# Advanced Computational Methods in Statistics

N. Kantas *

**Abstract**

The aim of this course is to introduce some advanced simulation methods. The course will introduce a variety of tools that will eventually be applied to performing inference for nonlinear non-Gaussian state-space models. These models are ubiquitous in Statistics, Econometrics, Engineering and Signal Processing, but performing inference with them can pose considerable challenges. Particle methods, also known as Sequential Monte Carlo (SMC) methods, provide reliable numerical approximations to the associated state inference problems, also known as the non-linear filtering. We will provide a comprehensive introduction to these algorithms and illustrate fundamental strengths and weaknesses. We will also discuss how these methods can be combined with other approaches such as Markov Chain Monte Carlo (MCMC). The course has an emphasis in the methodology and practical aspects of the method, but will also briefly touch on the theory behind the SMC and its relevance in improving the methodology.

## 1 Introduction to the course

State-space models, also known as hidden Markov models (HMMs), are a very popular class of time series models that have found numerous of applications in fields as diverse as Statistics, Ecology, Econometrics, Engineering and Environmental sciences; see [10], [28], [75]. Loosely speaking a HMM is a bivariate stochastic processes $\{X_n\}_{n\geq 0}$ and $\{Y_n\}_{n\geq 0}$, where $\{X_n\}_{n\geq 0}$ is the hidden component and $\{Y_n\}_{n\geq 0}$ is the sequence of observations. The popularity of state-space models stems from the fact that they are flexible and easily interpretable. They include models as diverse as smoothing splines, discretised stochastic differential equations (SDEs) and dynamic generalised linear models. Applications of state-space models include stochastic volatility models where $X_n$ is the volatility of an asset and $Y_n$ its observed log-return [46], biochemical network models where $X_n$ corresponds to the population of various biochemical species and $Y_n$ are imprecise measurements of the size of a subset of these species [77], neuroscience models where $X_n$ is a state vector determining the neuron's stimulus-response function and $Y_n$ some spike train data [59].

However, nonlinear non-Gaussian state-space models are also notoriously difficult to fit to data, and in many scientific disciplines this means one could not realise the potential of using HMM models. The aim of this course is to introduce very recent powerful simulation techniques to achieve this. The main aim of the course is to introduce: a) inference for HMMs b) the statistical methodology behind non-linear filtering, c) particle filtering and how it can applied to fit HMMs to data. Eventually the students following the course should be able to implement particle methods using a high level programming language, e.g. using R, Julia, Matlab or Python, understand how the methodology works and what are its limitations.

We will cover the following topics:

1. Overview of basic Monte Carlo techniques
   (a) Perfect Monte Carlo,
   (b) Importance Sampling,
   (c) Markov Chain Monte Carlo.
2. Introduction to HMMs and Bayesian Filtering
   (a) HMMs and State Space Models,
   (b) Recursive filtering using prediction and Bayes update,
   (c) The Kalman Filter and some of its extensions.

*Email: n.kantas@imperial.ac.uk, Affiliation: Dept. of Mathematics, Imperial College London.

3. Introduction Particle Filtering

   (a) Sequential Importance Sampling,

   (b) Sequential Importance Sampling Resampling, [29, 28, 48, 56],

   (c) Basic convergence results and the degeneracy problems; see [21] for a reference book.

4. Advanced Sequential Monte Carlo

   (a) Extensions to the basic method,

   (b) Particle smoothing,

   (c) Bayesian Inference for fixed state spaces, [15].

5. Parameter Estimation for state space models:

   (a) Likelihood and Bayesian Methods, [44],

   (b) Particle Markov Chain Monte Carlo, [5].

The course website is:
`https://github.com/nkantas/LTCC-Advanced-Computational-Methods-in-Statistics`
There one can find supporting material like video-lectures, code for some of the examples and soft copies of presentation slides. In addition, there will be information related to courseworks, streaming links for the lectures and other practicalities.

These notes contain also homework exercises that will not be marked, but are useful for discussion. The intention is that they assist you in understanding and completing the programming gradually and in particular to familiarise with implementation issues related to coding and debugging. The notes are meant to be fairly self contained and are complemented by presentation slides. Mostly the slides replicate what is included in the notes, but it is possible some extra points or ideas might appear there. This is part of an effort to link the course with active research and give a modern perspective with current application and methodological challenges. For interested readers, there are quite a few reference books one can use to complement the course notes and slides:

- [68] is a very nice introductory text-book at the right level and is available on-line at the following link:
  - `https://users.aalto.fi/~ssarkka/pub/cup_book_online_20131111.pdf`
- Another nice reference is [27] and also has some nice examples coded in R.
- [10] and [19] are more advanced books. [19] has comes with Python code available through a GitHub repository.

Finally there are some portals with a lot of resources, such papers, links to code and a good overview on various developments on SMC:

- The SMC page maintained by Arnaud Doucet: `http://www.stats.ox.ac.uk/~doucet/smc_resources.html`
- The portal by Pierre Del Moral: `http://people.bordeaux.inria.fr/pierre.delmoral/simulinks.html`

Note that Monte Carlo is not a black box method. One needs to pay attention to the problem and model at hand to design an efficient and effective method. The first time one tries to implement more advanced Monte Carlo algorithms, one might struggle and notice some gaps in terms of understanding the procedure, what the key points are and how to test the written code. I would encourage every student to avoid shortcuts and go through the effort of implementing these techniques on their own and avoid relying excessively on ready available code. The benefits will be obvious soon thereafter. The aim of this course is to gradually overcome these initial obstacles in terms of understanding the methodology and provide confidence in using advanced algorithms such as Particle MCMC or Sequential Monte Carlo Samplers.

# 2 Overview of basic Monte Carlo techniques

Monte Carlo usually refers to using simulation and sampling from complex high dimensional distributions in order to compute integrals. The topic has a long history during the last century and has been very influencial in scientific disciplines where computing is used. In Statistics it was crucial in the success of Bayesian Statistics, but as we will see in this course can be used also in other inferential procedures such as Maximum Likelihood (ML) estimation. For a nice historical account you could visit the Wikipedia lemma: `https://en.wikipedia.org/wiki/Monte_Carlo_method`.

The purpose of this section is eventually to provide a revision or overview of basic Monte Carlo techniques and register notation conventions that will follow later on. Consider an arbitrary distribution on $\mathcal{X}$ with a density $\pi$ w.r.t to $dx$, written as follows

$$\pi(x) = \frac{\gamma(x)}{Z}$$

where $Z$ an is an unknown normalising constant. Here $\gamma$ is an unnormalised density that is integrable, $\int_{\mathcal{X}} \gamma(x)dx = Z < \infty$, but $Z$ is not necesserily equal to 1 and in principle unknown and to be estimated. To perform most standard Monte Carlo methods we will require that one can compute $\gamma$ pointwise, which means that $\gamma(x)$ can be evaluated (by a computer) for any value of $x$. A classical example comes from Bayesian inference where $\gamma$ is the unnormalised posterior density

$$\gamma(x) = p(y|x)p(x)$$

with $p(y|x)$ being the likelihood of the aquired data $y$, $p(x)$ the prior density and

$$Z = p(y) = \int p(y|x)p(x)dx$$

the so called marginal likelihood or evidence used for model comparison.

Let $\varphi : \mathcal{X} \longrightarrow \mathbb{R}^d$ be a bounded $\pi$ measureable function (i.e. $\overline{\varphi} = \sup \varphi < +\infty$) and say we want to compute integrals of the form

$$\pi(\varphi) = \mathbb{E}_\pi[\varphi(X)] = \int_{\mathcal{X}} \varphi(x)\pi(dx). \tag{1}$$

Note that each of these notations for integrals will follow throughout the course and often when integrating over $\mathcal{X}$ we will omit the domain of integration and write $\pi(\varphi) = \int \varphi(x)\pi(dx)$. In addition, following a standard minor abuse of notation we will use the same letter $\pi$ to denote both the (normalised) density and the distribution itself, i.e. we write both $\pi(x)$ above for any $x \in \mathcal{X}$ as well as $\pi(A)$ for the probability measure evaluated at $A \subseteq \mathcal{X}$ or $\pi(dx)$ when writing the Lebesgue integral in (1).

Numerical integration and calculation or approximation of expectations as in (1) lies at the core of the problem of performing inference for state space models (and Bayesian inference in general). Capturing $\pi$ numerically is done by approximating moments, integrals of simple functions used in construction of histograms or probabilities of interesting events. Simulation methods and Monte Carlo use samples or approximate samples from $\pi$ to do this and are also a natural choice nowadays given computational power available. In contrast to standard deterministic methods, Monte Carlo has many advantages:

- choosing the number of samples provides an inherent control on the quality of the estimation.
- approach is simple and directly applicable in multivariate cases. Of course once dimension of the state space of $\mathcal{X}$ starts to grow beyond a point (say from $10-20$) a naive implantation of the methodology will suffer from the curse of dimensionality. These will manifest as increasing Monte Carlo variance in the resulting estimates.
- it is provably convergent/consistent and there is an extensive theory to explain the behaviour of the estimates.

We will begin with some basics on Perfect Monte Carlo, Rejection Sampling and Importance sampling and then present some elements of Markov Chain Monte Carlo.

## 2.1 Perfect Monte Carlo

Assume we can obtain $N$ i.i.d. samples $X^i \sim \pi$ (from now on we will often refer to samples also as *particles*). One can use the so called perfect (or naive) Monte Carlo approximation:

$$\widehat{\pi}(\varphi) = \int_{\mathcal{X}} \varphi(x)\widehat{\pi}(dx) = \frac{1}{N}\sum_{i=1}^{N} \varphi(X^i). \tag{2}$$

to approximate $\pi(\varphi)$. We are just using sample averages here, but it might be useful for intuition to view

$$\widehat{\pi}(dx) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X^i}(dx)$$

as a particle approximation of $\pi$, where here $\delta_{x'}(dx)$ denotes the Dirac measure, such that $\delta_{x'}(A) = 1_{x' \in A}$.

Note that the estimator $\widehat{\pi}(\varphi)$ is a random variable, sampled from $\prod_{i=1}^{N} \pi(dx^i)$, which we will refer to as the sampling distribution in this case. In addition $\widehat{\pi}(\varphi)$ has some very nice properties:

- It is consistent. By the (strong) Law of Large Numbers (LLN) $\widehat{\pi}(\varphi) \to_{N\to\infty} \pi(\varphi)$.
- It is unbiased $\mathbb{E}^N[\widehat{\pi}(\varphi)] = \pi(\varphi)$, where $\mathbb{E}^N[\cdot]$ will generally denote the expectation w.r.t the sampling distribution. This will depend in general on the sampling method used. Here for the simple i.i.d. perfect sampling case it is $\prod_{i=1}^{N} \pi(dx^i)$. It is not hard to check that due to i.i.d. sampling

$$\mathbb{E}^N\left[\sum_{i=1}^{N} \varphi\left(X^i\right)\right] = \sum_{i=1}^{N} \mathbb{E}_\pi\left[\varphi\left(X^i\right)\right] = N\mathbb{E}_\pi\left[\varphi\left(X\right)\right]$$

- The Monte Carlo variance decreases with $N$. It is given in fact by

$$\mathbb{V}ar^N[\widehat{\pi}(\varphi)] = \frac{1}{N}\mathbb{V}ar_\pi\left[\varphi(X^i)\right] = \frac{1}{N}\left(\pi(\varphi^2) - \pi(\varphi)^2\right),$$

  where $\mathbb{V}ar^N[\cdot]$ will always denote variance w.r.t. the sampling distribution (here $\prod_{i=1}^{N} \pi(dx^i)$). Note the rate of decrease with $N$ (i.e. the standard Monte Carlo rate $N^{-\frac{1}{2}}$) does not dependent on size of $\mathcal{X}$, but at the same time the numerator $\pi(\varphi^2) - \pi(\varphi)^2$ does!

- One can quantify the limiting distribution of $\widehat{\pi}(\varphi)$ using a Central Limit Theorem (CLT):

$$\sqrt{N}\left(\widehat{\pi}(\varphi) - \pi\left(\varphi\right)\right) \Rightarrow \mathcal{N}(0, \sigma_\varphi^2)$$

  as $N \to \infty$ with $\sigma_\varphi^2 = \pi(\varphi^2) - \pi(\varphi)^2$.

The problem is that in most interesting examples and applications we cannot simply sample directly from $\pi$. In some particular cases, even if this is possible, this could lead to a very high variance , e.g. for $\varphi = 1_A$ where $A$ is a set with very low probability.

**Example 1.** Consider the tail estimation problem for a continuous distribution $P$ with density $p(x)$ defined on the positive real line, i.e. $x > 0$. Suppose we are interested in computing a lower tail $p^* = P(X \leq c)$ whose true value is extremely low around $10^{-9}$. We will consider sampling i.i.d. $x^i \sim p(\cdot)$, $i = 1, \ldots, N$ and then computing the estimator

$$\widehat{p^*} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{x\leq c}(x^i).$$

The latter is consistent and will converge to $p^*$ as $N$ (and the computational effort) grows and we know that the variance of estimator is

$$\sigma^2_{\widehat{p^*}} = \frac{\mathbb{V}ar_P[\mathbf{1}_{x\leq c}]}{N} = \frac{p^* - {p^*}^2}{N}.$$

This means that the relative error $\sqrt{\mathbb{V}ar^N\left[\frac{\widehat{p^*}}{p^*}\right]}$ can be approximated by $\frac{1}{\sqrt{p^* N}}$, so one would require at least $N \sim 10^{11}$ to estimators of decent quality (e.g. around 0.1) and this in turn would require prohibitively long simulation times.

This example is meant to illustrate a few things:

- Variance of estimators is an important measure of efficiency. In practical or more interesting problems direct sampling is not possible and expressions or bounds for Monte Carlo variance can be much more challenging to derive, but are very useful to understand the performance and applicability of a method.
- Even when direct samples from $\pi$ are possible some test functions $\varphi$ can result to estimates with very high Monte Carlo variance. This means that in some cases indirect sampling will be better. To avoid any confusion from now on we will treat these problems as one with a more reasonable target distribution that is intractable (i.e. not possible to sample from). In Example 1 such a choice would be $\gamma(x) = \mathbf{1}_{x\leq c} \cdot p(x)$ or similar that should be tackled using indirect sampling.

## 2.2 Rejection Sampling

**Algorithm 1** A Rejection Sampling algorithm to obtain one sample from $\pi$

1. Sample $X \sim q$
2. Sample $U \sim U[0,1)$
3. **Accept** sample, $Y = X$ if $U < \frac{w(X)}{M}$
4. Repeat 1-3 until a sample is accepted.

Indirect sampling uses an instrumental distribution (other than $\pi$) to obtain samples. This is often refered to as the proposal and needs to satisfy certain requirements or properties. We will begin with a definition of the first one that is absolute continuity.

**Definition 1.** We will say $q$ is distribution that is absolutely continuous with $\pi$ and denote this as $\pi \ll q$, **if** $\pi(x) > 0$ implies $q(x) > 0$ for every $x$ where $\pi(x) > 0$.

In simpler terms this means that $q$ has heavier tails than $\pi$. In the definition above for simplicity we assume that both $\pi$ and $q$ have densities w.r.t. the same dominating measure (here we use $dx$). A more general definition can be posed as a requirement that the Radon Nikodym derivative $\frac{d\pi}{dq}(x)$ exists and is bounded. Recall in our case $\frac{d\pi}{dq}(x) = \frac{\gamma(x)}{Z \cdot q(x)}$.

So let $q$ be a density from a distribution that is absolutely continuous with $\pi$ and assume additionaly that you know a constant $M$ such that for all $x$:

$$w(x) = \frac{\gamma(x)}{q(x)} < M.$$

This is quite a strong assumption, but can be used to construct the Rejection Sampling algorithm presented in Algorithm 1. There, a sample $x$ from $q$ is accepted as a sample from $\pi$, with an acceptance probability $\frac{w(x)}{M}$. To justify this let $U$ be uniform random variable, $X$ be a sample from $q$ and consider the following simple conditioning argument:

$$\begin{aligned}
\mathbb{P}\left[X \in A | \, U \leq \frac{w(X)}{M}\right] &= \frac{\mathbb{P}\left[X \in A, U \leq \frac{w(X)}{M}\right]}{\mathbb{P}\left[U \leq \frac{w(X)}{M}\right]} \\
&= \frac{\int_A \int_0^1 q(x) 1_{u<w(x)/M} du dx}{\int_{\mathcal{X}} q(x) \left(\int_0^{w(x)/M} du\right) dx} \\
&= \frac{\int_A q(x) \frac{w(x)}{M} dx}{\int_{\mathcal{X}} q(x) \frac{w(x)}{M} dx} \\
&= \frac{\int_A \gamma(x) dx}{\int_{\mathcal{X}} \gamma(x) dx} \\
&= \pi(A).
\end{aligned}$$

The conditional probability law $\mathbb{P}\left[X \in A | \, U \leq \frac{w(X)}{M}\right]$ describes exactly the output of an accepted sample of Algorithm 1.

The disadvantage with this procedure is that

$$\mathbb{P}\left[U \leq \frac{w(X)}{M}\right] = \frac{Z}{M}$$

so method will not be very efficient if $M$ high and requires a sharp bound. For a reasonably efficient scheme we need $M$ to be close to $Z$. This means that $\pi$ should be very close to $q$ which is not easy in practice. There are also more advanced rejection methods based on more sophisticated constructions such as envelopes and adaptation; see [66] for more details. Whilst these methods were quite popular in the 90-s, currently they are not used often except in very simple settings due to their limited applicability and efficiency for a wide class of models and problems.

## 2.3 Importance Sampling

One alternative way to get around not being to sample for $\pi$ directly is to use Importance Sampling (IS). IS is more easy to generalise and we will see later in the course how it is used for sequential settings and particle filters.

Let $q$ be again absolutely continuous with $\pi$. Then one can write

$$\pi(x) = \frac{w(x)q(x)}{\int w(x)q(x)dx}$$

with

$$w(x) = \frac{\gamma(x)}{q(x)},$$

where here the instrumental distribution $q$ will play the role of the importance distribution and $w$ is the function for the un-normalised importance weights. Note absolute continuity conditions ensures $0 < w(x) < \infty$ (everywhere where $\pi > 0$).

Given $Z$ is assumed to be unknown, we will present what is commonly refered as self normalised IS. Assume we can obtain $N$ i.i.d. samples $X^i \sim q$, then approximate $\pi(\varphi)$ as

$$\widehat{\pi}(\varphi) = \sum_{i=1}^{N} W^i \varphi(X^i) \tag{3}$$

where $W^i = \frac{w(X^i)}{\sum_{j=1}^{N} w(X^j)}$ such that $\sum_{i=1}^{N} W^i = 1$. Similar to before one can write a particle approximation for $\pi$ as

$$\widehat{\pi}(dx) = \sum_{i=1}^{N} W^i \delta_{X^i}(dx).$$

The difference with perfect Monte Carlo is that now we have to weight the samples/particles to compensate for the fact that they are not perfect samples from $\pi$. In addition we can estimate the unknown normalising constant $Z$ as

$$\widehat{Z} = \int \frac{\gamma(x)}{q(x)} \widehat{q}(dx) = \frac{1}{N} \sum_{i=1}^{N} \frac{\gamma(X^i)}{q(X^i)} = \frac{1}{N} \sum_{i=1}^{N} w(X^i)$$

where $\widehat{q}(dx) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X^i}(dx)$. As $Z = \int \gamma(x)dx$ is an integral w.r.t $\gamma$ and not $\pi$, the estimate $\hat{Z}$ is a perfect Monte Carlo estimate of $Z$, which due absolute continuity of $q$ and $\pi$ can be also writen as

$$Z = \int \frac{\gamma(x)}{q(x)} q(dx) = \int w(x)q(dx).$$

So $\widehat{Z} = \frac{1}{N} \sum_{i=1}^{N} w(X^i)$ is not constructed using self normalised IS, but with perfect Monte Carlo (the construction here is the same as IS with known normalising constants). As a result one can easily show that $\hat{Z}$ is unbiased as

$$\mathbb{E}^N \left[ \sum_{i=1}^{N} \frac{\gamma(X^i)}{q(X^i)} \right] = \sum_{i=1}^{N} \mathbb{E}_q \left[ \frac{\gamma(X^i)}{q(X^i)} \right] = \sum_{i=1}^{N} \int \gamma(x^1)dx^1 = NZ$$

Note unbiasedness does not hold for $\widehat{\pi}(\varphi)$ due to the non-linear terms arising in $W^i$ when dividing by $\sum_{j=1}^{N} w(X^j)$.

Still when using (self normalising) IS we maintain some of the nice properties we had earlier

- It is asymptotically consistent, by LLN, and we can write the asymptotic bias

$$(\widehat{\pi}(\varphi) - \pi(\varphi)) = -\frac{1}{N} \int_{\mathcal{X}} \frac{\pi^2(x)}{q(x)} (\varphi(x) - \pi(\varphi)) \, dx$$

- A CLT holds $\sqrt{N} (\widehat{\pi}(\varphi) - \pi(\varphi)) \Rightarrow_{N \to \infty} \mathcal{N}(0, \sigma^2_{q,\varphi})$ with $\sigma^2_{q,\varphi} = \int_{\mathcal{X}} \frac{\pi^2(x)}{q(x)} (\varphi(x) - \pi(\varphi))^2 \, dx$.

- The estimate of $Z$ is unbiased $\mathbb{E}^N \left[ \widehat{Z} \right] = Z$.

- The Monte Carlo variance of $\pi(\varphi)$ decreases with $N$. In addition the variance of $Z$ is:

$$\mathbb{V}ar^N\left[\widehat{Z}\right] = \frac{Z^2}{N}\left(\int \frac{\pi^2(x)}{q(x)}dx - 1\right), \tag{4}$$

  so the $1/N$ rate of decrease of the variance is still relevant.

Recall that $\mathbb{E}^N[\cdot], \mathbb{V}ar^N$ denote throughout these notes the expectation and variance respectively w.r.t. the sampling distribution of $N$ samples or particles; here it is $\prod_{i=1}^N q(dx^i)$.

Note that from a practical perspective one needs to decide how to choose $q$. This is not always an easy task. One possible attempt is to select the $q$ which minimises the variance of the estimator $\hat{\pi}(\varphi)$ in (3). This is minimised by

$$q(x) = \frac{|\varphi(x)|\,\pi(x)}{\int\limits_{\mathcal{X}} |\varphi(x)|\,\pi(x)\,dx}.$$

This expression is not very useful in practice as we are interested in estimating expectations of several test functions (e.g. moments or simple functions) with the same set of samples. An alternative would be to minimise the variance of $\hat{Z}$ in (4). Then it is not hard to see that this will result to requiring sampling from $\pi$, which is not available. Nevertheless, this provides useful intuition that one should aim to find a $q$ that is very close to $q$ in some sense. One could construct $q$ as an approximation of $\pi$ using other methods based on the Laplace principle, Gaussian approximations etc.

Another interesting thing to notice is that minimising the variance of $\hat{Z}$ in (4) is equivalent to minimising the variance of the importance weights. Both give $q = \pi$, which makes sense as equal weights will mean we are back to the perfect Monte Carlo case.

We will conclude this section with the definition of the Effective Sample Size (ESS) for IS:

$$\text{ESS} = \frac{N}{1 + \mathbb{V}ar_q\left[\frac{w(X)}{Z}\right]},$$

which is an "efficiency" measure that can be interpreted as the number of perfect samples from $\pi$ that would result in the same Monte Carlo variance. This stems from the following approximation:

$$\mathbb{V}ar^N\left[\widehat{\pi}(\varphi)\right] \approx \frac{1}{N}\mathbb{V}ar_\pi\left[\widehat{\pi}(\varphi)\right]\left(1 + \mathbb{V}ar_q\left[\frac{w(X)}{Z}\right]\right),$$

which is valid asymptotically and can be shown using the delta method. We refer the interested reader for more details and derivation are presented in [55, pages 35-36]. The $ESS$ is in fact monitored using its Monte Carlo approximation

$$ESS = \frac{1}{\sum_{i=1}^N \left(W^i\right)^2} = \frac{\left(\sum_{i=1}^N w(X^i)\right)^2}{\sum_{i=1}^N w(X^i)^2},$$

which is a number in $[1, N]$ with higher $ESS$ indicating more efficient sampling.

IS when applied on its own as presented here will be effective only in low dimensional toy problems. Nevertheless it is still very useful and can be used within other more powerful methods. Later in Section 4.1 we will present an extension that implements IS sequentially. This is very useful extension for Bayesian inference problems where the data has some ordering and the likelihood a product form, which implies $\pi$ can be reached through a sequence of intermediate posterior distributions.

## 2.4 Markov Chain Monte Carlo

We will discuss now a different Monte Carlo approach for sampling from $\pi$, which is Markov chain Monte Carlo (MCMC). The principle is based on obtaining indirect samples from $\pi$ by sampling from an ergodic Markov chain with invariant distribution $\pi$. These samples are then used to provide sample averages and approximate expectations. This topic has a long history and a rich literature and has been for a long time a popular choice for performing Bayesian inference. Indicatively we mention [57, 38, 33, 71, 69].

Lets say we have access to an ergodic Markov Probability kernel $K$ such that

$$\int \pi(dx)K(x, dy) = \pi(dy) \tag{5}$$

and suppose we start from an initial distribution $\nu$ (possibly $\delta_x$). Then, a MCMC sampling procedure is to iteratively obtain a chain of samples $\{X_i\}_{i=1,\dots,N}$ (notice now sample index is now in subscript to follow the usual Markov Chain notation) as follows:

$$X_0 \sim \nu,\ X_1 \sim K(X_0, \cdot),\ X_2 \sim K(X_1, \cdot), \dots, X_N \sim K(X_{N-1}, \cdot), \dots$$

and approximate $\pi(\varphi)$ as

$$\widehat{\pi}(\varphi) = \frac{1}{N} \sum_{i=1}^{N} \varphi(X_i).$$

Often to allow for the Markov Chain to converge to $\pi$, the first $L$ iterations are discarded as a burn in (typically $L$ ranges between 2000-5000) and one uses $\widehat{\pi}(\varphi) = \frac{1}{N-L} \sum_{i=L+1}^{N} \varphi(X_i)$ instead. There are many ways to design such a $K$ for a given $\pi$, but here we will just briefly present the most popular ones, the Metropolis-Hastings sampler in Algorithm 2 and the Gibbs sampler in Algorithm 3. These algorithms are run usually for a large number of iterations $N$ that could range from $10^4$ to $10^6$ or higher depending on the difficulty of the problem, its dimensionality, etc.

**Algorithm 2** A Metropolis Hastings MCMC algorithm

Sample $X_0 \sim \nu$.
For $k \geq 1$

1. Sample a candidate proposal: $Y_k \sim Q(X_{k-1}, \cdot)$
2. Compute acceptance ratio
$$\alpha(X_{k-1}, Y_k) = 1 \wedge \frac{\gamma(Y_k) Q(Y_k, X_{k-1})}{\gamma(X_{k-1}) Q(X_{k-1}, Y_k)}$$
3. With probability $\alpha(X_{k-1}, Y_k)$
   - Set $X_k = Y_k$ **(accept proposal)**

   otherwise (with probability $1 - \alpha(X_{k-1}, Y_k)$)
   - Set $X_k = X_{k-1}$ **(reject sample)**

### 2.4.1 The Metropolis-Hastings (MH) sampler

The Metropolis-Hastings procedure is designed so that $K$ is reversible with $\pi$, i.e.

$$\pi(dx) K(x, dy) = \pi(dy) K(y, dx).$$

$Q$ is referred as the proposal. Note that we denote here $Q(X, X')$ the conditional density of moving from $X$ to $X'$. This could have also been denoted as $Q(X'|X)$, but when doing MCMC we will use the $Q(X, X')$ convention to distinguish with IS sampling proposals.

A very common choice for the MCMC proposal is a random walk (RW)

$$Y_k = X_{k-1} + \varrho Z_k, \quad Z_k \sim h(\cdot). \tag{6}$$

If $h$ is symmetric $q(x, y) = q(y, x)$ then the acceptance ratio takes the following convenient form:

$$\alpha(x, y) = 1 \wedge \frac{\gamma(y)}{\gamma(x)}.$$

Very often a Gaussian random walk is used, so it is common to use $h = \mathcal{N}(0, I)$. Here $\varrho$ is a tuning parameter for step size of the proposal and typically is used to set the acceptance ratio to some desired value, typically between 0.2-0.3. Note that there is a trade-off to be balanced: high step sizes lead to low acceptance rates and vice versa, so a compromise of this leads to these values for $\alpha$ between 0.2-0.3.

Random walk proposals are very common in practice due to their simplicity, but when the dimension of the state space $\mathcal{X}$ starts to become of some moderate size (e.g. more than 5) the tuning of $\rho$ and the covariance of $h$ in (6) becomes cumbersome. If some knowledge or approximation of the covariance or its diagonal of $\pi$ is available

then it is recommended to use it for the covariance of $h$ to provide a scaling of the coordinates in $X_k$ that is in alignment with $\pi$.

Note that in step 1 of Algorithm 2 one could replace the Markov kernel $Q(X_{k-1}, \cdot)$ with a distribution $Q$ that does not depend on $x_{k-1}$, and set $Y_k \sim Q(\cdot)$. This is known as the *independence sampler* and in step 2. we would have

$$\alpha(X_{k-1}, Y_k) = 1 \wedge \frac{\gamma(Y_k)Q(X_{k-1})}{\gamma(X_{k-1})Q(Y_k)}.$$

Similar to what we saw for IS and Rejection sampling one would require here $\frac{\gamma(x)}{q(x)} < \infty$. In addition, to accept often, $Q$ needs to be similar to $\pi$, which of course is quite hard to get in practical situations. On the other hand, if a proposal $Y_k$ is accepted it is sampled independently of $X_{k-1}$ so the correlations in the chain are much weaker.

In general, it is a good idea to have a proposal that contains information of $\pi$. A common proposal like this is the so called Metropolis adjusted Langevin algorithm (MALA), which arises from using an Euler discretisation of a Langevin SDE

$$Y_n = X_{n-1} + \delta \frac{1}{2} \nabla \log \gamma(X_{n-1}) + \sqrt{\delta} Z_n, \quad Z_n \sim \mathcal{N}(0, I).$$

Note that in this case $Q$ is **not** symmetric so the full acceptance ratio of step 3. in Algorithm 2 should be used. More proposals are summarised in [71] that include, autoregressive ones, deterministic ones (a special case of which is Hybrid or Hamiltonian Monte Carlo), using multiple or mixture of proposals. We will mention briefly the class of autoregressive proposals recently refered also as pre conditioned Crank Nicolson proposals in [20]. Lets say we have $\frac{d\pi}{d\lambda}(x) = \vartheta(x)$, which in Bayesian contexts would mean $\vartheta$ is a likelihood function and $\lambda$ is a prior. Assume that one can construct a proposal invariant to the prior $\lambda$, i.e. $\int \lambda(dx) Q(x, y) = \lambda(y)$, and this $Q$ is used as a proposal in step 1. of Algorithm 2. Then we can replace the acceptance ratio in step 2. with

$$\alpha(X_{k-1}, Y_k) = 1 \wedge \frac{\vartheta(Y_k)}{\vartheta(X_{k-1})},$$

which means that the algorithm uses only the likelihood ratio to accept or reject the proposed sample. This is very useful for a very high dimensional state space $\mathcal{X}$ such as the ones arising from space discretisations of PDEs/ODEs and Bayesian inverse problems, because in these models most of the discrepancy between the proposal and the posterior is due to accumulation of small discrepancies between the proposal and the prior over a large number of dimensions. When $\lambda$ is Gaussian and $\lambda = \mathcal{N}(\mu, S)$ then $Q(X_{n-1}, \cdot)$ can be implemented as

$$Y_n = \mu + \rho\,(X_{n-1} - \mu) + (1 - \rho^2)^{\frac{1}{2}} Z, \quad Z \sim \mathcal{N}(0, S)$$

with $\rho$ being a tuning variable taking values in $(0, 1)$. The case $\rho = 0$ corresponds to an independence sampler and the one where $\rho$ approaches 1 means that the step size is approaching 0.

### 2.4.2 The Gibbs sampler

In many problems one can derive conditional distributions for each variable given the rest, and this is exploited in the Gibbs sampler presented in Algorithm 3. Let $x = (x^1, \ldots, x^d)$ and $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_d$ and define also $x^{-i} = (x^1, \ldots, , x^{i-1}, x^{i+1}, \ldots, x^d)$. Assume that the so called *Gibbs full conditional distributions*

$$\pi_i(x^i | x^{-i}) = \frac{\gamma(x)}{\int \gamma(x) dx^i} = \frac{\pi(x)}{\pi_i(x^{-i})}$$

are available and it is possible to sample from them. Then, a Gibbs sampler samples iteratively from each conditional distribution. This sampler is only $\pi$-invariant and not reversible w.r.t. $\pi$, but is often very efficient and does not waste any simulation with rejection steps. When for some coordinate $i$ direct sampling from $\pi_i$ is not possible this can be substituted by a small number of MH iterations that are invariant to $\pi_i$. This is refered to as the Metropolis within Gibbs algorithm. Finally, other extensions look at avoiding to use full conditionals Algorithm 3, which is called the collapsed Gibbs sampler; see [55, Section 6.7] for an introduction and [72] for recent extensions. For this class of Gibbs samplers, in order to have a valid MCMC sampler, one needs to pay more attention in the order of the steps in Algorithm 3 and in many cases this can improve speed of mixing and performance.

**Algorithm 3** A Gibbs Sampler

Sample $X_0 \sim \nu$.
For $k \geq 1$. Sample:

- $X_k^1 \sim \pi_1(\cdot|X_{k-1}^{-i})$
- $X_k^2 \sim \pi_2(\cdot|X_k^1, X_{k-1}^3, \ldots, X_{k-1}^d)$
- $X_k^3 \sim \pi_3(\cdot|X_k^1, X_k^2, X_{k-1}^4, \ldots, X_{k-1}^d)$

  $\vdots$MH
- $X_k^d \sim \pi_d(\cdot|X_k^{-d})$

### 2.4.3 Theory and practice of MCMC

MCMC algorithms are justified by many theoretical results, that are derived from the convergence properties of Markov chains defined on general state spaces; see [71] for a nice review. The MCMC algorithm defines a Markov transition kernel and one aims to establish some form of ergodicity:

$$\left\|K^k(x_0, \cdot) - \pi\right\| \leq r(x_0, k), \quad r(x_0, k) \to_{k\to\infty} 0,$$

where we denote $K^k(x_0, dx_k)$ the probability distribution of $X_k$ ($K^k$ is the $k$-th iterate of $K$, $K^k(x, dz) = \int K^{k-1}(x, dx')K(x', dz)$ with $K^0$ being identity). The convergence here is w.r.t to some prespecified appropriate norm $\|\cdot\|$ for probability measures (e.g. $L^2$, total variation norm, or Wasserstein distance). Note the rate function $r(x_0, k)$ is important so for a given $\pi$ some algorithms are better than others and this can be checked also in practice. Using ergodicity together with other properties (e.g. minorisation, aperiodicity etc.), one can derive properties like

- a LLN $\widehat{\pi}(\varphi) \to_{N\to\infty} \pi(\varphi)$ for $\varphi \in L^1(\pi)$,
- a CLT for $\sqrt{N}(\widehat{\pi}(\varphi) - \pi(\varphi)) \to \mathcal{N}(0, v(\varphi, K))$, $\varphi \in L^2(\pi)$.

The CLT variance can be writen as

$$v(\varphi, K) = \mathbb{V}ar_\pi[\varphi] + 2\sum_{i\geq 1} \mathbb{C}ov[\varphi(X_0), \varphi(X_i)]$$

and is a useful to characterise the asymptotic sampling error in $\widehat{\pi}(\varphi)$. It is also used to derive an Effective Sample Size (ESS), which is different than then one used for IS. To obtain the ESS for MCMC, define first the *integrated auto-correlation time* for $\varphi$ as

$$\tau_\varphi = \frac{v(\varphi, K)}{\mathbb{V}ar_\pi[\varphi]} = 1 + 2\sum_{i\geq 1} Cor[\varphi(X_0), \varphi(X_i)].$$

A lower $\tau_\varphi$ means the MCMC chain decorrelates faster and this results to lower asymptotic Monte Carlo variance ($v(\varphi, K)$) and higher efficiency. Note that in the expression for $v(\varphi, K)$ the term $\mathbb{V}ar_\pi[\varphi]$ can be related to perfect Monte Carlo variance, so one could then use:

$$\text{ESS} = \frac{N}{\tau_\varphi},$$

which is a number in $[1, N]$ with higher ESS indicating more efficient sampling.

