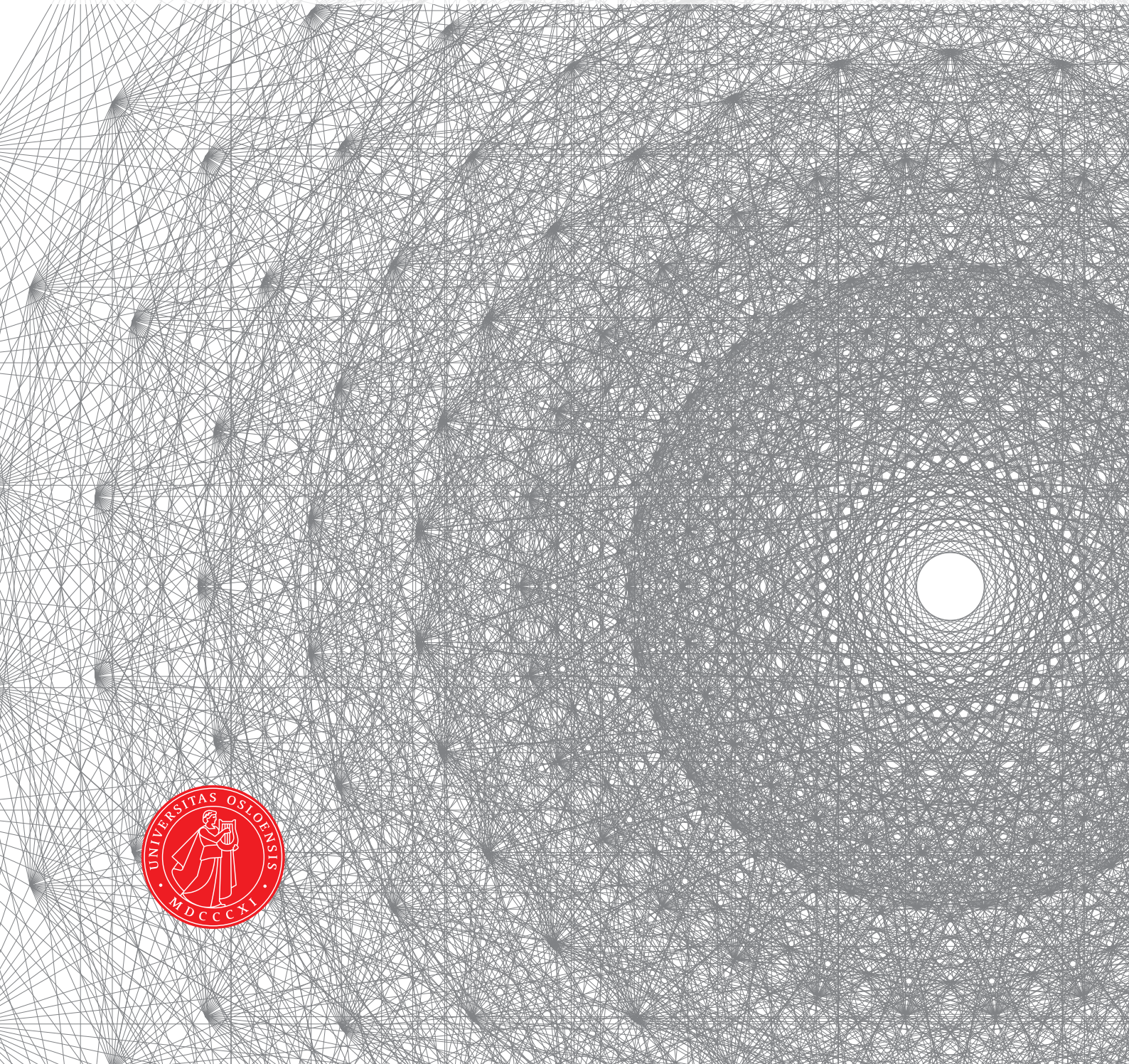
UiO : Department of Mathematics
University of Oslo

Sequential Monte Carlo in High Dimension

Challenges and Promising Algorithms

S M Mamun Ar Rashid

Master's Thesis, Spring 2024



This master's thesis is submitted under the master's programme Data Science, with program option Statistics and Machine Learning, at the Department of Mathematics, University of Oslo. The scope of the thesis is 60 credits.

The front page depicts a section of the root system of the exceptional Lie group $E8$, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Acknowledgements

I express my deepest gratitude to my supervisor, Geir Olve Storvik, who provided me with regular guidance and support throughout the year. His insights, suggestions, patience, and mentorship have made it possible for me to complete this work.

Contents

List of Figures

List of Tables

List of Symbols

$\{X_t^{d_x}\}$	Hidden stochastic process of dimension d_x
$\{Y_t^{d_y}\}$	Observed stochastic process of dimension d_y
$\mathcal{X}$	State space of a state-space model
$\mathcal{Y}$	State space of the observation process
$f_t(x_t \mid x_{t-1}; \theta)$	State transition density
$g_t(y_t \mid x_t; \theta)$	Observation/emission density
$\mathbb{R}$	Set of real numbers
$\mathbb{E}$	Expectation (of a random variable)
$\mathcal{N}_{d_x}$	d_x-dimensional Gaussian distribution
w_i	Weight of the i-th particle
d_x	Dimension of state x_t
d_y	Dimension of observation y_t
$\delta(x)$	Dirac delta measure
$\mathcal{O}$	Big O notation

CHAPTER 1

Introduction

Since their advent in the 1990s, Sequential Monte Carlo (SMC) methods, also known as particle filters, have found routine applications in many fields of science. Their great success stems from their conceptual simplicity, ease of implementation, and applicability in nonlinear and non-Gaussian systems. While greatly successfully in the estimation of low dimensional state spaces, SMC methods face fundamental challenges in the high dimensional, such as particle degeneracy, weight collapse, and high computational cost. For high dimensional state space models, it becomes crucial to design guided particle filters with novel construction of proposal distributions that alleviate weight degeneracy. There has as yet no universal recipe for generating such effective proposals. However, several interesting developments have emerged recently on particle filtering in the high dimensional setting. SMC in the high dimension remains a nascent branch of SMC for which we anticipate increasingly more research and breakthroughs.

Structure of the Report

This thesis is organized into three key parts. In the first, we review the foundational concepts of sequential Monte Carlo methods. Then we discuss limitations of these methods in making inferences on the high dimensional state spaces. Finally, we review a few classes of algorithms that have shown some promise in the high dimensional setting and compare their performances. For all numerical examples and experiments, we have used R and Python, and the codes are available at our GitHub repository https://github.com/smmarashid/smc_hd. Below we present a brief outline of the structure of report.

Background concepts

In chapter 2, we discuss state space models and the way Monte Carlo integration makes it possible to make inferences on these models in cases where the analytical method fails. We consider the importance sampling and resampling concepts and finally the SMC or particle filtering algorithms.

Challenges of SMC in high dimension

In chapter 3, we study the failure of particle filter in the high dimensional setting, even when the state space is linear and Gaussian, for which we have

analytical solutions available from the Kalman filter ((Kalman, 1960)).

Emerging algorithms

In chapter 4, we discuss a few classes of emerging algorithms that have shown some promise in making inferences on the high dimensional SSMs, particularly Block Particle Filter (Rebeschini and Van Handel (2015), Space-Time Particle Filter (Beskos et al. (2017), Nested Sequential Monte Carlo (C. Naesseth, Lindsten and Schon (2015), and Divide and Conquer Sequential Monte Carlo (Crucinio and Johansen (2022).

In chapter 5, we present the results from simulations by algorithms on a linear Gaussian state space model with different degrees of correlations among its dimensions. We compare the results based on several metrics, such as runtime, Wasserstein distance and Kolmogorov–Smirnov distance between distributions produced by the Kalman filter and those produced by these algorithms, and their relative mean square error compared to the estimates made by the Kalman filter.

In chapter 6, we summarize key findings, practical challenges we faced, the future works we could undertake given the opportunity.

Appendix A presents a few mathematical derivations to additionally clarify some points made in the main text.

CHAPTER 2

An Introduction to Sequential Monte Carlo

Sequential analysis of state space models (SSM) is the main application of Sequential Monte Carlo (SMC), and state space models are ubiquitous in many different fields of science. In this chapter, we start by defining state space models and reviewing its essential characteristics. We then discuss how Monte Carlo integration and the concept of importance sampling lead to numerical solutions to the inferencing problems, such as estimating the state and its parameters, in for which no analytical solutions exist.

2.1 State-Space Models

In many fields, such as statistics, econometrics, information engineering, signal processing, finance, and biology, we frequently encounter dynamic systems that evolve over time. Our primary objective when encountering such a system often becomes determining the underlying states that describe the system. However, the dynamic process that governs the states are often not directly observable but manifested by indirect, and essentially noisy, observations that are available to us.

A general, mathematical framework for modeling such a system that evolves over time is the class of State space models (SSM). An SSM is essentially a time series model that consists of two discrete-time processes $\{X_t\} \coloneqq (X_t)_{t\geq 0}, \{Y_t\} \coloneqq (Y_t)_{t\geq 0}$ that, respectively, take values in space $\mathcal{X}$ and space $\mathcal{Y}$ (Chopin and Papaspiliopoulos, 2020). $\{X_t\}$ represent the unknown model variables, generally referred to as latent or hidden variables, that describe the true state of the phenomena at time t and follow a Markov process, while $\{Y_t\}$ are the measured observations that are available to us and assumed to be conditionally independent given $\{X_t\}$. We have also parameters $\theta \in \Theta \subset \mathrm{R}^{d_\theta}$, which describe the model but are typically unknown or not sufficiently known. In addition, there might be explanatory or known variables that we do not model as stochastic. The spaces $\mathcal{X}$ and $\mathcal{Y}$ can be high-dimensional Euclidean spaces ($\mathcal{X} \subseteq \mathrm{R}^{d_x}, \mathcal{Y} \subseteq \mathrm{R}^{d_y}$), discrete spaces, or often other types of less-standard spaces, such as manifold space and function space.

Definition 2.1: Markov process (Pavliotis, 2016)

Let T be an ordered set of time indices and $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space where Ω is the set of outcomes or the sample space, $\mathcal{F}$ the σ-algebra of events, and $\mathbb{P}$ the probability measure on $(\Omega, \mathcal{F})$. The time set is either discrete ($T = \mathbb{N}$) or continuous ($T = [0, \infty)$). T is given the Borel σ algebra $\mathcal{T}$, and the time space $(T, \mathcal{T})$ has a natural measure: counting measure when the time set is discrete, or Lebesgue measure when the set is continuous. The set of states S is given the Borel σ-algebra $\mathcal{S}$, and $(S, \mathcal{S})$ is the state space. S can also be discrete and finite ($S = \mathbb{N}$) or continuous ($S = \mathbb{R}^d$). The stochastic process $\{X_t\}_{t \in T}$ is a Markov process if

$$\mathbb{P}(X_{t+k} \in A \mid \mathcal{F}) = \mathbb{P}(X_{t+k} \in A \mid X_t)$$

for all $t, k \in T$ and $A \in \mathcal{S}$.

Definition 2.1 implies that if we know the current state X_t, then any additional information about the past events is redundant in predicting the next state X_{t+1} or any future state X_{t+k}. In the definition we assume, according to the primary interest our study throughout the thesis, that $\{X_t\}_{t \in T}$ follow a discrete time Markov process ($T = \mathbb{N}$) with $\Omega = X_t$.

