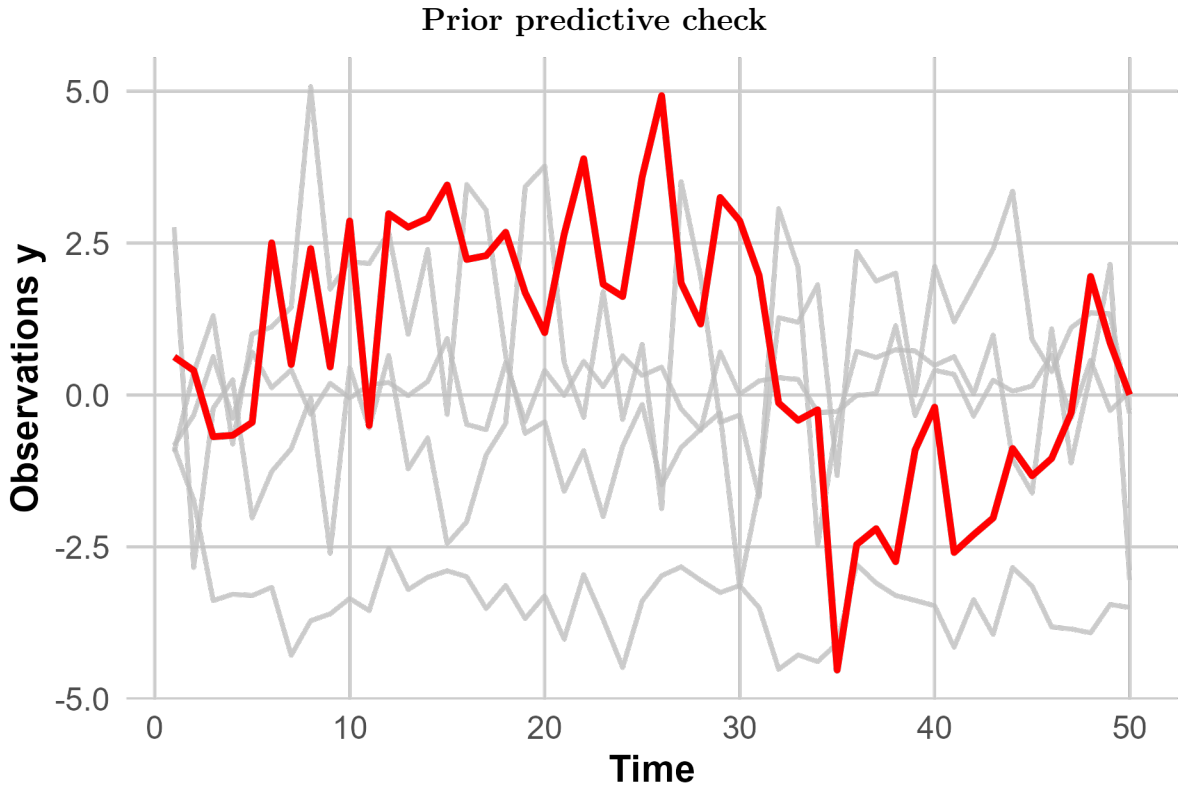

Figure 4.1: A simulated trajectory of the SSM in Example 4.2 alongside four simulated trajectories using the priors.

and Example 3.3, that is we have $\theta = (\phi, \sigma_x, \sigma_y)$ and we have the following SSM

$$\begin{aligned} X_1 &\sim N(0,\, 1) \\ X_t &= \phi X_{t-1} + \sin(X_{t-1}) + \sigma_x V_t, \quad V_t \sim N(0,\, 1) \\ Y_t &= X_t + \sigma_y W_t, \quad W_t \sim N(0,\, 1). \end{aligned}$$

We as before, set $\phi = 0.7, \sigma_x = 1, \sigma_y = 1$, with a time horizon of $T = 50$, and $N = 1000$ particles. We assume θ is unknown. Thus, we need to specify priors on these three parameters. We choose the following generic weakly informative priors

$$\begin{aligned} \phi &\sim N(0, 1), \\ \sigma_x &\sim \text{half-}N(1), \\ \sigma_y &\sim \text{half-}N(1). \end{aligned}$$

We do a prior predictive check, where we generate data from the prior and assess whether it aligns with the observed data. Data simulated from the DGP using the true values $\theta = (0.7, 1, 1)$ and four times from the prior is visualized in Figure 4.1. We see that the priors are reasonable, but note that if $|\phi| > 1$ by the nature of the DGP that the observations can easily explode.