In addition, the expression for both $\tau_\varphi$ and $v(\varphi, K)$ directly implies that monitoring autocorrelation function (ACF) plots for assessing mixing of the MCMC chain is sensible and is closely linked with Monte Carlo variance for very large number of samples. One could summarise the diagnostics we have mentioned so far as the ESS, ACF decay, the (average-in-time) acceptance ratio. In addition one could assess mixing by observing trace plots qualitatively or using other criteria such as approximations of the expected square jumping distance $\frac{1}{N}\sum_{n=1}^N (X_n - X_{n-1})^2$. More elaborate diagnostics that also check whether stationarity has been reached can be found in [66, Chapter 8], but these tend to be more compicated and require multiple MCMC chains.

### 2.4.4 The pseudo-marginal sampler

In many cases one cannot compute $\gamma$ pointwise. Most commonly this is due to intractable likelihoods in Bayesian inference. Recall $\gamma(x) = p(y|x)p(x)$ and suppose $p(y|x)$ is intractable. This means it is not possible to evaluate $p(y|x)$ for any $x$, either because the observation density for $y$ does not exist (stable or g and k distributions are common cases) or it is simply too expensive to evaluate (as in complicated and large scale models used in genetics or system biology). We will extend the MH sampler for cases where an unbiased estimate of $p(y|x)$ is available instead. We will also simplify our notation and denote $\mathcal{Z}_x = p(y|x)$ and let $\hat{\mathcal{Z}}_x$ be an unbiased estimate of it, i.e.

$$\mathbb{E}\left[\hat{\mathcal{Z}}_x\right] = \mathcal{Z}_x = p(y|x)$$

We will consider the case where $p(y|x)$ is a marginal likelihood of a model with latent variables $Z$:

$$p(y|x) = \mathcal{Z}_x = \int p(y, z|x)dz,$$

which is very common in hierarchical models, HMMs and other similar models. We will assume also that $p(y, z|x)$ can be computed pointwise for any $z, x$ (the data $y$ is fixed). Based on what we have seen earlier we can construct $\hat{\mathcal{Z}}_x$ using IS. Suppose we are given an arbitrary value for $x$ and then we:

1. sample $Z_n^i \sim q(\cdot|x)$, i.i.d. for $i = 1, \ldots, L$.
2. then compute

$$\hat{\mathcal{Z}}_x = \frac{1}{L}\sum_{i=1}^{L} w(y, Z_n^i, x),$$

with

$$w(y, z, x) = \frac{p(y, z|x)}{q(z|x)}. \tag{7}$$

As we saw in Section 2.3 $\hat{\mathcal{Z}}_x$ is unbiased and $\mathbb{E}^L\left[\hat{\mathcal{Z}}_x\right] = \mathcal{Z}_x$.

We present the Pseudo-marginal MH sampler in Algorithm 4 proposed in [4]. Comparing with standard MH and step 2. of Algorithm 2, given we cannot compute $\mathcal{Z}_x$ we have replace it with $\hat{\mathcal{Z}}_x$. This means we use

$$\widetilde{\alpha}(x, x') = 1 \wedge \frac{\hat{\mathcal{Z}}_{x'} p(x')Q(x', x)}{\hat{\mathcal{Z}}_x p(x)Q(x, x')}$$

instead of

$$\alpha(x, x') = 1 \wedge \frac{\mathcal{Z}_{x'} p(x')Q(x', x)}{\mathcal{Z}_x p(x)Q(x, x')}$$

Note that in our notation the proposed states in Algorithm 2 are denoted as $Y_k$ and the data used in $p(y|x)$ here is denoted by $y$. These two should not be confused with each other so we opt to use $X'_k$ for the proposal in Algorithm 4. Also note $Q$ is the MCMC proposal, $q(\cdot|x)$ the IS one and these are different. Finally it should be emphasised that one needs to compute each $\hat{\mathcal{Z}}_{X_k}$only once at step 2. and then stored and re-used in future iterations of the algorithm when computing $\widetilde{\alpha}$.

It turns out that Algorithm 4 is a valid MCMC scheme and has the right invariant distribution. More precisely admits $\pi$ as a marginal. In fact we are running an MCMC algorithm with state at each time $k$ being $(X_k, Z_k^1, \ldots, Z_k^L)$. The auxiliary simulation variables $Z_k^i$-s should be considered also when trying to find what the stationary distribution is. It turns out that Algorithm 4 is a standard MH algorithm, which is invariant and reversible to

$$\tilde{\pi}(x, z^1, \ldots, z^L) \propto p(x)\left(\frac{1}{L}\sum_{i=1}^{L} w(y, z^i, x)\right)\prod_{i=1}^{L} q(z^i|x),$$

where $\propto$ denotes proportional to. The structure of $\tilde{\pi}$ is not surprising. It is the product of the prior, the unbiased estimate of the likelihood $\hat{\mathcal{Z}}_x$ and the sampling density of the auxiliary $z^i$-s. This means that the latter are sampled perfectly within a MH scheme targetting $\tilde{\pi}$, so that part of the proposal and target densities will cancel out. In fact,

**Algorithm 4** The Pseudo-marginal MH algorithm

Sample $X_0 \sim \nu$.
For $k \geq 1$

1. Sample a candidate proposal: $X'_k \sim Q(X_{k-1}, \cdot)$
2. Sample $Z_n^i \sim q(\cdot|X'_k)$, i.i.d. for $i = 1, \ldots, L$. and compute

$$\hat{\mathcal{Z}}_{X'_k} = \frac{1}{L} \sum_{i=1}^{L} w(y, Z_n^i, X'_k)$$

   with $w(y, z, x)$ defined in (7).

3. Compute the acceptance ratio

$$\widetilde{\alpha}(X_{k-1}, X'_k) = 1 \wedge \frac{\hat{\mathcal{Z}}_{X'_k} p(X'_k) Q(X'_k, X_{k-1})}{\hat{\mathcal{Z}}_{X_{k-1}} p(X_{k-1}) Q(X_{k-1}, X'_k)}$$

4. With probability $\widetilde{\alpha}(X_{k-1}, X'_k)$
   - Set $X_k = X'_k$ and $\hat{\mathcal{Z}}_{X_k} = \hat{\mathcal{Z}}_{X'_k}$

   otherwise
   - Set $X_k = X_{k-1}$ and $\hat{\mathcal{Z}}_{X_k} = \hat{\mathcal{Z}}_{X_{k-1}}$.

Algorithm 4 is actually Algorithm 2 targetting $\tilde{\pi}$ with a proposal $Q(x, x') \prod_{i=1}^{L} q(Z_n^i|x')$. Then the MH acceptance ratio is simply

$$\begin{aligned}\widetilde{\alpha}(x, x') =& 1 \wedge \frac{p(x') \prod_{i=1}^{L} q(Z_n^i|x') \left( \frac{1}{L} \sum_{i=1}^{L} w(y, Z_n^i, x') \right)}{p(x) \prod_{i=1}^{L} q(Z_{n-1}^i|x) \left( \frac{1}{L} \sum_{i=1}^{L} w(y, Z_{n-1}^i, x) \right)} \cdot \frac{Q(x', x) \prod_{i=1}^{L} q(Z_{n-1}^i|x)}{Q(x, x') \prod_{i=1}^{L} q(Z_n^i|x')} \\ =& 1 \wedge \frac{\hat{\mathcal{Z}}_{x'} p(x') Q(x', x)}{\hat{\mathcal{Z}}_x p(x) Q(x, x')}.\end{aligned}$$

More importantly, because of unbiasedness of $\hat{\mathcal{Z}}_x$ we have:

$$\pi(x) = \int \tilde{\pi}(x, z^1, \ldots, z^L) dz^1 \cdots dz^L.$$

In simulation context marginalisation means we ignore samples for $z^1, \ldots, z^L$ and only use the marginal chain $(X_k; \quad k = 1, \ldots, L)$ for estimating $\widehat{\pi}(\varphi)$. The mixing of this pseudo-marginal algorithm will depend on

- the mixing of the true-marginal MCMC algorithm, i.e. Algorithm 2 applied to $\pi$ with proposal $Q$.
- the variance in the estimators of the marginal likelihood $\mathcal{Z}_x$. Overestimating $\mathcal{Z}_x$ for some state $x$ can cause a chain to spend a large number of time steps in state $x$ as it will require a certain number of iterations to get an estimator that will allow moving away from $x$. This will be improved by increasing $L$ to reduce the variance of $\hat{\mathcal{Z}}_x$.

The last point also opens the question on how should computation be split between the total number of MCMC iterations $N$ and auxiliary variables $L$. A simple strategy is to increase $L$ (iteratively not within the run of the algorithm) until one is finds trace plots, ACF plots and $\frac{\text{ESS}}{N}$ at a satisfactory level and at the same time aim for a long MCMC run. More thorough guidelines backed by rigorous analysis can be found in [31].

We conclude this section by clarifying that the method presented is valid also for other ways of estimating $\mathcal{Z}_x$ as long as $\hat{\mathcal{Z}}_x$ is unbiased. We will revisit this point later when dealing with HMMs and introducing particle MCMC, where particle filters will be used for the estimation of the marginal likelihood.

## 2.5 Discussion

A large number of the concepts and algorithms in this section are fairly basic and commonly taught as undergraduate courses in simulation methods. As a result there was an effort to maintain a concise presentation. In the lectures and slides you will find more details, numerical examples as well as some extensions such as how to estimate the normalising constant $Z$ with MCMC, and how to adapt the proposals of MCMC using the realisation of the chain (adaptive MCMC).

The first point we would like to make is that simulation methods such as IS, MCMC, SMC etc. should not always be viewed as antagonising each other. In fact modern developments seek to combine these so that each can be used within another resulting in more powerful algorithms. This is the case for pseudo-marginal algorithm and we will see more examples later when we introduce SMC.

We have already provided comments regarding the scope and applicability of perfect Monte Carlo, rejection sampling and IS, so will focus the remainder of this discussion on MCMC. MCMC is a very powerful and simple algorithm that can be used very widely. There is a large and still growing literature on both establishing advanced theory and pushing the methodology further. In general, MCMC as a method comes with some weaknesses. First it is quite hard to parallelise, estimation of $Z$ is not straightforward and often lacks in terms of accuracy and theoretical properties. Another common issue is that in multimodal distributions some MCMC techniques (and definitely the ones presented here) might require long time to travel between modes. Often this means in practice that certain important modes can be missed if the runs are not long enough. Finally, MCMC is an iterative method. In Bayesian context this means that it needs the all the data in each iteration as a batch. If new data arrives, we need to re-run everything again from the beginning for the augmented dataset. Despite these shortcomings MCMC is still extremely valuable, but we list them as motivation for using Sequential Monte Carlo and particle methods, which do not suffer from these issues.

### 2.5.1 Key points

Make sure you understand:

- Notation, high level purpose, and principles behind each Monte Carlo method.
- Why sampling is important for Bayesian inference.
- Basic structure, reasoning and justification for each algorithm.
- What is Monte Carlo variance, why is it important and how it manifests in practice in each algorithm.

### 2.5.2 Reading list

For a more detailed introduction you could consult [55, 66].

### 2.5.3 Homework

You are encouraged to work through the following the tasks:

1. Make a note of differences between a rejection sampler and an independence sampler.
2. Implement a Metropolis Hastings Random Walk sampler for the posterior $p(x|y)$, where for the prior we have $x \sim IG(a, b)$, with $a = b = 1$ and we obtain 5 independent observations $Y_i \sim \mathcal{N}(0, X)$, with $i = 1, \ldots, 5$. Reproduce some of the figures in the lecture slides.
3. Implement a Gibbs sampler for the target distribution $\pi = \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix})$. Reproduce some of the figures in the lecture slides.

# 3 Introduction to Non-linear Filtering

## 3.1 Hidden Markov models

A HMM is described in Figure 1 as a directed graphical model. To put it in more mathematical terms, consider a canonical probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with $\Omega = \prod_{n=0}^{\infty} (\mathcal{X} \times \mathcal{Y})^n$ and $\mathcal{F}$ the associated Borel $\sigma$ algebra. We can

write the HMM quite formally for $0 \leq n \leq T$ as:

$$\mathbb{P}\left[X_n \in A | (X_{0:T} = x_{0:T}, Y_{0:T} = x_{0:T})\right] = \int_A f_\theta(x|x_{n-1})dx,$$
$$\mathbb{P}\left[Y_n \in B | (X_{0:T} = x_{0:T}, Y_{0:T} = x_{0:T})\right] = \int_B g_\theta(y|x_n)dy,$$

This means that the hidden process $\{X_n\}_{n\geq 0}$ (that we will call the **state**) is a $\mathcal{X}$-valued latent Markov process of initial density $\eta_\theta(x)$ and Markov transition density $f_\theta(x'|x)$, that is

$$X_0 \sim \eta_\theta(x_0), \ \{X_n | (X_{0:T} = x_{0:T}, \ Y_{0:T} = y_{0:T})\} = \{X_n | (X_{n-1} = x_{n-1})\} \sim f_\theta(x_n|x_{n-1}), \tag{8}$$

whereas the $\mathcal{Y}$-valued observations $Y_n$ are conditional on $X_n$ i.i.d and satisfy:

$$\{Y_n | (X_{0:T} = x_{0:T}, \ Y_{0:T} = y_{0:T})\} = \{Y_n | (X_n = x_n)\} \sim g_\theta(y_n|x_n), \tag{9}$$

where $g_\theta(y|x)$ denotes the conditional likelihood density.

Here we use subscripts to denote $\theta \in \Theta$ being static parameters of the model. Every quantity with a subscript $\theta$ such as $p_\theta, f_\theta, g_\theta$ denote a conditional dependence on the value of $\theta$. When $\theta$ is known this is not so important, but will become so when we discuss its estimation methods in Section 7. Throughout this document we will use $z_{i:j}$ to denote $(z_i, z_{i+1}, ..., z_j)$ for any sequence $\{z_n\}$. In this course we will mainly take $\mathcal{X}$ and $\mathcal{Y}$ to be Euclidean spaces $\mathbb{R}^{d_x}$ and $\mathbb{R}^{d_y}$, but what follows applies to more general state spaces as well. Most of the times we will avoid excessive mathematical details and write casually:

$$\begin{aligned} X_n &\sim f_\theta(\cdot|x_{n-1}), \\ Y_n &\sim g_\theta(\cdot|x_n), \end{aligned} \tag{10}$$

In many practical situations, this model depends on an *unknown* vector parameter $\theta$ that needs to be inferred from the data either in an on-line or off-line manner. In fact, inferring the parameter $\theta$ is often the primary problem of interest. For example, in finance practitioners often want to predict future volatility based on recent asset returns (as observations), which requires some features of this latent process that are captured by parameters of the transition density $f_\theta(x'|x)$. Similarly for biochemical networks, we are not interested in the population of the species per se, but we want to infer some chemical rate constants, which are parameters of the transition prior $f_\theta(x'|x)$. In other applications the parameters $\theta$ are available and the interest is geared more towards inferring the hidden state sequence $\{X_n\}_{n\geq 0}$, e.g. target tracking and navigation in engineering, communications etc.

HMMs are often described in terms of nonlinear and non-Gaussian state space models

$$X_{n+1} = \psi_\theta(X_n, V_{n+1}), \ Y_n = \phi_\theta(X_n, W_n), \tag{11}$$

where $\{V_n\}_{n\geq 1}$ and $\{W_n\}_{n\geq 0}$ are arbitrary i.i.d. noise sequences and $(\psi_\theta, \ \phi_\theta)$ are nonlinear functions. Often these models are time discretisations of continuous time models, e.g. stochastic differential equations. We will see below some examples.

### 3.1.1 Examples of HMMs and state space models

**Linear Gaussian Model** We start by the Linear Gaussian State Space Model. Let $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ and consider

$$\begin{aligned} X_n &= \alpha X_{n-1} + \sigma_v V_n, \\ Y_n &= X_n + \sigma_w W_n, \end{aligned} \tag{12}$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, $X_0 \sim \mathcal{N}(0,1)$. In this case: $\theta = (\alpha, \sigma_v, \sigma_w)$. One can also define a multi-dimensional version with $\mathcal{X} = \mathbb{R}^{d_x}$ and $\mathcal{Y} = \mathbb{R}^{d_y}$

$$X_n = AX_{n-1} + BW_n, Y_n = CX_n + DV_n, \tag{13}$$

$W_n, V_n$ iid zero mean Gaussian vectors and $A, B, C, D$ are appropriate matrices. The Linear Gaussian model is quite a generic model that has been used in many applications, mainly because one can perform many calculations analytically.

**Stochastic Volatility Model** Another popular model with $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ is the Stochastic Volatility Model:

$$\begin{aligned} X_n &= \alpha X_{n-1} + \sigma_v V_n, \\ Y_n &= \beta \exp(\tfrac{X_n}{2}) W_n, \end{aligned} \tag{14}$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$. In this case: $\theta = (\alpha, \sigma_v, \beta)$. Here $X_n$ is the volatility of an asset and $Y_n$ its observed return [46].

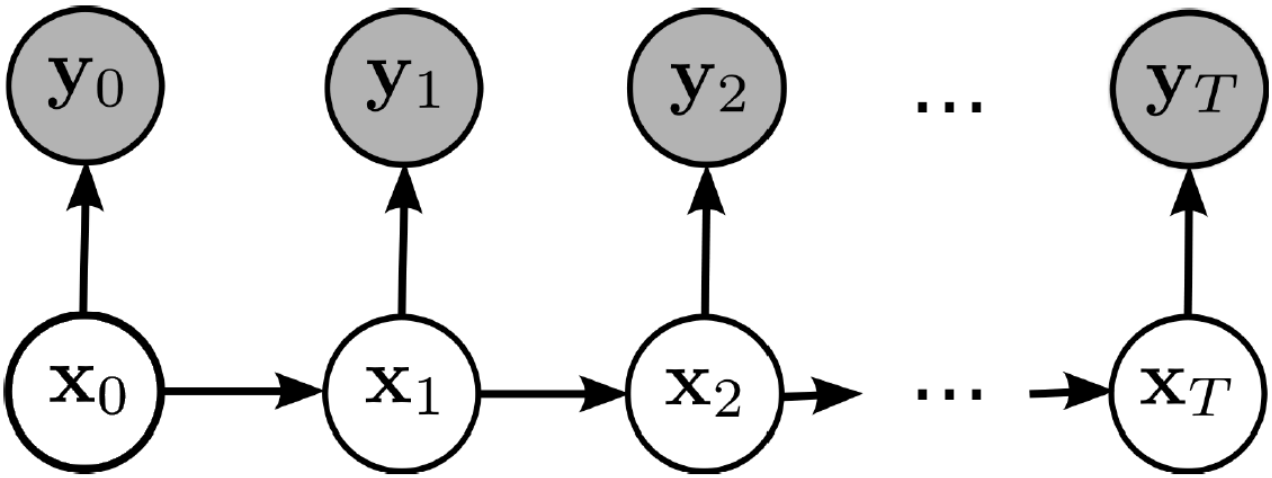


Figure 1: A Hidden Markov Model

**Bearings only tracking** Next we present a model from the engineering literature that relates to target tracking. Let $X_n$ denote the state of an arbitrary moving target on two dimensional plane that is composed of position and velocity in $x$- and $y$- directions, so $\mathcal{X} = \mathbb{R}^4$. Let also $A \in \mathbb{R}^2$ the position of a static observer that takes (passive) bearing observations with additive noise. So write

$$\begin{aligned} X_n &= GX_{n-1} + HW_n, \\ Y_n &= \tan^{-1}\left(\frac{X_n(1) - A(1)}{X_n(3) - A(2)}\right) + V_n, \end{aligned}$$

where $W_n, V_n$ i.i.d. zero mean noise sequences from arbitrary distributions and

$$G = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, H = \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix}.$$

$G$ and $H$ can be viewed as coming from appropriate discretisations of continuous time kinematic models with constant acceleration.

**Stochastic Epidemic model** Compartmental models play a very important role in prediction and understanding of epidemics. We will present a basic continuous time model for a population of fixed size $\mathsf{N}$. The state $X_t = (S_t, I_t, R_t)$ will be composed of the number of susceptible, infectious and recovered individuals and at any time we will have $S_t + I_t + R_t = \mathsf{N}$. This is also refered to as a SIR model. The state is a continuous time Markov process obeying:

$$\begin{aligned} S_t &= S_0 - \mathcal{P}^1\left(\frac{\lambda}{\mathsf{N}}\int_0^t S(r)I(r)dr\right) \\ I_t &= I_0 + \mathcal{P}^1\left(\frac{\lambda}{\mathsf{N}}\int_0^t S(r)I(r)dr\right) - \mathcal{P}^2\left(\gamma\int_0^t I(r)dr\right) \\ R_t &= R_0 + \mathcal{P}^2\left(\gamma\int_0^t I(r)dr\right) \end{aligned}$$

with $\mathcal{P}^1, \mathcal{P}^2$ being independent standard Poisson processes. Due to certain individuals lacking symptoms or not being assessed clinically or via testing, only a proportion of infected individuals is observed, so one can model the observations as

$$Y_{n\Delta} \sim \text{Bin}(I_{n\Delta}, p).$$

Here $\theta = (X_0, \lambda, \gamma, p)$, and note that many important epidemiological quantities require knowledge of $\theta$, e.g. the rate of infection $\mathcal{R}_0 = \frac{\lambda}{\gamma}$.

**Stochastic kinetic models** Finally, we present a class of models that is popular for modelling chemical reactions or predator-prey interactions between species. These models are also known as Lotka-Volterra or Predator-Prey models. Consider two species and let $X^1$ denote the number of prey individuals and $X^2$ the number of predators. One could also similarly use $X^1, X^2$ denote the number of molecules for reactants $A$ and $B$ in a chemical reaction that only changes the concentration of $A, B$. We then model $X^1$and $X^2$ as the continuous time Markov chain following

$$\begin{aligned}
\mathbb{P}\left[X^1_{t+\delta} = x^1_t + 1, X^2_{t+\delta} = x^2_t \middle| X^1_t = x^1_t, X^2_t = x^2_t\right] &= \alpha x^1_t \delta + o(\delta),\\
\mathbb{P}\left[X^1_{t+\delta} = x^1_t - 1, X^2_{t+\delta} = x^2_t + 1 \middle| X^1_t = x^1_t, X^2_t = x^2_t\right] &= \beta x^1_t x^2_t \delta + o(\delta),\\
\mathbb{P}\left[X^1_{t+\delta} = x^1_t, X^2_{t+\delta} = x^2_t - 1 \middle| X^1_t = x^1_t, X^2_t = x^2_t\right] &= \gamma x^2_t \delta + o(\delta).
\end{aligned}$$

We then observe $X^1$at discrete time intervals with additive noise:

$$Y_n = X^1_{n\Delta} + V_n,$$

where $V_n$ is an i.i.d. noise sequence of known density. Here $\theta = (\alpha, \beta, \gamma)$ containts the reaction rate constants.

## 3.2 Bayesian Filtering

Estimation of $\theta$ is a very challenging problem, so we will consider first the scenario where the parameter $\theta$ is *known*. We will introduce filtering as a sequential Bayesian inference problem on the latent state process $\{X_n\}$ given the observations $\{Y_n\}$ (and the known $\theta$). We will see that this is only feasible analytically for simple models such as linear Gaussian state-space models, so later we will introduce approximations based on simulation methods.

### 3.2.1 Filtering, Smoothing and Prediction

In the most general case, the object of interest is the conditional distribution of the whole path $X_{0:n}$ given the observations $Y_{0:n}$. We will call this the **joint filtering distribution** and write it as

$$\Pi_n[\cdot] = \mathbb{P}\left[X_{0:n} \in \cdot \middle| Y_{0:n}\right], \tag{15}$$

and we will denote its density as $p_\theta(x_{0:n}|y_{0:n})$. In other cases we will be only interested in its final marginal and the so called **filtering distribution** or **filter**

$$\pi_n[\cdot] = \mathbb{P}\left[X_n \in \cdot \middle| Y_{0:n}\right] \tag{16}$$

with density $p_\theta(x_n|y_{0:n})$. We would like at time $n$, as $y_n$ becomes available, to use $\Pi_{n-1}$ and $y_n$ to get $\Pi_n$ recursively (and similarly for $\pi_n$). *Filtering* usually denotes this task of estimating recursively in time the sequence of marginal posteriors $\{p_\theta(x_n|y_{0:n})\}_{n\geq 0}$. However we will adopt here a more general definition and will refer to filtering as the task of estimating the sequence of joint posteriors $\{p_\theta(x_{0:n}|y_{0:n})\}_{n\geq 0}$ recursively in time, but we will still refer to the marginals $\{p_\theta(x_n|y_{0:n})\}_{n\geq 0}$ as the filtering densities.

Similarly one can define other quantities such as:

- the joint smoothing density $p(x_{0:n}|y_{0:T})$, $T > n$ (or the marginal smoothing density $p(x_n|y_{0:T})$)
- the joint prediction density $p(x_{0:n+p}|y_{0:n})$, $p \geq 1$ (or the marginal prediction density $p(x_{n+p}|y_{0:n})$)

These are important quantities, whose computation hinges upon having $\Pi_n$ or $\pi_n$ available.

### 3.2.2 Filtering recursions using Bayes rule

Given observed data $y_{0:n}$, inference about the states $X_{0:n}$ may be based on the following posterior density

$$p_\theta(x_{0:n}|y_{0:n}) = \frac{p_\theta(x_{0:n}, y_{0:n})}{p_\theta(y_{0:n})} \tag{17}$$

where, from (9), the joint density of $(X_{0:n}, Y_{0:n})$ is

$$p_\theta(x_{0:n}, y_{0:n}) = \eta_\theta(x_0) \prod_{k=1}^{n} f_\theta(x_k|x_{k-1}) \prod_{k=0}^{n} g_\theta(y_k|x_k) \tag{18}$$

and the *marginal likelihood* is given by

$$p_\theta(y_{0:n}) = \int p_\theta(x_{0:n}, y_{0:n})\, dx_{0:n}. \tag{19}$$

It is easy to verify that the posterior density $p_\theta(x_{0:n}|y_{0:n})$ and the likelihood $p_\theta(y_{0:n})$ satisfy the following fundamental recursions: for $n \geq 1$,

$$p_\theta(x_{0:n}|y_{0:n}) = p_\theta(x_{0:n-1}|y_{0:n-1})\, \frac{f_\theta(x_n|x_{n-1})\, g_\theta(y_n|x_n)}{p_\theta(y_n|y_{0:n-1})} \tag{20}$$

and

$$p_\theta(y_{0:n}) = p_\theta(y_{0:n-1})\, p_\theta(y_n|y_{0:n-1}) \tag{21}$$

where

$$p_\theta(y_n|y_{0:n-1}) = \int g_\theta(y_n|x_n)\, f_\theta(x_n|x_{n-1})\, p_\theta(x_{0:n-1}|y_{0:n-1})\, dx_{0:n}. \tag{22}$$

Similarly to update the marginals, $\pi_n$, one can write a two step recursion for the densities:

$$\begin{aligned} p_\theta(x_n|y_{0:n-1}) &= \int p_\theta(x_n, x_{n-1}|y_{0:n-1})\, dx_{n-1} && (23)\\ &= \int f_\theta(x_n|x_{n-1})\, p_\theta(x_{n-1}|y_{0:n-1})\, dx_{n-1} && (24) \end{aligned}$$

and

$$\begin{aligned} p_\theta(x_n|y_{0:n}) &= \frac{p_\theta(x_n, y_n|y_{0:n-1})}{p_\theta(y_n|y_{0:n-1})} && (25)\\ &= \frac{p_\theta(x_n|y_{0:n-1})\, g_\theta(y_n|x_n)}{p_\theta(y_n|y_{0:n-1})} && (26)\\ &= \frac{p_\theta(x_n|y_{0:n-1})\, g_\theta(y_n|x_n)}{\int p_\theta(x_n|y_{0:n-1})\, g_\theta(y_n|x_n)\, dx_n} && (27) \end{aligned}$$

The Bayesian paradigm is natural here. In (17), $\eta_\theta(x_0) \prod_{k=1}^{n} f_\theta(x_k|x_{k-1})$ plays the role of the prior, $\prod_{k=0}^{n} g_\theta(y_k|x_k)$ the role of the likelihood and $p_\theta(y_{0:n})$ is the evidence or marginal likelihood. A sequential Bayesian formulation is also clear for the joint and marginal recursions. In (20), $p_\theta(x_{0:n-1}|y_{0:n-1})\, f_\theta(x_n|x_{n-1})$ is the prior, $g_\theta(y_n|x_n)$ is the likelihood and $p_\theta(y_n|y_{0:n-1})$ the evidence term. Similarly, in (27) $g_\theta(y_n|x_n)$ is again the likelihood and $p_\theta(x_n|y_{0:n-1})$ the prior. The posterior in time $n-1$, $p_\theta(x_{n-1}|y_{0:n-1})$, enters the prior at time $n$ through the prediction step in (24). Notice that the sequential evidence for both the full path and the marginal case is $p_\theta(y_n|y_{0:n-1})$. This is not surprising as the denominator of (27) is the same as (22). The latter could have been equivalently written as

$$\int g_\theta(y_n|x_n)\, f_\theta(x_n|x_{n-1})\, p_\theta(x_{n-1}|y_{0:n-1})\, dx_{n-1:n}$$

by marginalisation.

*Remark* 1. An important byproduct of this so-called filtering task from a parameter estimation viewpoint is that it provides us with a way to compute the likelihood of the observations given $\theta$. The marginal likelihood is a high dimensional integral of the form

$$p_\theta(y_{0:n}) = \int p_\theta(x_{0:n}, y_{0:n})\, dx_{0:n} = \int \eta_\theta(x_0) \prod_{k=1}^{n} f_\theta(x_k|x_{k-1}) \prod_{k=0}^{n} g_\theta(y_k|x_k)\, dx_{0:n}$$

The particle approximation of this likelihood is a key ingredient of numerous parameter inference techniques. The recursive formulation of the filtering problem allows to view $p_\theta(y_{0:n})$ as a product of lower dimensional integrals

$$p_\theta(y_{0:n}) = \prod_{k=0}^{n} p(y_k|y_{0:k-1}) = \prod_{k=0}^{n} \int g_\theta(y_k|x_k)\, p_\theta(x_k|y_{0:k-1})\, dx_k,$$

with $y_{-1:0}$ being used to denote $y_0$ for convenience. Being able to compute $p(y_n|y_{0:n-1})$ (using $p_\theta(x_k|y_{0:k-1})$ or an approximation of it) simplifies the problem and eventually we will approximate it using a Monte Carlo approach.

It should be clear by now that being able to compute integrals is an essential component of the filtering problem. This is rarely possible for general models, but there are essentially two classes of models for which $p_\theta(x_{0:n}|y_{0:n})$ and $p_\theta(y_{0:n})$ can be computed exactly: the class of linear Gaussian models, for which the above recursions may be implemented using Kalman techniques, and when $\mathcal{X}$ is a finite state-space; see for example [10]. For other models these quantities are typically intractable, i.e. the densities in (20)-(22) cannot be computed exactly, so one needs to resort to approximations.

### 3.3 The Kalman filter and its variants

Consider the Linear Gaussian model in (13). For convenience we omit using $\theta$ in the subscripts in this part. Assume that the initial distribution $\eta = \mathcal{N}(\mu_{0|0}, \Sigma_{0|0})$ and known. Given the initial distribution and $f, g$ in in (13) are all Gaussian, it is clear from standard conjugate properties that when performing recursively the calculations for (24)-(27), we will always obtain a Gaussian distribution. Lets write (24)-(27) as follows:

$$\begin{aligned} p(x_n|y_{0:n-1}) &= \int f(x_n|x_{n-1})p(x_{n-1}|y_{0:n-1})dx_{n-1} \qquad (28)\\ &= \mathcal{N}_{x_n}(\mu_{n|n-1}, \Sigma_{n|n-1}) \\ p(y_n|y_{0:n-1}) &= \int g_n(y_n|x_n)p(x_n|y_{0:n-1})dx_n \qquad (29)\\ &= \mathcal{N}_{y_n}(m_n, S_n) \\ p(x_n|y_{0:n}) &= \frac{g_n(y_n|x_n)p(x_n|y_{0:n-1})}{\int g_n(y_n|x_n)p(x_n|y_{0:n-1})dx_n} \qquad (30)\\ &= \mathcal{N}_{x_n}(\mu_{n|n}, \Sigma_{n|n}) \end{aligned}$$

The Kalman Filter (KF) computes $\mu_{n|n}, \Sigma_{n|n}, m_n, S_n, \mu_{n|n-1}, \Sigma_{n|n-1}$ recursively as follows:

$$\begin{aligned} \mu_{n|n-1} &= A\mu_{n-1|n-1} \qquad (31)\\ \Sigma_{n|n-1} &= A\Sigma_{n-1|n-1}A^T + BB^T \qquad (32)\\ m_n &= C\mu_{n|n-1} \qquad (33)\\ S_n &= C\Sigma_{n|n-1}C^T + DD^T \qquad (34)\\ K_n &= \Sigma_{n|n-1}C^T S_n^{-1} \qquad (35)\\ \mu_{n|n} &= \mu_{n|n-1} + K_n(Y_n - m_n) \qquad (36)\\ \Sigma_{n|n} &= \Sigma_{n|n-1} - K_n C\Sigma_{n|n-1} \qquad (37) \end{aligned}$$

It is quite easy (but tedious) to derive or check the calculations leading to (31)-(37). There are usually two approaches. One invokes properties of the multivariate Gaussian random variables, related to linear transformations (for (31)-(34)) and conditioning ((35)-(37)), and some linear algebra. Another one is more probabilistic and aims to compute the integrals and distributions in (28)-(30). In this case the main tools are:

- The Gaussian integral for a scalar

$$\int_{-\infty}^{\infty} e^{-ax^2+bx+c}\,dx = \sqrt{\frac{\pi}{a}}\,e^{\frac{b^2}{4a}+c}$$

  or in $N_x$ dimensions

$$\int e^{-\frac{1}{2}x^TAx+x^Tb}dx = \sqrt{\frac{(2\pi)^{N_x}}{\det A}}e^{\frac{1}{2}b^TA^{-1}b}.$$

  This is useful to derive (31)-(34) from (28)-(29).

- Taking log transforms and rearranging terms in (30) so that they gather in quadratic form $-\frac{1}{2}x^T\Sigma_n^{-1}x + x^T\mu_n$ and with some algebra one gets (35)-(37).

We leave it to the interested student to perform the full derivation.

*Remark* 2. A useful observation that is not sometimes emphasised enough in the literature is that the KF can be used to compute $p_\theta(y_{0:n}) = \prod_{k=0}^n p(y_k|y_{0:k-1})$ using (33)-(34). This is quite useful in the context of parameter estimation.

The KF recursion is deterministic update requires propagation of the the moments of $\pi_n$, i.e. deterministic quantities. This is easy to implement, and can be quite cheap computationally. Having said that (31)-(37) are not the only possible recursion. In fact there are alternative (correct) formulations such as the Information KF or the square root KF that avoid certain matrix inversion steps and hence could be more efficient in practice. Often in practice practitioners use the KF when $V_n, W_n$ are not actually Gaussian. Clearly this is an approximation whose performance will depend on "how far" the distributions of $V_n, W_n$ are from a Gaussian..

### 3.3.1 The Extended Kalman Filter (EKF)

It is possible to extend these ideas for non-linear state space models of the form of (11) using linearisations and first order Taylor approximations. This is known as the Extended Kalman Filter (EKF). To get the EKF one needs to replace (11) with an approximate linear Gaussian model

$$X_n = a_n + A_n X_{n-1} + B_n V_n,\ Y_n = c_n + C_n X_n + D_n W_n, \tag{38}$$

Linearisation of $\psi, \phi$ around $\mu_{n-1|n-1}$ and $\mu_{n|n-1}$ respectively gives

$$A_n = \nabla_x \psi_\theta|_{\mu_{n-1|n-1}}, \quad C_n = \nabla_x \phi_\theta|_{\mu_{n|n-1}}$$

and similarly assuming the noises are zero mean we can set

$$B_n = \nabla_V \psi_\theta|_0, \quad D_n = \nabla_W \phi_\theta|_0,$$

so ignoring the higher order terms of the Taylor expansion one gets

$$a_n = \psi_\theta\left(\mu_{n-1|n-1}, 0\right) - A_n \mu_{n-1|n-1}, \quad c_n = \phi_\theta\left(\mu_{n|n-1}, 0\right) - C_n \mu_{n|n-1}$$

The EKF computes $\mu_{n|n}, \Sigma_{n|n}, m_n, S_n, \mu_{n|n-1}, \Sigma_{n|n-1}$ recursively as

$$\mu_{n|n-1} = a_n + A_n \mu_{n-1|n-1} \tag{39}$$

$$\Sigma_{n|n-1} = A_n \Sigma_{n-1|n-1} A_n^T + B_n B_n^T \tag{40}$$

$$m_n = c_n + C_n \mu_{n|n-1} \tag{41}$$

$$S_n = C_n \Sigma_{n|n-1} C_n^T + D_n D_n^T \tag{42}$$

$$K_n = \Sigma_{n|n-1} C_n^T S_n^{-1} \tag{43}$$

$$\mu_{n|n} = \mu_{n|n-1} + K_n(Y_n - m_n) \tag{44}$$

$$\Sigma_{n|n} = \Sigma_{n|n-1} - K_n C_n \Sigma_{n|n-1} \tag{45}$$

The differences from the KF implementation are minor and due to the linearisations. These are the inclusion of the additive terms and $a_n, c_n$ and having to use time varying matrices $A_n, B_n, C_n, D_n$.