There are generally two common ways that are adopted to express an SSM mathematically, i.e., to relate the processes $\{X_t\}$ and $\{Y_t\}$. The first is through a set of deterministic functions that allow for necessary uncertainty:

$$X_0 = F_0(U_0, \theta) \tag{2.1a}$$
$$X_t = F_t(X_{t-1}, U_t, \theta), \quad t = 1, 2, 3, \ldots \tag{2.1b}$$
$$Y_t = G_t(X_t, V_t, \theta), \quad t = 0, 1, 2, \ldots \tag{2.1c}$$

where F_0, F_t, G_t are the deterministic functions, $\{U_t\}, \{V_t\}$ sequences of independent and identically distributed (iid) random variables, and θ the parameter vector of the model, which can be either known or unknown and often be the primary problem of interest, together with $\{X_t\}$.

The other way to express an SSM is through a set of probability densities that define the joint density of the processes via a factorisation:

$$p_0(x_0; \theta) \prod_{t=0}^{T} g_t(y_t \mid x_t; \theta) \prod_{t=1}^{T} f_t(x_t \mid x_{t-1}; \theta).$$

This joint density describes a generative probabilistic model that evolves from an initial state through two successive processes. At the onset ($t = 0$), X_0 is drawn according to the initial density $p_0(x_0; \theta)$, and then each X_t is drawn conditionally on the previously drawn $X_{t-1} = x_{t-1}$ according to the density $f_t(x_t \mid x_{t-1}; \theta)$, and each Y_t conditionally on the most recent $X_t = x_t$, according to the density $g_t(y_t \mid x_t; \theta)$.

$$X_0 \sim p_0(x_0; \theta). \tag{2.2a}$$
$$X_t \sim f_t(x_t \mid x_{t-1}; \theta), \quad t = 1, 2, 3, \ldots. \tag{2.2b}$$
$$Y_t \sim g_t(y_t \mid x_t; \theta), \quad t = 1, 2, 3, \ldots. \tag{2.2c}$$

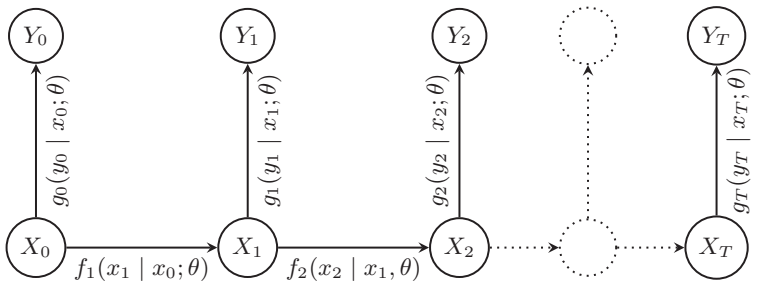


Figure 2.1: A schematic representation of a simplified state-space model

To summarize, an SSM is characterized by two principles: (i) the state process $\{X_t\}$ is assumed to be a Markov process and (ii) observations, $\{Y_t\}$, are independent given $\{X_t\}$. An assumption that might sound unsatisfactory in the second definition is that the conditional distribution of X_t given $X_{t-1} = x_{t-1}$ has a *density*, but the density might not be directly available. However, this assumption is not an essential requirement to the definition of the state space model and can be avoided with other, suitable formulations.

2.2 Analysis of State-Space Models

There are obviously several aspects of a state space model we are generally interested in making inference about. These aspects can be broadly classified into two major categories:

1. **State inference:** Learn the states $\{X_t\}$ given the available observational data $\{Y_t\}$. This objective can further be distinguished by several tasks as listed in the Table 2.1 below.

Table 2.1: Common tasks in State inference

Task	**PDF of interest**
Filtering	$p(x_t \mid y_{0:t})$
Joint filtering	$p(x_{0:t} \mid y_{0:t}), t = 0, 1, 2, \ldots$
State prediction	$p(x_{t+1} \mid y_{0:t})$ or $p(x_{t+1:t+h} \mid y_{0:t}), h \geq 1$
Joint smoothing	$p(x_{0:T} \mid y_{0:T})$
Marginal smoothing	$p(x_t \mid y_{0:T}), t \leq T$
Fixed lag smoothing	$p(x_{t-h:t} \mid y_{0:t}), h \geq 1$

2. **Parameter inference:** Learn the parameters θ given the observations. Parameter estimation can either be (i) *online*, in which model parameters are estimated, and the estimates are updated, as new data become sequentially available over time, or (ii) *offline*, in which all the observations become available before the parameters are estimated.

3. **Observation inference:** In addition to the inference about the states and the parameters, we might also be interested in the inference about the

observation process. For example, in predicting or forecasting future data, we are interested about the PDF $p(y_{t+1:t+h} \mid y_{0:t})$ where $h \geq 1$ and $t \geq 0$.

The first strategy of inference

The obvious starting point in making any inference in the state space system appears to be computing the posterior distribution of the states from the joint prior distribution $p(y_{0:T})$ of the states and the conditional likelihood $p(y_{0:T} \mid x_{0:T})$ of the observations, as given below (we skip θ for rotational simplicity):

$$p(x_{0:T}) = p_0(x_0) \prod_{t=1}^{T} f_t(x_t \mid x_{t-1}); \tag{2.3}$$

$$p(y_{0:T} \mid x_{0:T}) = p_0(x_0) \prod_{t=0}^{T} g_t(y_t \mid x_t). \tag{2.4}$$

Now for any given t, the posterior distributions of the states could simply be calculated by the Bayes' rule:

$$p(x_{0:t} \mid y_{0:t}) = \frac{p(y_{0:t} \mid x_{0:t})\, p(x_{0:t})}{p(y_{0:t})} \propto p(x_{0:t}, y_{0:t}). \tag{2.5}$$

Computing the full posterior at each time point is however a crude strategy that might be feasible only for a small amount of observations (it can still be difficult due to the integration of the denominator in (2.5)), but obviously not in real-time applications when a large number observations arrive with each time step.

The next strategy is to exploit the Markov property of the system and establish the possible recursions in the computation to ascertain that a constant number of computations is done on each time steps. We illustrate the strategy with its application on the filtering calculation, i.e. we are interested in marginal posterior distribution or filtering distribution of the state X_t at each time step t given the history of the measurements up to the time step t:

Theorem 2.2.1: State space filtering equations

The recursive Bayesian filtering equations for computing the predicted distribution $p(x_t \mid y_{0:t-1})$ and the filtering distribution $p(X_t \mid y_{0:t})$ at the time step t are given by the following Bayesian filtering equations.

- **Initialization.** The recursion starts from prior distribution of state $p(x_0)$ at $t = 0$.
- **Prediction.** The predictive distribution of the state x_t at the time step k, given the emission model, can be computed by the Chapman–Kolmogorov equation

$$p(x_t \mid y_{0:t-1}) = \int f(x_t \mid x_{t-1})\, p(x_{t-1} \mid y_{0:t-1})\, dx_{t-1}. \tag{2.6}$$

- **Update.** Given the measurement y_t at time step t the posterior distribution of the state $x_t$can be computed by Bayes' rule:

$$p(x_t \mid y_{0:t}) = \frac{g(y_t \mid x_t)\, p(x_t \mid y_{0:t-1})}{Z_t}, \tag{2.7}$$

  where the normalizing constant Z_t is given as

$$Z_t = \int g(y_t \mid x_t)\, p(x_t \mid y_{0:t-1}) dx_t. \tag{2.8}$$

  If some of the components of the state are discrete, the corresponding integrals are replaced with summations.

Proof. See Appendix A.1.

Implementing the recursions

Now that we have recursive formulas for predicting and updating the states, the next obvious strategy is to find whether these equations yield a closed form to provide an analytical solution. Since we cannot generally compute the normalizing constant $p(y_{0:t})$ and marginals of the posterior $p(x_t \mid y_{0:t})$, because these require an evaluation of complex, high-dimensional integrals, there are only a few restrictive cases that admit a close form of solution.

- *Posterior at each time is Gaussian.*
  This implies that the posterior is completely described by mean and covariance, and this case occurs when the state-space can be expressed by the functional form specified in equations (2.1a)–(2.1c), with the deterministic functions F_t and G_t being linear and the $\{U_t\}$ and $\{V_t\}$ Gaussian. This reduces the Bayesian filter of the state space to the famous Kalman filter (Kalman (1960)) that has the analytical solutions.
- *Domain of the state space X_t is discrete and finite.*
  Suppose that the state space at time $t-1$ can assume only one of a finite set of values:$\{X_{t-1,j}\}_{j=1,2,\ldots,N_s}$. Let $p(x_{t-1,j} \mid y_{0:t-1}) = w_{j,t-1|t-1}$. The

posterior distribution can therefore by expressed as a sum of Dirac delta functions:

$$p(x_{t-1} \mid y_{0:t-1}) = \sum_{j}^{N} w_{j,t-1|t-1}\delta(x_{t-1} - x_{t-1,j}) \tag{2.9}$$

The prediction will now be

$$\begin{aligned} p(x_t \mid y_{0:t-1}) &= \sum f(x_t \mid x_{t-1})\, p(x_{t-1} \mid y_{0:t-1})\, dx_{t-1} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} p(x_{t,i} \mid x_{t-1,j}) w_{j,t-1|t-1}\delta(x_{t-1} - x_{t-1,i}) \\ &= \sum_{i=1}^{N} w_{i,t|t-1}\delta(x_t - x_{t,i}) \end{aligned}$$

where $w_{i,t|t-1} = \sum_{j=1}^{N} p(x_{t,i} \mid x_{t-1,j}) w_{j,t-1|t-1}$. We note two things here: the new prior is also a weighted sum of the delta functions, and the new prior weights are arrived by reweighting the old posterior weights based on the state transition probabilities.

The update step becomes

$$\begin{aligned} p(x_t \mid y_{0:t}) &= \frac{p(y_t \mid x_t)\, p(x_t \mid y_{0:t-1})}{\sum p(y_t \mid x_t)\, p(x_t \mid y_{0:t-1})} \\ &= \sum_{i=1}^{N} w_{i,t|t}\delta(x_{t-1} - x_{i,t-1}) \\ w_{i,t|t} &= \frac{w_{i,t|t-1}\, p(y_t \mid x_{t,i})}{\sum_{j=1}^{N} p(y_t \mid x_{t,j})} \end{aligned}$$

Posterior weights are reweighting of prior weights using likelihoods and normalization.