We use the proposal distribution

$$q_{\text{pilot}}(\theta'|\theta) \sim N(\theta, 0.1 \cdot I),$$

for a pilot run with $N_{\text{pilot}} = 100$ particles and $M_{\text{pilot}} = 2000$ MCMC iterations, discarding the first 1000 samples as burn-in, and initializing θ from the priors. From this pilot

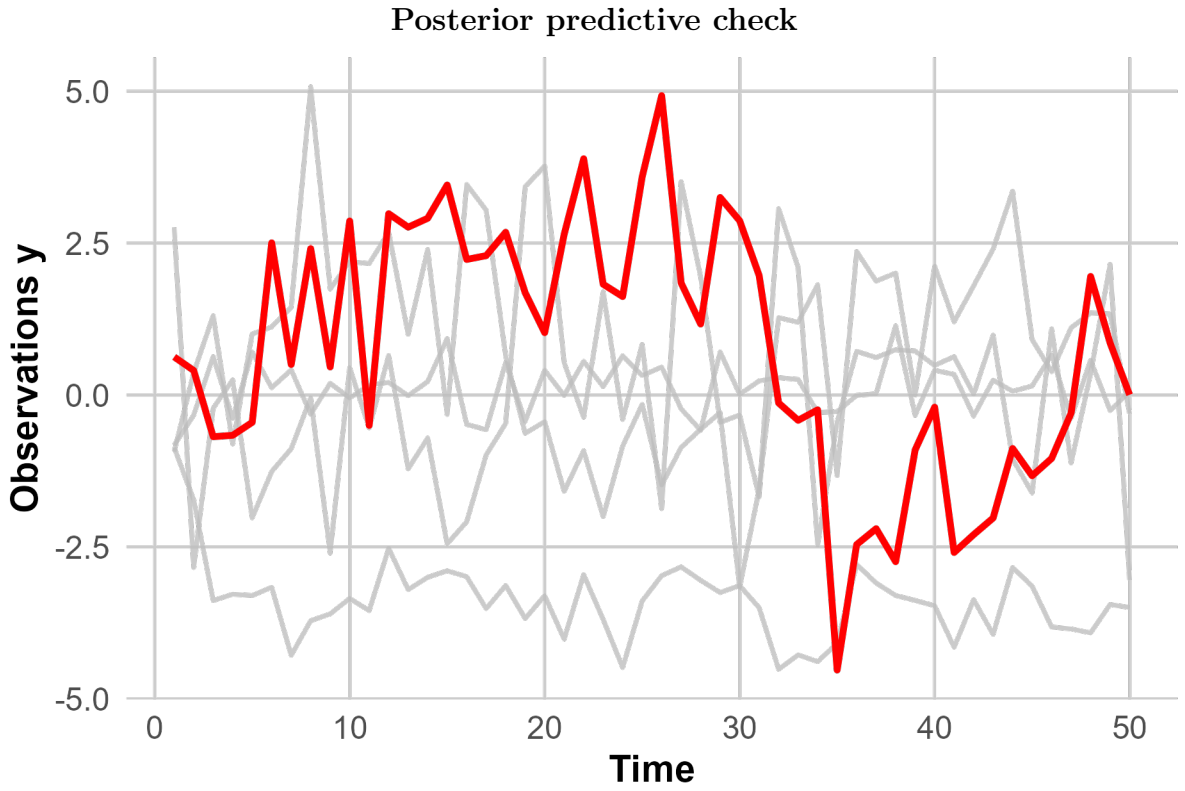

Figure 4.2: A simulated trajectory of the SSM in Example 4.2 alongside four sample paths generated from the posterior predictive distribution.

run, we estimate both the posterior covariance matrix, $\hat{\mathcal{P}}$, and the posterior mean, $\hat{\theta}$. Next, using the estimated posterior mean, we calculate an estimate of the variance of the log-likelihood, denoted by $\widehat{\mathrm{Var}}(\ell)$, by performing 10 repetitions with $N_{\text{pilot}} = 100$ particles each time. Finally, we set

$$N = \max\Big(N_{\text{pilot}} \cdot \widehat{\mathrm{Var}}(\ell),\, 100\Big),$$

so that the log-likelihood has approximately unit variance at the estimated posterior while ensuring that N is at least 100.

We then run four independent MCMC chains with the proposal distribution

$$q(\theta'|\theta) \sim N(\theta, \hat{\mathcal{P}}),$$

for $M = 15000$ MCMC iterations, discarding the first 2000 samples as burn-in, using the value of N from above and initializing θ as $\hat{\theta}$. The R code implementing this can be found at `https://github.com/BjarkeHautop/master-thesis/tree/main/R`; see also Appendix C for further implementation details.

An example chain is shown in Appendix D in Figure D.1, where we see no clear signs that the chain has not converged and is not mixing well. To ensure that the posterior distribution produces data consistent with the observed data, we perform a posterior predictive check, shown in Figure 4.2. The results appear reasonable, with the posterior values aligning well with the observed data.

To ensure that the chains agree and have converged to the same posterior we in Figure Effective Sample Size (ESS) show the density plot of the four chains for ϕ (the