The EKF is quite easy to implement as the recursion is deterministic and simply propagates the same quantities as the KF. Derivations related derivatives can be done off-line it can be quite cheap and fast computationally. Sometimes it can give reasonable answers, and this is typically for models like

$$X_n = \psi_\theta\left(X_{n-1}\right) + BV_n,\ Y_n = \phi_\theta\left(X_n\right) + DW_n,$$

where the nonlinearities in $\phi, \psi$ are smooth (so locally close to being linear), and $V_n, W_n$ Gaussian, so in some way we are "close to" a linear Gaussian model regime. On the other hand it is hard to predict whether it will work and how well. It consists of an approximation that is hard to justify in many cases: when the higher order terms in the Taylor series are significant, when the noise are non-Gaussian, or when $\phi, \psi$ are not differentiable everywhere. This has motivated the quest for improved non-linear filters since the 60-s. Currently and since the 90-s, *particle filters* are the state of the art. For completeness and motivation, in the following section we will present a short and critical overview of different avenues to solve the filtering problem without simulation methods.

## 3.4 Non-linear filters without using simulation

The question that rises is whether we can improve the performance EKF, while at the same time keeping its core idea of propagating moments or other sufficient statistics. In nonlinear non-Gaussian scenarios, over the past decades numerous approximation schemes following these principles have been proposed. Common ideas that appeared could be related to using improved deterministic numerical integration methods. We will look at some popular ones, such as the Gaussian sum filter [1] or the Unscented KF (UKF) [41, 42]. Not many details will be provided on the derivation or implementation of these schemes. We will restrict to a critical discussion to serve as a motivation for particle filters and using simulation.

### 3.4.1 Some finite dimensional filters

Consider for a minute the KF, which propagates the first two moments of $\pi_n$ recursively. In the linear Gaussian case due to conjugacy of Gaussian distribution, it is sufficient to propagate only the filter mean and covariances to capture the full distribution of $\pi_n$. Would it be possible to apply the same ideas related to conjugate priors for non-Gaussian cases and propagating fixed number of moments or other sufficient statistics to to capture a non Gaussian $\pi_n$ in closed form? Such filters are often called finite dimensional. In principle is a good idea but unfortunately there are very few cases that this is possible in practice: [67, 73, 74].

### 3.4.2 The Gaussian Sum filter

In the early 70-s Alspach and Sorenson proposed the Gaussian Sum filter in [1]. The key idea is to extend the KF/EKF by propagating a mixture of Gaussians, instead of a single Gaussian distribution in time. Use a linearisation similar to EKF, but construct an approximation based on the following mixture

$$\pi_{n|n-1} = \sum_{i=1}^{q'_n} w'_{n,i} \mathcal{N}_{x_n}(\mu'_{n,i}, \Sigma'_{n,i})$$
$$\pi_n = \sum_{i=1}^{q_n} w_{n,i} \mathcal{N}_{x_n}(\mu_{n,i}, \Sigma_{n,i})$$

where $\sum_{i=1}^{q'_n} w'_{n,i} = \sum_{i=1}^{q_n} w_{n,i} = 1$ and $\pi_{n|n-1}$ is used as shorthand for $p(x_n|y_{0:n-1})$. Note that now one needs to propagate in time the number of elements in the mixture, each weight as well as the mean and covariance for each component in the mixture. In [1], the authors present recursions for $w_n, q_n, \mu_{i,n}, \Sigma_{i,n}$ and $w'_n, q'_n, \mu'_{i,n}, \Sigma'_{i,n}$. They demonstrate that this approach does indeed address some limitations of EKF such as multimodality. On the other hand the approach is still based on linearisation (like the EKF) and $q_n$ increases linearly with time, so we end up with increasing computational cost per time. For a particular class of models (conditionally linear Gaussian models), these limitations were addressed much later in [14] and the so called Mixture Kalman Filter, which is uses simulation ideas and is similar with some of the particle methods we will see in this course.

### 3.4.3 The Unscented KF and other approaches using deterministic integration

At the heart of the filtering problem lies the problem of numerical integration. So one may wonder whether application of advanced quadrature or cubature techniques are useful. To get some of the intuition behind this approach, consider computing numerically integrals of the form:

$$\int_{\mathcal{A}} \varphi(x)dx \approx \sum_{i=1}^{N} \alpha_k \varphi(X_k).$$

On a very high level, these methods will provide a design for good choice $\{\alpha_k\}_{k=1}^N$ and placement of points $\{X_k\}_{k=1}^N$ (later referred as sigma points) using information on the structure of $\mathcal{A}$ and smoothness properties of $\varphi$. The integral above corresponds to integrating with the uniform distribution, but there are cubature extensions for Gaussian or other integrals that exploit symmetries, choice of appropriate basis functions and many sophisticated extensions.

Returning to the filtering problem, these ideas have appeared in [41, 42] as the Unscented KF (UKF) and were extended more recently in [7] under the name of Cubature KF (CKF). We will discuss briefly the UKF, as it is simpler and has been very popular especially for models that are "close" to Gaussian. The UKF builds upon the so the Unscented Transform (UT), which is a quadrature method that computes the mean and covariance of a Gaussian random variable that undergoes a nonlinear transformation.

Let $\mathtt{X}$ be a Gaussian random variable $\mathtt{X} \sim \mathcal{N}(\mu, QQ^T)$. Say we are interested to approximate the mean and covariance of $h(\mathtt{X})$, where $h$ is a smooth non-linear transform. The UT procedure is presented in Algorithm 5. Firstly we will form a set of $2L+1$ "sigma-points" to capture most of the mass in the support of the distribution of $\mathtt{X}$ (Step 1) and then we will propagate the sigma points through $h(\cdot)$ and use quadrature estimates (Step 2) to get an approximate mean and covariance of $h(\mathtt{X})$. Here, $\lambda, \kappa$ are tuning parameters that are used to improve on the performance of the algorithm; see [41, 42] for some guidelines on how to choose them.

The UT method of Algorithm 5 can be useful for extending and improving some of the steps in the EKF (the ones in (39)-(42)). Consider again the model in (11) and the EKF in (39)-(45). The UKF replaces (39)-(40) by applying Algorithm 5 with $\mathtt{X} = (X_{n-1}, V_n)$, $\mathtt{Y} = X_n$ and $h = \psi_\theta$ to get $\mu_{n|n-1}, \Sigma_{n|n-1}$. Similarly, it replaces

**Algorithm 5** The Unscented Transform for $\mathtt{Y} = h(\mathtt{X})$, where $\mathtt{X} \sim \mathcal{N}(\mu, \Sigma)$ with $\Sigma = QQ^T$ .

1. Initialise

$$\mathtt{X}^0 = \mu$$

For $i = 1, \ldots, L$ set

$$\begin{aligned} \mathtt{X}^i &= \mu + \sqrt{n+\lambda}\,[Q]_i \\ \mathtt{X}^{i+n} &= \mu - \sqrt{n+\lambda}\,[Q]_i \end{aligned}$$

with $\left[Q_{n|n}\right]_i$ denoting the i-th column of the matrix $Q$.

2. Then propagate the sigma points through $h(\cdot)$

$$\mathtt{Y}^i = h(\mathtt{X}^i), \quad i = 0, \ldots, 2L$$

Use quadrature estimates

$$\begin{aligned} \mathbb{E}\left[h(\mathtt{X})\right] = \mu_h &\approx \sum_{i=0}^{2L} W_m^i \mathtt{Y}^i \\ \mathbb{C}ov\left[h(\mathtt{X})\right] = \Sigma_h &\approx \sum_{i=0}^{2L} W_c^i \left(\mathtt{Y}^i - \mu_h\right)\left(\mathtt{Y}^i - \mu_h\right)^T \end{aligned}$$

with

$$\begin{aligned} W_m^0 &= \frac{\lambda}{L+\lambda} \\ W_c^0 &= \frac{\lambda}{L+\lambda} + \kappa \\ W_m^i &= W_m^i = \frac{1}{2\left(L+\lambda\right)}, \quad i = 1, \ldots, L \end{aligned}$$

(41)-(42), by applying again Algorithm 5 using $\mathtt{X} = (X_n, W_n)$, $\mathtt{Y} = Y_n$, $h = \phi_\theta$ to get $m_n, \Sigma_n$ . For the update steps both UKF and EKF use the same steps, i.e. (43)-(45).

## 3.5 Discussion

Many methods have been proposed over the past fifty years to solve the filtering problem, but these methods lack formal justification and can be unreliable in practice in terms of accuracy, while deterministic integration methods are difficult to implement. The more sophisticated extensions of EKF using numerical integration, such as the UKF/CKF can work much better than EKF in many cases. The UKF is said to work well when filter is close to a Gaussian distribution and one may view the CKF as its refinement from the perspective of numerical integration using advanced cubature methods. On the other hand we are still in a case that requires very strong assumptions on model, for example how close is the given model to a Gaussian one or how close $\phi_\theta, \psi_\theta$ are to polynomials or smooth functions. It is also worth noting that cubature and quadrature techniques are quite cumbersome to apply in higher than one dimensions and their performance can scale very poorly with dimension of $X_n, Y_n$. We have seen also the Gaussian sum filter that seem to relax some assumptions on the noise distributions, but its implementation might result to increasing computational cost per time (at least for the vanilla version presented here).

Based on what we have seen so far to solve numerically a general non-linear filtering problem, one needs to use simulation based methods. Markov chain Monte Carlo (MCMC) methods are a possible candidate and can obviously be used, but they are impractical for on-line inference. Even for off-line inference, it can be difficult to build efficient proposal distributions and MCMC algorithms for the high-dimensional target distributions that are needed for high $n$. For non-linear non-Gaussian state space models, *particle filtering* algorithms are the methods which have emerged as the most successful. The widespread popularity of these SMC methods is due to the fact that they are easy to implement, suitable for parallel implementation and more importantly, have been demonstrated in numerous settings to yield more accurate estimates than previously proposed alternatives; e.g. see [10], [21], [28],

[54].

### 3.5.1 Key points

Make sure you understand:

- The HMM structure and state space models.
- The framework of Bayesian inference for filtering when $\theta$ is known.
- The basic filtering recursions for the path space and the marginal filtering distributions.
- For linear and Gaussian state space models, filtering is analytically tractable. Familiarise with the KF and EKF, the quantities it computes and ensure that you are able implement a KF.

### 3.5.2 Reading list

If you find the topic in this section interesting, you could read further in:

- Sarkka [68]: Chapters 1, 2 for a general introduction, Chapter 4 for an introduction to filtering and the KF, and Chapter 5 for a discussion on the EKF and the UKF;
- Douc et. al. [27]: Chapter 3 for a review on state space models and interesting applications.

### 3.5.3 Homework

You are strongly encouraged to complete the tasks below, especially the ones that are not marked by (*). The tasks marked as (*) are more challenging, but will be useful later as the course develops.

For the scalar model $X_n = \rho X_{n-1} + \tau V_n, \quad Y_n = X_n + \sigma W_n$, where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, and $X_0 \sim \mathcal{N}(0,1)$:

1. Run the model to synthesise a data-set $y_{0:T}$ for $T = 50$, $\rho = 0.8, \tau = 1, \sigma = 0.1$. Store the real state trajectory $x^*_{0:T}$ for future comparisons.
2. Implement the Kalman filter and compare $\mu_{n|n}, \mu_{n|n-1}$ with $x^*_{0:T}$.
   - Plot estimated means, true state vs time $n$ together with confidence intervals from $\Sigma_{n|n-1}, \Sigma_{n|n}$ and notice any differences.
3. (*) Assume only $\rho, \tau$ are unknown and let $\theta = (\rho, \tau)$. Using the computed $p_\theta(y_{0:n})$ from the Kalman filter design a MCMC chain to target $p(\theta|y_{0:T})$. You may use a uniform prior on $[-1, 1]$ for $\rho$ and and an inverse gamma prior, $IG(1, 1)$, for $\tau^2$
4. (*) Validate your results in 4. using a grid method to compute the posterior. If computing the two dimensional posterior this way proves tricky, try it for just one unknown parameter.
5. Repeat some of the above steps for different data-sets, but same parameter values.

# 4 Introduction Particle Filtering

## 4.1 Sequential Importance Sampling (SIS) as a general procedure

We well return to Monte Carlo and focus on sequential problems like the ones encountered in filtering. Let say we are interested to do IS for the following arbitrary target density:

$$\pi(x_{0:T}) = \frac{\gamma(x_{0:T})}{Z},$$

We will present a generic approach to perform IS recursively. Define a sequence of target distributions $\{\pi_n(x_{0:n}) = \frac{\gamma_n(x_{0:n})}{\mathcal{Z}_n}\}_{n \leq T}$, with $\pi = \pi_T$. It is not restrictive to assume that for the target density the following product factorisation holds

$$\gamma_n(x_{0:n}) = \gamma_{n-1}(x_{0:n-1}) r_n(x_n | x_{0:n-1}),$$

**Algorithm 6** Sequential Importance Sampling: at each $n$ we we have available $\{X^i_{0:n-1}, W^i_{n-1}\}_{i=1}^N$ and then compute Steps 1. and 2. Note that for simplicity we are using the notation $q_0(\cdot|x_{-1}) = q_0(x)$.

For each $n \geq 0$

1. The sampling step: For $i = 1, \ldots, N$,
   (a) sample particles: $X^i_n \sim q_n\left(\cdot|X^i_{0:n-1}\right)$,
   (b) Augment the path of the state: $X^i_{0:n} = \left(X^i_{0:n-1}, X^i_n\right)$.
2. Compute weight: For $i = 1, \ldots, N$,
   (a) Compute un-normalised weight
   $$\widetilde{W}^i_n = W^i_{n-1} \frac{r_n(X^i_n|X^i_{0:n-1})}{q_n\left(X^i_n|X^i_{0:n-1}\right)}.$$
   (b) Compute normalised weight $W^i_n = \frac{\widetilde{W}^i_n}{\sum_{j=1}^N \widetilde{W}^j_n}$.

where $r_n(x_n|x_{0:n-1}) = \frac{\gamma_n(x_{0:n})}{\gamma_{n-1}(x_{0:n-1})}$. In a similar manner we can construct proposal or instrumental density as follows

$$q_n(x_{0:n}) = q_{n-1}(x_{0:n-1}) q_n(x_n|x_{0:n-1})$$

and then obtain a recursive expression for the IS weight

$$w_n(x_{0:n}) = w_{n-1}(x_{0:n-1}) \frac{r_n(x_n|x_{0:n-1})}{q_n(x_n|x_{0:n-1})}.$$

This simple construction is quite useful, because in this way the proposal simply extends the path $x_{0:n-1}$ with $x_n$, so that previous samples/particles can be reused. Then the weight can be computed recursively to correct for the fact that we use a different distribution for the proposal than the target. A generic SIS algorithm is presented in Algorithm 6. We will later apply this for the filtering problem targeting $\Pi_n$, but SIS could be applied to a variety of different problems, e.g sampling from a self avoiding random walk.

At time $n$, the approximations of $\pi_n$ and $\mathcal{Z}_n$ after the sampling step are given by

$$\widehat{\pi_n}\left(dx_{0:n}\right) = \sum_{i=1}^N W^i_n \delta_{X^i_{0:n}}\left(dx_{0:n}\right),$$
$$\widetilde{\mathcal{Z}_n} = \frac{1}{N} \sum_{i=1}^N w_n(X^i_{0:n}). \tag{46}$$

One can interpret this estimate of $\mathcal{Z}_n$ as a standard IS estimate:

$$\widetilde{\mathcal{Z}_n} = \int \frac{\gamma_n(x_{0:n})}{q_n(x_{0:n})} \widehat{q}_n(dx_{0:n}) = \frac{1}{N} \sum_{i=1}^N w_n(X^i_{0:n})$$

where $\widehat{q}_n(dx_{0:n}) = \frac{1}{N} \sum_{i=1}^N \delta_{X^i_{0:n}}(dx_{0:n})$ and the sampling distribution is given by: $X^{1:N}_{0:n} \sim \prod_{i=1}^N \prod_{k=0}^n q_n(dx^i_n|x^i_{0:n-1})$. There is nothing special about this estimator and and the intuition and results follow directly using standard IS arguments. $\widetilde{\mathcal{Z}_n}$ in (46) is unbiased and its relative variance is given by

$$\frac{\mathbb{V}ar^N\left[\widetilde{\mathcal{Z}_n}\right]}{\mathcal{Z}_n^2} = \frac{1}{N}\left(\int \frac{\left(\pi_n\left(x_{0:n}\right)\right)^2}{q_n(x_{0:n})} dx_{0:n} - 1\right)$$

Another estimator for the normalising constant can be constructed using

$$\widehat{\mathcal{Z}_n} = \prod_{k=0}^{n} \widehat{\frac{\mathcal{Z}_k}{\mathcal{Z}_{k-1}}}, \tag{47}$$

$$\widehat{\frac{\mathcal{Z}_n}{\mathcal{Z}_{n-1}}} = \sum_{i=1}^{N} W_{n-1}^i \frac{r_k(X_n^i | X_{0:n-1}^i)}{q_n\left(X_n^i | X_{0:n-1}^i\right)}. \tag{48}$$

This sequential estimator that will prove useful later and is motivated by considering the identity

$$\begin{aligned}
\mathcal{Z}_n &= \prod_{k=0}^{n} \frac{\mathcal{Z}_k}{\mathcal{Z}_{k-1}} \\
&= \prod_{k=0}^{n} \frac{\int \gamma_k(x_{0:k}) dx_{0:k}}{\int \gamma_{k-1}(x_{0:k-1}) dx_{0:k-1}} \\
&= \prod_{k=0}^{n} \int \frac{\gamma_{k-1}(x_{0:k-1})}{\int \gamma_{k-1}(x_{0:k-1}) dx_{0:k-1}} r_k(x_k | x_{0:k-1}) dx_{0:k} \\
&= \prod_{k=0}^{n} \int \pi_{k-1}(x_{0:k-1}) r_k(x_k | x_{0:k-1}) dx_{0:k}
\end{aligned}$$

We can use IS and rewrite the $k$-th term in the product as

$$\int \pi_{k-1}(x_{0:k-1}) r_k(x_k | x_{0:k-1}) dx_{0:k} = \int \frac{r_k(x_k | x_{0:k-1})}{q_k(x_k | x_{0:k-1})} \pi_{k-1}(x_{0:k-1}) q_k(x_k | x_{0:k-1}) dx_{0:k}$$

and then derive (48) as a consistent Monte Carlo approximation

$$\begin{aligned}
\widehat{\frac{\mathcal{Z}_n}{\mathcal{Z}_{n-1}}} &= \int \frac{r_n(x_n | x_{0:n-1})}{q_n(x_n | x_{0:n-1})} \left[\hat{\pi}_{n-1} \otimes \hat{q}_n\right](dx_{0:n}) \\
&= \sum_{i=1}^{N} W_{n-1}^i \frac{r_n(X_n^i | X_{0:n-1}^i)}{q_n\left(X_n^i | X_{0:n-1}^i\right)}
\end{aligned}$$

with $\left[\hat{\pi}_{n-1} \otimes \hat{q_n}\right](dx_{0:n}) = \sum_{i=1}^{N} W_{n-1}^i q_n(dx_n | x_{0:n-1}) \delta_{X_{0:n-1}^i}(dx_{0:n-1})$. Due to the use of the normalised weight $W_{n-1}^i$, it is clear that this estimator cannot be unbiased and that the (asymptotic) analysis of this estimator is more complicated. In any case, using this sequential approach will become more useful later when considering particle filtering.

### 4.1.1 SIS for the filtering problem

Lets apply SIS for the filtering on the path space. Recall $\Pi_n(\cdot) = \mathbb{P}\left[X_{0:n} \in \cdot \,|\, Y_{0:n}\right]$ and for the density of interest we have

$$\gamma_n(x_{0:n}) = \prod_{k=0}^{n} f_\theta\left(x_k | x_{k-1}\right) g_\theta\left(y_k | x_k\right)$$

and

$$\mathcal{Z}_n = p(y_{0:n}),$$

where for convenience we will write $f(x_0|x_{-1})$ to be $\eta(x_0)$. Given the Markov structure of the HMM we will use importance densities of the form $q_\theta\left(x_0 | y_0\right)$ and $q_\theta\left(x_n | y_n, x_{n-1}\right)$. So the importance weight can be written as

$$w_n(x_{0:n}) = \frac{\gamma_n(x_{0:n})}{q_n(x_{0:n})} = \prod_{k=0}^{n} \frac{f_\theta\left(x_k | x_{k-1}\right) g_\theta\left(y_k | x_k\right)}{q_\theta\left(x_k | y_k, x_{k-1}\right)} := \prod_{k=0}^{n} \omega_k(x_{k-1}, x_k),$$

where we have define the incremental importance weights as follows:

$$\omega_0\left(x_0\right) = \frac{\eta_\theta\left(x_0\right) g_\theta\left(y_0 | x_0\right)}{q_\theta(x_0 | y_0)}, \tag{49}$$

$$\omega_n\left(x_{n-1:n}\right) = \frac{r_n(x_n | x_{0:n-1})}{q_n(x_n | x_{0:n-1})} = \frac{f_\theta\left(x_n | x_{n-1}\right) g_\theta\left(y_n | x_n\right)}{q_\theta\left(x_n | y_n, x_{n-1}\right)} \text{ for } n \geq 1. \tag{50}$$

**Algorithm 7** SIS for filtering. In steps 1(b) and 2(b) we are presenting computation of un-normalised weight and normalisation in one go. One could look at Step 2. of Algorithm 6 for details.

1. At time $n = 0$
   (a) Sample $X_0^i \sim q_\theta(x_0|\, y_0)$.some
   (b) Compute the weights $w_0\left(X_0^i\right)$ and set $W_0^i \propto w_0\left(X_0^i\right)$, $\sum_{i=1}^N W_0^i = 1$.
2. At time $n \geq 1$
   (a) Sample $X_n^i \sim q_\theta(x_n|\, y_n, X_{n-1}^i)$ and set $X_{0:n}^i = \left(X_{0:n-1}^i, X_n^i\right)$.
   (b) Compute the weights $\omega_n\left(X_{n-1:n}^i\right)$ and set $W_n^i \propto W_{n-1}^i \omega_n\left(X_{n-1:n}^i\right)$, $\sum_{i=1}^N W_n^i = 1$.

An SIS algorithm for filtering is presented in Algorithm 7. At time $n$, the approximations of $p_\theta\left(x_{0:n}|\, y_{0:n}\right)$ and $p_\theta\left(y_n|\, y_{0:n-1}\right)$ after the sampling/weighting step (step 2. of Algorithm 7) are

$$\widehat{p}_\theta\left(dx_{0:n}|\, y_{0:n}\right) = \sum_{i=1}^N W_n^i \delta_{X_{0:n}^i}\left(dx_{0:n}\right), \tag{51}$$

$$\widehat{p}_\theta\left(y_n|\, y_{0:n-1}\right) = \sum_{i=1}^N W_{n-1}^i \omega_n\left(X_{n-1:n}^i\right). \tag{52}$$

Hence an estimate of the marginal likelihood is given either by

$$\widehat{p}_\theta\left(y_{0:n}\right) = \widehat{p}_\theta\left(y_0\right) \prod_{k=1}^n \widehat{p}_\theta\left(y_k|\, y_{0:k-1}\right). \tag{53}$$

or can be approximated unbiasedly using (46) with

$$\widetilde{p}_\theta\left(y_{0:n}\right) = \frac{1}{N} \sum_{i=1}^N \prod_{k=0}^n \omega_k(X_{k-1}^i, X_k^i), \tag{54}$$

where $\prod_{k=0}^n \omega_k(X_{k-1}^i, X_k^i)$ can be trivially computed recursively in time.

These approximations enjoy a similar theoretical justification as IS shown in Section 2.3. Let $\varphi : \mathcal{X}^n \to \mathbb{R}^d$ be a bounded measureable test function and let also the integral of interest be

$$I_n = \int \varphi(x_{0:n}) p_\theta\left(x_{0:n}|\, y_{0:n}\right) dx_{0:n}$$

and its particle approximation from SIS be

$$\begin{aligned} \hat{I}_n &= \int \varphi(x_{0:n}) \hat{p}_\theta\left(dx_{0:n}|\, y_{0:n}\right) \\ &= \sum_{i=1}^N W_n^i \varphi\left(X_{0:n}^i\right). \end{aligned}$$

So we maintain the following properties:

- $\hat{I}_n$ is asymptotically consistent as $N \to \infty$ with asymptotic bias

$$\left(\widehat{I}_n - I_n\right) = -\frac{1}{N} \int_{\mathcal{X}^n} \frac{\left(p_\theta\left(x_{0:n}|\, y_{0:n}\right)\right)^2}{q_n(x_{0:n})} \left(\varphi(x_{0:n}) - I_n\right) dx_{0:n}$$

- A CLT holds with

$$\sqrt{N}\left(\widehat{I}_n - I_n\right) \Rightarrow \mathcal{N}\left(0, \sigma_{SIS}^2(\varphi, q, n)\right)$$

  where

$$\sigma_{SIS}^2(\varphi, q, n) = \frac{1}{N} \left( \int_{\mathcal{X}^n} \frac{\left(p_\theta\left(x_{0:n}|\, y_{0:n}\right)\right)^2}{q_n(x_{0:n})} \left(\varphi(x_{0:n}) - I_n\right)^2 dx_{0:n} \right)$$

- The estimate $\tilde{p}(y_{0:n})$ in (54) is unbiased and will have a relative variance (the variance of $\widehat{Z_n}/Z_n$) as follows:

$$\frac{\mathbb{V}ar^N\left[\widehat{Z_n}\right]}{Z_n^2} = \frac{1}{N}\left(\int \frac{\left(p_\theta\left(x_{0:n}|\, y_{0:n}\right)\right)^2}{q_n(x_{0:n})} dx_{0:n} - 1\right). \tag{55}$$

These are the same properties seen earlier for IS. Essentially the only difference is we are computing the weight recursively. The only thing we have changed is to substitute expressions for $\pi$ with $p_\theta\left(x_{0:n}|\, y_{0:n}\right)$ in the IS expressions mentioned of Section 2.3.

Note that despite some initial reassurance and elegance, the consistency and asymptotic normality results, are not informative enough on whether Algorithm 7 is a practically useful algorithm. We have not mentioned how $\sigma^2_{SIS}(\varphi, q, n)$ behaves with time or how the speed of convergence of $\left(\widehat{I_n} - I_n\right)$ to 0 changes with $n$. Similarly we do not know how $\frac{\mathbb{V}ar^N\left[\widehat{Z_n}\right]}{Z_n^2}$ grows with $n$. These are important questions because in principle we want to run Algorithm 7 for a high $n$, so how the accuracy changes with $n$ is crucial. It turns out that these quantities grow too quickly with $n$ to be useful. This dependence could be exponential or polynomial. As $n$ increases $W_n^i \leq 1$, low weights will remain low for each particle (in fact most will decay exponentially) and due to normalisation most weight mass will concentrates to fewer and fewer particles (most likely at some point to one particle). As a result the variance of the weights will increase and hence the variance of the estimates eventually explodes.

SIS is a basic building block of modern particle filtering techniques, but on its own as illustrated Algorithm 7 will face serious issues related to efficiency and accuracy as $n$ increases. One should not expect this method to be effective beyond $n$ being 20 or 30. We conclude the presentation of SIS for filtering with some historical remarks. Very soon after Kalman a variant of this basic Monte Carlo filter appeared in [37]. This was the first SIS filter proposed in the literature (to the best of my knowledge) and [37] used $q_n = f_\theta$ in the proposal. We will later refer to this as the **bootstrap** proposal. This contribution was quite ahead of its time, and given the lack of significant computing resources at the time, it took some decades for resampling to be used and particle filters to develop.

## 4.2 Choosing Importance Proposals for HMMs

As in standard IS, the key to designing well performing algorithms is to find a good proposal $q_n$. We will focus on how this can be achieved for HMMs. Although so far, we have only presented SIS and not a full particle filtering algorithm, the problem of finding a good proposal is the same for both cases, because the IS step is common to both algorithms.

We now look at what criteria could be used for choosing $q_\theta\left(x_0|\, y_0\right)$ and $q_\theta\left(x_n|\, y_n, x_{n-1}\right)$ . We can rule out approaches that aim to minimise the asymptotic variance of the estimator $\hat{I}_n$ like

$$q(x_{0:n}) \propto \left|\varphi\left(x_{0:n}\right)\right| p_\theta\left(x_{0:n}|\, y_{0:n}\right)$$

as these are not useful in a filtering context, where we are interested in the expectations of several test functions (e.g. moments or simple functions). One simple and popular option is to use

$$q_\theta\left(x_n|\, y_n, x_{n-1}\right) = f_\theta\left(x_n|\, x_{n-1}\right)$$

and hope that $\omega_n\left(x_{n-1:n}\right) = g(y_n|x_n)$ will not have very high variance. This option is often called the bootstrap proposal and will perform well when $p_\theta\left(x_{0:n}|\, y_{0:n}\right)$ does not change very fast with $n$. This will correspond to $y_n$ being not very informative (i.e. having a fairly flat and not peaky likelihood function). When the observations are quite accurate this approach will not very effective and we will need a mechanism to guide the particles to areas of higher $p_\theta\left(x_{0:n}|\, y_{0:n}\right)$ using some information from $y_n$.

A better approach would be to attempt minimise the relative variance of the normalising constant $\hat{Z}_n$, which is equivalent to minimising the variance of the importance weights and will result in $q_n(x_{0:n})$ being very similar or close to $p_\theta\left(x_{0:n}|\, y_{0:n}\right)$. This can be achieved using the so called optimal proposal (w.r.t the variance of the weights) given as:

$$q_\theta^{opt}\left(x_n|\, y_n, x_{n-1}\right) = p_\theta\left(x_n|\, y_n, x_{n-1}\right).$$

This leads to weight ratio being

$$\omega_n^{opt}(x_{n-1}, x_n) = p_\theta(y_n|x_{n-1}).$$

Note both $\omega_n^{opt}, q_n^{opt}$ are not possible to compute analytically, so one needs to resort to approximations like the EKF or UKF to construct a good proposal that looks similar to $p_\theta\left(x_n|\, y_n, x_{n-1}\right)$ (but then the weight will be given by (50) and not by $\omega_n^{opt}$).

We will see an example of designing $q_\theta(x_n|y_n, x_{n-1})$ based on the linearisations similar to the EKF (but not the same). Consider the state usual state space model given by in (11) with $V_n, W_n$ both zero mean i.i.d noise sequences with identity variance. In particular we will use the Gaussian approximation

$$\begin{aligned} q_\theta(x_n|y_n, x_{n-1}) &\propto q_\theta^f(x_n|x_{n-1})\, q_\theta^g(y_n|x_n) \\ &= \mathcal{N}_{x_n}(\mathfrak{m}_n(x_{n-1}), \mathcal{S}_n(x_{n-1})) \end{aligned}$$

where here $q_\theta^f, q_\theta^g$ here denote Gaussian approximations of $f_\theta, g_\theta$ respectively. Recall, using EKF one can approximate $f(x_n|x_{n-1})$ as a Gaussian using a linearisation of $\psi$ around the noise and its mean 0. Then the following expressions

$$X_n = \psi_\theta(X_{n-1}, 0) + B_n V_n, \quad B_n = \nabla_V \psi_\theta(X_{n-1}, 0), \quad V_n \sim \mathcal{N}(0, I) \tag{56}$$

give $q_\theta^f(x_n|x_{n-1})$. Similarly a linearisation of $\phi$ around $\psi_\theta(X_{n-1}, 0)$ (the mean of $X_n$ in (56)) to get $q_\theta^g(y_n|x_n)$ as follows:

$$Y_n = \phi_\theta(\psi_\theta(X_{n-1}, 0), 0) + C_n(X_n - \psi_\theta(X_{n-1}, 0)) + D_n W_n,$$

with

$$W_n \sim \mathcal{N}(0, I), \quad C_n = \nabla_x \phi_\theta(\psi_\theta(X_{n-1}, 0), 0), \quad D_n = \nabla_W \phi_\theta(\psi_\theta(X_{n-1}, 0), 0).$$

Note $B_n, C_n, D_n$ are all functions of $X_{n-1}$. In contrast to the EKF here we want to maintain the dependence on $X_{n-1}$, because the proposal will need to move particle $i$ with a transition density that depends on the value of particle $X_n^i$. Straightforward calculations can give the desired expressions for the covariance and mean of $q_\theta(x_n|y_n, x_{n-1})$:

$$\mathcal{S}_n^{-1} = \left(B_n B_n^T\right)^{-1} + C_n^T \left(D_n D_n^T\right)^{-1} C_n$$

and

$$\mathfrak{m}_n(x_{n-1}) = \mathcal{S}_n \left( \left(B_n B_n^T\right)^{-1} \psi_\theta(X_{n-1}, 0) + C_n^T \left(D_n D_n^T\right)^{-1} (y_n - \phi_\theta(\psi_\theta(X_{n-1}, 0), 0) + C_n \psi_\theta(X_{n-1}, 0)) \right).$$

Alternatively one could use similar calculations to obtain a different Gaussian approximation based on the UKF, but these are omitted here.

Suppose we have set the proposal as described above, the weight will be given by (50) and one may run the SIS filter of Algorithm (7). How does one monitor the performance of the algorithm (and the chosen proposal)? We mentioned earlier that the variance of the weights is a measure of the variance of $\hat{Z}_n$, but it might be hard to interpret a given number for the variance of weights. To achieve interpretability a common good practice is to rescale this number to get the time varying ESS below

$$ESS_n = \frac{1}{\left(\sum_{i=1}^N (W_n^i)^2\right)}.$$

This is the same ESS as in IS in Section 2.3, which was designed to approximate the number of particles perfect Monte Carlo would require to result to the same Monte Carlo variance for $\hat{I}_n$. $ESS_n$ will take values in $[1, N]$ and a perfect sampler would achieve the maximum possible value. It can be shown that $\max ESS_n = N$ when $W_n^i = N^{-1}$ (so weights have zero variance). In the poor performance case when there exists $i$ such that $W_n^i \approx 1$ , and for $j \neq i$, $W_n^j \approx 0$ (so one particle occupies almost all the mass of the particle approximation), then we have that $ESS_n \approx 1$. As a rule of thumb, the higher the ESS the better our approximation, and this gives an indication of how efficient a particle approximation is. As you would expect similar to the construction of the proposal the ESS will be a useful tool for the standard particle filter (presented later) and more advanced implementations.

## 4.3 Discussion on SIS

### 4.3.1 A Numerical Example

We begin by showing some numerical results for a scalar linear Gaussian model;

$$X_n = \rho X_{n-1} + \tau V_n, \quad Y_n = X_n + \sigma W_n,$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, $X_0 \sim \mathcal{N}(0,4)$, $\rho = 0.6, \tau = 1, \sigma = \sqrt{2}$ and $n \leq T = 100$, $N = 1000$. The top left panel of Figure 2 shows a simulated data-set of observations $y_{0:T}$ and the top right one shows the true filter mean

$\mathbb{E}[X_n|Y_{0:n}]$ (for $n = 1, \ldots, T$) computed by the KF (in orange) together with the true simulated state sequence $x^*_{0:T}$ (in light blue) that was used to generate $y_{0:T}$. The bottom panel of Figure 2 shows the results for SIS when the bootstrap (i.e. $q_n = f_\theta$) and optimal proposals are used to estimate the filter mean $\mu_n = \mathbb{E}[X_n|Y_{0:n}]$ and variance $v_n = \mathbb{E}[(X_n - \mathbb{E}[X_n|Y_{0:n}])^2 |Y_{0:n}]$. Whereas for $\mu_n$ the results are fairly satisfactory (at least for the optimal proposal) the results for $v_n$ are quite poor compared to the true values obtained from the KF.