2.3 Monte Carlo Methods

Because we do not have a closed form of equations in most of the cases of state space models, we approximate the intractable integrals appearing in the equations. Let us consider any test function $\psi_t(x_{0:t})$ for some fixed t in which $x_{0:t}$ are random variables that follow the probability density $\pi_t(x_{0:t})$. $\pi_t(\cdot)$ here can be any of the distributions listed in Table 2.1 Thus, the expectation $\mathbb{E}_{\pi_t}[\psi_t(x_{0:t})]$, which is often a statistic of interest in statistical analysis, including the state space systems, is given by the following integral form:

$$I_t = \mathbb{E}_{\pi_t}[\psi_t(x_{0:t})] = \int_{\mathbb{X}} \psi_t(x_{0:t})\pi_t(x_{0:t})\, dx_{0:t}. \tag{2.10}$$

The basic Monte Carlo states that if we sample N independent random variables, $x_{0:t,i} \sim \pi_t(x_{0:t})$ for $i = 1, \ldots, N$, then I_t can be approximated by the Monte Carlo estimate $\hat{I}$:

$$\hat{I}_t^{\mathsf{MC}} = \frac{1}{N}\sum_{i=1}^{N} \psi_t(x_{0:t,i}) \approx I_t. \tag{2.11}$$

The reliability of the Monte Carlo estimate is based on the Law of large numbers (LLN) and the Central limit theorem (CLT). In general, we can make the following propositions about $\hat{I}_t^{\mathsf{MC}}$:

Propositions 2.3.1: Limit theorems about the MC estimate

(a) If $\mathbb{E}_{\pi_t}[\psi_t(x_{0:t})] < \infty$, then $\hat{I}_t^{\mathsf{MC}}$ is an unbiased and strongly consistent estimator of I_t, i.e., $\mathbb{E}[\hat{I}_t^{\mathsf{MC}}] = I_t$ and $\hat{I}_t^{\mathsf{MC}} \to I_t$ almost surely as $N \to \infty$.
Proof.

$$\mathbb{E}[\hat{I}_t^{\mathsf{MC}}] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\pi_t}[\psi_t(x_{0:t,i})] = \frac{1}{N}(I_t + \ldots + I_t) = I_t.$$

Strong consistency directly follows from the LLT.

(b) If $\mathbb{E}_{\pi_t}[\psi_t(x_{0:t})] < \infty$ and $\sigma^2 = \mathbb{V}[\psi_t(x_{0:t})] < \infty$, then $\mathbb{E}[(\hat{I}_t - I_t)^2] = \mathbb{V}[\hat{I}_{t,N}] = \sigma^2/N$ and $\sqrt{N}(\hat{I}_{t,N} - I_t)/\sigma \xrightarrow{\mathrm{d}} \mathcal{N}(0,1)$.
Proof.

$$\mathbb{V}[\hat{I}_{t,N}] = \mathbb{E}[(\hat{I}_{t,N} - \mathbb{E}[\hat{I}_{t,N}])^2] = \mathbb{E}[(\hat{I}_{t,N} - I_t)^2].$$

$$\begin{aligned}\mathbb{V}[\hat{I}_{t,N}] &= \mathbb{V}\left[\frac{1}{N}\sum_{i=1}^{N}\psi_t(x_{0:t,i})\right] = \frac{1}{N^2}\mathbb{V}\left[\sum_{i=1}^{N}\psi_t(x_{0:t,i})\right] \\ &= \frac{1}{N^2}\sum_{i=1}^{n}\mathbb{V}[\psi_t(x_{0:t,i})] = \frac{1}{N^2}N\sigma^2 = \sigma^2/N.\end{aligned}$$

By applying the CLT on $\psi_t(x_{1:t,i})$,

$$\begin{aligned}&\frac{\psi_t(x_{1:t,1}) + \ldots + \psi_t(x_{1:t,N}) - NI_t}{\sigma\sqrt{N}} \xrightarrow{\mathrm{d}} \mathcal{N}(0,1), \\ \Rightarrow &\frac{N\hat{I}_t^{\mathsf{MC}} - NI_t}{\sigma\sqrt{N}} \xrightarrow{\mathrm{d}} \mathcal{N}(0,1), \\ \Rightarrow &\sqrt{N}(\hat{I}_{t,N} - I_t)/\sigma \xrightarrow{\mathrm{d}} \mathcal{N}(0,1).\end{aligned}$$

The main advantage of Monte Carlo methods over standard approximation techniques is that the standard deviation and the variance of the approximation error decreases at a rate of $\mathcal{O}(N^{1/2})$ and $\mathcal{O}(N^{-1})$ respectively, regardless of the dimension d_x of the state space $\mathbb{X} = \mathbb{R}^{d_x}$ of x_t. That is, whether $d_x = 1$ or 10000, the error is always in σ/N. This quality of the Monte Carlo estimate seems to have been misinterpreted sometimes with the erroneous claim that the Monte Carlo estimates circumvents the Curse of dimensionality problem that arises in the high dimensional state space. Because σ generally depends on d_x, the curse of dimensionality must often be encountered and be attempted by other methods to circumvent it. There are at least three key problems with the basic Monte Carlo approach:

- Problem 1: If $\pi_t(x_{0:t})$ is a complex high-dimensional probability

distribution, we still may not be able sample from it directly.

- Problem 2: The posterior distribution $p(x_t \mid y_{0:t})$, as it often happens in Bayesian inference, may only be available up to a proportionality constant. That is, the normalizing constant constant Z_t in Equation 2.7 is often unknown, leaving us with

$$p(x_t \mid y_{0:t}) \propto g(y_t \mid x_t)\, p(x_t \mid y_{0:t-1}), \tag{2.12}$$

- Problem 3: Even if we can sample exactly from $\pi_t(x_{0:t})$, the computational complexity of such a sampling scheme is typically at least $\mathcal{O}t$. Moreover, this complexity is valid only for low dimensional sate space or, more specifically, when the dimension of x_t is 1. The computational complexity will increase exponentially with the dimension, as we would note in the later chapters.

2.4 Importance Sampling

The first two problems are improved by a sampling technique called Importance Sampling (IS), a fundamental method in Monte Carlo that has served as the basis of all the algorithms developed later on. Importance sampling functions by introducing an arbitrary density $q_t(\cdot)$ from which samples are drawn, rather than from the original, target density $\pi_t(\cdot)$. Such strategy might be needed, for example, when the target density is so complex that successful sampling from it is a complicated challenge, while the arbitrary density is easier to sample from but is nevertheless close enough to the target density. A change of distribution then will not necessarily increase the variance appreciably. Such scenario is especially common when π_t is approximately Gaussian; a perfect Gaussian is then the natural choice for the arbitrary distribution. The new density is referred to as importance density, proposal density, or instrumental density, and chosen such that

$$\pi_t(x_{0:t} \mid y_{0:t}) > 0 \rightarrow q_t(x_{0:t} \mid y_{0:t}) > 0$$

In other words, the support of $q_t(x_t \mid y_{0:t})$ must be greater than or equal to the support of $\pi_t(x_t \mid y_{0:t})$.

The notion of IS is based on the following decomposition of the expectation over the posterior probability density $\pi_t(x_t \mid y_{0:t})$:

$$\begin{aligned} I_t &= \int \psi_t(x_t)\pi_t(x_t \mid y_{0:t})\, dx_t \\ &= \int \left[\psi_t(x_t)\frac{\pi_t(x_t \mid y_{0:t})}{q_t(x_t \mid y_{0:t}}\right] q_t(x_t \mid y_{0:t})\, dx_t \end{aligned} \tag{2.13}$$

The right-hand side of the equation now shows the expectation of the term $\left[\psi_t(x)\frac{\pi_t(x_t|y_{0:t})}{q_t(x_t|y_{0:t}}\right]$ over the distribution $q_t(x_t \mid y_{0:t})$, so I_t can also be written as

$$I_t = \mathbb{E}_{p_t}[\psi_t(x_t)\pi_t(x_t \mid y_{0:t})] = \mathbb{E}_{q_t}\left[\psi_t(x_t)\frac{p(x_t \mid y_{0:t})}{q_t(x_t \mid y_{0:t}}\right], \tag{2.14}$$

which is called the *importance sampling fundamental identity*, (Robert and Casella, 2004) where $\mathbb{E}_{q_t}$ is the expectation for a probability measure for which the distribution is $q_t(\cdot)$ instead of $p_t(\cdot)$. Importance sampling can now be defined as

Definition 2.2: Importance sampling (Robert and Casella, 2004)

The method of importance sampling is an evaluation of (2.10) based on generating an ensemble $\{x_{t,i}\}_{i\geq 0}$ of particles from a given distribution q_t:

$$x_{t,i} \sim q_t(x_t \mid y_{0:t}), \qquad i = 1, \ldots, N, \tag{2.15}$$

and making the Monte Carlo approximation of the expectation

$$\hat{I}_t^{\mathsf{IS}} = \frac{1}{N}\sum_{i=1}^{N} \frac{\pi_t(x_{t,i} \mid y_{0:t})}{q_t(x_{t,i} \mid y_{0:t})}\psi_t(x_{t,i}) = \frac{1}{N}\sum_{i=1}^{N} \widetilde{w}_{t,i}\psi_t(x_{t,i}) \tag{2.16}$$

$$\approx I_t = \mathbb{E}_q[\widetilde{w}_t(x_{0:t})\psi_t(x_t \mid y_{0:t})] \tag{2.17}$$

where

$$\widetilde{w}_{t,i}(x_t) = \frac{\pi_t(x_{t,i} \mid y_{0:t})}{q_t(x_{t,i} \mid y_{0:t})} \tag{2.18}$$

The algorithm is described below:

Algorithm 2.1 Importance Sampling

1 Given a measurement model $p_t(y_{0:t} \mid x_t)$, a prior $\pi_t(x_t)$, the number of particles N, choose q_t such that $\text{supp}(q_t) \supset \text{supp}(\pi_t \cdot \psi_t)$.
2 **for** $i = 1, \ldots, N$ **do**
3 Draw $x_{t,i} \sim q_t(x_t \mid y_{0:t})$.
4 Set the weight $\widetilde{w}_{t,i}(x_t) = \dfrac{\pi_t(x_{t,i} \mid y_{0:t})}{q_t(x_{t,i} \mid y_{0:t})} = \dfrac{p_t(y_{0:t} \mid x_{t,i})\pi_t(x_{t,i})}{q_t(x_{t,i} \mid y_{0:t})}$.
5 Return an estimate to the posterior expectation of any function $\psi_t(x_t)$ by the following:

$$\mathbb{E}[\psi_t(x_t \mid y_{0:t})] \approx \frac{1}{N}\sum_{i=1}^{N} \widetilde{w}_{t,i}\psi_t(x_{t,i}).$$

In essence, importance sampling implies yielding a weighted ensemble $\{x_{t,i}, \widetilde{w}_{t,i}\}$ from $q_t(\cdot)$ instead of $pi_t(\cdot)$ and using the ensemble for estimating expectations and therefore probabilities and related measures. The weight, i.e., the ratio in Equation (2.18), is known as the importance ratio, importance weight, or the likelihood ratio. The term importance sampling derives from the notion that the most common values of x_t under the distribution $\pi_t(\cdot)$ may not necessarily be the most important ones. A well-chosen, alternative density $q_t(\cdot)$ might make the "important" values of x_t more likely.

Properties of importance sampling estimate

- $\hat{I}_t^{\mathsf{IS}}$ is consistent if $\text{supp}(q_t) \supset \text{supp}(p_t \cdot \psi_t)$ and $\mathbb{E}_{q_t}[\widetilde{w}_t(x_t) \cdot \psi_t(x_t)] < +\infty$, as $\hat{I}_t^{\mathsf{IS}} = \frac{1}{N}\sum_{i=1}^N \widetilde{w}_{t,i}\psi_t(x_{t,i}) \xrightarrow{\text{a.s.}} \mathbb{E}_{p_t}[\psi_t(x_t)]$. This property follows from the Law of large numbers.
- The expected value of the weights $\mathbb{E}_{q_t}[\widetilde{w}_t] = 1$.
- $\hat{I}_t^{\mathsf{IS}}$ is unbiased.