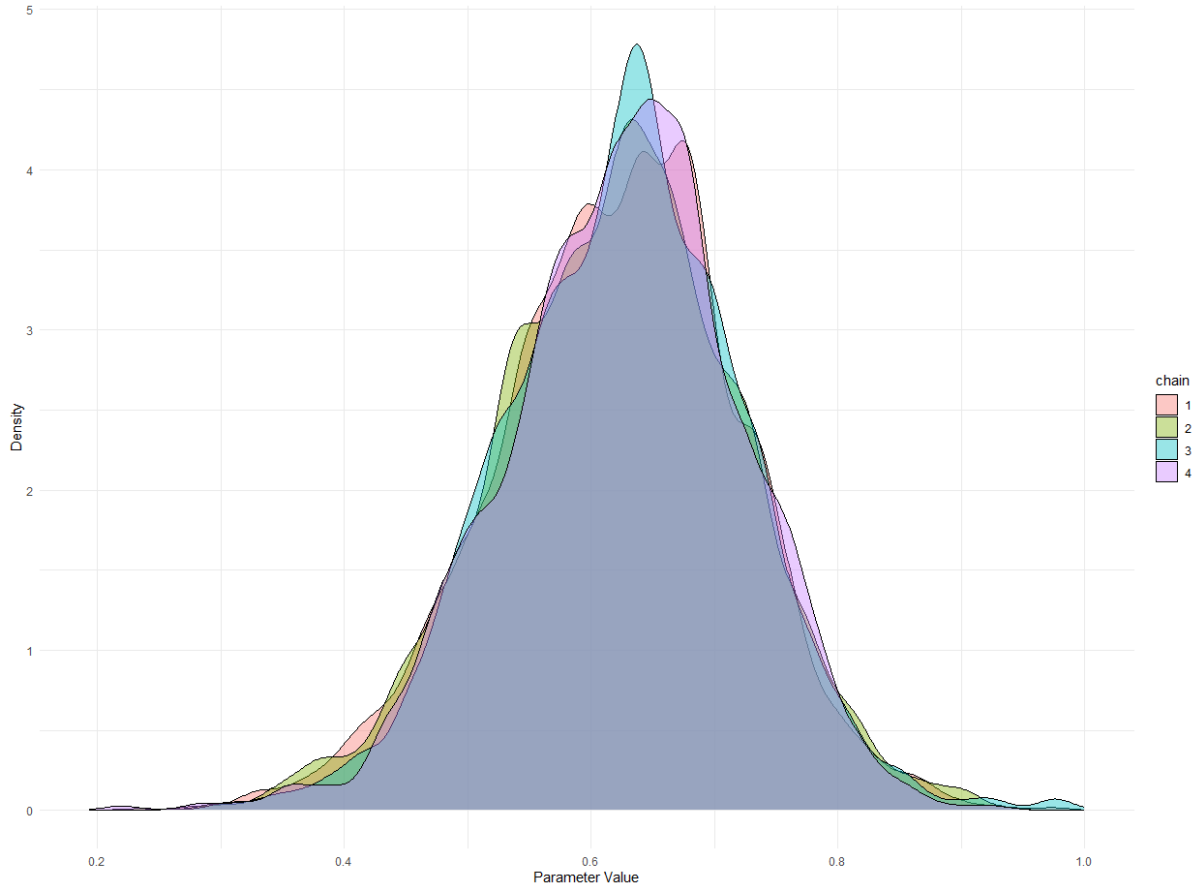

Figure 4.3: Density plot of the four chains for ϕ in Example 4.2. **UPDATE FIGURE TO HIGH QUALITY**

Table 4.1: ESS and split-$\widehat{R}$ for ϕ, σ_x, and σ_y

Parameter	ESS	split-$\widehat{R}$
ϕ	2609	1.002
σ_x	1806	1.002
σ_y	1304	1.003

figures for σ_x and σ_y can be found in D). We see that the four chains look very similar. We furthermore verify that we can trust the inference by computing the split-$\widehat{R}$ statistic and ESS (see Appendix A for definition of these), which is summarized in Table 4.1. We see, that the ESS is much larger than the recommendation of at least 400 and the split-$\widehat{R}$ are all below the recommended 1.01. Thus, we can use the MCMC samples for reliable inference.

The posterior mean and 95% credible intervals are given in Table 4.2. The RMSE of the latent state was 0.79, only slightly worse than the result of the filtering and smoothing algorithms from Table 3.2, where the parameters were known. **ONLY BASED ON 1 SAMPLE**

Table 4.2: Mean and Credible Interval for Parameters

Parameter	Mean	95% Credible Interval
ϕ	0.62	[0.42, 0.81]
σ_x	1.01	[0.64, 1.49]
σ_y	0.89	[0.45, 1.29]

Bibliography

C. Andrieu, A. Doucet, and R. Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(3):269–342, 05 2010. doi: 10.1111/j.1467-9868.2009.00736.x. URL https://doi.org/10.1111/j.1467-9868.2009.00736.x.

Y. BRESLER. Two-filter formulae for discrete-time non-linear bayesian smoothing. *International Journal of Control*, 43(2):629–641, 1986. doi: 10.1080/00207178608933489. URL https://doi.org/10.1080/00207178608933489.

J. Dahlin and T. B. Schön. Getting started with particle metropolis-hastings for inference in nonlinear dynamical models. *Journal of Statistical Software, Code Snippets*, 88(2):1–41, 2019. doi: 10.18637/jss.v088.c02. URL https://www.jstatsoft.org/index.php/jss/article/view/v088c02.

R. Douc, O. Cappé, and E. Moulines. Comparison of resampling schemes for particle filtering, 2005. URL https://arxiv.org/abs/cs/0507025.