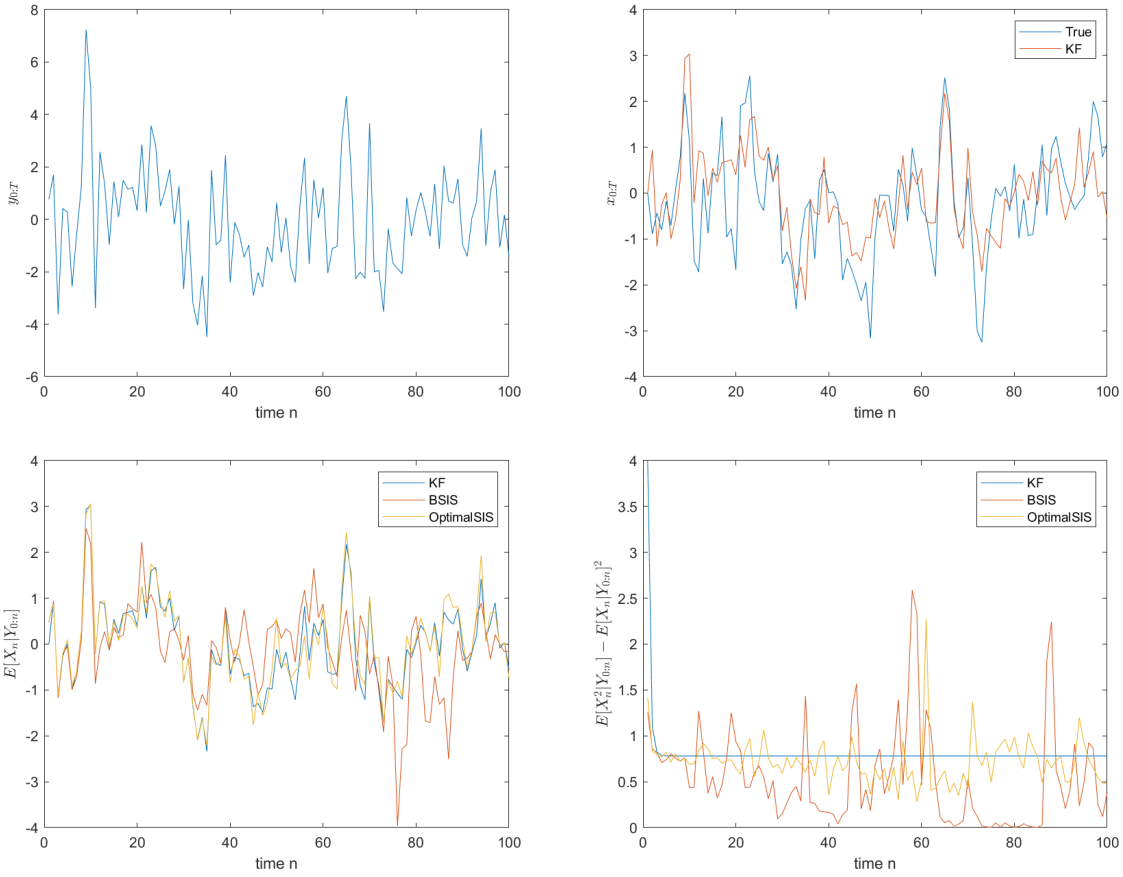


Figure 2: Top left: Generated observations $y_{0:T}$ . Top right: real state trajectory $x^*_n$ (blue) and KF mean $\mu_n = \mathbb{E}[X_n|Y_{0:n}]$ (orange). Bottom left: SIS estimates for the filter mean $\mathbb{E}[X_n|Y_{0:n}]$ at each $n = 1, \ldots, T$ for $q_n = f_\theta$ (BSIS - orange) and $q_n = q_n^{opt}$ (optimal - yellow); blue shows KF mean $\mu_n$. Bottom right: SIS estimates and KF computed filter variance $\mathbb{E}[(X_n - \mathbb{E}[X_n|Y_{0:n}])^2 |Y_{0:n}]$.

Figure 3 shows the results when SIS is used to estimate $p(y_{0:n})$ (top left panel). Both optimal and boostrap SIS filter provide reasonable answers compared to KF, but there is a large error that grows in time. The top right panel of Figure 3 shows the ESS, which in both cases exhibits an exponential decline with time. Finally, bottom panels of Figure 3 plot the trajectory of every particle of both the bootstrap and optimal proposals. A darker shading is used to encode higher values of weights. What seems to be happening is that the mean $\mu_n$ here is estimated by almost averaging from 3-4 strong particles for the bootstrap and 5-10 for the optimal proposal, which visually seem to concentrate well around the true mean indicated by the KF. Hence it is not surprising that with such a low number of effective particles (and hence low ESS) one cannot capture the variance of the filter accurately, as shown earlier in Figure 2. At the same time the support of all particles from the optimal proposal seems to have borders with a similar shape than $\mu_n$ as it uses information from $y_{0:n}$.

### 4.3.2 Discussion

The numerical results shown so far show the serious limitations of SIS when used for filtering, even in this most favourable example. Even when the optimal proposal is tractable one fails to capture the mean with high level of accuracy and cannot track the posterior variance. On the bright side the particles seem to explore adequately the state space, but this is not used efficiently due to the *weight degeneracy*. When the weight of a particle

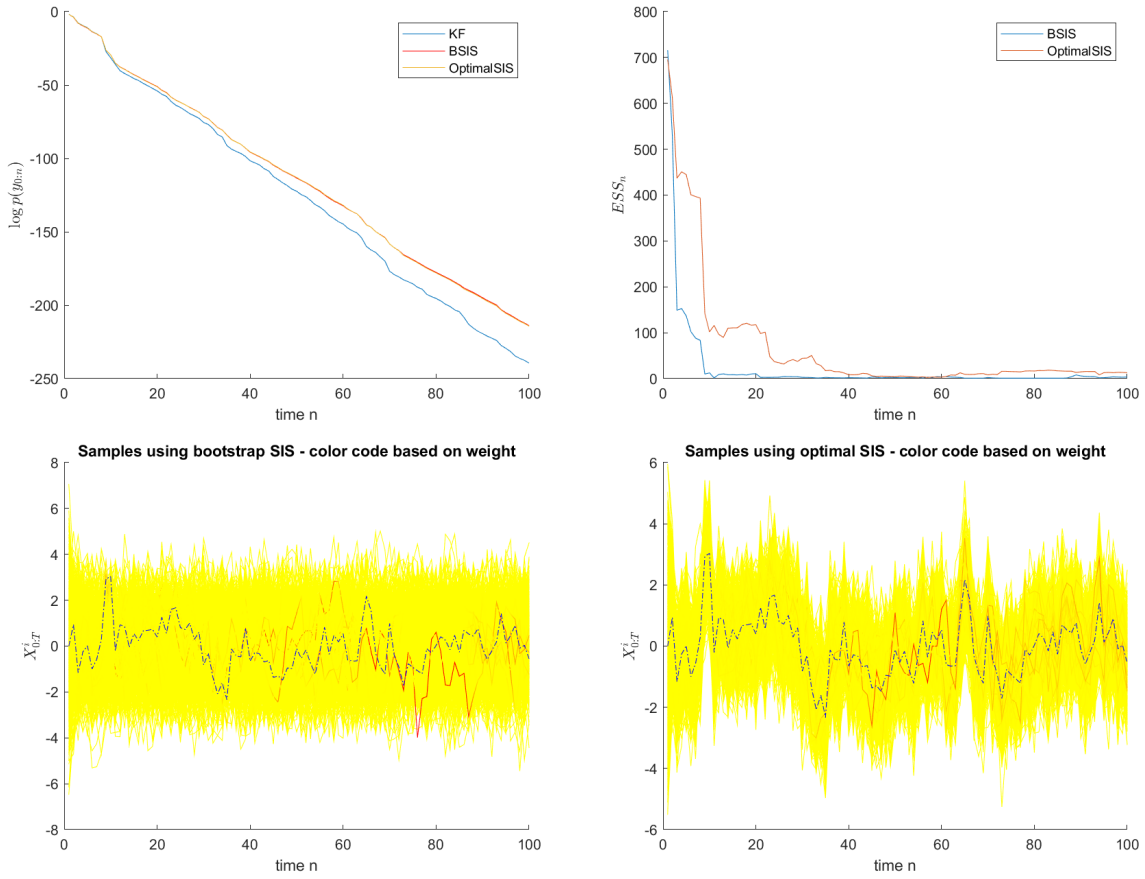


Figure 3: Top left: SIS and KF estimates the marginal likelihood $p(y_{0:n})$ (colour coding and proposals for SIS as in Figure 2 bottom plots). Top right: $ESS_n$ against $n$ for each SIS proposal. Bottom Left and Right: We plot the full simulated trajectories of the bootstrap and optimal proposal (respectively), $X^i_{0:T}$, $(i = 1, \ldots, N)$; darker and bolder yellow/orange lines indicate higher weights and dotted line is KF mean $\mu_n$ for comparison.

becomes low then there is an extremely low chance to increase in the future. As a result, the posterior mass in the particle approximations concentrates on few particles (here it seems at best 10 out of 1000) and the weight variance eventually explodes (and $ESS_n$ goes to 1). We conclude with a positive note: a well designed proposal is crucial and can make a difference in performance. Here the optimal proposal results seem more acceptable at least compared to the bootstrap ones. The necessity of a good importance proposal holds in general for IS based methods and is something that will be also useful in the particle filters we will present later on.

### 4.3.3 Reading List

For the topics in this section, you could find more material for further study in:

- Sarkka [68]: Chapter 7.0-7.3 for SIS
- Douc et. al. [27]: Section 10.2
- The paper in [29] is an excellent tutorial for the material so far and is available on-line in two versions:
  a) the actual paper
  `http://www.stats.ox.ac.uk/~doucet/doucet_godsill_andrieu_sequentialmontecarloforbayesfiltering.pdf` (material so far is covered up to Section III before Resampling in p. 201.) and
  b) the slightly more comprehensive preprint
  `http://www.cs.ubc.ca/~arnaud/doucet_tr310_sequentialmontecarlofiltering.pdf` (material so far is covered up to p. 12 before Resampling.)

### 4.3.4 Homework

For the following scalar model

$$X_n = \rho X_{n-1} + \sigma V_n, \quad Y_n = \beta \exp(\frac{X_n}{2}) W_n,$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, $X_0 \sim \mathcal{N}\left(0, \frac{\sigma^2}{1-\rho^2}\right)$.

1. Run the model to synthesise a data-set with $y_{0:T}$ for $T = 200$, $\rho = 0.91, \sigma = 1, \beta = 0.5$. Store the real state trajectory $x^*_{0:T}$ for future comparisons.
2. Implement the EKF and compare $\mu_{n|n}, \mu_{n|n-1}$ with $x^*_{0:T}$.
3. Implement a SIS filter and compare with EKF. Use $N = 50, 500, 2000$.
4. (*) Using a few multiple runs (50-500) compute an approximation the Monte Carlo variance of the estimator for the first and second moment of the filter as a function of time when computed with SIS.
5. Repeat some of the above steps for different data-sets or different parameter values.

## 4.4 Sequential Importance Resampling (SIR)

We identified an issue related to the low weights $W^i_n$ for some or many $i$-s not contributing much on the estimation procedure, especially for large $n$. A natural thing to do is to attempt to remove weak particles with low values of $W^i_n$ and replace them with stronger particles. This is reminiscent of some genetic algorithms, where only strong particles are allowed to generate new particles in the future and the computations focus only on the most promising paths. In the context of filtering it was first introduced in [36] (and coincidentally the first and last authors were members of our Statistics Section). In [36] the authors proposed to use $f_\theta$ as proposal $q_n$ and then considered sampling from the weighted population or approximation in (51) in a manner inspired by bootstrap techniques and hence coined the term bootstrap filter. This is what we refer today as resampling, which is a probabilistic selection method that only keeps particles with high weights $W^i_n$ and allows them to generate future samples/particles. Each particle is copied a number of times on average proportional to $NW^i_n$ and after resampling every particle is assigned a weight of $\frac{1}{N}$. Hence the weights can be stabilised and their exponential decay in time is avoided.

This approach was later studied in more detail and extended in many works, see [48, 56, 29, 11] for some early papers and [28] for an early edited volume with the state of the art around the late 90-s. In Algorithm 8 we present an algorithm that uses SIS and resampling, which often is referred as Sequential Importance Resampling (SIR). This

**Algorithm 8** Sequential Importance Resampling (SIR) for filtering

At time $n = 0$

1. Sample $X_0^i \sim q_\theta(x_0|\, y_0)$.
2. Compute the weights $\omega_0\left(X_0^i\right)$ and set $W_0^i \propto \omega_0\left(X_0^i\right)$, $\sum_{i=1}^N W_0^i = 1$.
3. Resample $\left\{W_0^i, X_0^i\right\}$ to obtain $N$ equally-weighted particles $\left\{\frac{1}{N}, \overline{X}_0^i\right\}$.

At time $n \geq 1$

1. Sample $X_n^i \sim q_\theta(x_n|\, y_n, \overline{X}_{n-1}^i)$ and set $X_{0:n}^i \leftarrow \left(\overline{X}_{0:n-1}^i, X_n^i\right)$.
2. Compute the weights $\omega_n\left(X_{n-1:n}^i\right)$ and set $W_n^i \propto \omega_n\left(X_{n-1:n}^i\right)$, $\sum_{i=1}^N W_n^i = 1$.
3. Resample $\left\{W_n^i, X_{0:n}^i\right\}$ to obtain $N$ new equally-weighted particles $\left\{\frac{1}{N}, \overline{X}_{0:n}^i\right\}$.

is a basic particle filtering or SMC algorithm. At time $n$, the approximations of $p_\theta\left(x_{0:n}|\, y_{0:n}\right)$ and $p_\theta\left(y_n|\, y_{0:n-1}\right)$ after the sampling step are

$$\widehat{p}_\theta\left(dx_{0:n}|\, y_{0:n}\right) = \sum_{i=1}^N W_n^i \delta_{X_{0:n}^i}\left(dx_{0:n}\right), \tag{57}$$

$$\widehat{p}_\theta\left(y_n|\, y_{0:n-1}\right) = \frac{1}{N}\sum_{i=1}^N \omega_n\left(X_{n-1:n}^i\right). \tag{58}$$

Hence an estimate of the marginal likelihood is given by

$$\widehat{p}_\theta\left(y_{0:n}\right) = \widehat{p}_\theta\left(y_0\right)\prod_{k=1}^n \widehat{p}_\theta\left(y_k|\, y_{0:k-1}\right). \tag{59}$$

After the resampling step, an alternative approximation of $p_\theta\left(x_{0:n}|\, y_{0:n}\right)$ is

$$\overline{p}_\theta\left(dx_{0:n}|\, y_{0:n}\right) = \frac{1}{N}\sum_{i=1}^N \delta_{\overline{X}_{0:n}^i}\left(dx_{0:n}\right). \tag{60}$$

### 4.4.1 The Resampling step

We will concentrate on the resampling step presented in steps 3 in Algorithm 8. In time $n$, after step 2 we have available an approximation like in (51) or (57). This will be a weighted sample $\left\{X_{0:n}^i, W_n^i\right\}_{i=1}^N$. Consider the problem of producing an equally weighted sample set of particles $\left\{\bar{X}_{0:n}^i, N^{-1}\right\}_{i=1}^N$ from the weighted sample $\left\{X_{0:n}^i, W_n^i\right\}_{i=1}^N$. This sampling (or sub-sampling) step is often of more interest than producing approximations like (57), as one may just wish to obtain samples from $\Pi_n$. So one needs to design a procedure that takes as an input $\widehat{p}_\theta\left(dx_{0:n}|\, y_{0:n}\right)$ in (57) and outputs another empirical approximation $\overline{p}_\theta\left(dx_{0:n}|\, y_{0:n}\right)$ shown in (60). The latter can be used to construct estimators of $\Pi_n(\varphi)$ as follows:

$$\begin{aligned}\bar{I}_n &= \int \varphi(x_{0:n})\bar{p}_\theta\left(dx_{0:n}|\, y_{0:n}\right)\\ &= \tfrac{1}{N}\sum_{i=1}^N \varphi\left(\bar{X}_{0:n}^i\right).\end{aligned}$$

Given, $\sum_{i=1}^N W_n^i = 1$ we can interpret $W_n^i$ as probability of selection, so one could assign each $\bar{X}_{0:n}^i$ in $\left\{\bar{X}_{0:n}^i, N^{-1}\right\}_{i=1}^N$, by copying from a suitable ancestor, $a_n(i)$, from $\left\{X_{0:n}^j\right\}_{j=1}^N$ according to

$$\bar{X}_{0:n}^i = X_{0:n}^{a_n(i)}.$$

**Algorithm 9** Multinomial resampling

1. Sample offsprings:
$$(o_n(1), \ldots, o_n(N)) \sim \mathcal{M}ultinomial(N; W_n^1, \ldots, W_n^N)$$
2. Copy particles: set $k = 0$; For $i = 1 : N$
   - IF $o_n(i) > 0$: For $j = 1 : o_n(i)$,
     - $\bar{X}_{0:n}^k = X_{0:n}^i$;
     - $k \leftarrow k + 1$

Note here $a_n(i)$ is a random variable such that:

$$\mathbb{P}\left[a_n(i) = j\right] = W_n^j.$$

That is for each $i = 1, \ldots, N$, we select the ancestor of $i$, $a_n(i)$, based on $\left\{W_n^j\right\}_{j=1}^N$ and then copy to get $\bar{X}_{0:n}^i = X_{0:n}^{a_n(i)}$. This is a simple random sampling with replacement scheme and in this example one could allow for instance $a_n(1), \ldots, a_n(N) \sim \mathcal{C}at(W_n^1, \ldots, W_n^N)$ (i.i.d sampling from a categorical distribution).

The procedure we just described comes under the name multinomial resampling, shown in Algorithm 9. There we interpret the resampling step as each $X_{0:n}^j$ in the weighted approximations producing a random number of offsprings. This is actually equivalently to trying to find ancestors for $\bar{X}_{0:n}$ and seems a more natural viewpoint from an algorithm design perspective. In this case, the offsprings will then follow a multinomial distribution. Let $o_n(i)$ denote number of offsprings of particle $i$. Then $(o_n(1), \ldots, o_n(N)) \sim \mathcal{M}ultinomial(N; W_n^1, \ldots, W_n^N)$ and one could construct the following particle approximation:

$$\begin{aligned} \overline{p}_\theta\left(dx_{0:n} \middle| y_{0:n}\right) &= \frac{1}{N}\sum_{i=1}^N o_n(i)\delta_{X_{0:n}^i}\left(dx_{0:n}\right). \\ &= \frac{1}{N}\sum_{i=1}^N \delta_{\overline{X}_{0:n}^i}\left(dx_{0:n}\right) \end{aligned}$$

Multinomial sampling is unbiased as $\mathbb{E}\left(o_n(i)\right) = NW_n^i$ and also ensures that we always have $N$ particles as $\sum_{j=1}^N o_n(j) = N$. On the other hand, subsampling of $\widehat{p}_\theta\left(dx_{0:n} \middle| y_{0:n}\right)$ to get $\bar{p}_\theta\left(dx_{0:n} \middle| y_{0:n}\right)$ adds some noise to the particle approximations, so one could assess a resampling scheme based on the variance of $o_n(i)$. From this perspective better performing resampling schemes exist such as residual or systematic resampling.

We would like to perform better in terms of the variance of estimators using resampled particles and at the same time we wish to preserve unbiasedness of $o_n(i)$ and the accuracy in estimates. There are quite a few methods that have been proposed in the literature, see [25] for a comparison and more detailed discussion. Here we will present briefly only two different methods: systematic resampling shown in Algorithm 10 and residual resampling shown in Algorithm 11. Systematic resampling is a common method used in discrete probability sampling and is quite convenient in terms of computational cost of implementation and accuracy. The downside is that its analysis is more complicated. Residual resampling can be viewed as a refinement of multinomial resampling, so that each particle $i$ is guaranteed to produce at least $\left\lfloor NW_n^i \right\rfloor$ offsprings, i.e. $o_n(i) \geq \left\lfloor NW_n^i \right\rfloor$ with $c = \lfloor x \rfloor$ being the highest integer such that $c \leq x$. This achieved by assigning first $\left\lfloor NW_n^i \right\rfloor$ offsprings and then calculating a residual weight, which is used in a multinomial distribution to assign more offsprings to each particle. In this way there is less variance in $o_n(i)$-s as one eliminates "unlucky" instances where $o_n(i) < \left\lfloor NW_n^i \right\rfloor$.

*Remark* 3. Multinomial resampling as presented in Algorithm 9 has a cost is proportional to $N \log N$, but there are better implementations with lower cost proportional to $N$. Systematic resampling has a cost proportional to $N$.

## 4.5 Discussion on SIR

### 4.5.1 Numerical Example

We will now apply the SIR algorithm with multinomial resampling and $N = 1000$ on the scalar linear Gaussian example of Section 4.3.1. In Figure 4 we see an massive improvement to the ones seen in Figure 2 both for the

**Algorithm 10** Systematic resampling

- Sample

$$U_1 \sim Uniform[0, \frac{1}{N}),$$

$$o_n(1) = \left\{ k : \sum_{l=1}^{k-1} W_n^l \leq U_1 \leq \sum_{l=1}^{k} W_n^l \right\}$$

- For $k = 2 : N$,

$$U_k = U_1 + \frac{k-1}{N},$$

$$o_n(k) = \left\{ j : \sum_{l=1}^{j-1} W_n^l \leq U_k \leq \sum_{l=1}^{j} W_n^l \right\}$$

- Copy particles: execute step 2 of Algorithm 9

**Algorithm 11** Residual resampling

- Let $\tilde{N}_n^i = \lfloor NW_n^i \rfloor$ and $\tilde{N}_n = \sum_{i=1}^{N} \lfloor NW_n^i \rfloor$
- Set $\bar{W}_n^i = W_n^i - \frac{\tilde{N}_n^i}{N}$ and normalise to get

$$\tilde{W}_n^i = \frac{\bar{W}_n^i}{\sum_{j=1}^{N} \bar{W}_n^j}$$

- Sample

$$(\tilde{o}_n(1), \ldots, \tilde{o}_n(N)) \sim \mathcal{M}ultinomial(N - \tilde{N}_n; \tilde{W}_n^1, \ldots, \tilde{W}_n^N)$$

- For $k = 1 : N$, compute $o_n(k) = \tilde{o}_n(k) + \tilde{N}_n^k$
- Copy particles: execute step 2 of Algorithm 9

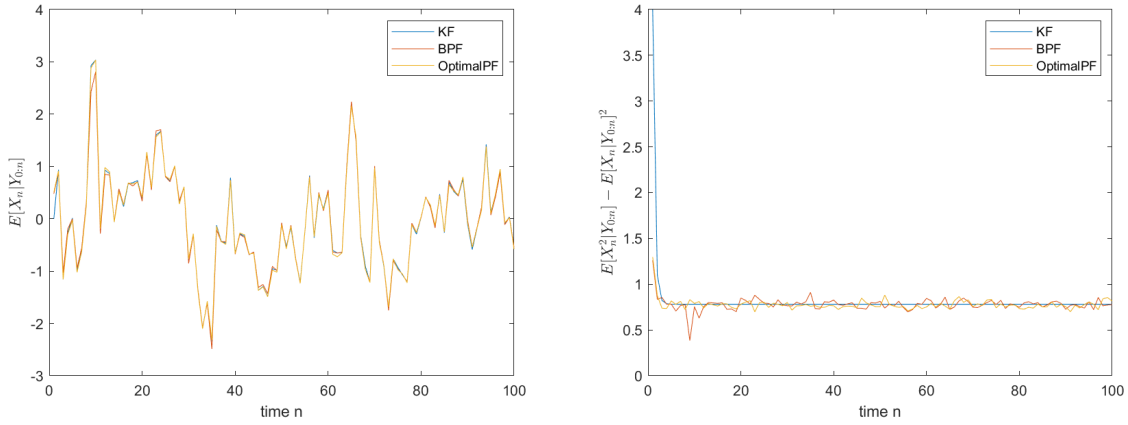


Figure 4: Left: SIR estimates for the filter mean $\mathbb{E}[X_n|Y_{0:n}]$ at each $n = 1, \ldots, T$ for $q_n = f_\theta$ (prior - red) and $q_n = q_n^{opt}$ (optimal - green); blue shows KF mean $\mu_n$. Right: SIR estimates and KF computed filter variance $\mathbb{E}[(X_n - \mathbb{E}[X_n|Y_{0:n}])^2 |Y_{0:n}]$.

filter mean and variance. The errors are smaller when the optimal proposal is used. Here this is apparent only for estimation of the filter variance, but one could check it is true for the mean by plotting errors separately.

Figure 5 shows the results when SIR is used to estimate $p(y_{0:n})$, where both proposal distributions seem to result to very good performance. The top right panel of Figure 5 shows $ESS_n$ versus $n$ which is clearly better for the optimal proposal. For simplicity in the bottom panel we consider only the case of the bootstrap PF. The bottom left panel of Figure 5 show $\bar{X}_n^i$ for each $i$ when generated at time $n$, which is used to provide the approximation for $\pi_n$. This can be interpreted as a random grid generated by the algorithm at each $n$, which provides the particles constructing the random probability measure $\overline{p}_\theta(dx_n|\, y_{0:n})$.

Finally, the bottom right panel of Figure 5 final simulated trajectories $\bar{X}_{0:T}^i$ that compose the approximation $\overline{p}_\theta(dx_{0:T}|\, y_{0:T})$. The plot is quite different that the one in the bottom left panel of Figure 3 and shows a tree that has many nodes near $T$, but whose diversity is gradually depleted going backwards for times $k < 70$. This phenomenon is called *path degeneracy* and has been generated by the successive resampling/selection steps applied to earlier parts of the path space. At each time $n$, one can imagine a tree like the bottom right plot of Figure 5 evolving forward. Taking a snapshot at time $n$, the endpoints of this tree are the outputs $\bar{X}_n^i$ of Algorithm 8 (shown as dots at time $n$ in bottom left plot of Figure 5). The resampling operation at time $n$ will affect all times $k \leq n$. As the tree moves forward in time, the random grid of time $k$ (shown in the bottom left plot of Figure 5) will have been subject to $n - k + 1$ resampling operations, so one should expect only a portion of the originally resampled points $\bar{X}_k^i$ (at time $k$) to survive in the approximation of $\bar{X}_{0:n}^i$ (at time $n$).

### 4.5.2 Discussion

The massive improvement of SIR in terms of performance over SIS is very clear. This appears when integral of interest is of the form

$$I_n = \int \varphi(x_n) p_\theta(x_{0:n}|\, y_{0:n})\, dx_{0:n}$$

This is because of an interesting property of SIR and particle filtering in general, which we will later attribute to the exponential forgetting property of $\pi_n$ and the HMM in question. At time $T$, while the approximation of $\pi_T$ can be very good, but in the approximation of $\Pi_T(x_{0:T})$ one relies on few or a single particle for $n \ll T$ and $p_\theta(x_{0:n}|\, y_{0:T})$ will eventually be approximated by a single unique particle as $T - n$ increases. This is a fundamental weakness of particle filtering approximations referred to as *path degeneracy*. For many tracking applications it not a major issue, but we will see later that it is a weakness that needs to be overcome when performing parameter estimation and smoothing. In the latter case, the diversity in the path space for all $k \leq n$ is crucial and the fact that the number of unique particles decreases as we go backwards in time is an issue. The bottom right panel in Figure 5 suggest that if $\int \varphi(x_{0:n}) p_\theta(x_{0:n}|\, y_{0:n})\, dx_{0:n}$ depends on the full path or a big part of it we should have a poor approximation when $N \ll n$. Given a fixed number of particles $N$, it is impossible to approximate $p_\theta(x_{0:n}|y_{0:n})$ "well" when $n$ is large (typically as soon as $n \approx N$). For more details we refer the interesting reader to [40] for a theoretical study

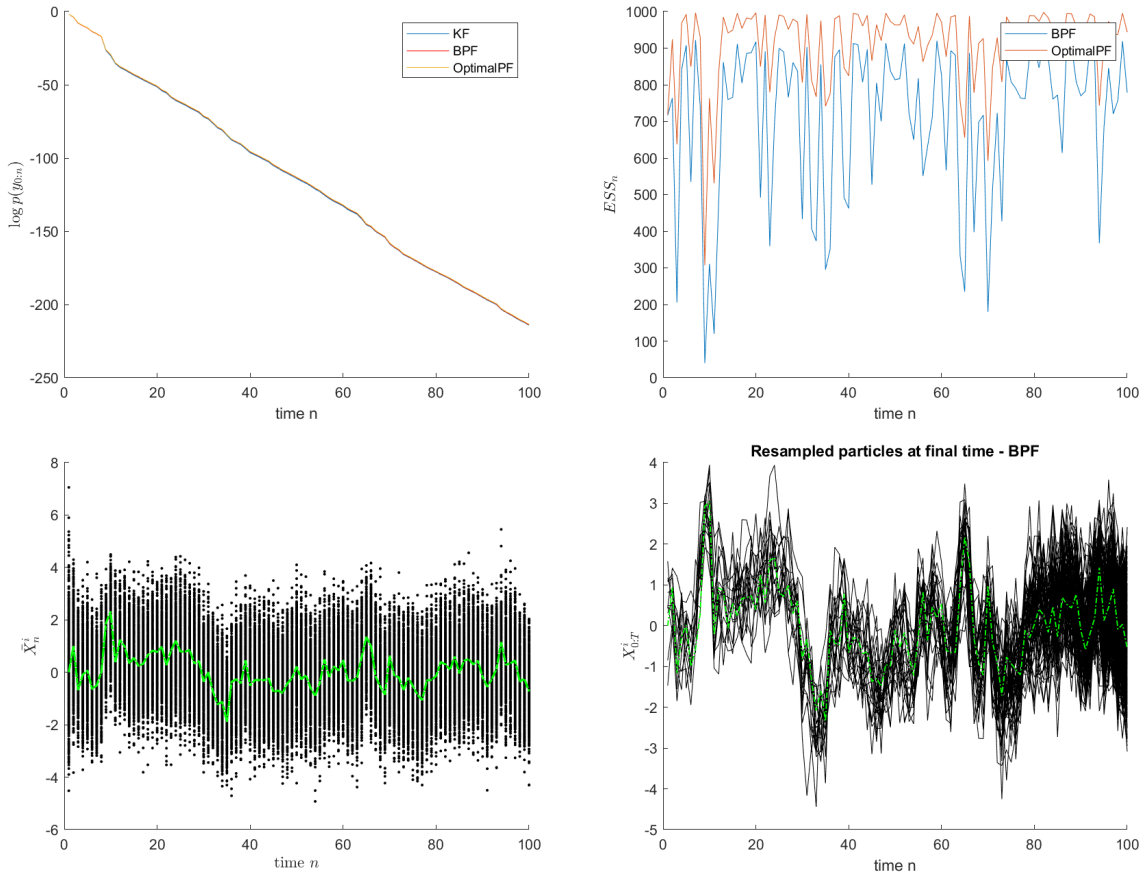


Figure 5: Top Left: SIR and KF estimates the marginal likelihood $p(y_{0:n})$ (colour coding and proposals for SIS as in Figure 4). Top Right: $ESS_n$ against $n$ for each SIR proposal. Bottom Left: we plot bootstrap PF particles $\bar{X}_n^i$ when generated at time $n$ and contrast with KF mean $\mu_n$ (in green). Bottom Right: we plot the final resampled trajectories with a bootstrap proposal, $\bar{X}_{0:T}^i$, $(i = 1, \ldots, N)$ and $\mu_n$ (in green).

showing the average backward coalescence time of the particle tree to a single path is proportional to $N \log N$ and the implications of this in terms of optimising memory management.

### 4.6 Some convergence results

We will present some basic convergence results and try to interpret them in terms of what we have seen so far. Let $\epsilon_{\theta,n}\left(dx_{0:n}\right) = \widehat{p}_\theta\left(dx_{0:n}|\, y_{0:n}\right) - p_\theta\left(dx_{0:n}|\, y_{0:n}\right)$. If $\omega_0\left(x_0\right)$ and $\omega_n\left(x_{n-1:n}\right)$ are upper bounded, then for any bounded test function $\varphi_n : \mathcal{X}^{n+1} \to \mathbb{R}$, there exists constants $C_{\theta,n,p} < \infty$ such that for any $p > 0$, fp

$$\mathbb{E}^N\left[\left|\int \varphi_n(x_{0:n})\epsilon_{\theta,n}\left(dx_{0:n}\right)\right|^p\right]^{\frac{1}{p}} \leq \frac{C_{\theta,n,p}\overline{\varphi_n}}{N^{1/2}}, \tag{61}$$

where $\overline{\varphi_n} = \sup_{x_{0:n}\in\mathcal{X}^{n+1}} |\varphi_n(x_{0:n})|$. In this $L_p$ bound the expectation $\mathbb{E}^N[\cdot]$ is taken w.r.t to the sampling distribution being the probability law of the particle filter (i.e. the joint distribution of all the simulated variables in the SIR algorithm). The $L_p$ bound in (61) is weak result because without making particular assumptions on $f_\theta, g_\theta$ one should expect in general $C_{\theta,n,p}$ to grow exponentially/ polynomially with $n$ and exponentially with $d_x$. This should not come as a surprise as the dimension of the target density $p_\theta\left(x_{0:n}|\, y_{0:n}\right)$ we are approximating is increasing with $n$. This also means that if one is interested in achieving a prescribed accuracy then it will not be possible to do this fixed $N$ as $n$ grows.

Fortunately this is not always the case. Many state-space models possess the so-called *exponential forgetting* property. For any $x_0, x_0' \in \mathcal{X}$ and observation record $y_{0:n}$,

$$\int \left|p_\theta\left(x_n|\, y_{0:n}, x_0\right) - p_\theta\left(x_n|\, y_{0:n}, x_0'\right)\right| dx_n \leq C\lambda^n, \tag{62}$$

where $\lambda \in [0,1)$ and $C$ is a constant. This means that if one computes $\pi_n$ for different initial conditions, say $\pi_n^x$ for $X_0 = x$ and $\pi_n^{x'}$ for $X_0 = x'$, then eventually $\pi_n^x$ and $\pi_n^{x'}$ will converge (as $n$ grows) and this will happen at an exponential rate. Then one can show that for a small integer $L > 0$ and any bounded test function $\varphi_L : \mathcal{X}^L \to \mathbb{R}$, there exists constants $D_{\theta,L,p} < \infty$ such that for any $p > 0$

$$\mathbb{E}^N\left[\left|\int \varphi_L(x_{n-L+1:n})\epsilon_{\theta,L}\left(dx_{n-L+1:n}\right)\right|^p\right]^{\frac{1}{p}} \leq \frac{D_{\theta,L,p}\overline{\varphi_L}}{N^{1/2}}, \tag{63}$$

where here $\epsilon_{\theta,L}\left(dx_{n-L+1:n}\right) = \int_{\mathcal{X}^{n-L+1}} \epsilon_{\theta,n}\left(dx_{0:n}\right)$ . Notice $D_{\theta,L,p}$ does not depend on $n$ and hence we have a uniform in time error bound. This means that any degree of accuracy can be achieved using a particular $N$ for any $n$. This explains why particle filters seem to work very well when estimating $p_\theta\left(x_n|y_{0:n}\right)$ or $p_\theta\left(x_{n-L:n}|y_{0:n}\right)$ for a low $L$.

Apart from the $L_p$ bound, there is a plethora of more convergence results in the literature, from mainstream ones to more exotic; see for instance the book by Del Moral [21] for a detailed analysis. We mention here briefly:

- a CLT also applies for SIR (and SMC in general) and its asymptotic variance is much lower than SIS, e.g. [16]. In fact, it can be even uniformly bounded with time under good mixing properties of the HMM [76].
- The SIR algorithm results in an unbiased estimation of the marginal likelihood

$$\mathbb{E}^N[\hat{p}_\theta\left(y_{0:T}\right)] = p_\theta\left(y_{0:T}\right) \tag{64}$$

  assuming the resampling method is unbiased, which holds for multinomial case. This is a non-trivial result, that will be extremely useful when discussing particle MCMC.
- $\widehat{p}_\theta\left(y_{0:n}\right)$ has a non-asymptotic relative variance (i.e. variance of $\frac{\widehat{p}_\theta(y_{0:n})}{p_\theta(y_{0:n})}$) that increases linearly with $n$, [13].

We conclude this section with a negative result, that motivates the use of more advanced smoothing methods. Even if (62) holds, then the asymptotic variance of the SMC estimate of an expected additive functional $\varphi_n(x_{0:n}) = \sum_{k=0}^n \varphi\left(x_k\right)$ will grow with a super-linear rate. Say we are interested in what is often refered to as a *smoothed additive functional:*

$$\mathcal{S}_n^\theta = \int \left[\sum_{k=0}^n \varphi\left(x_k\right)\right] p_\theta\left(x_{0:n}|\, y_{0:n}\right) dx_{0:n}, \tag{65}$$

and this estimated by

$$\widehat{\mathcal{S}}_n^\theta = \int \left[ \sum_{k=0}^{n} \varphi(x_k) \right] \widehat{p}_\theta(dx_{0:n} | y_{0:n}). \tag{66}$$

In [65] the authors show that

$$\mathbb{V}ar^N \left( \widehat{\mathcal{S}}_n^\theta \right) \geq D_\theta \frac{n^2}{N}, \tag{67}$$

where $D_\theta$ is a constant that does not depend on $n, N$. This motivates the use of dedicated smoothing algorithms. These will be presented below and are very important for parameter estimation of $\theta$.