Theorem 2.4.1: Bias and Variance of Importance Sampling

$$\mathbb{E}_{q_t}[\hat{I}_t^{\mathsf{IS}}] = I_t \tag{2.19}$$

$$\mathbb{V}_q[\hat{I}_t^{\mathsf{IS}}] = \frac{\mathbb{V}_q[\widetilde{w}(x_t)\psi_t(x_t)]}{N} \tag{2.20}$$

There is one key challenge to using Algorithm 2.1 in its current form: $q_t(.)$ itself can be available only up to a proportionality constant. Suppose, $\pi_t(x_t \mid y_{0:t}) = Z_t\gamma_t(x_t \mid y_{0:t})$, where the proportionality constant Z_t is unknown to us. Then

$$\begin{aligned}\hat{I}_t^{\mathsf{IS}} &= \frac{1}{N}\sum_{i=1}^N \widetilde{w}_{t,i}\psi_t(x_{t,i}) \\ &= \frac{1}{N}\sum_{i=1}^N \frac{Z_t\gamma_t(x_{t,i} \mid y_{0:t})}{q_t(x_{t,i} \mid y_{0:t})}\psi_t(x_{t,i}) = \frac{1}{N}\sum_{i=1}^N Z_t\widetilde{w}_{t,i}\psi_t(x_{t,i}),\end{aligned} \tag{2.21}$$

where we have weights $\{\widetilde{w}_t\}$ defined in terms of $\gamma_t(\cdot)$ and $q_t(\cdot)$. Since Z_t does not cancel out, knowing up to $\gamma_t(\cdot)$ is not enough to derive an estimate. We can however resolve the challenge if we replace the normalization by N by $\sum_{i=1}^N Z_t\widetilde{w}_{t,i}$ in the estimate so that

$$\mathbb{E}_{q_t}[\psi_t(x_t \mid y_{0:t})] \approx \frac{\sum_{i=1}^N Z_t\widetilde{w}_{t,i}\psi_t(x_{t,i})}{\sum_{i=1}^N Z_t\widetilde{w}_{t,i}} = \sum_{i=1}^N w_{t,i}\psi_t(x_{t,i}), \tag{2.22}$$

where $\{w_{t,i}\}$ are the standardized weights and defined as

$$w_{t,i} = \frac{Z_t\widetilde{w}_{t,i}}{\sum_{i=1}^N Z_t\widetilde{w}_{t,i}} = \frac{\widetilde{w}_{t,i}}{\sum_{i=1}^N \widetilde{w}_{t,i}} \tag{2.23}$$

have the property $\sum_{i=1}^N \widetilde{w}_{t,i} = 1$.
The change of the normalization now ensures that the estimate does not depend on Z_t, and it is enough to know $\pi_t(\cdot)$ up tp a multiplicative constant. This variant of importance sampling is referred to as self normalized importance sampling, and the resulting algorithm is presented below:

Algorithm 2.2 Importance sampling using self-normalized weights

1 Given a measurement model $p_t(y_{0:t} \mid x_t)$, a prior $\pi_t(x_t)$, and the number of particles N, choose q_t such that $\text{supp}(q_t) \supset \text{supp}(\pi_t \cdot \psi_t)$.
2 **for** $i = 1, \ldots, N$ **do**
3 Draw $x_{t,i} \sim q_t(x_t \mid y_{0:t})$.
4 Set the unnormalized weight
$$\widetilde{w}_{t,i}(x_t) = \frac{\pi_t(x_{t,i} \mid y_{0:t})}{q_t(x_{t,i} \mid y_{0:t})} = \frac{p_t(y_{0:t} \mid x_{t,i})\pi_t(x_{t,i})}{q_t(x_{t,i} \mid y_{0:t})}.$$
5 Compute the standardized weight $w_{t,i} = \dfrac{\widetilde{w}_{t,i}}{\sum_{i=1}^{N} \widetilde{w}_{t,i}}$.
6 Return an estimate to the posterior expectation of any function $\psi_t(x_t)$ by the following:

$$\mathbb{E}[\psi_t(x_t \mid y_{0:t})] \approx \hat{I}_t^{\mathsf{SNIS}} = \sum_{i=1}^{N} w_{t,i}\psi_t(x_{t,i}).$$

Properties of self-normalized estimate

- $\hat{I}_t^{\mathsf{SNIS}}$ is consistent if $\text{supp}(q_t) \supset \text{supp}(\pi_t \cdot \psi_t)$ and $\mathbb{E}_{q_t}[\widetilde{w}_t(x_t) \cdot \psi_t(x_t)] < +\infty$, as we note that

$$\hat{I}_t^{\mathsf{SNIS}} = \frac{\sum_{i=1}^{N} \widetilde{w}_{t,i}\psi_t(x_{t,i})}{N} \cdot \frac{N}{\sum_{i=1}^{N} \widetilde{w}_{t,i}}. \tag{2.24}$$

  The first term on the right hand side is the importance sampling estimate $\hat{I}^{\mathsf{IS}}$, which almost surely converges to $\mathbb{E}_{p_t}[\psi_t(x_t)]$, and the second term almost surely converges to the expectation $\mathbb{E}_{q_t}[\pi_t(x_t)/q_t(x_t)]$ or 1 as $n \to \infty$.

- $\hat{I}_t^{\mathsf{SNIS}}$ is unbiased, but asymptotically unbiased.

Theorem 2.4.2: Bias and Variance of self-normalized IS

$$\mathbb{E}_{q_t}[\hat{I}_t^{\mathsf{SNIS}}] = I_t + \frac{I_t\mathbb{V}_q[\widetilde{w}_t(x_t)] - \text{Cov}_q[\widetilde{w}_t(x_t), \widetilde{w}_t(x_t)\psi_t(x_t)]}{N} + \mathcal{O}(N^{-2}) \tag{2.25}$$

$$\mathbb{V}_q[\hat{I}_t^{\mathsf{SNIS}}] = \frac{\mathbb{V}_q[\widetilde{w}_t(x_t)\psi_t(x_t)] - 2I_t\text{Cov}_q[\widetilde{w}_t(x_t), \widetilde{w}_t(x_t)\psi_t(x_t)]}{N} + \frac{I_t^2\mathbb{V}_q[\widetilde{w}t(x_t]}{N} + \mathcal{O}(N^{-2}) \tag{2.26}$$

See (Iambartsev, 2018).

2.5 Sequential Importance Sampling

Importance sampling provides us with the numerical approximation for an intractable function. Sequential importance (see, e.g.,Doucet, De Freitas and N. Gordon (2001)) is a sequential version of importance sampling. The SIS algorithm can be used for generating importance sampling approximations to filtering distributions of generic state space models of the form

$$\begin{aligned} x_t &\sim f_t(x_t \mid x_{t-1}), \quad x_t \in \mathbb{R}^{d_x}, \\ y_t &\sim g_t(y_t \mid x_t), \quad y_t \in \mathbb{R}^{d_y}. \end{aligned}$$

The state and measurements may contain both discrete and continuous components.

The SIS algorithm uses a weighted set of N particles $(w_{t,i}, x_{t,i}), i = 1, \ldots, N$, that is, samples from an importance distribution and their weights, for representing the filtering distribution $p_t(x_t \mid y_{0:t})$ such that at every time step t the approximation to the expectation of an arbitrary $\psi(x_t)$ can be calculated as the weighted sample average

$$\mathbb{E}[\psi_t(x_t \mid y_{0:t})] \approx \sum_{i=1}^{N} \widetilde{w}_{t,i} \psi_t(x_{t,i})$$

Equivalently, SIS can be interpreted as forming an approximation to the filtering distribution as

$$p_t(x_t \mid y_{0:t}) \approx \sum_{i=1}^{N} \widetilde{w}_{t,} \delta(x_t - x_{t,i})$$

To derive the algorithm, we consider the full posterior distribution of states $x_{0:t}$ given the measurements $y_{0:t}$. By using the Markov properties of the model, we get the following recursion for the posterior distribution (to simplify the notation, we drop the time subscript from the symbol of the density functions here, but it is understood):

$$\begin{aligned} p(x_{0:t} \mid y_{0:t}) &= p(x_t \mid x_{0:t-1}, y_{0:t}) p(x_{0:t-1} \mid y_{0:t}) \\ &\propto p(y_t \mid x_t, x_{0:t-1}, y_{0:t-1}) p(x_t \mid x_{0:t-1}, y_{0:t-1}) \\ &\quad \times p(y_t \mid x_{0:t-1}, y_{0:t-1}) p(x_{0:t-1} \mid y_{0:t-1}) \\ &\propto p(y_t \mid x_t) p(x_t \mid x_{t-1}) p(y_t \mid x_{0:t-1}, y_{0:t-1}) p(x_{0:t-1} \mid y_{0:t-1}), \end{aligned}$$

Using a similar rationale as in the previous section, we can now construct an importance sampling method which draws samples from a given importance distribution $x_{0:t,i} \sim q(x_{0:t} \mid y_{0:t})$ and compute the importance weights

$$\hat{w}_{t,i} \propto \frac{p(y_t \mid x_{t,i}) p(x_{t,i} \mid x_{t-1,i}) p(x_{0:t-1,i} \mid y_{0:t-1})}{q(x_{0:t,i} \mid y_{0:t})} \tag{2.27}$$

If we form the importance distribution for the states x_t recursively as follows:

$$q(x_{0:t} \mid y_{0:t}) = q(x_t \mid x_{0:t-1}, y_{0:t}) q(x_{0:t-1} \mid y_{0:t-1}) \tag{2.28}$$

then the expressions of the weights can be written as

$$\widetilde{w}_{t,i} \propto \frac{p(y_t \mid x_{t,i})p(x_{t,i} \mid x_{t-1,i})}{q(x_{t,i} \mid x_{0:t-1,i}, y_{0:t})} \cdot \frac{p(x_{0:t-1,i} \mid y_{0:t-1})}{q(x_{0:t-1} \mid y_{0:t-1})} \tag{2.29}$$

Let us now assume that we have already drawn the samples $\{x_{0:t-1,i}\}$ from the importance distribution $q(x_{0:t-1} \mid y_{0:t-1})$ and computed the corresponding importance weights $\{\widetilde{w}_{t-1,i}\}$. We can now draw samples $\{x_{0:t,i}\}$ from $q(x_{0:t} \mid y_{0:t})$ by drawing the new state samples for the step t as $x_{t,i} \sim q(x_t \mid x_{0:t-1,i}, y_{0:t})$. The importance weights from the previous step are proportional to the last term in (2.29):

$$\widetilde{w}_{t-1,i} \propto \frac{p(x_{0:t-1,i} \mid y_{0:t-1})}{q(x_{0:t-1,i} \mid y_{0:t-1})} \tag{2.30}$$

Thus the weights satisfy the recursion

$$\widetilde{w}_{t,i} \propto \frac{p(y_t \mid x_{t,i})p(x_{t,i} \mid x_{t-1,i})}{q(x_{t,i} \mid x_{0:t-1,i}, y_{0:t})} \cdot \widetilde{w}_{t-1,i} \tag{2.31}$$

The generic sequential importance sampling algorithm can now be described as follows.