A. Doucet and A. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12, 01 2009.

A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, Jul 2000. ISSN 1573-1375. doi: 10.1023/A:1008935410038. URL https://doi.org/10.1023/A:1008935410038.

A. Gelman and D. B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457 – 472, 1992. doi: 10.1214/ss/1177011136. URL https://doi.org/10.1214/ss/1177011136.

A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. ISBN 9781439840955. URL https://books.google.dk/books?id=ZXL6AQAAQBAJ.

C. J. Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, 7(4):473 – 483, 1992. doi: 10.1214/ss/1177011137. URL https://doi.org/10.1214/ss/1177011137.

N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin. On Particle Methods for Parameter Estimation in State-Space Models. *Statistical Science*, 30(3):328 – 351, 2015. doi: 10.1214/14-STS511. URL https://doi.org/10.1214/14-STS511.

M. Kiderlen. Survey sampling and stereology lecture notes, 2022.

D. Kroese, T. Taimre, and Z. Botev. *Handbook of Monte Carlo Methods*. Wiley Series in Probability and Statistics. Wiley, 2013. ISBN 9781118014950. URL https://books.google.dk/books?id=Trj9HQ7G8TUC.

J. S. Liu and R. Chen. Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576, 1995. doi: 10.1080/01621459.1995.10476549. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476549.

P. Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Its Applications. Springer New York, 2004. ISBN 9780387202686. URL https://books.google.dk/books?id=8LypfuG8ZLYC.

M. K. Pitt, R. d. S. Silva, P. Giordani, and R. Kohn. On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012. doi: 10.1016/j.jeconom.2012.06.

C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2004.

Shimao. How is this minimum variance worked out for this importance sampling estimator? Cross Validated, https://stats.stackexchange.com/q/324834, 2018. (Version: 2018-01-24).

Stan Development Team. Stan modeling language users guide and reference manual, version 2.36, 2024. URL https://mc-stan.org.

D. Vats and C. Knudson. Revisiting the Gelman–Rubin Diagnostic. *Statistical Science*, 36(4):518 – 529, 2021. doi: 10.1214/20-STS812. URL https://doi.org/10.1214/20-STS812.

A. Vehtari, A. Gelman, D. Simpson, B. Carpenter, and P.-C. Bürkner. Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*, 16(2):667 – 718, 2021. doi: 10.1214/20-BA1221. URL https://doi.org/10.1214/20-BA1221.

A. Vehtari, D. Simpson, A. Gelman, Y. Yao, and J. Gabry. Pareto smoothed importance sampling, 2024. URL https://arxiv.org/abs/1507.02646.

A MCMC

Markov Chain Monte Carlo (MCMC) methods are used to generate samples from complex probability distributions when direct sampling is infeasible. Let $\pi(x)$ be a density that we want to sample from. An MCMC algorithm constructs a Markov chain $\{X_t\}_{t\geq 1}$ with transition kernel $P(x, A)$ which has $\pi(x)$ as stationary distribution.

One widely used MCMC method is the Metropolis-Hastings algorithm. It generates a sequence of samples from $\pi(x)$ by proposing a candidate x' from a proposal distribution $q(x'|x_t)$ based on the current state x_t. The candidate is accepted with probability:

$$\alpha = \min\left(1, \frac{\pi(x')q(x_t|x')}{\pi(x_t)q(x'|x_t)}\right)$$

If the candidate is accepted, the next state is set to $x_{t+1} = x'$; otherwise, $x_{t+1} = x_t$.

A key property ensuring that the chain has the desired stationary distribution is that the algorithm satisfies the detailed balance condition with respect to $\pi(x)$, that is

$$\pi(x_t)q(x'|x_t)\alpha(x_t, x') = \pi(x')q(x_t|x')\alpha(x', x_t),$$

for all states x_t and x'. Provided that the Markov Chain satisfies some required regularity conditions on the proposal distribution it has the correct stationary distribution. A sufficient condition is that

$$\pi(x) > 0 \implies q(x|x') > 0 \quad \text{for any } x, x',$$

ensuring irreducibility, and that the chain is aperiodic (Robert and Casella [2004]).

In practice, when using MCMC methods for inference the first part of the chain is discarded to allow the chain to converge to the stationary distribution. This is referred to as *burn-in*.

MCMC diagnostics

Since convergence of MCMC is only guaranteed asymptotically, we must rely on diagnostic methods to assess convergence when working with a finite number of samples. Assume that we do MCMC to do inference about a parameter θ, which, for ease of notation, we suppose is a scalar. For models involving multiple parameters, the diagnostic methods described below are applied to each parameter individually.

Potential Scale Reduction

A method to evaluate the convergence of a MCMC chain is to run several independent chains and compare the behavior between them. This is the idea of the *potential scale*

reduction statistic $\widehat{R}$ first introduced in Gelman and Rubin [1992]. The potential scale reduction statistic compares the between-chain variance to the within-chain variance. If all chains are at equilibrium, these will be the same, and $\widehat{R}$ will be one. If they haven't converged to a common distribution, the $\widehat{R}$ statistic will be greater than one.