### 4.6.1 Key points

Make sure you understand:

- particle filtering and its ingredients, Importance Sampling and Resampling. Try to understand how resampling addresses address weight degeneracy of SIS when $n$ is large. Also spend some time to think how the algorithms presented here can be implemented in practice. For resampling you might find routines related to sampling from discrete probability distributions useful.
- the strengths of particle filtering when estimating $p_\theta(x_n|y_{0:n})$ or $p_\theta(x_{n-L:n}|y_{0:n})$ for a low $L$ and appreciate that this is inherited from the exponential property of $\pi_n$ and the mixing in the HMM.
- the possible weaknesses of the method. There are two types of degeneracy:
  - weight degeneracy due to Importance Sampling. This is due to a mismatch between the proposal and evolved particles and the target distribution in one step of the SMC algorithm. This can be crucial when $d_x$ is high.
  - path degeneracy: due to successive resampling selections in the particle approximation. This results in the particle approximation having a diverse population only for close to final times $n$ (or $T$).

### 4.6.2 Reading List

You could read further in:

- Sarkka [68]: Chapter 7
- Doucet et. al. 1998 [29] (except sections IV, V)
- A more modern tutorial on filtering found in [30]: Sections 1, 2.1, 2.2, 2.4, 3.1-3.5, 4.1, 4.3, 6. Can be found on-line in `http://www.stats.ox.ac.uk/~doucet/doucet_johansen_tutorialPF2011.pdf`

### 4.6.3 Homework

For the following scalar model

$$X_n = \rho X_{n-1} + \sigma V_n, \quad Y_n = \beta \exp(\frac{X_n}{2}) W_n,$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, $X_0 \sim \mathcal{N}\left(0, \frac{\sigma^2}{1-\rho^2}\right)$.

1. Run the model to synthesise a data-set with $y_{0:T}$ for $T = 200$, $\rho = 0.91, \sigma = 1, \beta = 0.5$. Store the real state trajectory $x^*_{0:T}$ for future comparisons.
2. Implement the SIR with $q(x_n|y_n, x_{n-1}) = f(x_n|x_{n-1})$ and compare estimated of the mean with EKF, SIS and $x^*_{0:T}$.
3. Implement the SIR with $q(x_n|y_n, x_{n-1})$ obtained from an approximation of the posterior using EKF and compare with previous estimates.
4. (*) Using a few multiple runs (50-500) compute an approximation the Monte Carlo variance of the estimator for $p(y_{0:n})$ a function of time when computed SIR.
5. Repeat some of the above steps for different data-sets or different parameter values.

# 5 Advanced Sequential Monte Carlo

There are more elaborate particle filtering algorithms that can work better in terms of variance of estimators, ESS, accuracy than the SIR. The methods we present here will improve the performance in terms of path degeneracy significantly, but do not address the problem completely. Very often they postpone it for an adequate amount of time.

We will look first at the following algorithms

- adaptive resampling,
- the resample move PF,
- the auxiliary particle filter.

These are presented separately but can be combined together in many ways and this will result to much more powerful algorithms. We will then conclude this section with a brief introduction to SMC sampling methods for static problems.

### Notation change alert

In the previous section we used $\omega_n$ to denote the incremental weights so that we can distinguish with the SIS importance weight on the path space $w_n$. Given resampling will be used from now on we will change our notation and denote the incremental weight instead with $w_n$ as follows:

$$w_0(x_0) = \frac{\eta_\theta(x_0)\, g_\theta(y_0|\, x_0)}{q_\theta(x_0|\, y_0)}, \tag{68}$$

$$w_n(x_{n-1:n}) = \frac{f_\theta(x_n|\, x_{n-1})\, g_\theta(y_n|\, x_n)}{q_\theta(x_n|\, y_n, x_{n-1})} \text{ for } n \geq 1. \tag{69}$$

The reason behind this swap is that this notation is most common in the literature.

## 5.1 Adaptive Resampling

It is clear that although resampling stabilises the weights and is important to maintain good particle system approximations as $n$ grows, it causes degeneracy when estimating $\Pi_n$ and tends to leave early states being represented by one or few particles.. One approach around this is to use it only when necessary, that is resample only when $ESS_n \leq \alpha N$, e.g. $\alpha = 1/2$ or $2/3$. If at some $n$, $ESS_n > \alpha N$ then one can apply SIS steps. This adaptive resampling procedure is presented in Algorithm 12 for $n \geq 1$. The particle approximations for the joint filtering density $\Pi_n$ after step 2 will be

$$\widehat{p}_\theta(dx_{0:n}|\, y_{0:n}) = \sum_{i=1}^{N} W_n^i \delta_{X_{0:n}^i}(dx_{0:n})$$

and for $p_\theta(y_n|\, y_{0:n-1})$

$$\widehat{p}_\theta(y_n|\, y_{0:n-1}) = \sum_{i=1}^{N} W_{n-1}^i w_n\left(X_{n-1:n}^i\right).$$

Note that if at time $n-1$ one has performed the resampling step then $W_{n-1}^i = \frac{1}{N}$. Only when resampling is executed in step 3 of the algorithm, one can obtain an unweighted particle approximation for $\Pi_n$ as in (60). Of course if this is desired one can always resample $\widehat{p}_\theta(dx_{0:n}|\, y_{0:n})$ separately and independently from the evolution of Algorithm 12. For computing the marginal likelihood one can also keep track of the resampling times say $\tau_1, \tau_2, \ldots, \tau_l, \ldots$, compute

$$\widehat{p}_\theta\left(y_{\tau_{l-1}+1:\tau_l}\middle|\, y_{0:\tau_{l-1}}\right) = \frac{1}{N}\sum_{i=1}^{N} \prod_{m=\tau_{l-1}+1}^{\tau_l} w_m\left(X_{m-1:m}^i\right)$$

and then recursively calculate the product over to get $\widehat{p}_\theta(y_{0:\tau_n})$.

Adaptive resampling is useful as a simple change in the algorithm overall results to less randomness in the particle system. It results to better performance in terms of combining accuracy and variance of the estimates, but it is worth pointing out that the normalising constant is no longer unbiased, i.e. (64) does not hold when adaptive resampling is performed. This means that this approach is not appropriate to be used within pseudo-marginal and particle MCMC algorithms.

**Algorithm 12** Adaptive Resampling PF; for convenience we do not show initialisation for $n = 0$.

At time $n \geq 1$

1. Sample $X_n^i \sim q_\theta(x_n|\, y_n, X_{n-1}^i)$ and set $X_{0:n}^i \leftarrow \left(X_{0:n-1}^i, X_n^i\right)$.
2. Compute the weights $w_n\left(X_{n-1:n}^i\right)$ as in (69) and set $W_n^i \propto W_{n-1}^i w_n\left(X_{n-1:n}^i\right)$, $\sum_{i=1}^N W_n^i = 1$.
3. IF $ESS_n \leq \alpha N$
   - resample $\left\{W_n^i, X_{0:n}^i\right\}$ to obtain $N$ new equally-weighted particles $\left\{\frac{1}{N}, \overline{X}_{0:n}^i\right\}$.
   - set $X_{0:n}^i \leftarrow \overline{X}_{0:n}^i$, $W_n^i \leftarrow \frac{1}{N}$

## 5.2 Adding MCMC steps: the Resample Move PF

One way to view path degeneracy, is that the number of unique particles is decreasing after the resampling step (and often by a large amount) and hence we have some loss in sample diversity. One way around it would be to aim to re-introduce some of this lost diversity by moving the particles. Of course, this needs to be done carefully. For instance, if one moves the particles at time $n$ after resampling using ad-hoc noise (e.g. adding a zero mean Gaussian random variable), then this will result to having a sample whose statistical properties are are distorted (with reference to the target $\Pi_n$) and hence there will be a bias in the Monte Carlo estimates.

One way to avoid this is to move the particles using well designed MCMC moves, whose target density is

$$p_\theta\left(x_{0:n}|\, y_{0:n}\right) \propto \eta_\theta\left(x_0\right) \prod_{k=1}^{n} f_\theta\left(x_k|\, x_{k-1}\right) \prod_{k=0}^{n} g_\theta\left(y_k|\, x_k\right).$$

This was proposed in [34] under the name Resample-Move PF. The idea is to improve on path degeneracy by re-inserting lost diversity in the particles using appropriate MCMC moves on the target $\Pi_n$ (see Section 2.4 for some basics on MCMC). Note we are using the path space to define the variable of interest, $X_{0:n}$, as this is where SMC operates on and where path degeneracy manifests. Let $K_n$ be an appropriate MCMC kernel, which can be a small number of iterations from any valid $\Pi_n$-invariant MCMC algorithm, e.g. random walk Metropolis, Gibbs, Hybrid Monte Carlo etc. The Resample-Move PF is presented in Algorithm 13.

**Algorithm 13** The Resample Move PF

At time $n \geq 1$

- Sample $X_n^i \sim q_\theta(x_n|\, y_n, \tilde{X}_{n-1}^i)$ and set $X_{0:n}^i \leftarrow \left(\tilde{X}_{0:n-1}^i, X_n^i\right)$.
- Compute the weights $w_n\left(X_{n-1:n}^i\right)$ and set $W_n^i \propto w_n\left(X_{n-1:n}^i\right)$, $\sum_{i=1}^N W_n^i = 1$.
- Resample $\left\{W_n^i, X_{0:n}^i\right\}$ to obtain $N$ new equally-weighted particles $\left\{\frac{1}{N}, \overline{X}_{0:n}^i\right\}$.
- Move particles by independently (for each $i$) sampling

$$\tilde{X}_{0:n}^i \sim K_n(\cdot|\overline{X}_{0:n}^i)$$

It is important to emphasise that when using a MCMC proposal $K_n$ within SMC, we are only interested in the invariance property of $K_n$ with $\Pi_n$, as shown earlier in (5). We are using $K_n$ just to provide a jitter in the particle

population and add diversity (while preserving the statistical properties of sample), but we are not interested in the ergodicity properties of $K_n$ or to compute ergodic averages (w.r.t to the number of MCMC iterations). This means that $K_n$ needs not be ergodic and does not need to move the whole trajectory $X_{0:n}$. Moving only $X_{n-L+1:n}$ for some small lag $L$ can still lead to correct algorithm that is $\Pi_n$-invariant. An example of such $K_n$ is shown in Algorithm 14, where we are implementing Metropolis-Hastings based on a Gaussian Random Walk proposal. This is useful in practice because we do not want from a computational perspective to run at every $n$ a MCMC algorithm on the full path $X_{0:n}$.

**Algorithm 14** A Random Walk MCMC kernel that can be used for filtering.

Set $\Upsilon_{0:n} = \overline{X}^i_{0:n}$

- For $m = 1, \ldots, M$
  - Sample $H \sim \mathcal{N}(0, S)$, with $S$ of appropriate dimension
  - Propose $\mathfrak{Z}_{n-L+1:n} = \Upsilon_{n-L+1:n} + \varrho H$
  - Compute acceptance ratio

$$\alpha = 1 \wedge \frac{\prod_{k=n-L+1}^{n} f_\theta\left(\mathfrak{Z}_k \middle| \mathfrak{Z}_{k-1}\right) g_\theta\left(y_k \middle| \mathfrak{Z}_k\right)}{\prod_{k=n-L+1}^{n} f_\theta\left(\Upsilon_k \middle| \Upsilon_{k-1}\right) g_\theta\left(y_k \middle| \Upsilon_k\right)}$$

  - with probability $\alpha$:
    - accept $\Upsilon_{0:n} \leftarrow (\Upsilon_{0:n-L}, \mathfrak{Z}_{n-L+1:n})$
    - otherwise reject proposal and$\Upsilon_{0:n}$ remains the same
- Set $\tilde{X}^i_{0:n} = \Upsilon_{0:n}$.

We conclude this part with some practical details on the tuning of $K_n$ based on the presentation in Algorithm 14 (but the intuition/comments will still apply for other implementations). Here, $M$ can be quite small 1-5 for simple problems to $20-30$ for harder ones. In addition, one can use the particles to tune this algorithm. For example one can use the empirical covariance of the particles after resampling $\{\overline{X}^i_n\}$ to design $S$ and $\varrho$ can be tuned for average acceptance ratio around $0.2-0.4$.

## 5.3 The auxiliary particle filter

Resample Move and adaptive resampling are meant to improve path degeneracy by attempting to increase particle diversity, but both do not affect the way the weights are computed and their behaviour. When presenting IS, we discussed on how effective good proposals are in terms of addressing weight degeneracy. The next extension will discuss a possible way to change the weights so that they exhibit less variance and hence one ends up with better estimates. This idea was originally proposed as the auxiliary particle filter in [62], where using likelihood informed proposals and transforming the weights were employed. The main idea can be summarised as using a different target sequence of distributions than $\{p_\theta\left(x_{0:n} \middle| y_{0:n}\right)\}_{n\geq 0}$ to propagate the particles and then post-process the particles to derive particle approximations for $\{p_\theta\left(x_{0:n} \middle| y_{0:n}\right)\}_{n\geq 0}$.

Recall the Bayesian recursion for $p_\theta\left(x_{0:n} \middle| y_{0:n}\right)$ in (20)-(22):

$$p_\theta\left(x_{0:n} \middle| y_{0:n}\right) = \frac{1}{Z_n} p_\theta\left(x_{0:n-1} \middle| y_{0:n-1}\right) f_\theta\left(x_n \middle| x_{n-1}\right) g_\theta\left(y_n \middle| x_n\right)$$

with $Z_n = p_\theta\left(y_n \middle| y_{0:n-1}\right)$. When using a bootstrap PF we move/mutate the particles every time with $f_\theta\left(x_n \middle| x_{n-1}\right)$ and weight with $g_\theta\left(y_n \middle| x_n\right)$. An alternative route would be to use the optimal proposal seen in Section 4.2, so given approximations for $p_\theta\left(x_{0:n-1} \middle| y_{0:n-1}\right)$ we can first weight with $p_\theta\left(y_n \middle| x_{n-1}\right)$ and then move with $p_\theta\left(x_n \middle| x_{n-1}, y_n\right)$. Note we have reversed here the usual steps and weight first as the weight will depend only on $x_{n-1}$ and recall that

$$p_\theta\left(x_n \middle| x_{n-1}, y_n\right) = \frac{f_\theta\left(x_n \middle| x_{n-1}\right) g_\theta\left(y_n \middle| x_n\right)}{p_\theta\left(y_n \middle| x_{n-1}\right)}. \tag{70}$$

Given it is possible to reverse the weight and sampling steps, one has some flexibility on when to resample too. One can either:

1. weight with $p_\theta(y_n|x_{n-1})$, move $p_\theta(x_n|x_{n-1},y_n)$ and then resample. This is equivalent to the standard PF in Algorithm 8 targeting $\{p_\theta(x_{0:n}|y_{0:n})\}_{n\geq 0}$ with the optimal importance proposal.

2. move $p_\theta(x_n|x_{n-1},y_n)$, weight with $p_\theta(y_{n+1}|x_n)$ and then resample. Moving the resampling step later means that we are actually using an implementation of Algorithm 8 targeting $\{p_\theta(x_{0:n}|y_{0:n+1})\}_{n\geq 0}$.

Note we have

$$p_\theta(x_{0:n}|y_{0:n+1}) \propto \eta_\theta(x_0)\prod_{k=1}^{n} f_\theta(x_k|x_{k-1})\left(\prod_{k=0}^{n} g_\theta(y_k|x_k)\right) p_\theta(y_{n+1}|x_n). \tag{71}$$

When this target distribution is targeted using a IS proposal composed of $\prod_{k=0}^{n} p_\theta(x_k|x_{k-1},y_k)$, the resulting weight will be $p_\theta(y_{n+1}|x_n)\prod_{k=0}^{n} p_\theta(y_k|x_{k-1})$; you can check this using (70). Option 2 above uses an incremental weight of $p_\theta(y_{n+1}|x_n)$ based on the ordering implied by the resampling step. Why is this useful? The answer is that targeting $\{p_\theta(x_{0:n}|y_{0:n+1})\}_{n\geq 0}$ uses a one step lookahead in the weights, so implicitly incorporates information on the next observation when resampling the particles (and informative observations are taken into account earlier). Then, one can still use another weight as in (69) to compute particle approximations for $\{p_\theta(x_{0:n}|y_{0:n})\}_{n\geq 0}$ separately.

While this is quite elegant, in general $p_\theta(x_n|x_{n-1},y_n)$ and $p_\theta(y_n|x_{n-1})$ are intractable, so one needs to resort to some form approximations. Lets say we obtain approximations $q_\theta(x_n|x_{n-1},y_n)$ and $q_\theta(y_{n+1},x_n)$, where $q_\theta(x_n|x_{n-1},y_n)$ is a good importance distribution. Note here $q_\theta(y_{n+1},x_n)$ is not necessarily required to be a pdf, but just an easy to evaluate non-negative function defined on $(x_n,y_{n+1})$ that takes into account the current observation (it is sometimes called a score-function for the weights.) Instead of (71) consider the target:

$$\tilde{\pi}_n(x_{0:n}|y_{0:n}) \propto p_\theta(x_{0:n}|y_{0:n})\, q_\theta(y_{n+1},x_n). \tag{72}$$

One may re-write

$$\begin{aligned}\tilde{\pi}_n(x_{0:n}|y_{0:n+1}) &\propto \eta_\theta(x_0)\, g_\theta(y_0|x_0)\, q_\theta(y_1,x_0)\\ &\times \prod_{k=1}^{n} f_\theta(x_k|x_{k-1})\, g_\theta(y_k|x_k)\,\frac{q_\theta(y_{k+1},x_k)}{q_\theta(y_k,x_{k-1})}\end{aligned} \tag{73}$$

and note that by construction the $q_\theta$-s cancel out:

$$q_\theta(y_1,x_0)\prod_{k=0}^{n}\frac{q_\theta(y_{k+1},x_k)}{q_\theta(y_k,x_{k-1})} = q_\theta(y_{n+1},x_n). \tag{74}$$

Similarly one can write $\tilde{\pi}_n$ recursively as:

$$\tilde{\pi}_n(x_{0:n}|y_{0:n+1}) \propto \tilde{\pi}_{n-1}(x_{0:n-1}|y_{0:n})\, f_\theta(x_n|x_{n-1})\left(g_\theta(y_n|x_k)\,\frac{q_\theta(y_{n+1},x_n)}{q_\theta(y_n,x_{n-1})}\right)$$

The Auxiliary PF (APF) is an implementation of Algorithm 8 targeting $\{\tilde{\pi}_n(x_{0:n}|y_{0:n+1})\}_{n\geq 0}$ using $q_\theta(x_n|y_n,x_{n-1})$ as a proposal. This leads to the following weights:

$$\tilde{w}_n(x_n,x_{n-1}) = \frac{f_\theta(x_k|x_{k-1})\, g_\theta(y_k|x_k)\, q_\theta(y_{n+1},x_n)}{q_\theta(y_n,x_{n-1})\, q_\theta(x_n|y_n,x_{n-1})}$$

We will implement this PF and then reweight to get approximations for original $\Pi_n$ that is actually of interest. This can be done by using the following set of weights

$$\begin{aligned} w_0(x_0) &= \frac{g_\theta(y_0|x_0)\,\eta_\theta(x_0)}{q_\theta(x_0|y_0)},\\ w_n(x_{n-1:n}) &= \frac{g_\theta(y_n|x_n)\, f_\theta(x_n|x_{n-1})}{q_\theta(x_n,y_n|x_{n-1})} \text{ for } n\geq 1\end{aligned}$$

where we denote for $n\geq 1$,

$$q_\theta(x_n,y_n|x_{n-1}) = q_\theta(x_n|y_n,x_{n-1})\, q_\theta(y_n,x_{n-1}).$$

These weights are consistent with (69) and (72) explains why we have removed the term $q_\theta(y_{n+1}, x_n)$ compared to $\tilde{w}_n(x_n, x_{n-1})$.

The Auxiliary PF (APF) is presented in Algorithm 15. For convenience we write the evaluation of the weight $\tilde{w}_n$ as a function of $w_n$, which needs to be computed anyway to approximate $\{p_\theta(x_{0:n}|y_{0:n})\}_{n\geq 0}$. Not surprisingly based on the discussion above, [62] recommends using if available $q_\theta(x_n|y_n, x_{n-1}) = p_\theta(x_n|y_n, x_{n-1})$ and $q_\theta(y_n, x_{n-1}) = p_\theta(y_n|x_{n-1})$ or approximations of them.

**Algorithm 15** The Auxiliary PF (APF)

At time $n = 0$ , for all $i \in \{1, ..., N\}$:

1. Sample $X_0^i \sim q_\theta(x_0|y_0)$.
2. Compute $\overline{W}_1^i \propto w_0\left(X_0^i\right) q_\theta\left(y_1, X_0^i\right)$, $\sum_{i=1}^N \overline{W}_1^i = 1$.
3. Resample $\overline{X}_0^i \sim \sum_{i=1}^N \overline{W}_1^i \delta_{X_0^i}(dx_0)$.

At time $n \geq 1$ , for all $i \in \{1, ..., N\}$:

1. Sample $X_n^i \sim q_\theta(x_n|y_n, \overline{X}_{n-1}^i)$ and set $X_{0:n}^i \leftarrow \left(\overline{X}_{0:n-1}^i, X_n^i\right)$.
2. Compute $\overline{W}_{n+1}^i \propto w_n\left(X_{n-1:n}^i\right) q_\theta\left(y_{n+1}, X_n^i\right)$, $\sum_{i=1}^N \overline{W}_{n+1}^i = 1$.
3. Resample $\overline{X}_{0:n}^i \sim \sum_{i=1}^N \overline{W}_{n+1}^i \delta_{X_{0:n}^i}(dx_{0:n})$.

The approximations of $p_\theta(x_{0:n}|y_{0:n})$ and $p_\theta(y_n|y_{0:n-1})$ are given by:

$$\widehat{p}_\theta(dx_{0:n}|y_{0:n}) = \sum_{i=1}^N W_n^i \delta_{X_{0:n}^i}(dx_{0:n}), \tag{75}$$

$$\widehat{p}_\theta(y_n|y_{0:n-1}) = \left(\frac{1}{N}\sum_{i=1}^N w_n\left(X_{n-1:n}^i\right)\right)\left(\sum_{i=1}^N W_{n-1}^i q_\theta\left(y_n, X_{n-1}^i\right)\right) \tag{76}$$

where

$$W_n^i \propto w_n\left(X_{n-1:n}^i\right), \sum_{i=1}^N W_n^i = 1$$

and

$$\widehat{p}_\theta(y_0) = \frac{1}{N}\sum_{i=1}^N w_0\left(X_0^i\right).$$

We conclude the discussion of the APF by a summary of the procedure and the anticipated benefits. We are changing carefully the weights by multiplying with something and dividing at the next step (notice the cancellations in (74)). The aim is to obtain weights with less variance and a PF that is more stable numerically. The hope is that the new likelihood $g_\theta(y_n|x_n)\frac{q_\theta(y_{n+1},x_n)}{q_\theta(y_n,x_{n-1})}$ will be less "peaky" or informative and as a result $\tilde{\pi}_n$ is closer to $\tilde{\pi}_{n-1}$. This strategy can be quite effective when the dynamics of $X_n$ mix slowly or $g_\theta$ too informative. Finally, note the effectiveness of the APF will depend on how strongly characteristics appear in a particular problem, see for instance [45] for a thorough comparison with the bootstrap PF.

*Remark* 4. For problems that involve discretisations of continuous time models like SDEs one could even set

$$\frac{q(y_{k+1}, x_k)}{q(y_k, x_{k-1})} = \prod_{m=1}^M \frac{\mathfrak{r}_{k,m}(y_{k+1}, y_k, x_{k,m})}{\mathfrak{r}_{k,m-1}(y_{k+1}, y_k, x_{k,m-1})}$$

and then use $f(X_k|X_{k-1}) = \prod_{m=1}^M f_m(X_{k,m}|X_{k,m-1})$ with $X_k = X_{k,M}$ and $X_{k-1} = X_{k,0}$. This can be very useful as it allows to progressively process an observation and implement a multi-step version of the APF; see [24] for more details.

*Remark* 5. Following the previous remark, one could also use a tempering implementation

$$\mathfrak{r}_{k,m} = g(Y_{k+1}|x_{k,m})^{\phi_m}, \quad r_{k,0} = 1,$$

with $\phi_M = 1$ and $0 < \phi_1 < \phi_2 < \ldots < \phi_m$. This was proposed originally in [35]. In the absence of expressions for $f_m$ in Remark 4 one could use instead appropriate MCMC moves instead of natural dynamics, similarly to the resample-move PF. In addition, one can decide values for each $\phi_m$ on the fly according to the ESS falling below some prescribed value. This combination of ideas from adaptive resampling, using MCMC steps and progressive weighting has been very popular recently and can be very effective in high dimensional problems such as in [43].

## 5.4 SMC for static problems

So far we have looked at problems targeting a sequence of distributions like $\{p_\theta(x_{0:n}|y_{0:n})\}_{n\geq 0}$ defined on on a state space of increasing dimension $\mathcal{X}^n$. We will now present briefly some ideas on how the SMC methodology is useful also on static problems defined on a state space of fixed dimension. The ideas were introduced [15] and later extended significantly in [22].

As an example consider the problem of performing Bayesian inference for

$$\begin{aligned}\grave{\pi}_n(\theta) &= p(\theta|y_{0:n}) \\ &\propto p(\theta)\prod_{k=0}^{n} p(y_k|y_{0:k-1},\theta).\end{aligned}$$

This could be an inference problem for $\theta$ in a HMM or any other statistical model for $\theta$ and $y_{0:n}$. Suppose that one can design MCMC kernels that are invariant to $\grave{\pi}_n(\theta)$ using an approach from Section 2.4 or a more advanced one. Lets denote such a MCMC kernel as $\grave{K}_n(\theta, d\theta')$. One can re-write $\grave{\pi}_n$ as

$$\grave{\pi}_n(\theta')d\theta' \propto p(y_n|y_{0:n-1},\theta')\int \grave{\pi}_{n-1}(\theta)K_{n-1}(\theta,d\theta'),$$

which is a recursion like (24) and (27) combined, with $K_{n-1}$ providing the dynamics and $p(y_n|y_{0:n-1},\theta')$ the conditional likelihood. Then one can implement Algorithm 8 using $K_{n-1}$ instead of $q$ and $p(y_n|y_{0:n-1},\theta')$ for $w_n$, which is a SMC algorithm for $\grave{\pi}_n(\theta)$. We have limited the presentation here on the marginal filters, which for each $n$ coincide with $\grave{\pi}_n(\theta)$. One expand this discussion to include what is happening on the path space but this goes beyond the purpose of this course.

Similarly to the discussion for the resample-move PF, one has benefit of being able to use the particles to design more efficient MCMC proposals. Note that in general SMC for static problems will be a sequential but not on-line algorithm. This is because one needs to compute $p(y_n|y_{0:n-1},\theta')$ at each $n$ and design MCMC steps that target a product $p(\theta)\prod_{k=0}^{n} p(y_k|y_{0:k-1},\theta)$ of increasing size (with $n$). In this case, one cannot use the fixed lag trick we mentioned earlier for the resample-move PF, so typically the computational cost would be proportional to $Nn^2$.

## 5.5 Discussion

Path degeneracy can be addressed partially by adaptive resampling (applying resampling only when necessary,) or by adding MCMC moves to jitter the particles and reintroduce lost diversity in particle approximations. This can improve the estimation procedure, but to a large extent path degeneracy will be still present. In addition, it is clear that weight degeneracy can be addressed by good selection of importance proposals. If this is not sufficient due to having too informative observations, or poorly mixing signals, then there is always an option to change the target sequence of distribution, so that the particles propagation to an easier problem as in APF. Finally, we emphasise that these ideas can be very useful in problems defined on static problems, with no dynamic behaviour. The SMC algorithm can be then combined nicely with MCMC steps, but its main structure and properties are very similar to the when used to tackle problems with HMMs.

### 5.5.1 Key points

Make sure you understand:

- what each extension of the basic PF is trying to achieve (i.e. whether it tackles weight or path degeneracy) and what is the mechanism in terms of the SMC algorithm.

- APF, Resample move and Adaptive resampling are generic approaches that can be combined together. This can lead to improved performance, albeit with a more complicated algorithm.
- In order to achieve strong performance for a given HMM, one still needs to spend effort to customise and design the algorithmic steps for the problem at hand, e.g. design $q_\theta\left(x_n|\, x_{n-1}, y_n\right)$ or $q_\theta\left(y_{n+1}, x_n\right)$ in the APF case.
- Paying attention to the model at hand is crucial for the performance of any Monte Carlo algorithm. This extends to designing IS proposals and assessing when APF is better than a bootstrap PF. In the vast majority of cases, there are no universal guarantees that one PF design is better than another for the same amount of computation. This will depend on the characteristics of the problem at hand.
- SMC can be used for static problems too when MCMC procedures are available. The advantage of SMC as an alternative to standard iterative MCMC is that the computation can be parallelised, which is hard with MCMC, and one can easily monitor as $n$ evolves how the posterior changes with the more observations. Note that the total computational cost involved is a super-linear and proportional to $n^2N$, so this is not a truly on-line approach.

### 5.5.2 Reading List

A good reference to use for this section is the [30] (Sections 1, 2.1, 2.2, 2.4, 3.1-3.5, 3.7, 4.1, 4.2, 4.3, 4.4, 6) and for the static SMC you could look at [15] and [22] for a more advanced treatment.

### 5.5.3 Homework

For the following scalar model

$$X_n = \rho X_{n-1} + \tau V_n, \quad Y_n = X_n + \sigma W_n, \tag{77}$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}\left(0,1\right)$, $X_0 \sim \mathcal{N}\left(0,1\right)$.

1. Synthesise a data-sets $y_{0:T}$ for $T = 5000$, $\rho = 0.8, \tau = 1$ with varying $\sigma = 0.001,\ 0.01,\ 0.1,\ 1,\ 10$. Store the real state trajectory $x^*_{0:T}$ for future comparisons in each case.
   (a) Implement the auxiliary PF (APF) for bootstrap or optimal importance proposals.
   (b) Compare the APF with bootstrap PF and with SIR with optimal proposal in terms of accuracy for filter mean and variance, as well as Monte Carlo variance of the marginal likelihood.
   (c) How small does $\sigma$ needs to get so that the APF shows superior performance?
2. For some cases, e.g. $\sigma = 0.1$
   (a) implement the resample move PF for $L = 1$, $M = 3$. Plot the ESS for the resample move and compare with APF, bootstrap PF, and optimal proposal PF.
   (b) repeat the above using adaptive resampling PF.

# 6 Smoothing Algorithms

Smoothing algorithms are focused in designing better particle approximations for $\left\{p_\theta\left(x_n|\, y_{0:T}\right)\right\}_{n=0}^T$. This could be either because one is interested in these smoothing marginals or because they arise in parameter estimation. In order to be able to relate later clearer with summary statistics that arise in parameter estimation, first we will slightly extend the definition of the smoothed additive functional presented earlier in (65) by letting $s_k$ depending both on $x_k$ and $x_{k-1}$. Then we will also manipulate the expression in (65) by reversing sum and integral and integrating out terms to get:

$$\mathcal{S}_T^\theta = \sum_{k=0}^{T} \int \left[s_k\left(x_k, x_{k-1}\right)\right] p_\theta\left(x_{k-1}, x_k|\, y_{0:n}\right) dx_{k-1:k}. \tag{78}$$

The simplest particle approximation for $\mathcal{S}_T^\theta$ would involve running a standard PF such as Algorithm 8 and using:

$$\begin{aligned}\widehat{\mathcal{S}_T^\theta} &= \int \left[\sum_{k=0}^{T} s_k\left(x_k, x_{k-1}\right)\right] \widehat{p}_\theta\left(dx_{0:T}\middle|\, y_{0:T}\right), \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{k=0}^{T} s_k\left(\bar{X}_k^i, \bar{X}_{k-1}^i\right),\end{aligned}$$

where here $\bar{X}_k^i$ is $k$-th time step of the final path particle $\bar{X}_{0:T}^i$ (and not the resampled particle at time $k$). This is exactly the particle approximation noted in (66), which was shown in (67) to suffer from super-quadratic increase in the Monte Carlo variance due to path degeneracy.

Having particle approximations $\{\hat{p}_\theta\left(x_n\middle|\, y_{0:T}\right)\}_{n=0}^T$ and $\{\hat{p}_\theta\left(x_n, x_{n+1}\middle|\, y_{0:T}\right)\}_{n=0}^T$ that do not suffer from path degeneracy then will be very useful to get estimates of $\mathcal{S}_T^\theta$ with much lower Monte Carlo variance. This will eventually translate better (or less variable) parameter estimates. Some popular approaches are to use (a) fixed lag smoothing, (b) forward filtering backward sampling, and (c) forward filtering backward smoothing.

## 6.1 Fixed lag approximations

Using fixed lag approximations for smoothing originates from [49]. For state-space models with "good" forgetting properties if $L$ large enough then the following approximation might be reasonable:

$$p_\theta\left(x_{0:n}\middle|\, y_{0:T}\right) \approx p_\theta\left(x_{0:n}\middle|\, y_{0:(n+L)\wedge T}\right),$$

i.e. the observations collected at times $k > n + L$ do not bring any significant additional information about $X_{0:n}$. This means that at times $k > n + L$ one needs only to resample $X_{k-L+1:k}^i$. The fixed lag method does not resample the components $X_{0:n}^i$ of the particles $X_{0:k}^i$ obtained by particle filtering. This has appeared in [49] and was used to provide approximations $\{\hat{p}_\theta\left(x_n\middle|\, y_{0:T}\right)\}_{n=0}^T$ and $\{\hat{p}_\theta\left(x_n, x_{n+1}\middle|\, y_{0:T}\right)\}_{n=0}^T$. The algorithm differs from standard SIR implementation only in the resampling step. The approach can work in practice, but method is asymptotically biased (in $N$) and it might be hard to tune $L$. A related theoretical study can be found in [58].

## 6.2 The backward interprentation of $p_\theta\left(x_{0:T}\middle|\, y_{0:T}\right)$

The joint smoothing distribution $p_\theta\left(x_{0:T}\middle|\, y_{0:T}\right)$ can be expressed as a function of the filtering distributions $\{p_\theta\left(x_n\middle|\, y_{0:n}\right)\}_{n=0}^T$ as follows:

$$p_\theta\left(x_{0:T}\middle|\, y_{0:T}\right) = p_\theta\left(x_T\middle|\, y_{0:T}\right)\prod_{n=0}^{T-1} p_\theta\left(x_n\middle|\, y_{0:n}, x_{n+1}\right) \tag{79}$$

where we define a backward Markov density:

$$\begin{aligned}p_\theta\left(x_n\middle|\, y_{0:n}, x_{n+1}\right) &= \frac{f_\theta\left(x_{n+1}\middle|\, x_n\right) p_\theta\left(x_n\middle|\, y_{0:n}\right)}{p_\theta\left(x_{n+1}\middle|\, y_{0:n}\right)}. \\ &= \frac{f_\theta\left(x_{n+1}\middle|\, x_n\right) p_\theta\left(x_n\middle|\, y_{0:n}\right)}{\int f_\theta\left(x_{n+1}\middle|\, x_n\right) p_\theta\left(x_n\middle|\, y_{0:n}\right) dx_n}.\end{aligned} \tag{80}$$

The expression in (79) suggests we can obtain a sample from $p_\theta\left(x_{0:T}\middle|\, y_{0:T}\right)$ by sampling initially from $p_\theta\left(x_T\middle|\, y_{0:T}\right)$ and then sequentially from the backward Markov transition density $p_\theta\left(x_n\middle|\, y_{0:n}, x_{n+1}\right)$ then a sample from. Clearly in general it is not possible to perform this simulation directly, but we can use particle approximations instead that pass through from the random points or grids simulated each time $n$ during a forward pass of SIR Algorithm 8.