Algorithm 2.3 Sequential Importance Sampling

1 Given a measurement model $p(y_{0:t} \mid x_t)$ and a prior $p(x_t)$.
2 Sample $x_{0,i}$ from the initial prior: $x_{0,i} \sim p(x_0), i = 1, \ldots, N$.
3 Set $w_{0,i} = 1/N$ for all $i \in N$.
4 **for** $t = 1, \cdots, T$ **do**
5 Sample $x_{t,i}$ from the importance distributions

$$x_{t,i} \sim q(x_t \mid x_{0:t-1,i}, y_{0:t}), i = 1, \ldots, N.$$

6 Set $w_{t,i} = 1/N$ for all $i \in N$.
7 Return x_t, i

Weight degeneracy in SIS

One challenge with SIS is that the variance of the weights increase as the time step t also increase, as illustrated below. From the general rule of conditional means and conditional variances, we have

$$\mathbb{V}[P] = \mathbb{E}[\mathbb{V}[P \mid Q]] + \mathbb{V}[\mathbb{E}[P \mid Q]] \geq \mathbb{V}[\mathbb{E}[P \mid Q]]$$

Putting $P = w_t, Q = x_{0:t-1}$ and using (2.31), we get

$$\begin{aligned}
\mathbb{E}[\widetilde{w}_t \mid x_{0:t-1}] &= \mathbb{E}\left[\frac{p(y_t \mid x_t)p(x_t \mid x_{t-1})}{q(x_t \mid x_{0:t-1}, y_{0:t})} \cdot \widetilde{w}_{t-1}\right] \\
&= \widetilde{w}_{t-1} \cdot \mathbb{E}\left[\frac{p(y_t \mid x_t)p(x_t \mid x_{t-1})}{q(x_t \mid x_{0:t-1}, y_{0:t})} \mid x_{0:t-1}\right] \\
&= \widetilde{w}_{t-1}.1 = \widetilde{w}_{t-1}
\end{aligned}$$

Thus, $\mathbb{V}[\widetilde{w}_t] \geq \mathbb{V}[\widetilde{w}_{t-1}]$. More specifically, it is shown that the variance of the weight increases exponentially with t and to control the variance with t, the number of Monte Carlo estimates need to be chosen as exponential of t.

2.6 Sequential Importance Resampling

The weight degeneracy in SIS indicates that the variance of the weights will continue to increase at each time-step t. The consequence of this uncontrolled variance is that only a few particles will dominate the entire ensemble in terms of the weight as time increases, consequently increasing the variability of the Monte Carlo estimate. The degeneracy problem can be in part solved by using *resampling*, a procedure which works as described below:

1. Assume that each weight $\widetilde{w}_{t,i}$ is the probability of obtaining the index i in the ensemble of particles $\{x_{t,i}\}$.
2. Draw N new particles from this discrete distribution and replace the old ensemble with the new ensemble.
3. Set each new weight $w_{t,i} = \frac{1}{N}$.

Resampling in essence transforms the weighted ensemble $\{\widetilde{w}_{t,i}, x_{t,i}\}_{i=1,\ldots,N}$ of particles into an unweighted or, in other words, a equally-weighted, ensemble $\{\frac{1}{N}, x_{t,i}\}$, without changing the distribution. The motivation behind the resampling procedure is to remove particles which have negligible weights and are, in general, not a good representative of the true state, and to replicate particles with large weights, according to weight of each particle before resampling. Resampling does not change the theoretical distribution represented by the weighted set of the ensemble, because we are merely choosing particles from among the already existing ensemble, but it introduces additional variance to the estimate of state. This variance caused by the resampling procedure can be reduced by proper choice of the resampling method.

Adding a resampling step to the sequential importance sampling algorithm leads to sequential importance resampling (SIR), which is the algorithm usually referred to as the particle filter.

Common resampling methods

The resampling step plays a central role in a sequential Monte Carlo algorithm, but it is computationally expensive. While there are many schemes to implement resampling, there are four widely used resampling methods in the context of sequential Monte Carlo.

1. **Multinomial resampling**
   This is the most commonly used resampling technique in SMC, the procedures is described below:

   Draw N random numbers $\{\widetilde{u}_m\}_{m=1,\ldots,N}$, ordered and indexed by m, from the Uniform distribution: $\widetilde{u}_m \sim U[0,1)$.

   Set $u_N = \widetilde{u}_N^{1/N}$ and successively compute $u_m = \widetilde{u}_{m+1}\widetilde{u}_m^{1/m}$.

Select new particle x_m according to the multinomial probability distribution formed by the normalized weights of the particles before resampling and the value of u_m. This selection procedure, which is common to several other resampling methods, works as follows:

Make the cumulative probability distribution $\{C_i\}_{i=1,\ldots,N}$ of the normalized weights.

Find the index value i such that $C_{i-1} \leq u_m < C_i$.
Select new the particle $x_m = x_i$.

2. **Stratified resampling**
   Draw $\widetilde{u}_m \sim U[0,1)$ for $m = 1, \ldots, N$.
   Compute $u_m = \dfrac{(m-1) + \widetilde{u}_m}{N}$.
   Select new particle x_m according to the multnomial distribution of the normalized weights, as describe above.

3. **Systematic resampling**
   The key difference between stratified and systematic resampling is that the random numbers generated from the Uniform distribution are not indexed or ordered in case of the letter. Thus, systematic resampling works as follows:
   Draw $\widetilde{u} \sim U[0,1)$. Compute $u_m = \dfrac{(m-1) + \widetilde{u}}{N}$ for $m = 1, \ldots, N$.
   Select new particle x_m according to the multinomial distribution remains the same.

4. **Residual resampling**
   Create $n'_i = \lfloor N w_i \rfloor$ replicates of the particle x_i, for $i = 1, \ldots, N$, where w_i is the normalized weight and $\lfloor\ \rfloor$ is the floor function, and directly transfer these replicates to the new distribution, which is to be the distribution after resampling. To select the remaining $N - \sum_i n'_i$ samples,
   Compute the residual weight w_i^{res} of particle x_i as $w_i^{\mathsf{res}} = N w_i - \lfloor N w_i \rfloor$, for $i = 1, \ldots, N$.
   Use these weights $\{w_i^{\mathsf{res}}\}$ as probabilities to select the additional samples from $\{x_i\}$, using one of the resampling procedures described above.

All of the four resampling algorithms are unbiased. Because the random numbers u_m are ordered, they can be implemented in $\mathcal{O}(n)$ time. However, because the process of generating u_m varies, they have different computational complexities and consequently different runtimes.

Choice of the resampling method

Systematic resampling is often recommended because of its speed as well as better empirical performance (in terms of estimates with lower a variance) than the other methods. Here, we apply the four resampling methods on a non-linear, 1-D state space model and record their performances. The model comes from

population ecology, and we have taken it from Chopin and Papaspiliopoulos (2020) in the following form:

$$\begin{aligned} X_0 &\sim N(0,1) \\ X_t &= \tau_0 - \tau_1 e^{\tau_2 X_{t-1}} + U_t, \quad U_t \sim N(0, \sigma_X^2) \\ Y_t &= X_t + V_t, \quad V_t \sim N(0, \sigma_Y^2) \end{aligned}$$

In our simulation, we have run a sequential importance resampling algorithm on this model for different numbers (N) of particles, while keeping $T = 100$. For each value of N, we have repeated the algorithm 100 times and computed the average time for a single run. We present the results in Figure 2.2. The

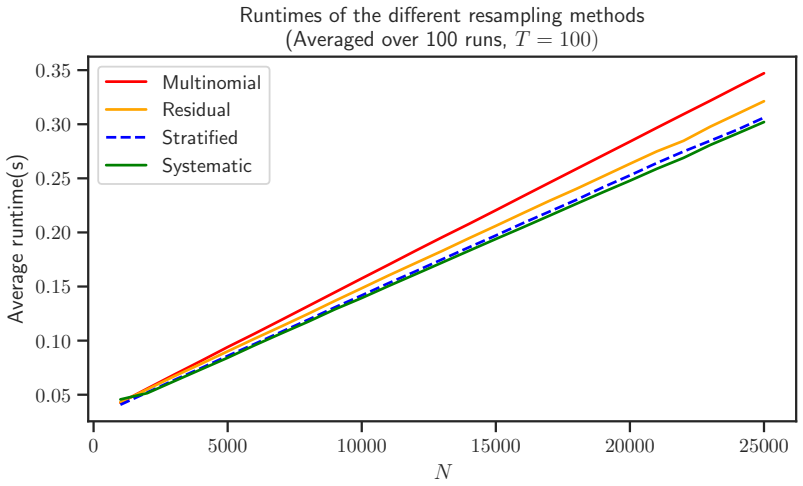


Figure 2.2: Average runtime for the four common resampling methods in sequential Monte Carlo

state space model on which we have tested the resampling schemes is non-linear but is a very low dimensional. Therefore, computational time was small for all cases. It is nevertheless obvious that multinomial resampling has the highest computational cost, while systematic the lowest. cost.

In SIR, resampling is generally not performed at every time step because of two reasons. First, resampling is the most computationally costly step of an SIR algorithm. Second, resampling at every time step might even be redundant, for example, at the time when the particles do not suffer from any degeneracy. Therefore, resampling is performed only when it is deemed necessary by some criteria. One such criteria could be resampling at the every mth time step, where m is some predefined constant. Another way is adaptive resampling, in which the effective sample size (ESS), a popular metric of quality or efficiency of Monte Carlo estimates that is based on weighted samples and estimated from the variance of the particle weights (Kong, Liu and Wong (1994)), is used to monitor the need for resampling. The estimate $\widehat{ESS}$ for the particles can be computed as:

$$\widehat{\mathrm{ESS}}(w_{t,1:n}) = \frac{1}{\sum_{i=1}^{N}(w_{t,i})^2} = \frac{\left[\sum_{i=1}^{N}(\widetilde{w}_{t,i})\right]^2}{\sum_{i=1}^{N}(\widetilde{w}_{t,i})^2} \tag{2.32}$$

where $\{w_{t,i}\}$ and $\{\widetilde{w}_{t,i}\}$ are respectively the normalized and unnormalized weights.

The SIR algorithm can be summarized as

Algorithm 2.4 Sequential Importance Resampling

1 Given a measurement model $p(y_{0:t} \mid x_t)$, a prior $p_0(x_0)$, and the number of particles N.

2 Draw N values from the initial prior:

$$x_{0,i} \sim p_0(x_0), i = 1, \ldots, N,$$

and set weight $w_{0,i} = 1/N$, for all $i = 1, \ldots, N$.

3 **for** *each* $t = 1, \ldots, T$ **do**

4 Draw samples $x_{t,i}$ from the importance distribution

$$x_{t,i} \sim q_t(x_t \mid x_{t-1}, y_{0:t}), \quad i = 1, \ldots, N$$

5 Calculate new, unnormalized weights

$$\widetilde{w}_{t,i} \propto w_{t-1,i} \frac{g_t(y_t \mid x_{t,i}) f_t(x_{t,i} \mid x_{t-1,i})}{q_t(x_{t,i} \mid x_{t-1,i}, y_{0:t})}$$

6 Normalize the weights

$$w_{t,i} = \frac{\widetilde{w}_{t,i}}{\sum_{i=1}^{N} \widetilde{w}_{t,i}}$$

7 If $\widehat{ESS}$ is less than a specific threshold value, perform resampling.

Performance of the SIR algorithm, like other algorithm that relies on the importance sampling, naturally depends on the quality of the importance distribution $q_t(\cdot)$. Its functional form should be such that we can easily draw samples from it and can evaluate the probability densities of the sample points. The optimal importance distribution in terms of variance (see, e.g., Doucet, De Freitas and N. Gordon (2001)) is

$$q_t(x_t \mid x_{0:t-1}, y_{0:t}) = p(x_t \mid x_{t-1}, y_t) \tag{2.33}$$

If the optimal importance distribution however cannot be directly used, efficient importance distributions can sometimes be obtained by indirect methods. One such method is the local linearization in which a mixture of Laplace Approximation, extended Kalman filters (EKF), unscented Kalman filters (UKF), or other types of non-linear filters are used to form the importance distribution (C. A. Naesseth, Lindsten, Schön et al. (2019)). It is also possible to use a Metropolis–Hastings step after, or in stead of, the resampling step to smooth the resulting distribution (Wan and Van Der Merwe (2000)). A particle filter with UKF importance distribution is also referred to as the unscented particle filter (UPF). Similarly, a particle filter with the Gauss–Hermite Kalman filter importance distribution can be termed the Gauss–Hermite particle filter

(GHPF) and one with the cubature Kalman filter importance distribution the cubature particle filter (CPF). However, such a mixture of importance distribution can also lead to the failure of particle filter convergence, demanding care and caution to avoid such a pitfall. Rather than forming the importance distribution by using the Gaussian approximation provided by the EKF, UKF, or other Gaussian filter, it may be recommended to artificially increase the covariance of the distribution or to replace the Gaussian distribution with a Student's t distribution with a suitable number of degrees of freedom (see Cappé, Moulines and Rydén (2009)). By tuning the resampling algorithm to specific estimation problems and possibly changing the order of weight computation and sampling, accuracy and computational efficiency of the algorithm can be improved.