Suppose we have a set of K Markov chains θ_k which each has M samples $\theta_k^{(m)}$. The between-chain variance estimate is

$$B = \frac{M}{K-1}\sum_{k=1}^{K}(\overline{\theta}_k^{(\cdot)} - \overline{\theta}_\cdot^{(\cdot)})^2,$$

where $\overline{\theta}_k^{(\cdot)}$ is the mean for chain k

$$\overline{\theta}_k^{(\cdot)} = \frac{1}{M}\sum_{m=1}^{M}\theta_k^{(m)},$$

and $\overline{\theta}_\cdot^{(\cdot)}$ is the overall mean of the chains

$$\overline{\theta}_\cdot^{(\cdot)} = \frac{1}{K}\sum_{k=1}^{K}\overline{\theta}_k^{(\cdot)}.$$

The within-chain variance is averaged over the chains,

$$W = \frac{1}{K}\sum_{k=1}^{K}s_k^2,$$

where

$$s_k^2 = \frac{1}{M-1}\sum_{m=1}^{M}(\theta_k^{(m)} - \overline{\theta}_k^{(\cdot)})^2.$$

The variance estimator is given by a mixture of the within-chain and cross-chain sample variances,

$$\widehat{\mathrm{var}}^{+}(\theta) = \frac{M-1}{M}W + \frac{1}{M}B, \tag{A.1}$$

This weighted combination accounts for the uncertainty in both the within-chain and between-chain variances. Finally, we can define the potential scale reduction statistic as

$$\widehat{R} = \sqrt{\frac{\widehat{\mathrm{var}}^{+}(\theta)}{W}}.$$

Gelman et al. [2013] introduced split-$\widehat{R}$ as a more sensitive diagnostic for convergence in MCMC sampling. The method involves splitting each of the K chains into two halves: the first $M/2$ samples and the last $M/2$ samples, giving $2K$ chains. The standard $\widehat{R}$ statistic is then computed across these $2K$ chains. By comparing the two halves of each chain, split-$\widehat{R}$ can detect issues such as slow mixing or nonstationarity within individual chains that might be overlooked when only comparing different chains.

A typical guideline is that $\widehat{R}$ values above 1.01 indicates that further sampling is needed (Stan Development Team [2024] and Vehtari et al. [2021]).

Effective Sample Size

Not the same or related to ESS for resampling condition in particle filter.
An $\widehat{R}$ close to 1 does not guarantee that an MCMC sample is reliable (Vats and Knudson [2021]). A sufficiently large ESS is also required to obtain stable inferences for the quantities of interest. The MCMC samples will typically be positively autocorrelated within a chain. The ESS is the number of independent samples with the same estimation power as the M autocorrelated samples. We follow the definitions given in Stan Development Team [2024] and Vehtari et al. [2021]. We again let K be the number of chains each consistent of M samples and assume all the chains have reached the stationary distribution $p(\theta)$ with mean μ and variance σ^2. The autocorrelation ρ_t at lag $t \geq 0$ is defined as SETUP

$$\begin{aligned}\rho_t &= \frac{1}{\sigma^2}\int (\theta^{(n)} - \mu)(\theta^{(n+t)} - \mu)p(\theta)\, d\theta \\ &= \frac{1}{\sigma^2}\int \theta^{(n)}\theta^{(n+t)}p(\theta)\, d\theta,\end{aligned}$$

where we used that $\theta^{(n)}$ and $\theta^{(n+t)}$ have the same stationary distribution. We then define the effective sample size M_{eff} of M samples by

$$M_{\text{eff}} = \frac{M}{1 + 2\sum_{t=1}^{\infty}\rho_t}.$$

For independent draws (i.e., $\rho_t = 0$ for $t \geq 1$), we recover $M_{\text{eff}} = M$. When draws are positively correlated, however, M_{eff} is smaller, reflecting the reduced information content of the chain, while if they were negatively correlated, the effective sample size will exceed the number of iterations. The ESS is a particular important measure, since the standard error of the estimate of a parameter decreases by $1/\sqrt{M_{\text{eff}}}$ and not $1/\sqrt{M}$.

In practice, the integral of the joint distribution $p(\theta)$ is intractable and thus we need to estimate the effective sample size. We can estimate the autocorrelation ρ_t by

$$\hat{\rho}_t = 1 - \frac{W - \frac{1}{K}\sum_{k=1}^{K}\hat{\rho}_{t,k}}{\widehat{\text{var}}^{+}(\theta)},$$

where $\hat{\rho}_{t,k}$ is an estimate of the autocorrelation at lag t for the kth Markov chain, and $\widehat{\text{var}}^{+}(\theta)$ is defined in Equation (A.1). Because of the increased noise of $\hat{\rho}1_t$ as t increases, in practice a truncated sum of $\hat{\rho}_t$ is used. We apply Geyer's initial monotone sequence criterion, as defined in Geyer [1992], to ensure stability. The effective sample size is estimated by

$$\widehat{M}_{\text{eff}} = \frac{KM}{\hat{\tau}},$$

where

$$\hat{\tau} = 1 + 2\sum_{t=1}^{2k+1}\hat{\rho}_t = 1 + 2\sum_{t=0}^{m}\hat{P}_t - \hat{\rho}_0 = -1 + 2\sum_{t=0}^{m}\hat{P}_t,$$

where $\hat{P}_t = \hat{\rho}_{2t} + \hat{\rho}_{2t+1}$. Summing over pairs starting from lag 0 ensures that the sequence $\hat{P}_t$ values is non-negative and non-increasing (Geyer [1992]). So, if we observe negative estimates of the autocorrelations it is due to finite-sample noise. Thus, we define an initial positive sequence by choosing the largest m such that $\hat{P}_t > 0$ for all $t \in \{1, \ldots, m\}$. We also enforce monotonicity by modifying the sequence $\hat{P}_t$ so that it does not exceed the smallest preceding value, ensuring a non-increasing sequence.