## 6.3 Forward Filtering Backward Sampling (FFBSa)

Lets say we have run a PF from time $n = 0$ to $T$, and in addition to the usual procedure in Algorithm 8 we also store every approximate filtering distribution $\left\{\widehat{p}_\theta\left(dx_n\middle|\, y_{0:n}\right) = \sum_{i=1}^N W_n^i \delta_{X_n^i}\left(dx_n\right)\right\}_{n=0}^T$. For every $n$, the $\left\{X_n^i\right\}_{i=1}^N$ can be viewed as a random sample from which one can sample from. If the probability of sampling is given by $W_n^i$ then this is equivalent to sampling from $\widehat{p}_\theta\left(dx_n\middle|\, y_{0:n}\right)$ or resampling. We can use a similar approach to sample from

**Algorithm 16** Forward Filtering Backward Sampling (FFBSa)

- Run a particle filter from time $n = 0$ to $T$, storing the approximate filtering distributions $\{\widehat{p}_\theta(dx_n|y_{0:n})\}_{n=0}^T$.
- Sample $\tilde{X}_T \sim \widehat{p}_\theta(dx_T|y_{0:T})$ and
- For $n = T-1, T-2, ..., 0$ sample

$$\tilde{X}_n \sim \widehat{p}_\theta\left(dx_n|y_{0:n}, \tilde{X}_{n+1}\right)$$

a particle approximation of $p_\theta(x_n|y_{0:n}, x_{n+1})$. Let $\hat{p}_\theta(dx_n|y_{0:n}, x_{n+1})$ denote this particle approximation, where this distribution is obtained by substituting $\widehat{p}_\theta(dx_n|y_{0:n})$ for $p_\theta(dx_n|y_{0:n})$ in (80):

$$\widehat{p}_\theta(dx_n|y_{0:n}, X_{n+1}) = \frac{\sum_{i=1}^N W_n^i f_\theta(X_{n+1}|X_n^i)\delta_{X_n^i}(dx_n)}{\sum_{j=1}^N W_n^j f_\theta(X_{n+1}|X_n^j)}. \tag{81}$$

Given a value for $X_{n+1}$, obtaining one sample from $\hat{p}_\theta(dx_n|y_{0:n}, x_{n+1})$ is equivalent to picking can sample from $\left\{X_n^i\right\}_{i=1}^N$ with probabilities $\left\{\frac{W_n^i f_\theta(X_{n+1}|X_n^i)}{\sum_{j=1}^N W_n^j f_\theta(X_{n+1}|X_n^j)}\right\}_{i=1}^N$. One can then proceed backwards from time $T$ to time 0 to get a sample $\tilde{X}_{0:T} \sim \hat{p}_\theta(x_T|y_{0:T}) \prod_{n=0}^{T-1} \hat{p}_\theta(x_n|y_{0:n}, x_{n+1})$. This procedure is referred to as Forward Filtering Backward Sampling (FFBSa) and is described in Algorithm 16. Note that this procedure generates a single sample for $\tilde{X}_{0:T}$ so has to be repeated $N$ times if one wants to get $N$ equally weighted samples $\left\{\tilde{X}_n^i\right\}_{i=1}^N$. These samples can be used to approximate $\mathcal{S}_T^\theta$ with

$$\widehat{\mathcal{S}_T^\theta} = \frac{1}{N}\sum_{k=0}^T \sum_{i=1}^N \left[s_k\left(\tilde{X}_k^i, \tilde{X}_{k-1}^i, y_k\right)\right].$$

This approach will address the path degeneracy issue, but will lead to a computational cost proportional to $N^2T$. Of course, at this cost one is able to choose randomly from $N^T$ possible paths (in contrast to $NT$ ones that Algorithm 8 would allow) and this can explain intuitively how path degeneracy is addressed. Recently, in [61] proposed an efficient simulation method to bypass the operations with $N^2$ cost, so one can perform FFBSa with a much lower computational cost than presented here.

## 6.4 Forward Filtering Backward Smoothing (FFBSm)

Note that the backward Markov kernel in (80) can be also used to provide a backward in time recursion for $\{p_\theta(x_n|y_{0:T})\}_{n=0}^T$. It follows by integrating out $x_{0:n-1}$ and $x_{n+1:T}$ in (79) while applying (80):

$$\begin{aligned} p_\theta(x_n|y_{0:T}) &= \int p_\theta(x_n, x_{n+1}|y_{0:T})\, dx_{n+1} \\ &= \int p_\theta(x_n|y_{0:n}, x_{n+1})\, p_\theta(x_{n+1}|y_{0:T})\, dx_{n+1} \\ &= \int \frac{f_\theta(x_{n+1}|x_n)\, p_\theta(x_n|y_{0:n})}{p_\theta(x_{n+1}|y_{0:n})} p_\theta(x_{n+1}|y_{0:T})\, dx_{n+1}. \end{aligned}$$

So the backward in time recursion for $\{p_\theta(x_n|y_{0:T})\}_{n=0}^T$ is:

$$p_\theta(x_n|y_{0:T}) = p_\theta(x_n|y_{0:n}) \int \frac{f_\theta(x_{n+1}|x_n)\, p_\theta(x_{n+1}|y_{0:T})}{p_\theta(x_{n+1}|y_{0:n})} dx_{n+1}. \tag{82}$$

This backward recursion can lead to a particle approximation $\{\overline{p}_\theta(x_n|y_{0:T})\}_{n=0}^T$. Assume at time $n$ we have an approximation

$$\overline{p}_\theta(dx_{n+1}|y_{0:T}) = \sum_{i=1}^N W_{n+1|T}^i \delta_{X_{n+1}^i}(dx_{n+1})$$

where we initialise at time $T$ with $W^i_{T|T} = W^i_T$. Then using (82) and (81) together, we obtain the approximation

$$\overline{p}_\theta\left(dx_n|\, y_{0:T}\right) = \sum_{i=1}^N W^i_{n|T}\delta_{X^i_n}(dx_n)$$

with

$$W^i_{n|T} = W^i_n \cdot \sum_{j=1}^N \frac{W^j_{n+1|T} f_\theta\left(X^j_{n+1}|X^i_n\right)}{\sum_{l=1}^N W^l_n f_\theta\left(X^j_{n+1}|X^l_n\right)}. \tag{83}$$

This is a standard particle approximation for $\{p_\theta\left(x_n|\, y_{0:T}\right)\}_{n=0}^T$, so $\sum_{i=1}^N W^i_{n|T}\varphi(X^i_n)$ can be used to approximate integrals of the form $\int \varphi(x_n)p_\theta\left(x_n|\, y_{0:T}\right)dx_n$. Note that this simply uses earlier samples $\left\{X^i_n\right\}_{i=1}^N$ that were generated during the usual forward PF of Algorithm 8 together with the new weights $W^i_{n|T}$. The approach is called Forward Filtering Backward Smoothing (FFBSm) and is presented in Algorithm 17. Due to the form of (83), the computational cost is proportional to $N^2T$ operations in total.

**Algorithm 17** Forward Filtering Backward Smoothing (FFBSm)

Run a particle filter from time $n = 0$ to $T$, storing the approximate filtering distributions $\{\widehat{p}_\theta\left(dx_n|\, y_{0:n}\right)\}_{n=0}^T$,

- Initialise backward pass: $W^i_{T|T} = W^i_T$
- For $n = T-1, T-2, ..., 0$ compute weights

$$W^i_{n|T} = W^i_n \cdot \sum_{j=1}^N \frac{W^j_{n+1|T} f_\theta\left(X^j_{n+1}|X^i_n\right)}{\sum_{l=1}^N W^l_n f_\theta\left(X^j_{n+1}|X^l_n\right)}.$$

  and obtain the approximation

$$\overline{p}_\theta\left(dx_n|\, y_{0:T}\right) = \sum_{i=1}^N W^i_{n|T}\delta_{X^i_n}(dx_n)$$

We return to the issue for approximating the smoothed additive functional $\mathcal{S}^\theta_n$ in (78). This requires specifying a particle approximation of $p_\theta\left(dx_n, dx_{n+1}|\, y_{0:T}\right)$ use within (78). Notice that in earlier calculations we used

$$\begin{aligned} p_\theta\left(x_n, x_{n+1}|\, y_{0:T}\right) &= p_\theta\left(x_n|\, y_{0:n}, x_{n+1}\right) p_\theta\left(x_{n+1}|\, y_{0:T}\right) \\ &= \frac{f_\theta\left(x_{n+1}|\, x_n\right) p_\theta\left(x_n|\, y_{0:n}\right)}{p_\theta\left(x_{n+1}|\, y_{0:n}\right)} p_\theta\left(x_{n+1}|\, y_{0:T}\right) \\ &= \frac{f_\theta\left(x_{n+1}|\, x_n\right) p_\theta\left(x_n|\, y_{0:n}\right)}{\int f_\theta\left(x_{n+1}|\, x_n\right) p_\theta\left(x_n|\, y_{0:n}\right) dx_n} p_\theta\left(x_{n+1}|\, y_{0:T}\right). \end{aligned} \tag{84}$$

Suppose we have an approximation

$$\overline{p}_\theta\left(dx_{n+1}|\, y_{0:T}\right) = \sum_{i=1}^N W^i_{n+1|T}\delta_{X^i_{n+1}}(dx_{n+1})$$

given from Algorithm 17 and are interested to obtain an approximation of the form:

$$\overline{p}_\theta\left(dx_n, dx_{n+1}|\, y_{0:T}\right) = \sum_{i=1}^N \tilde{W}^i_{n,n+1|T}\delta_{X^{a(i)}_n, X^i_{n+1}}(dx_n).$$

One can substitute $\overline{p}_\theta\left(dx_{n+1}|\, y_{0:T}\right)$ and $\hat{p}_\theta\left(dx_n|\, y_{0:n}\right)$ in equation (84) to get

$$\tilde{W}^i_{n,n+1|T} = W^{a_n(i)}_n \frac{W^i_{n+1|T} f_\theta\left(X^i_{n+1}|X^{a_n(i)}_n\right)}{\sum_{l=1}^N W^l_n f_\theta\left(X^i_{n+1}|X^l_n\right)}. \tag{85}$$

where $X_n^{a_n(i)}$ is the ancestor of $X_{n+1}^i$ generated by the resampling algorithm during the forward pass. One can then approximate approximate $\mathcal{S}_n^\theta$ with

$$\widehat{\mathcal{S}_T^\theta} = \sum_{k=0}^{T} \sum_{i=1}^{N} \tilde{W}_{k-1,k|T}^i s_k \left( X_k^i, X_{k-1}^{a_k(i)} \right). \tag{86}$$

## 6.5 Discussion

Assuming the HMM possesses some exponential forgetting properties one can show that approximations of $\mathcal{S}_n^\theta$ based on the fixed-lag approximation will have an asymptotic variance with rate $n/N$ with a non-vanishing (as $N \to \infty$) bias proportional to $n$ and a constant decreasing exponentially fast with $L$; see [58]. In addition, the asymptotic bias and variance of the particle estimate of $\mathcal{S}_n^\theta$ computed using the forward-backward procedures (FFBSa or FFBSm) satisfy:

$$\left| \mathbb{E}^N \left( \widehat{\mathcal{S}}_n^\theta \right) - \mathcal{S}_n^\theta \right| \leq F_\theta \frac{n}{N}, \;\; \mathbb{V}ar^N \left( \widehat{\mathcal{S}}_n^\theta \right) \leq H_\theta \frac{n}{N}. \tag{87}$$

but note this is using algorithms at cost of $N^2 T$ operations. The notable thing to notice here is that the expression for bias for a standard SMC algorithm and the forward-backward estimators of $\mathcal{S}_n^\theta$ are actually equal, [23]. So a standard SMC method achieves the same bias with less computational cost, but at the same time suffers from very high increase in variance with time. Choosing the right method for the a particular problem at hand will depend on many factors: the size of $T$, the model, the allowed computational budget for $N$, the required accuracy. Depending one all these choosing which approach is more suitable requires balancing the aforementioned bias-variance trade-off.

### 6.5.1 Numerical examples

We will consider again the simple scalar linear Gaussian state space model for the different comparisons to follow:

$$X_n = \rho X_{n-1} + \tau W_n, \; Y_n = X_n + \sigma V_n \tag{88}$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and $\rho \in [-1,1]$. The KF recursions can to provide a benchmark for the summary statistics $\mathcal{S}_n^\theta$. In this rather simple model it is straightforward to present numerical evidence of some effects of path degeneracy, which will be very relavant for parameter estimation of $\theta$. We will show how it can be overcome by choosing a particle smoothing method. For convenience we choose FFBSm, although one could opt here for the much more efficient method in [61]. Given use of particle methods does not impose much restrictions on the model dynamics choice, most conclusions can be extended to more complex and high dimensional settings.

We begin by investigating the empirical variance when estimating the smoothed additive functionals $\mathcal{S}_n^\theta$ given in (78). The first method we will consider is method like Algorithm 8 that does not employ any smoothing and uses a computational cost $\mathcal{O}(N)$ per time. One can use the output of Algorithm 8, and then use $\widehat{p}_\theta(dx_{0:n} | y_{0:n})$ in (57) to approximate $\mathcal{S}_n^\theta$ as $\widehat{\mathcal{S}}_n^\theta$ recursively by storing and updating $\sum_{k=1}^n s_k(X_{k-1}^i, X_k^i)$ at each time $n$; note that $\sum_{k=1}^n s_k(X_{k-1}^i, X_k^i)$ needs to be resampled together with $X_{0:n}^i$, see [10, Section 8.3] for more details. The second method is the forward only implementation of FFBSm presented in Algorithm 17 using (86) to estimate of $\mathcal{S}_n^\theta$. Recall that the latter method has a computational cost that is $\mathcal{O}(N^2)$ per time and provides the same estimates as the standard forward-backward implementation of FFBSm.

For the model described in (88) we will set $s_k(x_{k-1}, x_k) = x_{k-1} x_k$ and compute $\mathcal{S}_n^\theta$. We will use a simulated data-set of size $6 \times 10^4$ obtained using $\theta^* = (\rho^*, \tau^{2^*}, \sigma^{2^*}) = (0.8, 0.1, 1)$ and then generate 300 independent replications of each method in order to compute the empirical bias and variance of $\hat{\mathcal{S}}_n^\theta$ when $\theta$ is fixed to $\theta^*$. In order to make a comparison that takes into account the computational cost, we will use $N^2$ particles for the $\mathcal{O}(N)$ method (that uses SIR, Algorithm (8)) and $N$ for the $\mathcal{O}(N^2)$ one (that uses FFBSm, Algorithm (17)). We will look separately at the behavior of the bias of $\hat{\mathcal{S}}_n^\theta$ and the variance and mean squared error (MSE) of the re-scaled estimates $\hat{\mathcal{S}}_n^\theta / \sqrt{n}$. The results are presented in Figure 6 for $N = 50, 100, 200$.

For both methods the bias grows linearly with time, this growth being higher for the $\mathcal{O}(N^2)$ FFBSm method. This is in agreement with the theoretical results in the literature implying that we expect a growth proportional to $n/N^2$ for the $\mathcal{O}(N)$ method and to $n/N$ for the $\mathcal{O}(N^2)$ method. For the variance of $\hat{\mathcal{S}}_n^\theta / \sqrt{n}$, we observe a linear growth with time for the $\mathcal{O}(N)$ standard method with $N^2$ particles whereas this variance appears roughly constant for the $\mathcal{O}(N^2)$ FFBSm method. This figure is also in agreement with theoretical results as the variance $\hat{\mathcal{S}}_n^\theta$ is supposed to grow at least at the rate $n^2/N$ when the $\mathcal{O}(N)$ method is used (see [65]) and at the rate $n/N$ when the $\mathcal{O}(N^2)$ method is used [23], [26]. Finally, the MSE of $\hat{\mathcal{S}}_n^\theta / \sqrt{n}$ grows for both methods linearly as expected. In this particular scenario, the constants of proportionality are such that the MSE is lower for the $\mathcal{O}(N)$ method than

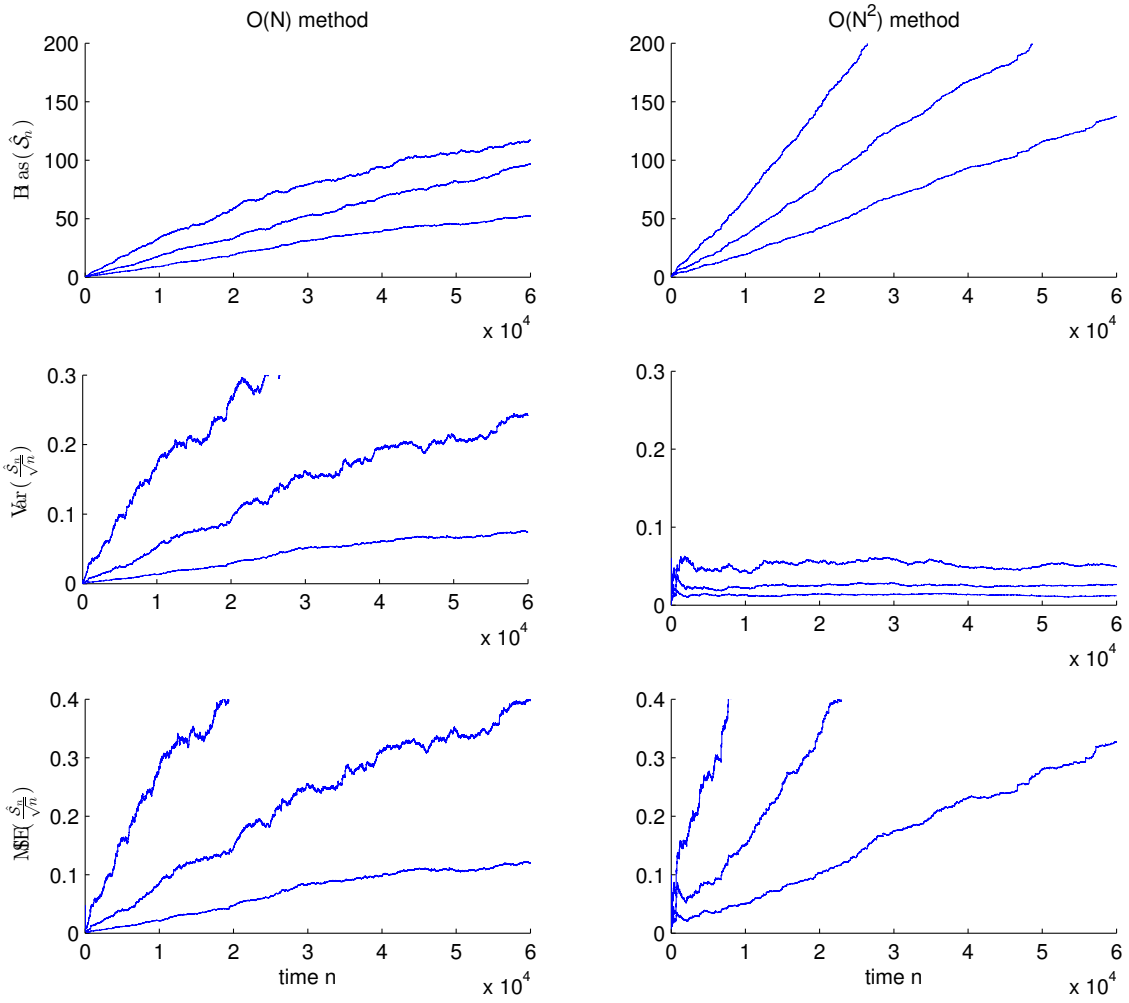


Figure 6: Estimating smoothed additive functionals: Empirical bias of the estimate of $\mathcal{S}_n^\theta$ (top panel), empirical variance (middle panel) and mean squared error (bottom panel) for the estimate of $\mathcal{S}_n^\theta/\sqrt{n}$. Left column: $\mathcal{O}(N)$ method using $N^2 = 2500, 10000, 40000$ particles. Right column: $\mathcal{O}(N^2)$ method using $N = 50, 100, 200$ particles. In every subplot, the top line corresponds to using $N = 50$, the middle for $N = 100$ and the lower for $N = 200$.

for the $\mathcal{O}(N^2)$ method. In general, we can expect that the $\mathcal{O}(N)$ method will be superior in terms of the bias and the $\mathcal{O}(N^2)$ method superior in terms of the variance. This motivates the development of bias reduction schemes for the $\mathcal{O}(N^2)$ method.

### 6.5.2 Key points

Make sure you understand:

- There are dedicated smoothing algorithms (some with higher computational cost) that are not affected by path degeneracy.
- To compute $\widehat{\mathcal{S}}_n^\theta$ one can implement a particle method with cost $N^2T$ either (a) simple particle filter with $N^2$ particles or (b) a FFBSa/m particle filter with $N$ particles. Then (a) suffers from path degeneracy, but exhibits a bias of order $T/N^2$ and variance at least of order $T^2/N^2$. On the other hand (b) has bias of order $T/N$ and variance of order $T/N$. So one method is better in terms of the bias and another in terms of the variance, but both will have a similar MSE.

- The tradeoff in the bullet point immediately above becomes much more in favour of using particle smoothing, when faster and cheaper smoothing implementations are used like the one in [61] instead of the $\mathcal{O}(N^2)$ cost implementations like Algorithms 16 and 17.

### 6.5.3 Reading List

You could read further in [68] Chapter 11, [29] Section V, and [30] Section 5.

### 6.5.4 Homework

For the following scalar model

$$X_n = \rho X_{n-1} + \sigma V_n, \quad Y_n = \beta \exp(\frac{X_n}{2}) W_n,$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$, $X_0 \sim \mathcal{N}\left(0, \frac{\sigma^2}{1-\rho^2}\right)$, synthesise a data-set with $y_{0:T}$ for $T = 50$, $\rho = 0.91, \sigma = 1, \beta = 0.5$. Store the real state trajectory $x^*_{0:T}$ for future comparisons. Then implement the SIR with $q_n(x_n|y_n, x_{n-1}) = f_\theta(x_n|x_{n-1})$ and in addition each smoothing method (fixed lag, FFBSa, FFBSm) with $N = 100$. Compare the estimates of $\mathbb{E}\left[X_n \middle| Y_{0:T}\right]$ and $\mathbb{E}\left[X_n^2 \middle| Y_{0:T}\right]$ in each case. Produce plots of the estimates of $\mathbb{E}\left[X_n \middle| Y_{0:T}\right]$, $\mathbb{E}\left[X_n \middle| Y_{0:n}\right]$ with $n$ and compare them with $x^*_{0:T}$.

# 7 Parameter Estimation for State Space models

So far we have managed to get very good approximations of $p_\theta(x_n|y_{0:n})$ but only when $\theta$ is known. Estimation of $\theta$ is often known as parameter inference for HMMs, model calibration, system identification. It is very crucial in practice, as without the static parameters you cannot perform filtering, prediction and smoothing.

In applications, often ad-hoc calibration methods are used, but here we will focus on principled statistical inference approaches. We have chosen to broadly classify the methods as follows: Bayesian or Maximum Likelihood (ML) and whether they are implemented off-line (batch) or on-line (recursively). In the Bayesian approach, the unknown parameters are assigned prior distributions and the target is to compute or approximate the posterior distribution of these parameters given the noisy observations. In the ML approach, the parameter estimates are the maximising arguments of the log-likelihood of the data. Both these inference procedures can be carried out off-line or on-line. Specifically, in an off-line framework we infer the parameter by iterating over a fixed observation record $y_{0:T}$. In contrast, on-line methods update the parameter estimate sequentially as observations $\{y_n\}_{n\geq 0}$ become available.

We need to use PFs within algorithms that are meant to perform inference for $\theta$. Although it is possible to define an extended state including the original state $X_n$ and the static parameter $\theta$ and then apply standard particle methods to perform parameter inference, it was recognised very early on in [47] that this naive approach is problematic due to the parameter space, $\Theta$, not being explored adequately. One could relate this point to loss of ergodicity and forgetting or exponential stability in (62) when $X_n$ is augmented with a static parameter. This has motivated over the past twenty years the development of many particle methods specific to the parameter estimation problem, but numerically robust methods have only been proposed fairly recently. The main objective of this section is to provide an introduction to the problem and familiarise with some methods that can be used.

## 7.1 Bayesian approach

We begin by looking at Bayesian inference using particle methods. In this case parameter $\theta$ is a random variable and given a a suitable prior density $p(\theta)$ one aims to compute or approximate the posterior $p(\theta|\, y_{0:n})$ is given by

$$p(\theta|\, y_{0:n}) \propto p_\theta(y_{0:n})\, p(\theta). \tag{89}$$

This is hard to implement directly as $p_\theta(y_{0:n})$ is intractable, but one could use the fact that $p_\theta(x_{0:n}, y_{0:n})$ can be computed point-wise and aim to approximate $p(x_{0:n}, \theta|\, y_{0:n})$ instead. This task can be performed either:

- Off-line: given a batch of data-points $y_{0:T}$ one aims to compute the joint posterior density $p(x_{0:T}, \theta|\, y_{0:T})$. We will look at the so called particle MCMC (PMCMC) algorithms and in particular pseudo-marginal algorithms that use particle approximations for $p_\theta(y_{0:n})$.

- On-line case: compute the sequence of posterior densities $\{p(x_{0:n}, \theta | y_{0:n})\}_{n=0,\ldots,T}$. For each $n = 1, \ldots, T$ one can use the previous posterior obtained at time $n$ together with $y_n$ to update the posterior. We will see that whilst this is possible implement using simple extensions to the methodology presented so far, the approach can result to biased estimates or high Monte Carlo variance that increases with time.

Before we proceed we remind the reader that in the notation $p_\theta$, the subscripts $\theta$ are used throughout to denote conditioning with $\theta$. For example $p_\theta(y_{0:n})$ is used to denote $p(y_{0:n}|\theta)$.

### 7.1.1 Sequential Bayesian estimation

One can straightforwardly augment state the $x_{0:n}$ with $\theta$ and attempt to implement directly a particle filtering on the extended state $(X_n, \theta_n)$. One can construct a HMM using an initial density $p(\theta_0)\eta_{\theta_0}(x_0)$ and propagate the hidden state with a transition density like $f_{\theta_n}(x_n|x_{n-1})\delta_{\theta_{n-1}}(\theta_n)$, so that $\theta_n = \theta_{n-1}$. One can then implement any particle filtering method. This approach of course is a bit naive. Applying a standard SMC algorithm like Algorithm 8 to the degenerate Markov process $\{X_n, \theta_n\}_{n\geq 0}$ will result to the parameter space only being explored at the initialisation of the algorithm. The successive resampling steps will deplete the diversity of the particle population, so after a certain time $n$, the approximation $\widehat{p}(d\theta_n | y_{0:n})$ will only contain a very few or even a single unique value for $\theta$.

From the perspective of the relevant theory, setting $\theta_n = \theta_{n-1}$ results to loss of ergodicity for both the augmented state and filter so the time uniform error bounds of (63) no longer apply. In addition, when the parameter is fixed to some sampled value, say $\theta = \theta^i$, the variance of $\hat{p}_{\theta^i}(y_{0:n})$increases linearly with $n$. If one is to maintain a certain performance threshold for the estimation of $\theta$, this implies we either need an increasing number of particles with $n$ for each $\theta^i$ or we need to impose some artificial dynamics for $\theta$, so that we are dealing with a problem PF are strong at.

This means that when $\theta_n$ is used as a hidden state one needs to inject some noise in the parameter to maintain some diversity in the particle population. An early pragmatic solution proposed in [47, 39, 54] was to use artificial dynamics for $\theta$, for example

$$\theta_n = \theta_{n-1} + \epsilon_n$$

with $\epsilon_n$ being zero mean noise with small variance. In [54] one can find some interesting approach to tune variance of $\epsilon_n$ from the particles using shrinkage ideas. The resulting estimates for $\theta$ based on $\widehat{p}(d\theta_n, dx_{0:n} | y_{0:n})$ will have a bias that is hard to quantify in general, but this method is quite easy to implement and in many cases has produced reasonable results.

Another approach could be to tweak the resampling procedure. For a sufficiently large $n$ the following approximation

$$p_\theta(x_{0:n-L}|y_{0:n-L}) \approx p_\theta(x_{0:n-L}|y_{0:n})$$

might be reasonable, so one could advocate that there is no need to resample the full path $X^i_{0:n}$ at every $n$ and just set

$$\left(\bar{\theta}^i_n,\, \bar{X}^i_{n-L+1:n}\right) = \left(\theta^{a_n(i)}_n,\, X^{a_n(i)}_{n-L+1:n}\right),$$

where $a_n(i)$ is the sampled ancestor of the resampled particle $i$. This way we leave early parts of the path space untouched and have $\bar{X}^i_{n-L+1:n} = X^i_{n-L+1:n}$. This will introduce a discontinuity in each particle path and a bias that will depend on the value of $L$. The latter requires some tuning for the method to be effective and achieve good balance of the resulting tradeoffs . If $L$ is small the bias will be large, but if it is large then the method will suffer from path degeneracy more. The approach can be complemented with MCMC move steps like the resample move PF in Algorithms 13-14, wherein one needs to modify the target distributions to account for augmenting $X_{0:n}$ with $\theta$. This fixed lag approach with MCMC steps was proposed in [64] under the name *practical filtering*.

Both approaches above will introduce some bias. If one wants to avoid this, one needs to resample $\theta_n$ together with the full path $X_{0:n}$ and one way to add noise to $\theta$ would be to use MCMC steps, like in the resample move PF presented in Section 5.2. With reference to Algorithm 13 changed so that it targets $(\theta, X_{0:n})$ one can use a MCMC kernel with invariant density $p(x_{0:n}, \theta | y_{0:n})$,

$$\left(X^{(i)}_{0:n}, \theta^{(i)}_n\right) \sim K_n\left(\cdot, \cdot | \overline{X}^i_{0:n}, \overline{\theta}^i_n\right)$$

where by construction $K_n$ satisfies

$$p(x'_{0:n}, \theta' | y_{0:n}) = \int p(x_{0:n}, \theta | y_{0:n}) K_n(x'_{0:n}, \theta' | x_{0:n}, \theta) d(x_{0:n}, \theta).$$

If the proposal is based on a Metropolis-Hastings method, one could set the proposal to move $\theta_n^{(i)}$ and $X_{n-L+1:n}^{(i)}$ only as done in Algorithm 14. In fact, one could even take this approach further and only move $\theta_n^{(i)}$ in the proposal of the MCMC step. For some models one can even use Gibbs steps to update the parameter values

$$K_n\left(dx'_{0:n}, d\theta' \middle| x_{0:n}, \theta\right) = \delta_{x_{0:n}}\left(dx'_{0:n}\right) p(\theta' | x_{0:n}, y_{0:n}) d\theta',$$

where

$$p\left(\theta \middle| y_{0:n}, x_{0:n}\right) = p(\theta | s_n(x_{0:n}, y_{0:n}))$$

with $s_n(x_{0:n}, y_{0:n})$ being fixed dimension sufficient statistic required to execute the Gibbs conditional sampling step. With some variation these ideas have appeared many times in the literature, e.g. [2, 70, 32, 12], with the most recent name being *particle learning* coined by [12].

As opposed to the methods relying on kernel or artificial dynamics, these MCMC-based approaches have the advantage of adding diversity to the particles approximating $p\left(\theta | y_{0:n}\right)$ without perturbing the target distribution. Whilst the method is very elegant, path degeneracy is still relevant and will play a role when $s_n(X_{0:n}^i, y_{0:n})$ is computed recursively. Unfortunately, these algorithms rely implicitly on the particle approximation of the density $p\left(x_{0:n} | y_{0:n}\right)$ even if algorithmically it is only necessary to store some fixed-dimensional sufficient statistics $\left\{s_n(X_{0:n}^i, y_{0:n})\right\}$. Hence in this respect they still suffer from the path degeneracy problem. This was noticed as early as in [2]; see also the word of caution in the conclusion of [32], [3] and [17]. The practical implications are that one observes empirically that the resulting Monte Carlo estimates can display quite a lot of variability over multiple runs; we will see this later in Figure 10. This should not come as a surprise as the sequence of posterior distributions does not have exponential forgetting properties, hence there is an accumulation of Monte Carlo errors over time. One will rely on SMC approximations of $p(s_n(x_{0:n}, y_{0:n}) | y_{0:n})$, which for fixed $N$ will exhibit an error due to path degeneracy that increases with $n$. This will result to Monte Carlo variance and sensitivity to parameter initialisation. Of course bearing all this in mind, these methods can be still useful as long as they are used with caution and validated carefully.

### 7.1.2 Batch estimation using Particle MCMC

**Algorithm 18** Particle Marginal Metropolis Hastings (PMMH)

At iteration $k = 0$,

- Set $\theta(0) \sim p(\cdot)$.
- Run an SMC algorithm targeting $p\left(x_{0:T} \middle| y_{0:T}, \theta\right)$, sample $X_{0:T}(0) \sim \widehat{p}\left(dx_{0:T} \middle| y_{0:T}, \theta(0)\right)$, and compute estimate $\widehat{Z}_T\left(\theta(0)\right)$

At iteration $k \geq 1$

- Sample a proposal $\theta' \sim q\left(\theta \middle| \theta(k-1)\right)$.
- Run an SMC algorithm targeting $p\left(x_{0:T} \middle| y_{0:T}, \theta'\right)$, sample $X'_{0:T} \sim \widehat{p}\left(dx_{0:T} \middle| y_{0:T}, \theta'\right)$, and compute estimate $\hat{p}_{\theta'}\left(y_{0:T}\right)$.
- Set $\theta(k) = \theta'$, $X_{0:T}(k) = X'_{0:T}$, and $\hat{p}_{\theta(k)}\left(y_{0:T}\right) = \hat{p}_{\theta'}\left(y_{0:T}\right)$ with probability

$$1 \wedge \frac{\hat{p}_{\theta'}\left(y_{0:T}\right) p(\theta') q(\theta(k-1) | \theta')}{\hat{p}_{\theta(k-1)}\left(y_{0:T}\right) p(\theta(k-1)) q(\theta' | \theta(k-1))},$$

  otherwise set $\theta(k) = \theta(k-1)$, $X_{0:T}(k) = X_{0:T}(k-1)$, and $\hat{p}_{\theta(k)}\left(y_{0:T}\right) = \hat{p}_{\theta(k-1)}\left(y_{0:T}\right)$.

Given a batch of data-points $y_{0:T}$ one can use an iterative method to infer $\theta$. Here we will focus on using MCMC and in particular on the particle MCMC (PMCMC) framework proposed in [5]. If one attempts to use MCMC directly for $p\left(\theta \middle| y_{0:T}\right)$ then (89) suggests that one will have to compute $p_\theta\left(y_{0:T}\right)$. This will be necessary when computing the acceptance ratio of a Metropolis Hastings method such as then one presented in Algorithm 2:

$$\alpha(\theta, \theta') = 1 \wedge \frac{p_{\theta'}\left(y_{0:T}\right) p\left(\theta'\right) q(\theta | \theta')}{p_\theta\left(y_{0:T}\right) p\left(\theta\right) q(\theta' | \theta)}$$

But this is not possible to compute directly as $p_\theta(y_{0:T})$ is intractable. In the same spirit as in the pseudo-marginal method presented in Section 2.4.4, we will use the unbiased particle approximations for $p_\theta(y_{0:T})$ (recall (64)).

We have already seen it is hard to justify such a sampler by targeting directly $p(\theta|\, y_{0:n})$ or the joint posterior density $p(x_{0:T}, \theta|\, y_{0:T})$. We will proceed with this route regardless, in order to develop some intuition. We will first consider the MH case and Algorithm 2 when targetting $p(x_{0:T}, \theta|\, y_{0:T})$. One would need to define a proposal for $(x_{0:T}, \theta)$, $q((x'_{0:T}, \theta')|\,(x_{0:T}, \theta))$, and then compute the acceptance ratio

$$\begin{aligned}\alpha((x_{0:T}, \theta), (x'_{0:T}, \theta')) &= 1 \wedge \frac{p(x'_{0:T}, \theta'|\, y_{0:T})\, q((x_{0:T}, \theta)|\,(x'_{0:T}, \theta'))}{p(x_{0:T}, \theta|\, y_{0:T})\, q((x'_{0:T}, \theta')|\,(x_{0:T}, \theta))} \\ &= 1 \wedge \frac{p_{\theta'}(x'_{0:T}, y_{0:T})\, p(\theta')\, q((x_{0:T}, \theta)|\,(x'_{0:T}, \theta'))}{p_\theta(x_{0:T}, y_{0:T})\, p(\theta)\, q((x'_{0:T}, \theta')|\,(x_{0:T}, \theta))}\end{aligned}$$

where $p_\theta(x_{0:T}, y_{0:T})$ is given by (18), so can be computed point-wise. This is an algorithm that can be implemented in practice, but a naive implementation will result in mixing and performance that will deteriorate rapidly with increasing $T$. The problem is that $x_{0:T}$ and $\theta$ are strongly correlated, and each $x_n$ is strongly correlated with $x_{n-1}$, so naive proposals based on random walks this will make the MCMC sampler very inefficient.