The bootstrap filter (N. J. Gordon, Salmond and Smith (1993)) is a variation of SIR where the dynamic model $f_t(x_t \mid x_{t-1})$ is used as the importance distribution. This makes the implementation of the algorithm easy, but inefficiency of the importance distribution may warrant a very large number of Monte Carlo samples for the estimation. In the bootstrap particle filter, resampling is normally performed at each time step.

Algorithm 2.5 Bootstrap Particle Filter

Given a measurement model $p(y_{0:t} \mid x_t)$ and a prior $p_0(x_0)$.
Sample N particles from the initial prior:

$$x_{0,i} \sim p_0(x_0), i = 1, \ldots, N,$$

Set weight $w_{0,i} = 1/N$, for all $i = 1, \ldots, N$.
for *each* $t = 1, \ldots, T$ **do**
  Draw a new particle $x_{t,i}$ from the dynamic model for each $x_{t-1,i}$
  $x_{t,i} \sim f_t(x_t \mid x_{t-1,i}), \quad i = 1, \ldots, N$
  Calculate new, unnormalized weights

$$\widetilde{w}_{t,i} \propto g_t(y_t \mid x_{t,i})$$

  Normalize the weights

$$w_{t,i} = \frac{\widetilde{w}_{t,i}}{\sum_{i=1}^{N} \widetilde{w}_{t,i}}$$

  Resample with replacement N particles from $x_{0:t,i}$ based on $\widetilde{w}_{t,i}$

Another variation of SIR is the auxiliary SIR (ASIR) filter (Pitt and Shephard (1999)). The ley idea of the ASIR is to mimic the availability of the optimal importance distribution by performing the resampling at step $t-1$ using the available measurement at time t.

2.7 SMC for parameter estimation

While discussing inferences in the state space, we have so far restricted our focus on the estimation of the latent variables $\{x_t\}$ that represent the states, assuming that the parameters of the model are known. In this section, we consider the parameters θ unknown and discuss their estimation, which is often another goal in the analysis of state space models. There are generally two approaches in parameter estimation: Maximum Likelihood Estimation and Bayesian Estimation (see, e.g., (Storvik, 2017).

Maximum Likelihood Estimation

Given a sequence of observations $\{y_{0:t}\}$ and a generic statistical model with likelihood function $\theta \to p(y_{0:t}; \theta)$, a maximum likelihood estimator (MLE) is defined as

$$\widehat{\theta} \in \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(y_{0:t}; \theta). \tag{2.34}$$

With the state space model defined in (2.2a)–(2.2c), the objective amounts to maximizing

$$p(y_{0:t} \mid \theta) = \int_{\mathrm{x}} p(y_{0:t} \mid x_{0:t}; \theta) p(x_{0:t}|\theta)\, dx_t \tag{2.35}$$

The integral is often intractable, so we resort to sequential Monte Carlo methods, and the main approach in this setting is to set

$$p(y_{0:t} \mid \theta) = p(y_0 \mid \theta) \prod_{s=0}^{t} p(y_s \mid y_{0:s-1}; \theta) \tag{2.36}$$

Applying the law conditional probability, $p(y_s \mid y_{0:s-1}; \theta)$ can be written as

$$p(y_s \mid y_{0:s-1}) = \int_{x_s} p(y_s \mid x_s; \theta)\, p(x_s \mid y_{0:s-1})\, dx_s \tag{2.37}$$

The right hand side can now be approximated recursively by an ensemble of particles

$$\int_{x_s} p(y_s \mid x_s; \theta)\, p(x_s \mid y_{0:s-1})\, dx_s \approx \sum_{i=1}^{N} w_{t-1,i}\, p(y_s \mid x_{s,i}; \theta) \tag{2.38}$$

Bayesian Estimation

In the Bayesian approach, we view θ as the realization of a random variable Θ, with a prior distribution $p(\theta)$ that reflects our available knowledge about it. Our aim is then to compute the posterior distribution of Θ, i.e., the distribution of Θ conditional on data $\{y_{0:t}\}$. To proceed on this aim, we start with the joint conditional distribution of the state variables $\{x_t\}$ and θ given $\{y_{0:t}\}$, i.e., $p(x_t, \theta \mid y_{0:t})$ and can assume that at time $(t-1)$ there exist a weighted sample $\{(x_{t-1,i}, \theta_i, w_{t-1,i})\}$ with respect to $p(x_{t-1}, \theta | y_{1:t-1})$. We can then use

the recursive Bayes filtering equation, now including θ as well,

$$\begin{aligned} p(x_t, \theta \mid y_{0:t-1}) &= \int_{x_{t-1}} p(x_t \mid x_{t-1}, \theta) p(x_{t-1}, \theta \mid y_{0:t-1}) \, dx_{t-1} \\ &\approx \sum_{i=1}^{N} w_{i,t-1} p(x_t | x_{i,t-1}, \theta_i) \delta_\theta(\theta_i) \end{aligned}$$

and

$$p(x_t, \theta \mid y_{0:t}) \approx Z \cdot \sum_{i=1}^{N} w_{t-1,i} \, p(x_t \mid x_{t-1,i}, \theta_i) \delta_\theta(\theta_i) \, p(y_t \mid x_t, \theta_i) \tag{2.39}$$

Updated samples $\{\theta_i, x_{i,t}, w_{i,t}\}$ are obtained by simulating $x_{i,t} \sim p(x_t | x_{i,t-1}, \theta_i)$ and update the weights as $w_{t,i} \propto w_{t-1,i} \, p(y_t \mid x_{t,i}, \theta_i)$, The proportionality constant can be taken care of by normalized weights.

CHAPTER 3

Challenges of Sequential Monte Carlo in High Dimension

Since their development in 1993, particle filters have become a standard tool with an immense success for inference and optimal estimation in general state space hidden Markov models, while showing successful applications in ever more complex scenarios (see, e.g., Godsill (2019)). This success has, however, remained limited only to the applications in the low dimensional state space models in general. Particle filters face a fundamental obstacle in the high-dimensional systems. Bootstrap particle filter, for example, perform poorly when the data-points are very informative. The intuitive explanation behind this drawback is that the algorithm samples particles $\{x_t^{d_x}\}$ from $f_t(x_t^{d_x} \mid x_{t-1}^{d_x})$, where d_x is the dimension of the state space, in an unguided way, implying that there is no control mechanism in it to ensure that many of the simulated particles be compatible with the observation $\{y_t^{d_y}\}$, where d_y is the dimension of the observation variable. The problem becomes especially pronounced when the observations are very informative and result in the likelihood $g_t(y_t^{d_y} \mid x_t^{d_x})$ to be a peaked function. This phenomenon happens particularly when d_y is high.

In this chapter, we illustrate the difficulty of sequential Monte Carlo in high-dimensional estimation by studying two linear state space models, one with the transition density and the observation density following the Gaussian distribution and the other with the densities following the Cauchy distribution. These models are characterized by simple dynamics, but still suffer from the weight degeneracy and high estimation error resulting from the curse of dimensionality in the high dimension. We explore, among others, the results of Snyder et al. (2008) and Bengtsson, Bickel and Li (2008) in light of our experimental simulation and analyze their key theoretical results that highlight the difficulty of simulation in high dimensions.

3.1 Model Setting

Our models are described by the following density functions, in line with expressions for a general SSM that we presented in (2.2a)–(2.2c):

$$X_0^{d_x} \sim p_0(x_0^{d_x}; \theta), \tag{3.1a}$$

$$X_t^{d_x} \sim f_t(x_t^{d_x} \mid X_{t-1}^{d_x}; \theta), \quad t \geq 1, \tag{3.1b}$$

$$Y_t^{d_y} \sim g_t(y_t^{d_y} \mid x_t^{d_x}; \theta), \quad t \geq 0. \tag{3.1c}$$

That is, we assume that $X_t^{d_x}$ is drawn conditionally from a prior or proposal distribution $f_t(x_t^{d_x} \mid x_{t-1}^{d_x})$, while new observational data $Y_t^{d_y}$ is related to the state $x_t^{d_x}$ by the conditional density $g_t(y_t^{d_y} \mid x_t^{d_x})$. We assume that this relationship between the states and the observations can be completely described by a deterministic function. Mathematically, $Y_t^{d_y} = G_t(X_t^{d_x}) + \varepsilon_t^{d_y}$, where $\varepsilon_t^{d_y}$ is an additive noise that is taken to be independent of the state $X_t^{d_x}$. Our goal is to approximate, for some function $\psi(\cdot)$, the following posterior expectation

$$\mathbb{E}\left[\psi(x_t^{d_x}) \mid y_{0:t}^{d_y}\right] = \int \psi(x_t^{d_x})\, p(x_t^{d_x} \mid y_{0:t}^{d_y})\, dx_t^{d_x} \tag{3.2}$$

$$= \int \psi(x_t^{d_x}) \frac{g_t(y_{0:t}^{d_y} \mid x_t^{d_x})\, p(x_t^{d_x})}{\int g_t(y_{0:t}^{d_y} \mid x_t^{d_x})\, p(x_t^{d_x}) dx_t^{d_x}} dx_t^{d_x}. \tag{3.3}$$

by the following estimate using importance ratio:

$$\mathbb{E}\left[\psi(x_t^{d_x}) \mid y_{0:t}^{d_y}\right] \approx \hat{E}\left[\psi(x_t^{d_x}) \mid y_{0:t}^{d_y}\right] = \sum_{i=1}^{N} \psi(x_{t,i}^{d_x}) \frac{g_t(y_{0:t}^{d_y} \mid x_{t,i}^{d_x})}{\sum_{j=1}^{n} g_t(y_{0:t}^{d_y} \mid x_{t,j}^{d_x})} \tag{3.4}$$

In this model setting, the normalized, posterior weight $w_{t,i}(x_{t,i}^{d_x})$ associated with ensemble member $x_{t,i}^{d_x}$ is given by

$$w_{t,i}(x_{t,i}^{d_x}) = \frac{g_t(y_{0:t}^{d_y} \mid x_{t,i}^{d_x})}{\sum_{j=1}^{n} g_t(y_{0:t}^{d_y} \mid x_{t,j}^{d_x})}$$