B Bayesian model validation

This Chapter contains some common tools used for validating a Bayesian model.

Prior predictive check

Before fitting a Bayesian model it is useful to assess whether the prior distributions are reasonable in the context of the model. This can be done using a prior predictive check, where data is simulated using parameters drawn from the prior. If the generated data is implausible, it suggests that the priors may be too weakly or strongly informative.

Posterior predictive check

After fitting a Bayesian model a posterior predictive check is used to evaluate how well the fitted model explains the observed data. Here, data is simulated using parameter values sampled from the posterior distribution. If the replicated data fails to resemble the observed data, it suggests that the model may not capture key aspects of the underlying data-generating process.

Prior Sensitivity Analysis

Prior sensitivity analysis is assessing how sensitive the model's inferences are to the choice of prior distributions. It involves re-running the model with different reasonable priors and comparing the resulting posterior distributions. This analysis helps determine how large of an effect the prior has on the posterior.

Since it is expensive to fit the same model many times with different priors, in practice it is often done using importance sampling, see for example Vehtari et al. [2024].

C Numerical stability tricks

This is a collection of some of the numerical tricks used in the implementation of the algorithms to avoid underflow and improve efficiency.

Log-Sum-Exp Trick

When aggregating the contributions of multiple particles to the log-likelihood, we need to compute a sum of the form

$$L_t = \frac{1}{N} \sum_{i=1}^{N} \exp(\ell_i),$$

where N is the number of particles and ℓ_i is the log-weights. Direct exponentiation of ℓ_i can lead to numerical underflow if ℓ_i is very small. To mitigate this, we use the log-sum-exp trick (Stan Development Team [2024]). Define

$$M = \max_{1 \le i \le N} \ell_i.$$

Then,

$$\log \left(\sum_{i=1}^{N} \exp(\ell_i) \right) = M + \log \left(\sum_{i=1}^{N} \exp(\ell_i - M) \right).$$

Thus, the incremental log-likelihood is computed as

$$\log L_t = -\log N + M + \log \left(\sum_{i=1}^{N} \exp(\ell_i - M) \right).$$

This approach rescales the log-likelihoods so that the exponential terms do not vanish numerically.

Transformation of Parameters

When parameters are defined on constrained domains, it is often beneficial to transform them into an unconstrained space to facilitate efficient MCMC proposals. Using a standard proposal distribution, such as the normal distribution, directly in the constrained space can lead to proposed values that lie outside the domain, resulting in a likelihood of zero. Suppose we have a vector of parameters

$$\theta = (\theta_1, \theta_2, \ldots, \theta_n)$$

where some components are defined on a constrained domain. For those components that are already unconstrained, we simply use the identity mapping, i.e.,

$$g_i(\theta_i) = \theta_i.$$

For the constrained parameters, we introduce an invertible and differentiable transformation g_i that maps the constrained θ_i into an unconstrained space:

$$\phi_i = g_i(\theta_i), \quad i = 1, 2, \ldots, n.$$

A common example is proposing values for a standard deviation, which must be positive, and we can then instead propose values in log-space.

If $p(\theta)$ denotes the joint density of θ and $\theta_i = g_i^{-1}(\phi_i)$ is the inverse transformation, the joint density in the ϕ-space is given by the change-of-variables formula:

$$p_\phi(\phi) = p\big(g_1^{-1}(\phi_1), \ldots, g_n^{-1}(\phi_n)\big) \prod_{i=1}^{n} \left| \frac{d}{d\phi_i} g_i^{-1}(\phi_i) \right|.$$

Taking logarithms yields the transformed log density:

$$\log p_{\boldsymbol{\phi}}(\boldsymbol{\phi}) = \log p\big(g_1^{-1}(\phi_1), \ldots, g_n^{-1}(\phi_n)\big) + \sum_{i=1}^{n} \log \left| \frac{d}{d\phi_i} g_i^{-1}(\phi_i) \right|.$$