In general for high dimensional problems, MCMC proposals aim to exploit hierarchical structure and 'break' conditional dependencies. One could improve the approach presented above by choosing the following proposal density:

$$q((x'_{0:T}, \theta')|\,(x_{0:T}, \theta)) = q(\theta'|\,\theta)\, p(x'_{0:T}|\, y_{0:T}, \theta') \tag{90}$$

Then the acceptance probability is

$$\begin{aligned}&1 \wedge \frac{p(x'_{0:T}, \theta'|\, y_{0:T})\, q(\theta|\,\theta')\, p(x_{0:T}|\, y_{0:T}, \theta)}{p(x_{0:T}, \theta|\, y_{0:T})\, q(\theta'|\,\theta)\, p(x'_{0:T}|\, y_{0:T}, \theta')} \\ =&1 \wedge \frac{p_{\theta'}(y_{0:T})\, p(\theta')\, q(\theta|\theta')}{p_\theta(y_{0:T})\, p(\theta)\, q(\theta'|\theta)}.\end{aligned} \tag{91}$$

This is referred to as the ideal marginal Metropolis Hastings (IMMH) sampler. The proposal manages to de-couple the correlation between $x_{0:T}$ and $\theta$, but the problem is that neither sampling from $p(x'_{0:T}|\, y_{0:T}, \theta')$ nor computing $p_{\theta'}(y_{0:T})$ is possible to do exactly.

What we will do instead is sample from a particle filter $\hat{p}(dx'_{0:T}|\, y_{0:T}, \theta')$ (or more precisely from the law of the particle filter $\hat{p}\left(\left\{\left\{x_n^i, o_n(i)\right\}_{i=1}^N\right\}_{n=1}^T \middle| y_{0:T}, \theta'\right)$ with $\left\{\left\{X_n^i, O_n(i)\right\}_{i=1}^N\right\}_{n=1}^T$ being all the sampled particles and offsprings in the SMC algorithm) and then use the PF estimate $\hat{p}_{\theta'}(y_{0:T})$ instead of $p_{\theta'}(y_{0:T})$ in (91). This is a PMCMC algorithm that is usually referred to as particle marginal Metropolis Hastings (PMMH). The approach follows from the pseudo-marginal framework of [4], which uses appropriate auxiliary variables that are integrated out in MCMC. Here the variables used to construct SMC algorithm $\left\{\left\{X_n^i, O_n(i)\right\}_{i=1}^N\right\}_{n=1}^T$ can be included together with $\theta$ as auxiliary variables and then integrated out. The inclusion of these auxiliary variables in most setups will not affect the mixing of the PMCMC chain w.r.t. $\theta$ if a sufficient value for $N$ is used. This can be attributed to the HMM structure and the efficiency of using a proposal very close to (90).

The PMMH algorithm is presented in Algorithm 18. It consists of a standard MCMC targeting

$$p\left(\left\{\left\{x_n^i, o_n(i)\right\}_{i=1}^N\right\}_{n=1}^T, \theta \middle| y_{0:T}\right),$$

which denotes the joint density of the parameter $\theta$ and all the simulated variables in the SMC algorithm. One can show that because of the unbiasedness of likelihood estimate,

$$\mathbb{E}^N[\hat{p}_\theta(y_{0:T})] = p_\theta(y_{0:T}),$$

if one marginalises out $\left\{\left\{X_n^i, O_n(i)\right\}_{i=1}^N\right\}_{n=1}^T$ from $p\left(\left\{\left\{x_n^i, o_n(i)\right\}_{i=1}^N\right\}_{n=1}^T, \theta \middle| y_{0:T}\right)$ then the MCMC procedure of Algorithm 18 satisfies detailed balance marginally with $p(\theta|\, y_{0:T})$ and hence generates valid MCMC samples from $p(\theta|\, y_{0:T})$. This means that as far $\theta$ is concerned Algorithm 18 is an exact MCMC algorithm targeting $p(\theta|\, y_{0:T})$ for any value of $N$. Algorithm 18 also provides a MCMC sample sequence $\{\theta(k), X_{0:T}(k)\}_{k\geq 0}$ that can be treated as a sequence of approximate samples from $p(x_{0:T}, \theta|\, y_{0:T})$. In [5], one can more details and also a particle Gibbs sampler implementation.

*Remark* 6. Note that the notation in Algorithm 18 and this section differs from Algorithm 2 and Section 2.4. First we use $\{\theta(k), X_{0:T}(k)\}_{k\geq 0}$ to denote the MCMC sequence and $k$ here is the MCMC iteration index. We keep subscripts for time indices of the HMM. Secondly, for the MCMC proposal we use $q\left(\left(x'_{0:T}, \theta'\right)\middle|\left(x_{0:T}, \theta\right)\right)$ and $q(\theta|\theta')$ instead of $Q\left(\left(x_{0:T}, \theta\right), \left(x'_{0:T}, \theta'\right)\right)$ and $Q(\theta', \theta)$. This choice was motivated in order to stay closer with the notation used in [5] and subsequent literature.

*Remark* 7. In Algorithm 18, at each time $k$ one should use a previously stored value for $\hat{p}_{\theta(k-1)}\left(y_{0:T}\right)$ and not repeat the PF sampling again with $\theta = \theta(k-1)$.

### 7.1.3 Discussion

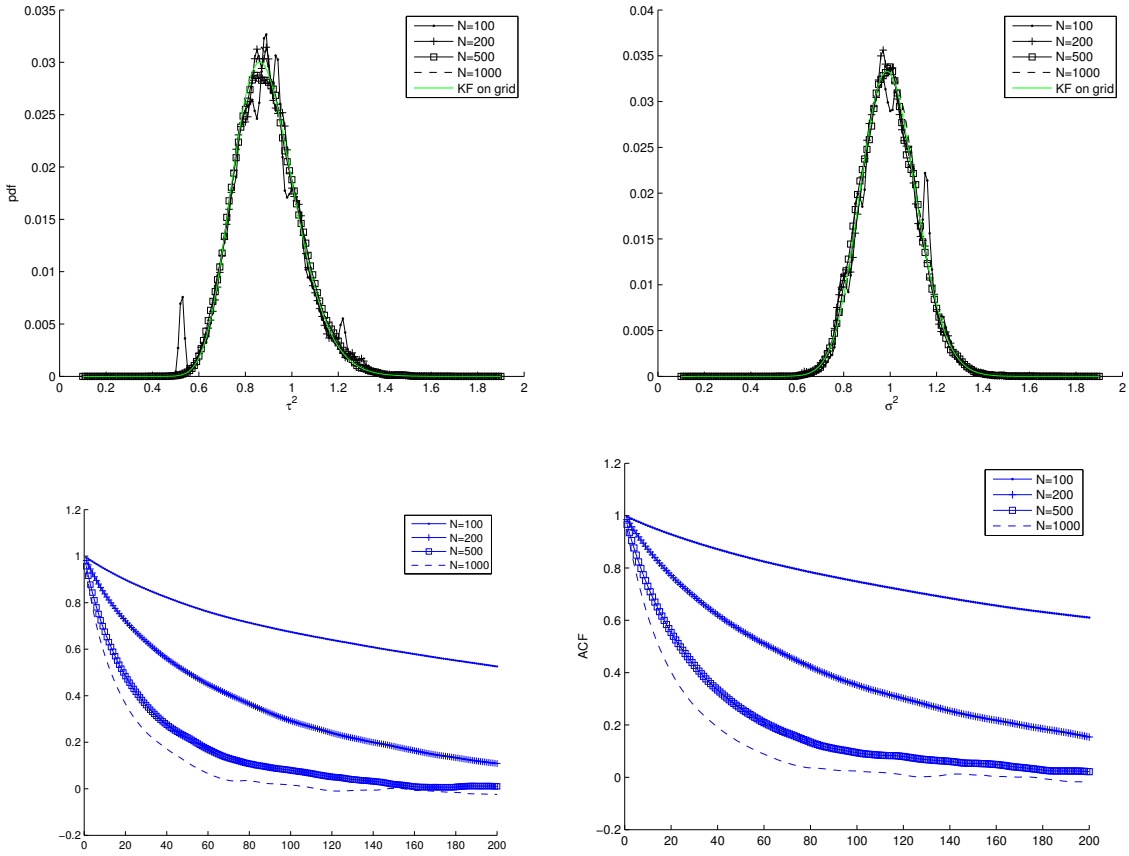


Figure 7: Particle MCMC: top row: estimated posteriors for $\tau^2$ (left) and $\sigma^2$ (right) obtained form a single run of PMMH with $N = 100$ ($*$), 200 ($+$), 500 ($\square$), 1000 (dashed) together with computed using Kalman filter on a grid (solid green line); bottom: autocorrelation plots.

The remarkable feature of this algorithm is that the marginal invariant distribution of the Markov chain $\{X_{0:T}(k), \theta(k)\}$ is $p\left(\theta\middle|\, y_{0:T}\right)$ whatever being $N$. SMC approximations do not introduce any bias and minimal tuning required compared to usual MCMC. Of course the higher $N$ is, then the better the mixing properties of the algorithm will be. In fact, one can notice that for relatively small values of $N$ the algorithm often gets stuck for small intervals of time. This is due to the Monte Carlo variance in $\hat{p}_{\theta'}\left(y_{0:T}\right)$, which can result in an evaluation significantly higher than $p_{\theta'}\left(y_{0:T}\right)$. One simple way to improve on this is to increase $N$. This comes at an extra computational cost, so a trade-off needs to be balanced as after some value of $N$ the reduction of the variance of $\hat{p}_{\theta'}\left(y_{0:T}\right)$ might not have a drastic effect for a fixed $T$. A recent rule of thumb for tuning $N$ was proposed in [63] and there the authors recommend setting $N$ to achieve for $\log \hat{p}_{\theta'}\left(y_{0:T}\right)$ a Monte Carlo variance near 1. As $T$ increases, under favourable mixing assumptions for the HMM, the variance of the acceptance rate of the PMMH sampler is proportional to $T/N$. So $N$ should roughly increase linearly with $T$ and the total computational cost required for a batch of $T$ observations should be proportional to $T^2$.

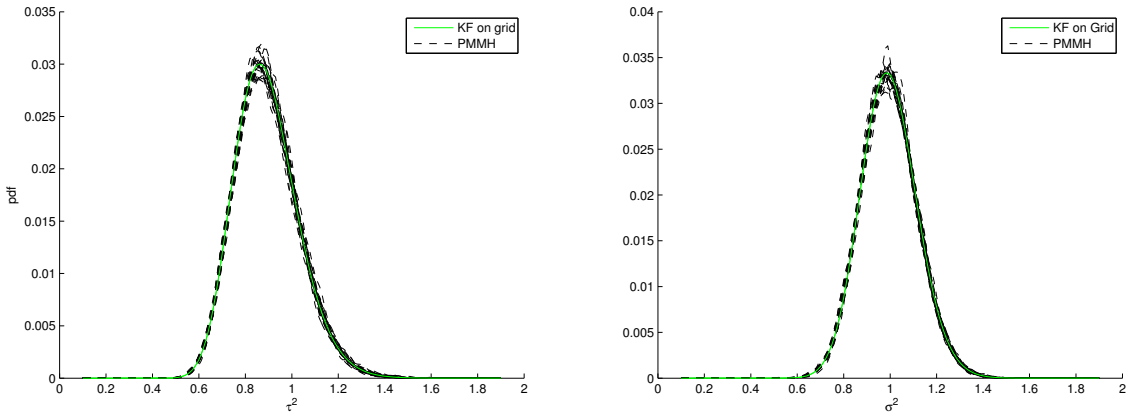


Figure 8: Particle MCMC: estimated pdfs for $\tau^2$ (left, black dashed line) and $\sigma^2$ (right, black dashed line) obtained from 25 independent replications of the PMMH algorithm with $N = 500$ together with pdf computed using Kalman filter on a grid (solid green line).

### 7.1.4 Numerical examples

As before we will consider the scalar linear Gaussian state space model for the different comparisons to follow:

$$X_n = \rho X_{n-1} + \tau W_n,\ Y_n = X_n + \sigma V_n \tag{92}$$

where $W_n, V_n \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and $\rho \in [-1,1]$. The main reason for choosing this model is that the Kalman filter recursions can be combined with relatively inexpensive grid computations to provide a benchmark for the true posterior densities $p(\theta|\, y_{0:n})$. (For the same reason, we will use the same model later also for the ML estimation case). In this rather simple model it is straightforward to present numerical evidence of some effects of path degeneracy for parameer estimation and show how it can be overcome by choosing an appropriate particle method (PMCMC). Given particle methods do not impose restrictions on the model dynamics choice, most conclusions can be extended to more complex and high dimensional settings.

We still consider the model in (92) but simplify it further by fixing either $\rho$ or $\tau$. This is done mainly in order to keep the computations of the benchmarks that use Kalman computations on a grid relatively inexpensive. For those parameters that are not fixed, we shall use the following (independent) priors: a uniform on $[-1,1]$ for $\rho$, and inverse gamma for $\tau^2, \sigma^2$ with the shape and scale parameter pair being $(a,b)$ and $(c,d)$ respectively with $a = b = c = d = 1$. In all the subsequent examples, we will initialize the algorithms by sampling $\theta$ from the prior.

**Off-line case with PMMH** We begin with a brief illustration of PMMH in Algorithm 18 when estimating the posterior of $\theta = (\tau^2, \sigma^2)$ with $\rho = 1$ held fixed and known. We will use a simulated data-set with $T = 500$ and set a random walk MCMC proposal for $\tau^2$ with a standard deviation of 0.07. In Figure 7 we display the posterior densities estimated from the histogram of samples together with the corresponding autocorrelation function plots for each case when $N = 100, 200, 500, 1000$. In each case we used $5 \times 10^5, 2 \times 10^5, 1.5 \times 10^5, 10^5$ iterations of PMMH respectively and the average acceptance ratio was 0.04, 0.13, 0.31 and 0.44. The accuracy of the estimated densities and the mixing of the associated Markov chain clearly improve with $N$ and the estimated densities are quite accurate for $N = 500, 1000$. On the other hand, the relative improvement from $N = 500$ to $N = 1000$ seems marginal relative to the added computational burden. This is in agreement with [5], where the authors emphasize on how computational savings can be made by monitoring the acceptance ratio and careful choosing of $N$ together with the tuning of the Metropolis-Hastings proposal; see [63] for detailed guidelines.

In Figure 8 we plot the estimated posterior densities from $10^5$ samples with $N = 500$ obtained from 25 independent runs of the algorithm. Again we display the posterior densities computed using Kalman filtering on a grid. We observe minimal variability across runs.

**On-line case: particle methods with MCMC steps** We proceed to examine the combination of particle method with MCMC methods, i.e. augmenting $\theta$ with $X_{0:n}$. We focus on an efficient implementation of this idea discussed in [12] which can be implemented for the simple model under consideration. We investigate the effect

of the degeneracy problem in this context. We first focus of the estimate of the posterior of $\theta = (\tau^2, \sigma^2)$ given a long sequence of simulated observations with $\tau = \sigma = 1$. In this scenario, $p_\theta(x_{0:n}, y_{0:n})$ admits the following two-dimensional sufficient statistics, $s_n(x_{0:n}, y_{0:n}) = \left(\sum_{k=1}^n (x_k - x_{k-1})^2, \sum_{k=0}^n (y_k - x_k)^2\right)$, and $\theta$ can be updated using Gibbs steps. We use $T = 5 \times 10^4$ and $N = 10^4$. We ran the algorithm over 100 independent runs over the same data-set. The top panel of Figure 9 show the box plots for the estimates of the posterior mean. In the middle row of Figure 9 we show how the corresponding relative variance of the estimator for the posterior mean evolves with time. Here the relative variance is defined as the ratio of the empirical variance (over different independent runs) of the posterior mean estimates at time $n$ over the true posterior variance at time $n$, which in this case is approximated using a Kalman filters on a fine grid. This quantity exhibits a steep increasing trend when $n \geq 15000$ and confirms the aforementioned variability of the estimates of the posterior mean. In the bottom row of Figure 9 we plot the average (over different runs) of the estimators of the variance of $p(\theta|y_{0:n})$. This average variance is also scaled/normalized by the actual posterior variance. The latter is again computed using Kalman filtering on a grid. This ratio between the average estimated variance of the posterior over the true one decreases with time $n$ and it shows that the supports of the approximate posterior densities provided by this method cover on average only a small portion of support of the true posterior. Both the experiments in the middle and bottom row of Figure 9 confirm that in this example the particle method with MCMC steps fails to adequately explore the space of $\theta$. Although the box plots provide some false sense of security, the relative and scaled average variance clearly indicate that any posterior estimates obtained from a single run of particle method with MCMC steps should be used with caution.

One might argue that these methods are meant to be used with larger $N$ and/or shorter data sets $T$. We shall consider this time a slightly different example where $\tau = 0.1$ is known and we are interested in estimating the posterior of $\theta = (\rho, \sigma^2)$ given a sequence of observations obtained using $\rho = 0.5$ and $\sigma = 1$. In that case, the sufficient statistics are $s_n(x_{0:n}, y_{0:n}) = \left(\sum_{k=1}^n x_{k-1}x_k, \sum_{k=0}^{n-1} x_{k-1}^2, \sum_{k=0}^n (y_k - x_k)^2\right)$, and the parameters can be rejuvenated through a single Gibbs update. In addition, we let $T = 5000$ and use $N = 10^4$ particles. In Figures 10 we display the estimated marginal posteriors $p(\rho|\, y_{0:n})$ and $p\left(\sigma^2\middle|\, y_{0:n}\right)$ obtained from 50 independent replications of the particle method. On this simple problem, the estimated posteriors seem to be consistently inaccurate for $\rho$, whereas they perform better for $\sigma^2$ but with some non-negligible variability over runs. Clearly, one expects this variability will decrease as $N$ increases and increases as $T$ increases. To investigate this further we will consider the same example for $T$=1000 and compare with estimates provided by the Particle Gibbs sampler of [53] using the same computational cost. In Figure 11 and 12 we plot the resulting posteriors when the sequential particle method with MCMC steps uses $N = 7.5 \times 10^4$ and $N = 6 \times 10^5$ respectively. In the same plots we present in the lower panels the results using the Particle Gibbs sampling using $N = 50$ particles with 3000 and 24000 iterations respectively. The estimates provided by the Particle Gibbs sampler appear to display less variability. For a higher dimensional parameter $\theta$ and/or very vague priors, this comparison would be much more favorable to the Particle Gibbs sampler. Of course, both methods require increasing $N$ with $T$ to achieve a specified level of accuracy.

Finally, in other related results not shown here, one can also investigate experimentally the empirical variance of the marginal likelihood estimates $\{\hat{p}(y_{0:n})\}_{n\geq 0}$. This variance increases quadratically with $n$ for the particle method with MCMC moves instead of linearly as it does for state-space models with good mixing properties. This is once more due to the degeneracy problem; i.e. the implicit reliance of such algorithms on the particle estimates of the joint distributions $\{p(dx_{0:n}|\, y_{0:n})\}_{n\geq 0}$.

### 7.1.5 Key points

Make sure you understand:

- for simple HMMs how to implement PMMH, PFs with $\theta$ added to the state by adding noise or using MCMC steps or using fixed lag resampling ideas.
- the limitations of sequential Bayesian inference with PFs and the role of path degeneracy.
- why independent multiple runs are necessary to assess the performance of different algorithms.
- computational considerations and trade-offs when implementing PMCMC. The question often is whether one should use higher $N$ or more MCMC iterations. A simple initial approach would be to use $N$ in the order of $T$ and then spend more computation in MCMC steps.

### 7.1.6 Reading List

You can complement your reading on PMCMC by looking first at the comprehensive book chapter in [6] and then at [5]. For a general discussion on Bayesian inference and its challenges using particle methods you can look at Sections 6.1, 6.2.1 of [44].

## 7.2 Maximum Likelihood (ML) estimation

Computational methods to perform ML estimation can be either based on a) direct optimisation approaches, b) gradient ascent, and c) Expectation Maximisation (EM). In addition, similar to the Bayesian case, ML can be performed either offline or online. In both cases, gradient ascent and EM relies on computing smoothed additive functionals, so the particle smoothing methods presented in Section 6 are very important. In contrast to what we saw in the Bayesian case, certain on-line ML implementations using particle methods (and forward only smoothing implementations) manage to overcome path degeneracy and are numerically very stable. This results in them being very useful in many applications with very long data sequences. Here, for the sake of simplicity we will focus more on offline methods. Towards the end of the section we will sketch some ideas related to on-line approaches and point to relevant references.

In the off-line case one is given a batch of data-points $y_{0:T}$ and aims to estimate of $\theta$ as the maximising argument of the marginal log-likelihood of the observed data:

$$\theta^{\dagger} = \underset{\theta \in \Theta}{\arg\max} \;\; \ell_T(\theta) \tag{93}$$

where

$$\ell_T(\theta) = \log \; p_\theta(y_{0:T}). \tag{94}$$

We will see different ways on how to approximate $\theta^{\dagger}$ using particle approximations.

### 7.2.1 Direct Optimisation of log-likelihood

One can use the particle approximation $\hat{p}_\theta(y_{0:T})$ to estimate $\ell_T(\theta)$ as

$$\hat{\ell}_T(\theta) = \log \widehat{p_\theta(y_{0:T})}.$$

and then perform the following maximisation directly

$$\widehat{\theta} = \underset{\theta \in \Theta}{\arg\max} \;\; \hat{\ell}_T(\theta) \tag{95}$$

Various off-the-shelf optimisation methods can be employed, from using grid methods on $\theta$ to more elaborate such as the popular Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS).

This approach will face two challenges:

- the Monte Carlo variance of $\hat{p}_\theta(y_{0:T})$ might act as clutter on the maximising surface $\hat{\ell}_T(\theta)$, so $\hat{\theta}$ might be very different from $\theta^{\dagger}$.
- $\hat{p}_\theta(y_{0:T})$ might not be a smooth or continuous function of $\theta$.
- It is not be possible to maintain unbiasedness when transforming $\hat{p}_\theta(y_{0:T})$. This means:

  $$\mathbb{E}^N[\log \hat{p}_\theta(y_{0:T})] \neq \log p_\theta(y_{0:T})$$

  so even if $\hat{p}_\theta(y_{0:T})$ is unbiased, $\log \hat{p}_\theta(y_{0:T})$ is a biased estimator of the log-likelihood.

We will expand on each of these points separately.

Regarding the variance of $\hat{p}_\theta(y_{0:T})$, one can describe this particle approximation as a noisy evaluation of the true likelihood $p_\theta(y_{0:T})$

$$\hat{p}_\theta(y_{0:T}) = p_\theta(y_{0:T}) + \mathcal{V}_T^N(\theta)$$

with $\mathcal{V}_T^N(\theta)$ being some non-trivial zero mean noise (due to unbiased-ness) depending on $T, N$, $\theta$ (and the chosen model). In fact, we know that (non-asymptotic) variance of $\mathcal{V}_T^N$ that increases linearly with $T$ (with the proportionality constant being proportional to $p_\theta(y_{0:T})^2$). This Monte Carlo variability due $\mathcal{V}_T^N$ is quite an issue for finding maximum over $\theta$, because it can distort the surface of $p_\theta(y_{0:T})$ and it can be very hard to retrieve an estimate

$\hat{\theta}$ close to $\theta^\dagger$. Some brute-force fixes around this issue could include using very high values for $N$, smoothing the approximation as a function of $\theta$ (e.g. using some kernel smoothing method), using common random seeds for each $\theta$ evaluation and also averaging over independent runs for every $\theta$ value. Another costly possibility is to resort to multiple independent runs of PFs and then average over their results.

When optimising a function calculated with a Monte Carlo error, a popular strategy is to make the evaluated function continuous by using common random numbers over different evaluations to ease the optimization. Unfortunately, this strategy is not sufficient in the particle context. Indeed, in the resampling stage, a piece-wise constant and hence discontinuous cumulative distribution function (cdf) is defined by the weights $\{W_n^i\}_{i=1}^N$ and particles $\{X_n^i\}_{i=1}^N$. A small change in $\theta$ will cause a small change in the importance weights $\{W_n^i\}_{i=1}^N$ and this will potentially generate a different set of resampled particles. As a result, the likelihood function estimate $\widehat{\ell_T}(\theta)$ will not be continuous in $\theta$ even if the true likelihood is.

A solution to this problem was proposed in [39] by using an importance sampling method but it has computational complexity $\mathcal{O}\left(N^2(T+1)\right)$ and can only provide low variance likelihood estimates in the neighborhood of a suitably pre-selected parameter value. When $\mathcal{X} \subseteq \mathbb{R}$, an elegant solution to the discontinuity problem was proposed in [60]. The method based on common random numbers introduces a "continuous" version of the resampling step by ordering the particles $\{X_n^i\}_{i=1}^N$ and defining a piece-wise linear resampling cdf. This method requires $\mathcal{O}(N(T+1)\log N)$ operations due to the sorting of the particles. The resulting continuous estimate of the likelihood function can be maximized using standard optimisation techniques. An extension to the multivariate case when $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ (with $d_x > 1$) has been proposed in [50].

The next thing to address is how we correct or reduce the bias of $\log \hat{p}_\theta(y_{0:T})$. A simple approach is for a given random variable $Z$ to use a Taylor series around $Z'$:

$$\log(Z) = \log Z' + \frac{1}{Z'}(Z - Z') - \frac{1}{2Z'^2}(Z - Z')^2 + O(Z^3)$$

Let $Z' = \mathbb{E}[Z]$ and after ignoring the higher then second order terms, we get:

$$\mathbb{E}\left[\log(Z)\right] = \log \mathbb{E}[Z] - \frac{1}{2\mathbb{E}[Z]^2}\mathbb{V}ar^N[Z]$$

Setting above $Z = \hat{p}_\theta(y_{0:T})$ and $Z' = p_\theta(y_{0:T})$ we get:

$$\mathbb{E}\left[\log\ \hat{p}_\theta(y_{0:T})\right] = \log p_\theta(y_{0:T}) - \frac{\mathbb{V}ar^N\left[\hat{p}_\theta(y_{0:T})\right]}{2p_\theta(y_{0:T})^2}.$$

This means that $\log\ \hat{p}_\theta(y_{0:T}) + \frac{1}{2}\mathbb{V}ar^N\left[\frac{\hat{p}_\theta(y_{0:T})}{p_\theta(y_{0:T})}\right]$ will exhibit a reduced bias coming only from the higher order terms (we are using a second order bias reduction scheme). To implement such as scheme in practice we need to approximate the relative variance $\frac{\mathbb{V}ar^N[\hat{p}_\theta(y_{0:T})]}{2p_\theta(y_{0:T})^2}$, whose accurate estimation is a very difficult task. One simple (but quite crude) possibility would be to use an expression like (55) and let

$$\begin{aligned}
\mathbb{V}ar^N\left[\frac{\hat{p}_\theta(y_{0:T})}{p_\theta(y_{0:T})}\right] &\approx \frac{1}{N}\left(\int \frac{\left(p(x_{0:T}|y_{0:T})\right)^2}{q(x_{0:T})}dx_{0:T} - 1\right) \\
&\approx \frac{1}{N}\left(\int \left(\prod_{n=0}^{T} w_n(x_{n-1:n})\right) p(x_{0:T}|y_{0:T})dx_{0:T} - 1\right)
\end{aligned}$$

Then one may use the particle approximation of

$$\hat{W}_T = \frac{1}{N}\sum_{i=1}^{N}\left(\prod_{n=0}^{T} w_n(\bar{X}_{n-1:n}^i)\right)$$

and use

$$\log\ \widehat{p_\theta(y_{0:T})} = \log\ \hat{p}_\theta(y_{0:T}) + \frac{\hat{W}_T - 1}{2N}$$

as a bias reduced estimator for $\ell_T$. The downside of this is that we are overestimating the variance of $\hat{p}_\theta(y_{0:T})$ by using (55) that is an identity for SIS. A refinement of this approach that works around this issue has appeared recently in [51], but its presentation is beyond the scope of this course.

### 7.2.2 Expectation Maximisation

Expectation Maximisation (EM) algorithm is a very popular procedure for maximising $\ell_T(\theta)$. The procedure is presented in Algorithm (19). The sequence $\{\ell_T(\theta_k)\}_{k\geq 0}$ generated by this algorithm is non-decreasing.

**Algorithm 19** Expectation Maximisation (EM) algorithm

At iteration $k+1$, we set

$$\theta_{k+1} = \arg\max_{\theta} \; Q(\theta_k, \theta) \tag{96}$$

where

$$Q(\theta_k, \theta) = \int \log p_\theta(x_{0:T}, y_{0:T}) \; p_{\theta_k}(x_{0:T}|y_{0:T}) dx_{0:T}. \tag{97}$$

In particular if $p_\theta(x_{0:T}, y_{0:T})$ belongs to the exponential family, then the EM consists of computing a $d_s$-dimensional summary statistic, $\mathcal{S}_T^\theta$, that is a smoothed additive functional of the following form

$$\mathcal{S}_T^\theta = \int \left[\sum_{k=0}^{T} s_k\left(x_k, x_{k-1}, y_k\right)\right] p_\theta\left(x_{0:T} \middle| y_{0:T}\right) dx_{0:T}, \tag{98}$$

Then the maximising argument of $Q(\theta_k, \theta)$ can be characterised explicitly through a suitable function $\Lambda : \mathbb{R}^{d_s} \to \Theta$, i.e.

$$\theta_{k+1} = \Lambda\left(\mathcal{S}_T^{\theta_k}\right). \tag{99}$$

A particle version of EM uses instead $\theta_{k+1} = \Lambda\left(\widehat{\mathcal{S}_T^{\theta_k}}\right)$, where $\widehat{\mathcal{S}_T^{\theta_k}}$ is a particle approximation (ideally using particle smoothing). We have already discussed in Section 6 in detail the options and issues that result when designing particle approximations for $\mathcal{S}_T^\theta$. The simplest particle implementation of EM consists running for each $k$ a standard PF such as Algorithm 8 and approximating $\mathcal{S}_T^{\theta_k}$ using:

$$\begin{aligned}\widehat{\mathcal{S}_T^\theta} &= \int \left[\sum_{k=0}^{T} s_k\left(x_k, x_{k-1}, y_k\right)\right] \widehat{p}_\theta\left(dx_{0:T} \middle| y_{0:T}\right), \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{k=0}^{T} s_k\left(\bar{X}_k^i, \bar{X}_{k-1}^i, y_k\right).\end{aligned}$$

The issue with doing this is that due to path degeneracy the asymptotic variance of the SMC estimate satisfies $\mathbb{V}ar^N\left(\widehat{\mathcal{S}_n^\theta}\right) \geq D_\theta \frac{n^2}{N}$, so this might cause some issues in cases where $T$ is large. This will be addressed using the dedicated smoothing algorithms we presented in Section 6, for which this increase in the variance is only linear. As a result, we recommend using particle smoothing techniques even if this comes at a certain expense in terms of computational cost.

### 7.2.3 Gradient ascent methods

The log-likelihood may be maximised with the following steepest ascent algorithm: a at iteration $k+1$

$$\theta_{k+1} = \theta_k + \gamma_{k+1} \; \nabla_\theta \ell_T(\theta)|_{\theta=\theta_k}, \tag{100}$$

where $\nabla_\theta \ell_T(\theta)|_{\theta=\theta_k}$ is the gradient of $\ell_T(\theta)$ w.r.t $\theta$ evaluated at $\theta = \theta_k$ and $\{\gamma_k\}$ is a sequence of small positive real numbers, called the step-size sequence, that needs to satisfy $\sum_k \gamma_k = \infty$ and $\sum_k \gamma_k^2 < \infty$. To obtain the *score* vector $\nabla_\theta \ell_T(\theta)$ we can use Fisher's identity

$$\nabla_\theta \ell_T(\theta) = \int \nabla_\theta \log p_\theta\left(x_{0:T}, y_{0:T}\right) \; p_\theta\left(x_{0:T}|y_{0:T}\right) dx_{0:T}. \tag{101}$$

Given (18), it is easy to check that the score is of the form (98). We have

$$\begin{aligned}\nabla_\theta \log p_\theta\left(x_{0:n}, y_{0:n}\right) &= \nabla_\theta \log \prod_{p=0}^{n} f_\theta\left(x_p \middle| x_{p-1}\right) g_\theta\left(y_p \middle| x_p\right) \\ &= \sum_{p=0}^{n} (\nabla \log f_\theta\left(x_p \middle| x_{p-1}\right) + \nabla \log g_\theta\left(y_p \middle| x_p\right))\end{aligned}$$

where here

$$s_p(x_{p-1:p}) = \nabla \log f_\theta\left(x_p \middle| x_{p-1}\right) + \nabla \log g_\theta\left(y_p \middle| x_p\right)$$

and we denote $\eta_\theta$ as $f_\theta(x_0|x_{-1})$. As a result, the discussion for EM above applies also here. One should use particle approximations to estimate $\mathcal{S}_n^\theta$ that avoid path degeneracy and the corresponding variance increase with $n$, which can become more pronounced here due to using gradients in $s_k$. This means it is recommended using either very high values for $N$ or the smoothing algorithms of 6.

### 7.2.4 Discussion

We have looked at how filtering and particle methods can be used for ML estimation using direct optimisation approaches, gradient ascent, and EM. From the point of view of practitioners, direct optimisation is usually very convenient, but we saw it has some weaknesses. Notice that Monte Carlo variance affects the estimation even when one aims to optimise for $p_\theta(y_{0:T})$ instead of its log-function. Using log transforms typically results in functions that much easier to optimise, but there are some issues related to bias reduction. Implementation of gradient ascent and EM using particle approximations is straightforward and the smoothing algorithms of 6 should be prefered. Gradient ascent algorithms can be sometimes numerically unstable as they require to scale carefully the components of the score vector. Hence, EM is usually favoured by practitioners whenever it is applicable as it is numerically more stable than gradient techniques, but both can result in very good results.

## 7.3 Further topics on on-line parameter inference

### 7.3.1 On-line likelihood methods

For a long observation sequence computing the gradient of $\ell_T(\theta)$ or performing EM iterations can be computationally prohibitive. In other cases, we might be interested to know how the parameter estimate changes with time. In this case we should use a recursive method that updates at each $n$ the ML estimate, $\theta_n$, of the model parameter after receiving the new data $y_n$. For these methods to be theoretically justified, it is crucial for the observation process to be stationary for the limiting averaged likelihood function $\ell_T(\theta)/T$ to have a well-defined limit as $T \to +\infty$.

An online gradient algorithm bypassing this problem has been proposed in the literature for a finite state-space latent process in [52], but can easily be extended for general state spaces using particle methods. It relies on the following update scheme

$$\theta_{n+1} = \theta_n + \gamma_{n+1} \nabla \log p_{\theta_{0:n}}(y_n|y_{0:n-1}) \tag{102}$$

where the subscript $\theta_{0:n}$ is used to denote the 'time-varying' score which is computed with a filter using the parameter $\theta_p$ at time $p$ (to avoid having to compute the filter from time 0 to time $n$ using the current parameter value $\theta_n$). As in the off-line case, the Fisher identity (101) can illustrate how particle approximate can be used to compute $\nabla p_{\theta_{0:n}}(y_n|y_{0:n-1})$. A method that implements this using a forward only version of smoothing is proposed in [65]. The resulting method has a computational cost proportional to $N^2T$, but using it is essential for accurate estimates of $\theta_n$.

We return to EM and mention some ideas on how it can be implemented on-line. At time $n$, let $\{\theta_p\}_{0\le p\le n}$ be the sequence of earlier parameter estimates of the on-line EM algorithm computed sequentially based on $y_{0:n-1}$. When $y_n$ is received, we can use stochastic approximation (or Robbins Monroe, [8]) to update the approximation the smoothed additive functional in (78) as follows:

$$\begin{aligned}\mathcal{S}_{\theta_{0:n}} &= \gamma_{n+1} \int s_n\left(x_{n-1:n}\right) p_{\theta_{0:n}}(x_{n-1}, x_n|y_{0:n}) dx_{n-1:n} \\ &+ (1-\gamma_{n+1}) \sum_{k=0}^{n} \left(\prod_{i=k+2}^{n} (1-\gamma_i)\right) \gamma_{k+1} \int s_k\left(x_{k-1:k}\right) p_{\theta_{0:k}}(x_{k-1:k}|y_{0:k}) dx_{k-1:k},\end{aligned} \tag{103}$$

where $\{\gamma_n\}_{n\ge1}$ are standard step sizes that satisfy $\sum_n \gamma_n = \infty$ and $\sum_n \gamma_n^2 < \infty$. Then a particle approximation of $\mathcal{S}_{\theta_{0:n}}$ can be used in the standard maximisation step (99) to provider $\theta_{n+1}$; see [23] or [9] for details. One could either use a standard PF with many particles or a forward only implementation of FFBSm found in [23]. The bias variance trade-off mentioned earlier will then apply for the estimates $\theta_n$. We will illustrate this bias variance trade-off in the example below.