We do not put the dimension to the notation of the weight to indicate that the quantity is a scalar. Our primary objective is to study how the set of N weights behave as the value of d_x or d_y increases. In particular, we would like to verify that one of the particles will dominate the entire ensemble with a very high likelihood $g_t(y_{0:t}^{d_y} \mid x_{t,i}^{d_x})$ compared to the other particles regardless of the time step and, consequently, result in $\max_i(\{w_{t,i}\}) \to 1$ when d_x or d_y becomes high. For high-dimensional state space systems, weight degeneracy is pervasive and appears to hold for a wide variety of prior and likelihood distributions

3.2 Parameter Setting

We mentioned in the previous section that d_y-dimensional observation vector $y_t^{d_y}$ is related to the state variable $x_t^{d_x}$ through a function $G_t(\cdot)$. More specifically, we assume this function to be a simple, linear operator H and model the observation $y_t^{d_y}$ as follows:

$$y_t^{d_y} = H x_t^{d_x} + \varepsilon_t^{d_y}$$

We further assume the following in the *Gaussian* case:

- The prior proposal $p_t(x_t^{d_x})$ is multivariate, isotropic Gaussian, i.e., each component $x_{t,d}, d \in d_x$, of $x_t^{d_x}$ is conditionally independent of all other components, given $x_{t-1,d}$. In other words,

$$x_0^{d_x} \sim \mathcal{N}_{d_x}(\mu_x, I_{d_x})$$

$$x_{t,d} \mid x_{t-1}^{d_x} \sim \mathcal{N}(x_{t-1,d}, 1)$$

For additional simplicity in terms of the computational cost, we take the mean μ_x of the Gaussian to be a d_x dimensional vector of 0 and variance σ_x^2 of each component to be 1, reducing the covariance matrix to the d_x-dimensional identity matrix $\mathbf{I}_{d_x}$. Thus, we have

$$\mathbb{E}\left[x_t^{d_x}\right] = \mu_x = 0, \quad \mathbb{E}\left[x_t^{d_x} x_t^{d_x\intercal}\right] = \sigma_x^2 I_{d_x} = I_{d_x}$$

- In our simulation, we have taken H to be a d_x-dimensional identity matrix. This makes each component of $y_t^{d_y}$ dependent only on the corresponding component of $x_t^{d_x}$.
- We take the additive noise to be Gaussian with mean 0 and covariance matrix an identity matrix.

$$\begin{aligned} &\varepsilon_t^{d_y} \sim \mathcal{N}_{d_y}(\mu_\varepsilon, \mathbf{I}_{d_y}) \\ &\Rightarrow \mathbb{E}\left[\varepsilon_t^{d_y}\right] = \mu_\varepsilon = 0, \quad \mathbb{E}\left[\varepsilon_t^{d_y} \varepsilon_t^{d_y\intercal}\right] = \sigma_x^2 I_{d_y} = I_{d_y} \end{aligned}$$

- Finally, we have stipulated $d_x = d_y$ for additional simplicity.

In the Cauchy case, we take all the densities to follow Cauchy distribution including the noise term. Thus, we have the following transition and observation densities:

$$\begin{aligned} x_0^{d_x} &\sim \text{Cauchy}(0, 1). \\ x_t^{d_x} &= x_{t-1}^{d_x} + U_t, \quad U_t \sim \text{Cauchy}(0, 1) \\ y_t^{d_y} &= H x_t^{d_x} + V_t, \quad V_t \sim \text{Cauchy}(0, 1) \end{aligned}$$

3.3 Simulations and Outcomes

With the model presented above, we have run a bootstrap particle filter for nine sets of state-space dimension and the number of particles: (d_x, N). We first present results from the Gaussian case, followed by those from the Cauchy case later in the section.

3.3.1 The Gaussian case

Weight degeneracy

For each set, we have let the process evolve till $T = 50$ and computed the N weights of the particles at the final time step, stored them in a list, and noted the maximum weight in the list. We then repeated the filtering algorithm for a total of 1000 times. Each histogram in Figure 3.1 summarizes the 1000 maximum weights for one set of (d_x, N) in the Gaussian case. The figure depicts the challenges posed by the dimension of the state space on the performance of the bootstrap particle filter.

When the dimension is small, e.g., when $d_x = 5$, most of the maximum weights fall between 0 and 0.2, indicating that there is no single particle that

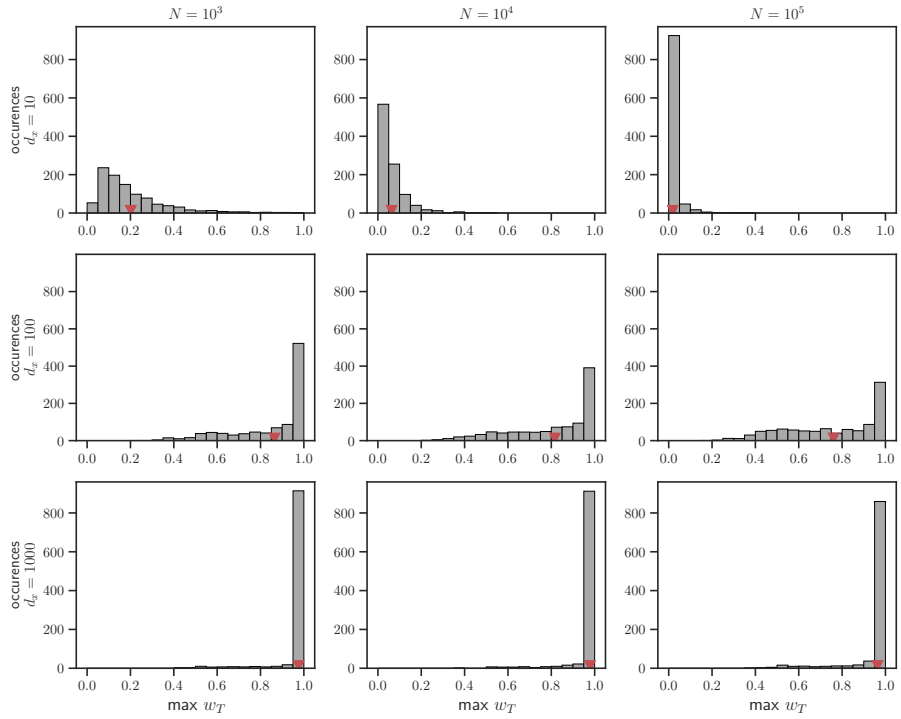


Figure 3.1: Sampling distribution of the maximum weight of the particles for nine sets of (d_x, N). Each histogram is based on 1000 simulations of the process at time $T = 50$. In each simulation, the maximum weight, max w_T, of the ensemble of particles is computed. The red triangle points to the average of all the 1000 maximum weights.

dominates the particle ensemble in terms of the weight. As the dimension starts to increase, some particles start experiencing higher weights, but the scenario is not so severe even at $d_x = 10$ and can further be ameliorated by increasing the size N of the ensemble. However, when the dimension of the state-space becomes 1000, the number of particles even at the order of 10^5 is not enough to improve performance of the algorithm. We summarize the proportion of the dominating cases in which $\max_i(w_i) > 0.5$ in Table 3.1:

Table 3.1: Percentages of cases where a single particle dominates (with normalized weight $> .5$) the entire particle ensemble in the bootstrap filter.

max $w_T > 0.5$		N_x		
		10^3	10^4	10^5
d_x	5	0.2%	0%	0%
	10	5.8%	0.1%	0%
	100	95.3%	90.5%	83.8%
	1000	99.9%	99.6%	99.2%

As we see from the table, when $d_x = 5$, only in 0.2% cases, one of the particles have weight greater than 0.5 if we choose $N_x = 10^3$. However, with $d_x = 10$ and $N_x = 10^3$, we observe that 5.8% of the cases have max$W_T > .5$. The situation improves if we increase the number of particles toward 10^5. However,

the algorithm becomes unreliable in almost 100% of the time even when the number of particles is 10^5. By applying a better proposal distribution, one can reduce the problem, but the problem will essenially stay.

Runtime

We have also recorded the runtimes of 1000 runs of the algorithm for the nine sets of (d_x, N). Table 3.2 below presents the average time spent in one run.

Table 3.2: Average runtime of a bootstrap particle filter in linear state space with Gaussian prior for different sets of state space dimension and number of particles.

Runtime (sec)		N		
		10^3	10^4	10^5
d_x	5	7.0×10^{-2}	2.5×10^{-1}	2.2×10^{0}
	10	1.2×10^{-1}	4.6×10^{-1}	4.1×10^{0}
	50	5.7×10^{-1}	2.3×10^{0}	2.2×10^{1}
	100	1.1×10^{0}	4.5×10^{0}	4.3×10^{1}
	1000	1.1×10^{1}	5.0×10^{1}	4.5×10^{2}

The table shows that the runtime is approximately directly proportional to the size of the dimension of the state space for a given number of particles. This simple proportional relationship might stem from the fact that we have stipulated the components of the latent variable to be independent of each other, and so was the case for the components of the observation variable. If the covariance matrix of the multivariate Gaussian or the linear operator H were not an identity matrix, the computational cost could have increased at a greater rate than a simple proportion due to more interactions among the variables.

On the other hand, the runtime shows some non-linear relationship to the number of particles. The algorithm takes about 4 times as much time to complete one run with $N = 10^4$ as with $N = 10^3$; the runtime becomes 36 times with $N = 10^5$ compared to the runtime with $N = 10^3$. This is mainly due to the resampling step, although we have chosen systematic resampling, the fastest among the four commonly used resampling methods.

Lack of particle diversity

Another problem, related to weight collapse, of the particle filter is lack of diversity among the particles as they evolve over time, as illustrated by the particle ancestry or genealogical tree in Figure 3.2. The tree shows that when the dimension of the state space is low, particles evolve from many different ancestors, ensuring diversity in the particle population. However, even when the dimension is moderate, for example, when $d_x = 100$ in our experiment, all the particles up to time $t = 48$ evolve from a single ancestor.