D Supplementary Figures

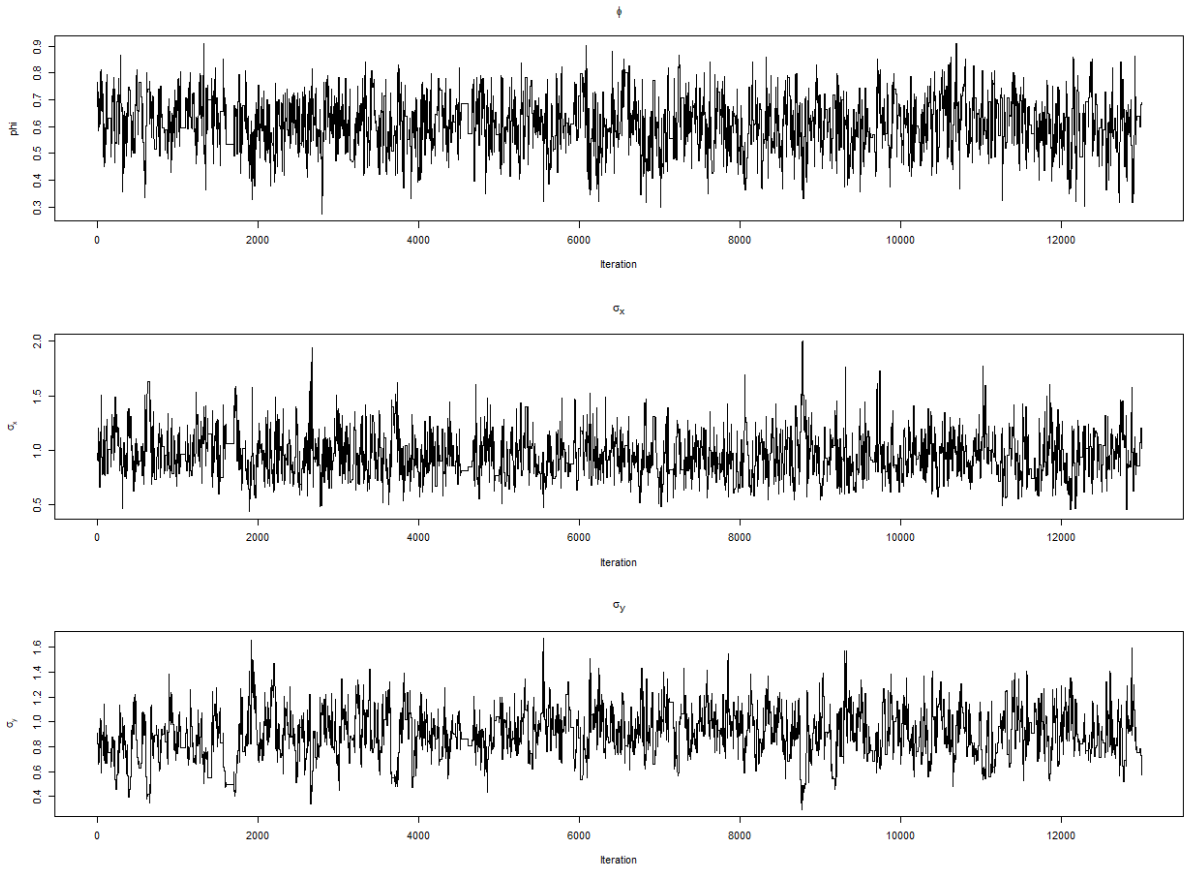

Figure D.1: A trace plot of an MCMC chain for the parameter vector $\theta = (\phi, \sigma_x, \sigma_y)$ in Example 4.2**UPDATE FIGURE TO HIGH QUALITY**