### 7.3.2 Numerical Results for On-line implementation of EM

We will continue the numerical case study in Section 6.5.1 to see how the bias and variance of the estimates $\mathcal{S}_n^\theta$ can affect ML estimation, when the former are used within an on-line EM algorithm. For the model in (92) the

E-step corresponds to computing $\mathcal{S}_n^\theta$ where $s_k(x_{k-1}, x_k, y_k) = \left((y_k - x_k)^2, x_{k-1}^2, x_{k-1}x_k, x_k^2\right)$ and the M-step update function is given by $\Lambda(z_1, z_2, z_3, z_4) = \left(\frac{z_3}{z_2}, z_4 - \frac{z_3^2}{z_2}, z_1\right)$. We will compare the estimates of $\theta^*$ when the E-step is computed using the $\mathcal{O}(N)$ and the $\mathcal{O}(N^2)$ methods described in the previous section with $150^2$ and 150 particles respectively. We will also contrast these results with the output of the ideal on-line EM algorithm based on KF type calculations for the smoothing problem. A simulated data-set for $\theta^* = (\rho^*, \tau^*, \sigma^*) = (0.8, 1, .2)$ will be used. In both cases we will initialize the algorithm using $\theta_0 = (0.1, 0.1, 0.2)$ and assume $\sigma^*$ is known. In Figures 13 and 14 we present the the results obtained using 100 independent replications of the on-line EM algorithm, where the step size is set as $\gamma_n = n^{-0.8}$ and for the first 5000 iterations no M-step update is performed.

Both methods yield fairly accurate results. Experimentally, the properties of the estimates of $\mathcal{S}_n^\theta$ discussed earlier appear to translate into properties of the resulting parameter estimates: the $\mathcal{O}(N)$ method provides estimates with less bias but more variance than the $\mathcal{O}(N^2)$ method.

### 7.3.3 The SMC$^2$ algorithm

We return to sequential Bayesian estimation. Earlier in Section 5.4 we mentioned how particle methods can be used to approximate sequentially $\{p(\theta|\, y_{0:n})\}_{n\geq 0}$. The SMC$^2$ algorithm proposed in [18] may be considered as the particle equivalent of PMCMC. It mimics the "ideal" particle algorithm proposed in [15] (presented here in Section 5.4) where the $N_\theta$ particles (in the $\theta$-space) at time $n$ are re-weighted according to $p_\theta(y_{0:n+1})/p_\theta(y_{0:n})$ at time $n+1$. As these likelihood terms are unknown, we substitute to them $\hat{p}_\theta(y_{0:n+1})/\hat{p}_\theta(y_{0:n})$ where $\hat{p}_\theta(y_{0:n})$ is a particle approximation of the marginal likelihood $p_\theta(y_{0:n})$, obtained by a running a particle filter of $N_x$ particles in the $x-$dimension, up to time $n$, for each of the $\theta-$particles. When particle degeneracy (in the $\theta-$dimension) reaches a certain threshold, $\theta-$particles are refreshed through the succession of a resampling step, and a MCMC step, which in these particular settings takes the form of a PMCMC update. The cost per iteration of this algorithm is not constant, therefore it cannot be used in truly on-line scenarios. Yet there are practical situations where it may be very useful to approximate jointly all the posteriors $p(\theta|y_{0:n})$, for $1 \leq n \leq T$, for instance to assess the predictive power of the model.

## 7.4 Summary

### 7.4.1 Key points

- Path degeneracy is a bottleneck for parameter estimation of HMMs with particle methods. It manifests as increasing Monte Carlo variance with time of certain sufficient statistics that are used for Gibbs update steps in the Bayesian case and EM or gradient updates for the likelihood inference case.
- PMCMC is a very robust and easy to use tool for Bayesian off-line inference. It can be used in a similar manner to standard MCMC algorithms and has very good performance results. The theoretical properties of PMCMC algorithm is currently very active topic topic of study in Statistics and Applied Probability.
- There is not a fully satisfying solution for online Bayesian inference. This remains largely an open problem. All proposed methods will be affected by path degeneracy when a computational cost of order $\mathcal{O}(N)$ per time is used. They can be quite easy to implement and can produce good results in some cases especially for short $T$ or informative priors, but it is advised to be used with caution.
- For on-line likelihood methods, there there are well-performing and numerically stable methods. One can implement using forward only implementations of FFBSm for either gradient or EM approaches [65, 23]. For the on-line gradients using this is crucial in terms of performance, see [65] for details. For on-line EM, one could also use a standard PF with many particles. The bias variance trade-off mentioned earlier for $\widehat{\mathcal{S}}_n^\theta$ will then apply also for $\theta_n$ estimates.

### 7.4.2 Reading List

A reference with more material for this section is [44]. The numerical examples are taken from there and this section is aimed as an introductory version of the material contained there.

### 7.4.3 Homework

Implement some the examples in this Section and [44] and try to validate some of the conclusions mentioned here. The code used for [44] can be found in `http://wwwf.imperial.ac.uk/~nkantas/code_stat_science.zip`

# 8 Concluding remarks

We have reviewed the various algorithms: IS, MCMC, SMC and combinations of them. Following a fairly traditional route we have presented these methods for HMMs and moved from solving the filtering problem to various ways of estimating static parameters in general HMMs. Most methods proposed originally in the literature suffered from the path degeneracy problem, so several smoothing algorithms have been proposed to deal with this problem. The methodology presented here is also relevant to other statistical problems that are not HMMs; we briefly discussed how SMC can be a general alternative to MCMC in Section 5.4.

These days most applications require solutions and answers that can be addressed only via some form of numerical methods and approximations. Simulation based methods and Monte Carlo are a mature topic that contains a rich and generic toolbox that is well understood. One should see all the methods presented in this course and others beyond as complementary and each meant for specific cases and necessarily to antagonise each other. Compared to previous decades, nowadays applications tend to be more demanding due to more complex and sophisticated modelling with larger number of variables and data points of various type. Demanding problems require often custom algorithmic solutions and in this spirit one should be creative in combining known methods, devising new ones and extending what has appeared already in the literature. This requires understanding of strengths and weaknesses of each method. In this course we tried to emphasise, both pros and cons. Computational Statistics is very active research area with many problems waiting to be solved from a both computational and theoretical point of view.

# References


[1] Alspach, D. and Sorenson, H. (1972). Nonlinear Bayesian estimation using Gaussian sum approximations, *IEEE Trans. Autom. Control*, 17(4), 439 - 448.

[2] Andrieu, C., De Freitas, J.F.G. and Doucet, A. (1999). Sequential MCMC for Bayesian model selection. *Proc. IEEE Workshop Higher Order Statistics*, 130–134.

[3] Andrieu, C., Doucet, A. and Tadić, V. B. (2005). On-line parameter estimation in general state-space models. *Proc. 44th IEEE Conf. on Decision and Control*, 332–337.

[4] Andrieu, C., & Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. The *Annals of Statistics*, 697-725.

[5] Andrieu, C., Doucet, A. and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *J. Royal Stat. Soc.* B (with discussion), 72, 269-342.

[6] Andrieu, C., Doucet, A., & Holenstein, R. (2009). Particle Markov chain Monte Carlo for efficient numerical simulation. In *Monte Carlo and quasi-Monte Carlo methods 2008* (pp. 45-60). Springer Berlin Heidelberg.

[7] Arasaratnam, I., & Haykin, S. (2009). Cubature kalman filters. *IEEE Transactions on Automatic Control*, 54(6), 1254-1269.

[8] Benveniste, A., Métivier, M. and Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximation*. New York: Springer-Verlag.

[9] Cappé, O. (2011). Online EM Algorithm for Hidden Markov Models, *J. Comput. Graph. Statist.,* 20, 728-749.

[10] Cappé, O., Moulines, E. and Rydén, T. (2005). *Inference in Hidden Markov Models*. New York: Springer-Verlag.

[11] Carpenter, J., Clifford, P. and Fearnhead, P. (1999). An improved particle filter for non-linear problems. *IEE proceedings - Radar, Sonar and Navigation*, 146, 2-7.

[12] Carvalho, C., Johannes, M., Lopes H. and Polson, N. (2010). Particle learning and smoothing. *Stat. Science*, 25, 88-106.

[13] Cérou, F., Del Moral, P. and Guyader, A. (2011). A non asymptotic variance theorem for unnormalized Feynman-Kac particle models. *Annales de l'Institut Henri Poincaré*, 47, 629-649.

[14] Chen, R. and Liu, J.S. (2000). Mixture Kalman filters. *J. Royal Stat. Soc.* B, 62, 493–508.

[15] Chopin, N. (2002). A sequential particle filter method for static models, *Biometrika* 89, 539-552.

[16] Chopin, N. (2004). Central limit theorem for sequential Monte Carlo and its application to Bayesian inference. *Ann. Statist.*, 32, 2385-2411.

[17] Chopin, N., Iacobucci, A. Marin, J.M. Mengersen, K., Robert, C.P., Ryder R., Schäfer C. (2011). On particle learning. In *Bayesian Statistics* 9 (Bernardo et al. eds), Oxford University Press, 317-360.

[18] Chopin, N., Jacob, P. and Papaspiliopoulos, O. (2013). SMC$^2$: A sequential Monte Carlo algorithm with particle Markov chain Monte Carlo updates. *J. Royal Stat. Soc.* B, 75, 397-426.

[19] Chopin, N., & Papaspiliopoulos, O. (2020). *An introduction to sequential Monte Carlo*, Springer Series in Statistics.

[20] Cotter, S. L., Roberts, G. O., Stuart, A. M., & White, D. (2013). MCMC methods for functions: modifying old algorithms to make them faster. *Statistical Science*, 424-446.

[21] Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. New York: Springer-Verlag.

[22] Del Moral, P., Doucet, A., & Jasra, A. (2006). Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3), 411-436.

[23] Del Moral, P., Doucet, A. and Singh. S.S. (2009). Forward smoothing using sequential Monte Carlo. Technical report 638, CUED-F-INFENG, Cambridge University, arXiv:1012.5390v1.

[24] Del Moral, P., & Murray, L. M. (2015). Sequential Monte Carlo with highly informative observations. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1), 969-997.

[25] Douc, R., Cappé, O., Moulines, E. (2005). Comparison of resampling schemes for particle filtering. I*n Proceedings of the 4th IEEE International Symposium In Image and Signal Processing and Analysis ISPA 2005*, pp. 64-69).

[26] Douc, R., Garivier, A., Moulines, E. and Olsson, J. (2011). Sequential Monte Carlo smoothing for general state space hidden Markov models. *Annals Applied Probab.*, 21, 2109-2145.

[27] R. Douc, E. Moulines, & D. S. Stoffer (2014). *Nonlinear Time Series Theory, Methods and Applications with R Examples*, CRC Press.

[28] Doucet, A., De Freitas, J.F.G. and Gordon N.J. (eds.) (2001). *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag.

[29] Doucet, A., Godsill, S. J. and Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Stat. Comput.*, 10, 197–208.

[30] Doucet, A. and A.M. Johansen (2011). A tutorial on particle filtering and smoothing: fifteen years later. In *Oxford Handbook of Nonlinear Filtering*, Oxford University Press.

[31] Doucet, A., Pitt, M. K., Deligiannidis, G., & Kohn, R. (2015). Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2), 295-313.

[32] Fearnhead, P. (2002). MCMC, sufficient statistics and particle filters. *J. Comp. Graph. Statist.*, 11, 848–862.

[33] Gelfand, A. E., & Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410), 398-409.

[34] Gilks, W. R. and Berzuini, C. (2001). Following a moving target - Monte Carlo inference for dynamic Bayesian models. *J. Royal Stat. Soc.* B, 63, 127-146.

[35] Godsill, S., & Clapp, T. (2001). Improvement strategies for Monte Carlo particle filters. In [28].

[36] N.J. Gordon, D. Salmond and A.F.M. Smith (1993) Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proc. F.*

[37] J. E. Handschin and D. Q. Mayne (1966). Monte Carlo Techniques to Estimate the Conditional Expectation in Multistage Nonlinear Filtering, *Int. J of Control.*

[38] Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1), 97-109.

[39] Hürzeler, M. and Künsch, H.R. (2001). Approximation and maximising the likelihood for a general state-space model. In [28].

[40] Jacob, P. E., Murray, L. M., & Rubenthaler, S. (2015). Path storage in the particle filter. *Statistics and Computing*, 25(2), 487-496.

[41] Julier, S. J., & Uhlmann, J. K.(1997). A new extension of the Kalman Filter to nonlinear systems. *In SPIE proceedings,* series (pp. 182-193).

[42] Julier, S. J., & Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3), 401-422.

[43] Kantas, N., Beskos, A., & Jasra, A. (2014). Sequential Monte Carlo Methods for High-Dimensional Inverse Problems: A Case Study for the Navier--Stokes Equations. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1), 464-489.

[44] Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., & Chopin, N. (2015). On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3), 328-351.

[45] Johansen, A.M. and Doucet, A. (2008). A note on auxiliary particle filters. *Statist. Probab. Letters*, 78, 1498–1504.

[46] Kim, S., Shephard, N. and Chib, S. (1998). Stochastic volatility: likelihood inference and comparison with ARCH models. *R. Econ. Studies*, 65:3, 361–393.

[47] Kitagawa, G. (1998). A self-organizing state-space model. *J. Am. Statist. Ass.*, 93, 1203-1215.

[48] Kitagawa G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state-space models. *J. Comput. Graph. Statist.*, 5, 1-25.

[49] Kitagawa, G. and Sato, S. (2001). Monte Carlo smoothing and self-organising state-space model. In [28], 178–195. New York: Springer.

[50] Lee, A. (2008). Towards smoother multivariate particle filters. M.Sc. Thesis, University of British Columbia.

[51] Lee, A., & Whiteley, N. (2018). Variance estimation in the particle filter. *Biometrika*, 105(3), 609-625.

[52] Le Gland, F. and Mevel, M. (1997). Recursive identification in hidden Markov models. *Proc. 36th IEEE Conf. Decision and Control*, 3468-3473.

[53] Lindsten, F, Jordan, M.I. and Schön, T.B. (2014). Particle Gibbs with ancestor sampling. *J. Machine Learning Res.*

[54] Liu J. and West M. (2001). Combined parameter and state estimation in simulation-based filtering, In [28].

[55] Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*. New York: Springer-Verlag.

[56] Liu J.S. and Chen R. (1998). Sequential Monte Carlo methods for dynamic systems. *J. Am. Statist. Ass.*, 93, 1032-1044.

[57] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6), 1087-1092.

[58] Olsson, J., Cappé, O., Douc, R. and Moulines, E. (2008). Sequential Monte Carlo smoothing with application to parameter estimation in non-linear state space models. *Bernoulli*, 14, 155-179.

[59] Paninski, L., Ahmadian, Y., Ferreira, D.G., Koyama, S., Rad. S.R, Vidne, M., Vogelstein, J. and Wu, W. (2010). A new look at state-space models for neural data. *J. Computational Neuroscience*, 29, 107-126.

[60] Malik, S., & Pitt, M. K. (2011). Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, 165(2), 190-209.

[61] Olsson, J., & Westerborn, J. (2017). Efficient particle-based online smoothing in general hidden Markov models: the PaRIS algorithm. *Bernoulli*, 23(3), 1951-1996.

[62] Pitt, M.K. and Shephard, N. (1999). Filtering via simulation: auxiliary particle filter. *J. Am. Statist. Ass.*, 94, 590-599.

[63] Pitt, M. K., dos Santos Silva, R., Giordani, P., & Kohn, R. (2012). On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2), 134-151.

[64] Polson, N.G., Stroud J.R. and Müller P. (2008). Practical filtering with sequential parameter learning. *J. Royal Stat. Soc.* B, 70, 413-428.

[65] Poyiadjis, G., Doucet, A. and Singh, S.S. (2011). Particle approximations of the score and observed information matrix in state-space models with application to parameter estimation. *Biometrika*, 98, 65-80.

[66] Robert, C. P., & Casella, G. (2004). *Monte Carlo Statistical Methods,* Springer, New York, Chicago.

[67] Runggaldier and Fabio Spizzichino (2001) Sufficient conditions for finite dimensionality of filters in discrete time: a Laplace transform-based approach, *Bernoulli*.

[68] S. Sarkka (2013) *Bayesian filtering and smoothing*, CUP Cambridge, 2013.

[69] Smith, A. F., & Roberts, G. O. (1993). Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, 3-23.

[70] Storvik, G. (2002). Particle filters in state space models with the presence of unknown static parameters. *IEEE. Trans. Signal Proc.*, 50, 281–289.

[71] Tierney, L. (1994). Markov chains for exploring posterior distributions. *Annals of Statistics*, 1701-1728.

[72] Van Dyk, D. A., & Park, T. (2008). Partially collapsed Gibbs samplers: Theory and methods. *Journal of the American Statistical Association,* 103(482), 790-796.

[73] Vidoni (1999) Exponential family state space models based on a conjugate latent process, *J. Royal Stat. Soc.* B.

[74] Vidoni and Ferrante (1998) Finite dimensional filters for nonlinear stochastic difference equations with multiplicative noises, *Stochastic Processes and Applications.*

[75] West, M. and Harrison, P.J. (1997). *Bayesian Forecasting and Dynamic Models.* New York: Springer-Verlag.

[76] Whiteley, N. (2013). Stability properties of some particle filters. *The Annals of Applied Probability*, 23(6), 2500-2537.

[77] Wilkinson, D.J. (2012). *Stochastic Modelling for Systems Biology.* CRC Press, 2nd edition.

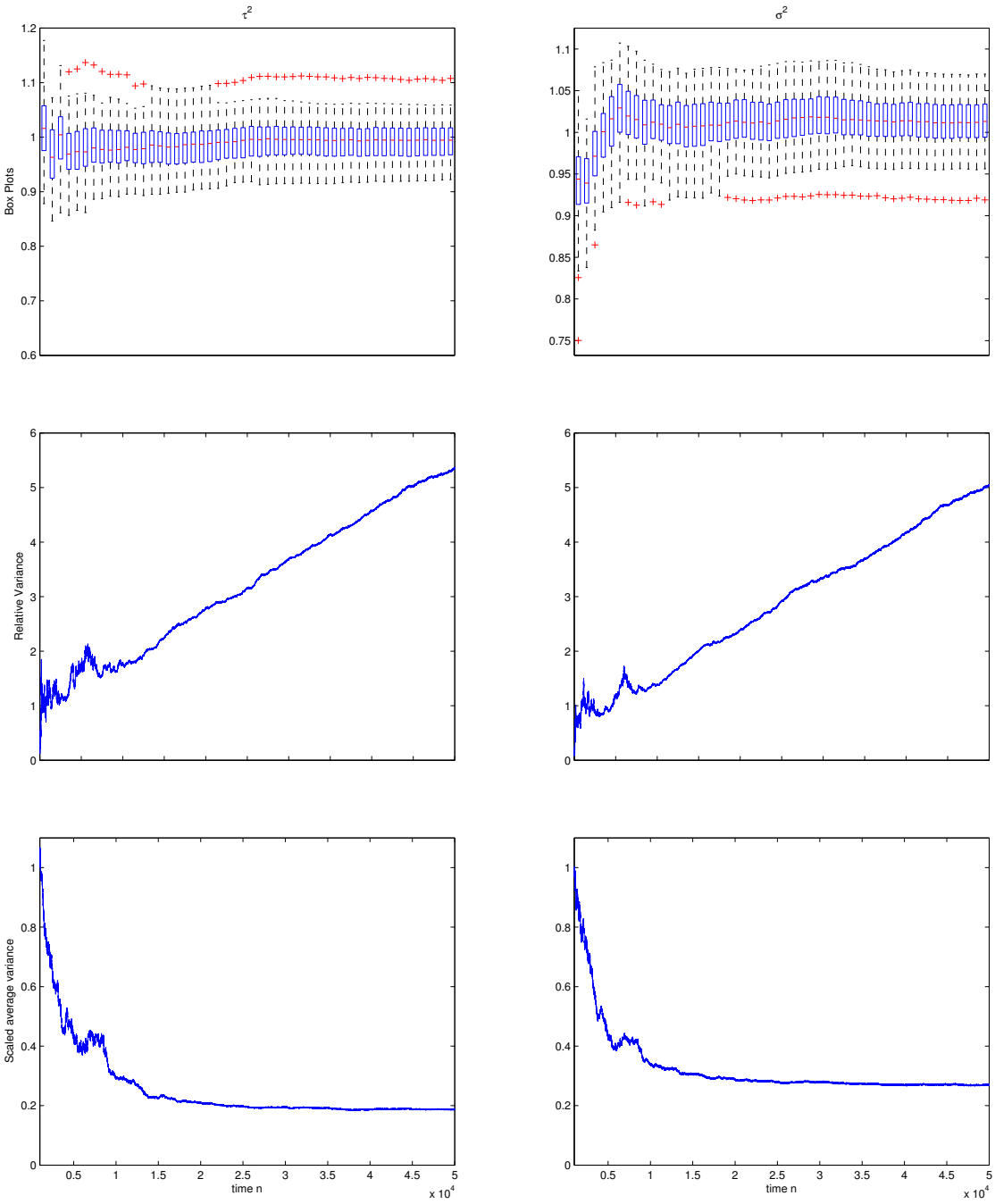


Figure 9: Top row: box plots for estimates of posterior mean at $n = 1000, 2000, \ldots 50000$; middle row: relative variance, i.e. empirical variance (over 100 different independent runs) for the estimator of the mean of $p(\theta|y_{0:n})$ using particle method with MCMC normalized with the true posterior variance computed using Kalman filtering on a grid; bottom row: average (over 100 different independent runs) of the estimated variance of $p(\theta|y_{0:n})$ using particle method with MCMC normalized with the true posterior variance. Left column: plots for $\tau^2$; right: plots for $\sigma^2$. All plots are computed using $N = 5000$.

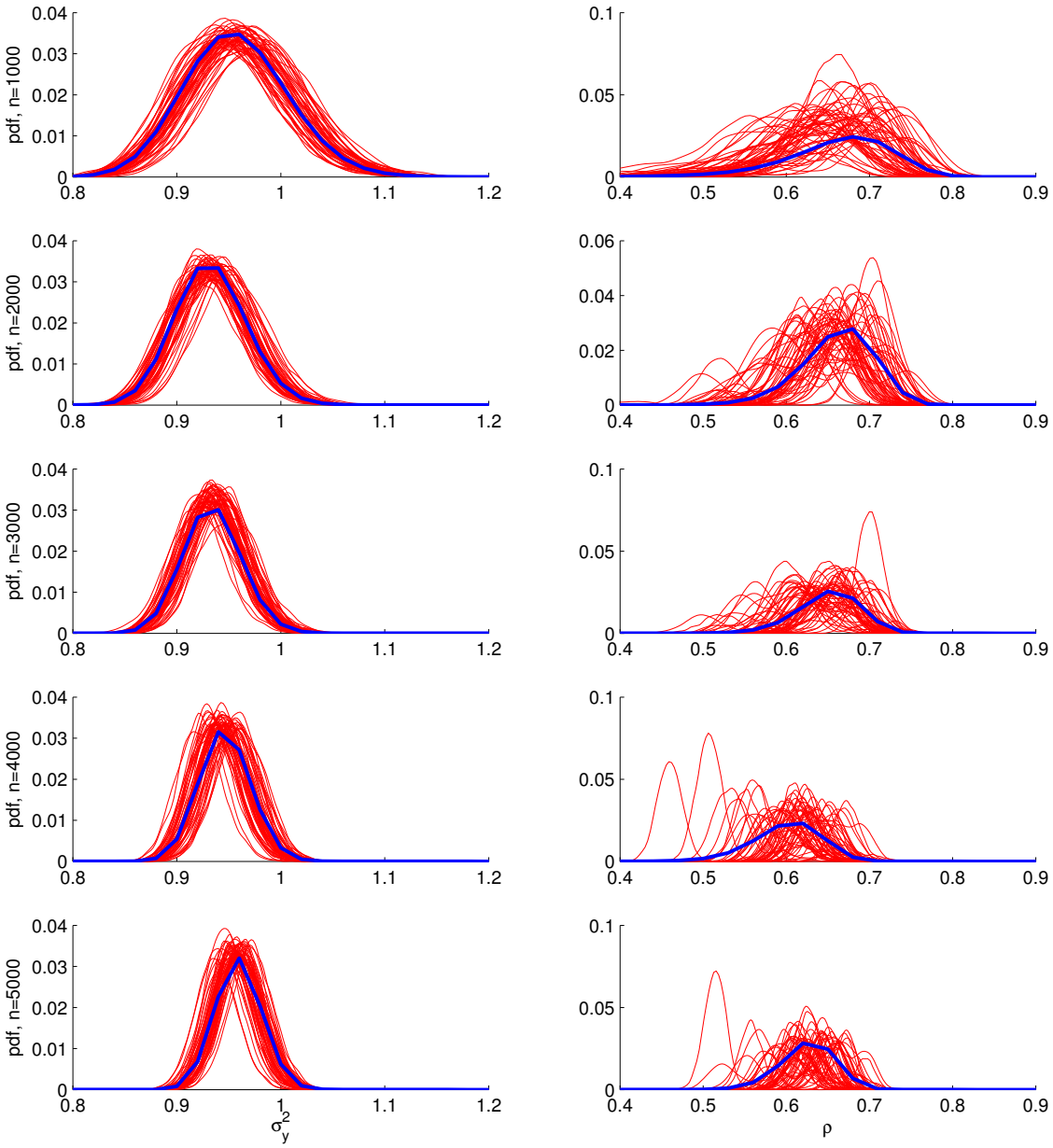


Figure 10: Particle method with MCMC, $\theta = (\rho, \sigma^2)$; estimated marginal posterior densities for $n = 10^3, 2 \times 10^3, \ldots, 5 \times 10^3$ over 50 runs (red) versus ground truth (blue).

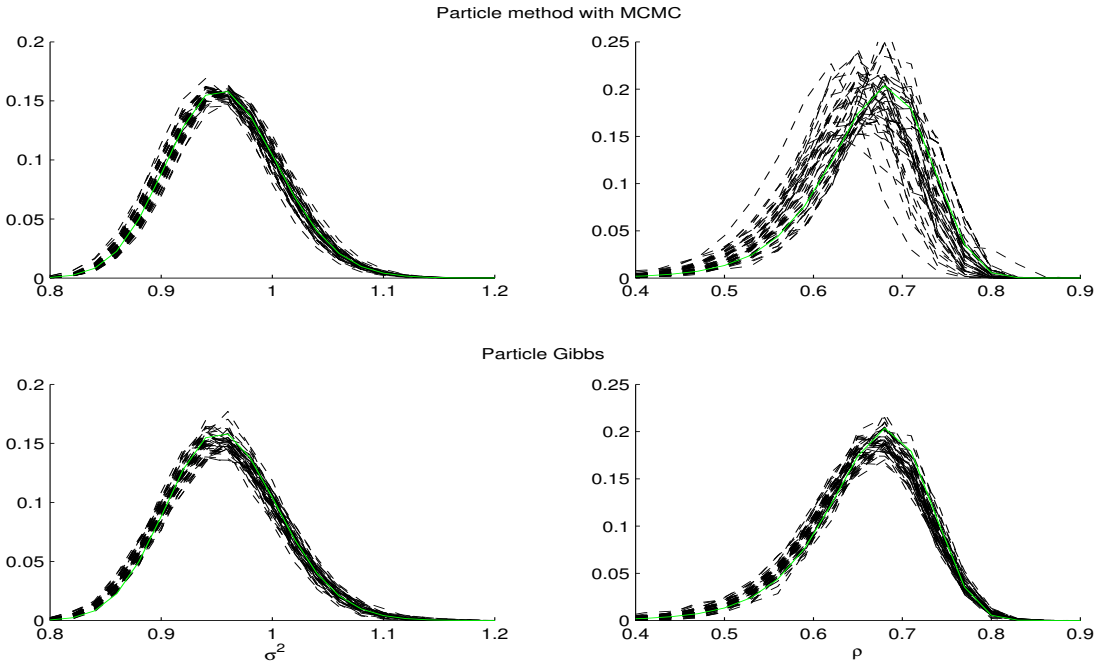


Figure 11: Estimated marginal posterior densities for $\theta = (\rho, \sigma^2)$ with $T = 10^3$ over 50 runs (black-dotted) versus ground truth (green). Top: Particle method with MCMC, $N = 7.5 \times 10^4$. Bottom: Particle Gibbs with 3000 iterations and $N = 50$.

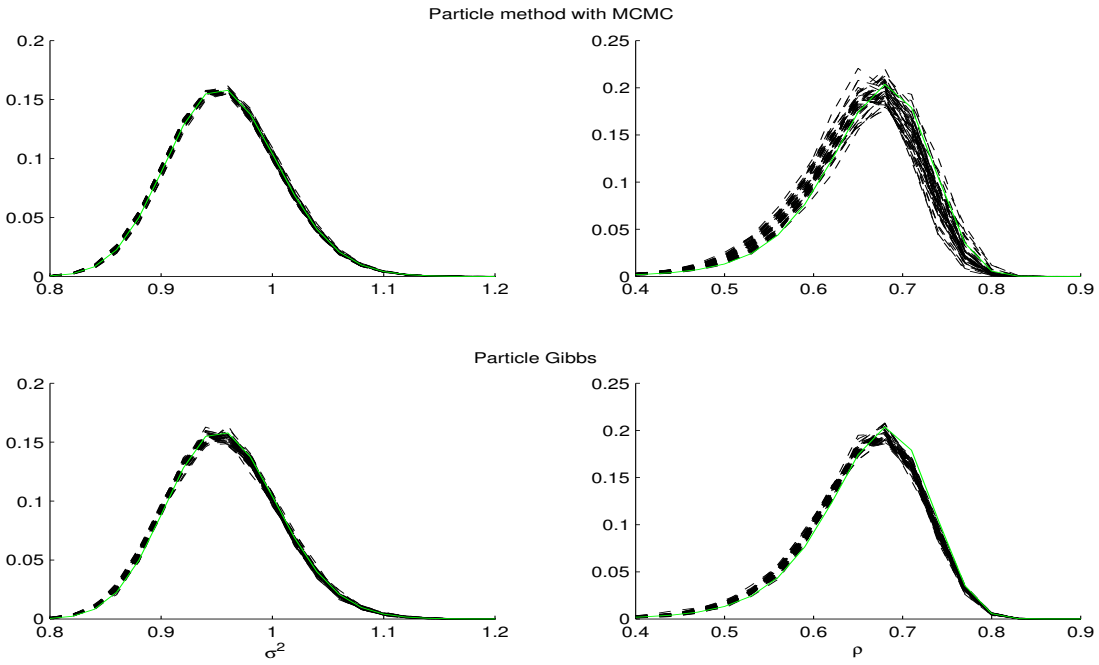


Figure 12: Estimated marginal posterior densities for $\theta = (\rho, \sigma^2)$ with $T = 10^3$ over 50 runs (black-dotted) versus ground truth (green). Top: Particle method with MCMC, $N = 6 \times 10^5$. Bottom: Particle Gibbs with 24000 iterations and $N = 50$.

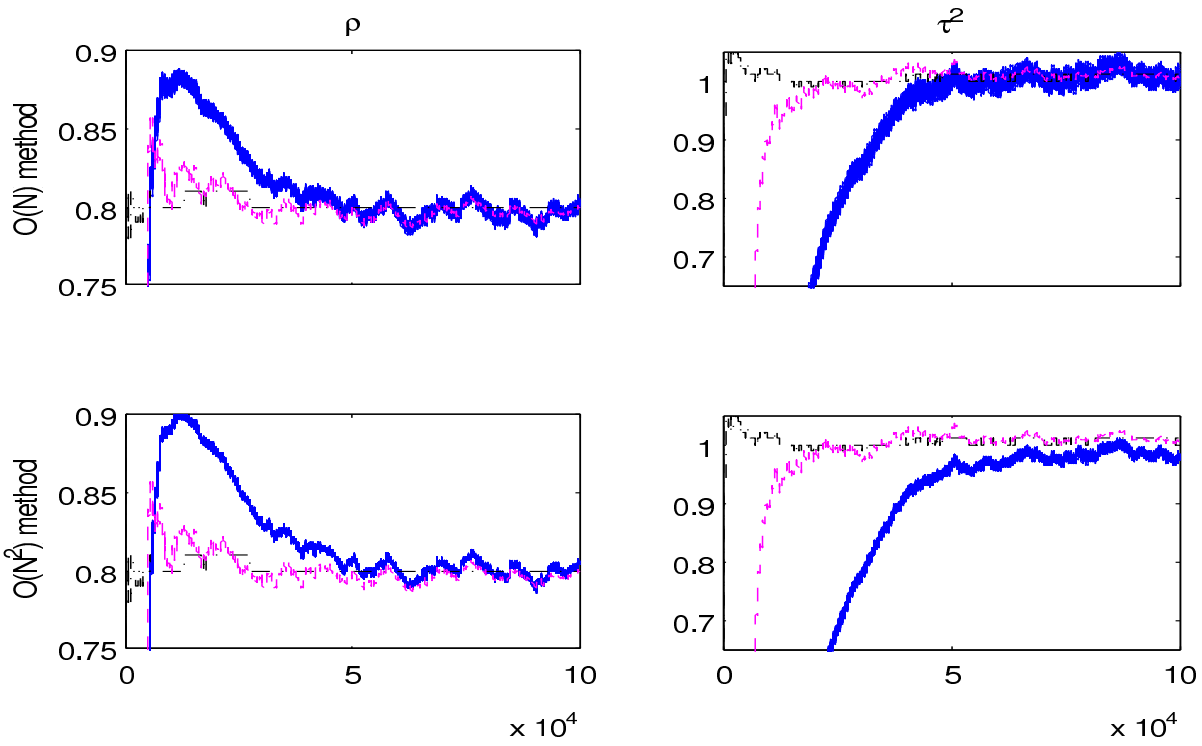


Figure 13: On-line EM: Trace of $\hat{\theta}_n$ for each of the 100 realizations of the algorithms (blue). Left: $\hat{\rho}_n$, Right: $\hat{\tau}_n^2$. Top row: $\mathcal{O}(N)$ method using $150^2$ particles. Bottom row: $\mathcal{O}(N^2)$ method using 150 particles. We also plot also the exact ML estimate at time $n$ obtained using Kalman filtering on a grid (black) as well as the ideal output of on-line EM obtained analytically (magenta).

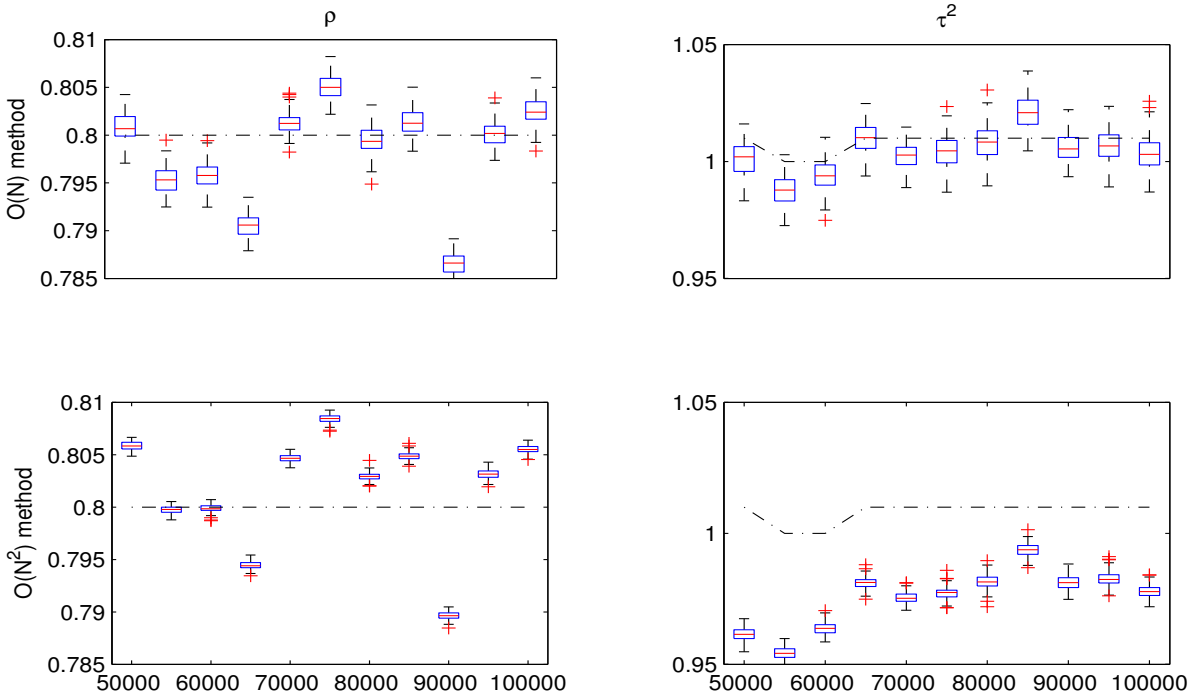


Figure 14: On-line EM: Box-plots of $\hat{\theta}_n$ for $n \geq 5 \times 10^4$ using 100 realizations of the algorithms. Details similar to Figure 13.