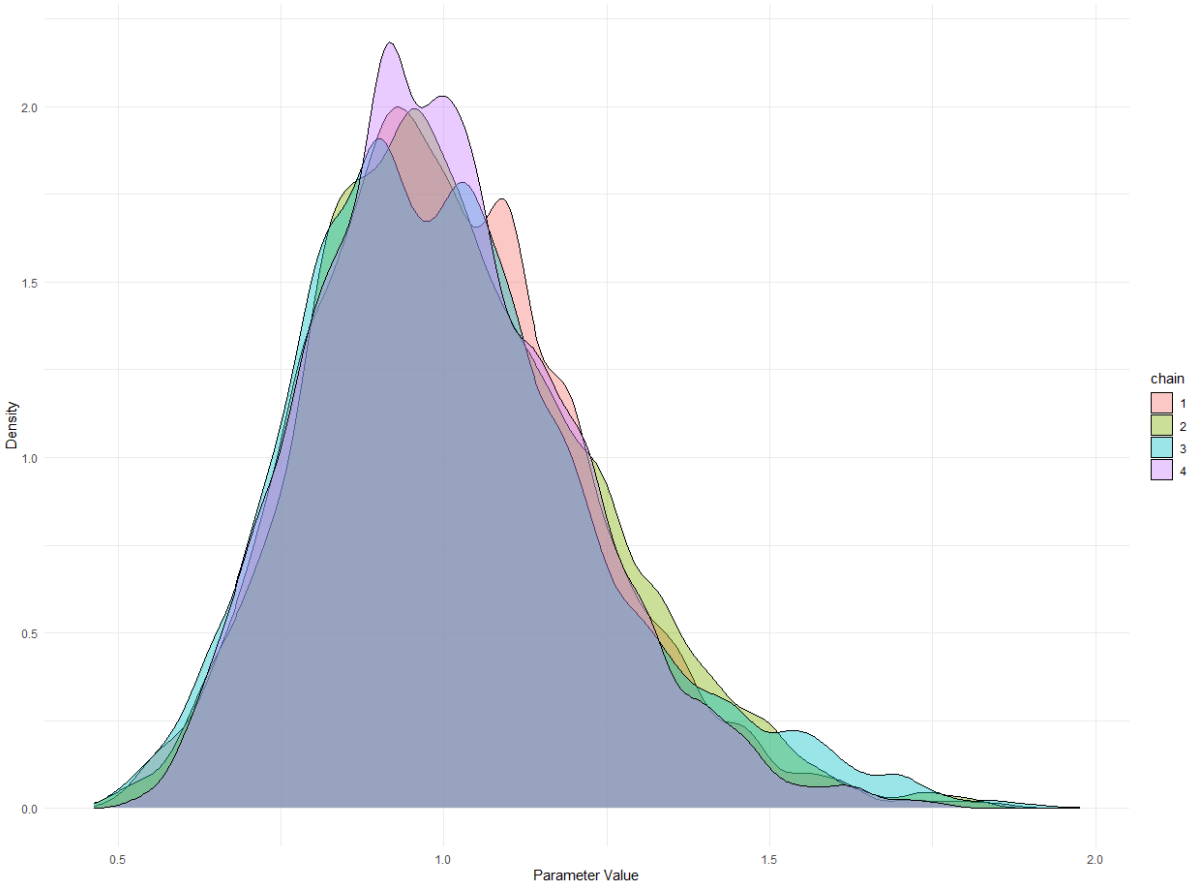

Figure D.2: Density plot of the four chains for σ_x in Example 4.2**UPDATE FIGURE TO HIGH QUALITY**

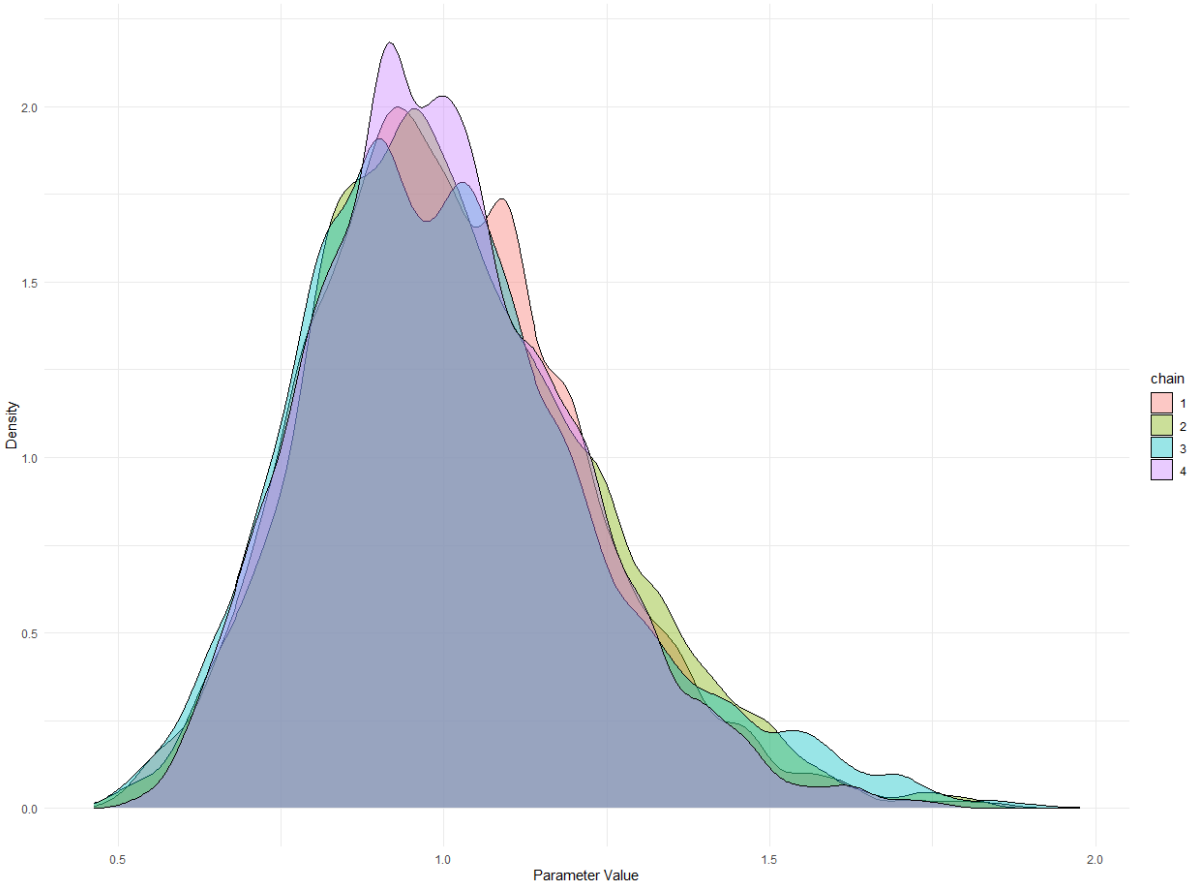

Figure D.3: Density plot of the four chains for σ_y in Example 4.2**UPDATE FIGURE TO HIGH QUALITY**