Effects of weight collapse

We examine the effects of weight collapse on the posterior mean and posterior variance of states. For our linear Gaussian model, we note (we drop the notations

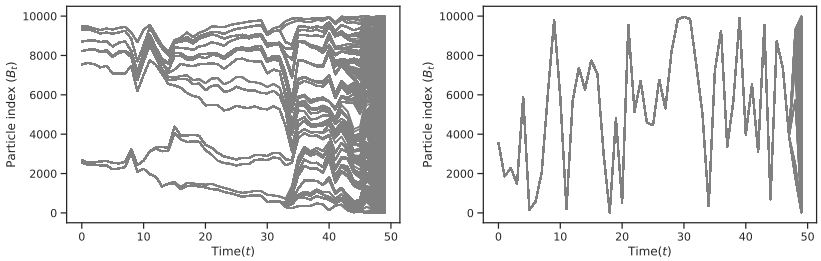


Figure 3.2: Ancestry trees generated by the bootstrap particle filter using multinomial resampling and $N = 10^4$ particles and $T = 50$ time steps. (left) tree with $d_x = 5$ and (right) tree with $d_x = 100$.

d_x, d_y for the dimensions, which are to be understood here unless explicitly stated otherwise):

$$\begin{aligned}\Sigma_{x_t y_t} &= \mathbb{E}[x_t y_t^\intercal] = \mathbb{E}[x_t(Hx_t + \varepsilon_t)^\intercal] \\ &= \mathbb{E}[x_t x_t^\intercal H^\intercal] + \mathbb{E}[x_t \varepsilon_t^\intercal] \\ &= \mathbb{E}[x_t x_t^\intercal]\mathbb{E}[H^\intercal] + \mathbb{E}[x_t]\mathbb{E}[\varepsilon_t^\intercal] \\ &= I_{d_x} \times I_{d_x} + 0 \times 0 = I_{d_x}. \end{aligned} \tag{3.5}$$

$$\begin{aligned}\Sigma_{y_t y_t} &= \mathbb{E}[y_t y_t^\intercal] = \mathbb{E}[(Hx_t + \varepsilon_t)(Hx_t + \varepsilon_t)^\intercal] \\ &= \mathbb{E}[Hx_t x_t^\intercal H^\intercal] + \mathbb{E}[Hx_t \varepsilon_t^\intercal] \\ &\qquad + \mathbb{E}[\varepsilon_t x_t^\intercal H^\intercal] + \mathbb{E}[\varepsilon_t \varepsilon_t^\intercal] \\ &= I_{d_x} + 0 + 0 + I_{d_x} = 2I_{d_x}. \end{aligned} \tag{3.6}$$

According to the Gauss-Markov theorem, the posterior mean vector $x_t^{\mathsf{c}} \coloneqq \mathbb{E}[x_t \mid y_t]$ at time t is given by

$$\begin{aligned} x_t^{\mathsf{c}} &= \mu_{x_t} + \Sigma_{x_t y_t}\Sigma_{y_t y_t}^{-1}\,(y_t - \mu_{x_t}) \\ &= 0 + I_{d_x}(2I_{d_x})^{-1}(y_t - 0) = y_t/2. \end{aligned} \tag{3.7}$$

In our bootstrap particle filtering experiment, x_t^{c} is estimated as

$$x_t^{\mathsf{c}} \approx \hat{x}_t^{\mathsf{c}} = \sum_{i=1}^{N} w_i x_{t,i}^{\mathsf{p}} \tag{3.8}$$

where the superscript p indicates a prior quantity.

The poster variance at time t is given by

$$\begin{aligned}\mathbb{V}[x_t \mid y_t] &= I_{d_x} - I_{d_x}H^\intercal[HI_{d_x}H^\intercal + I_{d_x}]^{-1}HI_{d_x} \\ &= I_{d_x} - I_{d_x}[2I_{d_x}]^{-1}I_{d_x} = I_{d_x}/2 \end{aligned} \tag{3.9}$$

The expected square loss of x_t^{c} over all the d_x components

$$\mathbb{E}\left[|x_t^{\mathsf{c}} - x_t|^2\right] = \mathbb{E}\left[\left|\frac{x_t^{\mathsf{p}} + y_t}{2} - x_t\right|^2\right]$$

$$
\begin{aligned}
&= \mathbb{E}\left[\left|\frac{x_t^{\mathsf{p}} - x_t}{2} - \frac{y_t - x_t}{2}\right|^2\right] \\
&= \mathbb{E}\left[\left|\frac{x_t^{\mathsf{p}} - x_t}{2}\right|^2\right] + \mathbb{E}\left[\left|\frac{y_t - x_t}{2}\right|^2\right]
\end{aligned}
$$

Due to that $y_t - x_t = \varepsilon_t$ which is independent of x_t. Further, since $\mathbb{E}|x_t^{\mathsf{p}} - x_t|^2 = \mathbb{E}|y_t - x_t|^2 = d_x$, we get

$$\mathbb{E}(|x_t^{\mathsf{c}} - x_t|^2) = \frac{1}{4}d_x + \frac{1}{4}d_x = \frac{1}{2}d_x$$

The correct posterior mean estimated by the filter

$$\hat{x_t}^{\mathsf{c}} = \sum_{i=1}^{N} w_{t,i} x_{t,i}^{\mathsf{p}}$$

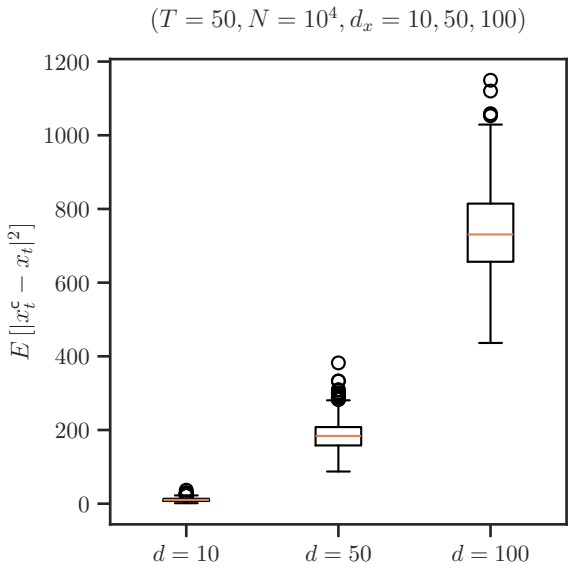


Figure 3.3: Expected squared error of the posterior mean x^{c} in the bootstrap particle filter on a linear Gaussian state space with three different cardinalities of dimension.

In our experiment, we have fixed $T = 50$ and $N = 10^4$ and run the filtering algorithm 1000 times. In each run at the final time step, we have computed the d_x dimensional Monte Carlo estimate of the state by taking the weighted average of the particles and compared this estimate with the d_x dimensional true state to obtain the aggregated squared error over all dimensions (square of the L_2-norm between the estimated state and the true state). We have repeated the 1000 runs for $d_x = 2, 5, 10, 50, 100,$ and 1000.

Figure 3.3 presents the histogram of the squared errors for each value of d_x, except for 1000, which we exclude from the plot because of their large errors affecting the vertical scale of the plot. Averaging the squarer error over the 1000 runs, we have got the mean squared error of $10.8, 186.0, 737.7,$ and 42109.2 for

$d_x = 10, 50, 100,$ and 1000, respectively. Thus, x^{c} is a very poor estimator of the posterior mean even for small dimensions and as the dimension increases, the squarer error increases at an exponential rate of the dimension.

Effect on ensemble size on posterior mean

To study the effect of the number of particles on the squared error of the estimated posterior mean, we have chosen a set of three dimension sizes $d_x = 10, 20, 50$, and for each value of d_x, we have run the bootstrap particle filter with a different number of particles between $N = 100$ and $N = 10^5$, inclusive. For each combination of (d_x, N), we have repeated the algorithm 100 times and taken the mean squared error of posterior mean described earlier. The scatter plot in Figure 3.4 below shows the rate of reduction of the squared error with the increase in the number of particles..

We find that the squared error reduces at an increasingly lower rate as the ensemble size increases and requires an exponentially high number of particles to keep the reduction at an appreciable level. For example, for $d_x = 10$, there is a very little reduction in the error if we choose $N = 10^5$, compared to $N = 10^4$. For $d_x = 20$, the squared error decreases from 29.3 with $N = 10^5$ to 23.9 with $N = 10^3$, amounting to an 18% gain in the performance; the error is nevertheless significantly higher that theoretical error we derived above.

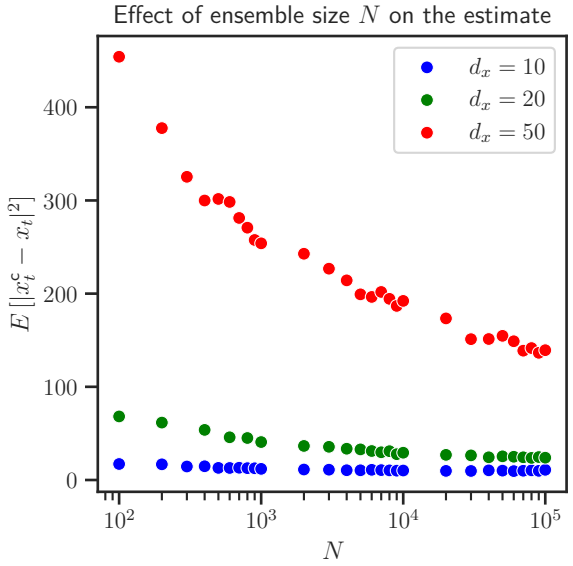


Figure 3.4: Effect of the ensemble size three different dimensions $d_x = 10, 20, 30$. Each scatter point is an average squared error based on 100 simulations of the process at time $T = 50$.

3.3.2 The Cauchy case

Weight degeneracy

We present the sampling distributions of the weights for the nine sets of (d_x, N) in Figure 3.5.

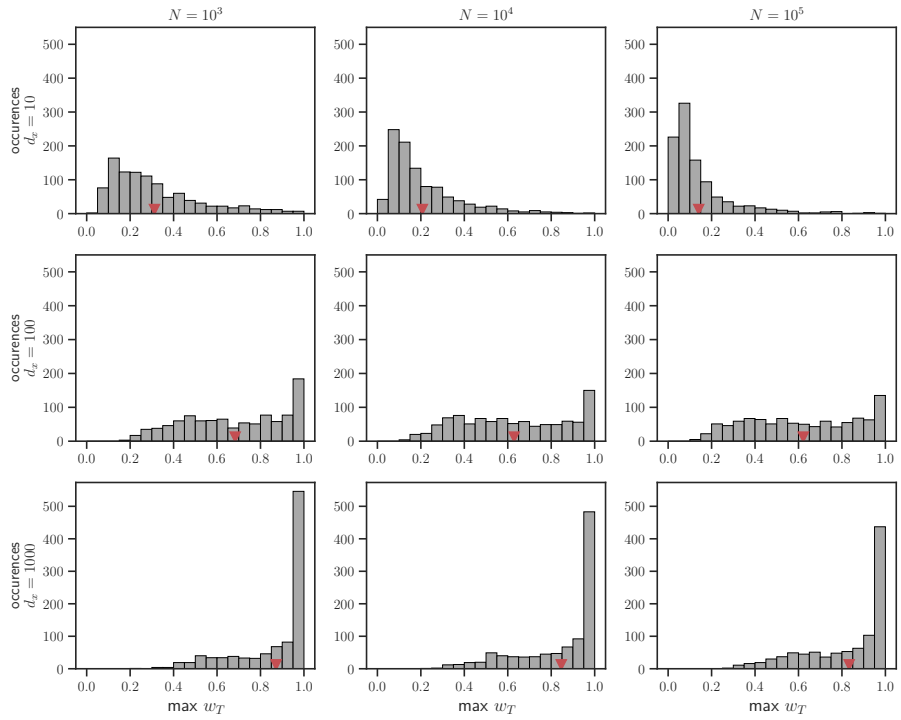


Figure 3.5: Sampling distribution of the maximum weight of the particles for nine sets of (d_x, N). Each histogram is based on 1000 simulations of the process at time $T = 50$. In each simulation, the maximum weight, max w_T, of the ensemble of particles is computed. The red triangle points to the average of all the 1000 maximum weights.

While the bootstrap particle filter with a Cauchy prior suffers, quite expectantly, from the weight collapse in high dimension, the weight collapse is, interestingly, not as severe as in the Gaussian case, as evidenced by the plot. When the dimension of the state space is 1000, the filter with Gaussian prior had a very negligible percentage of cases with the maximum weight of a particle greater than 0.5. The percentage of cases for Cauchy prior, through not significant, was not close to zero. We summarize the proportion of dominating particles when $\max_i(w_i) > 0.5$ in Table 3.3.

Runtime

By comparing the runtimes for the Cauchy case with those in the Gaussian case from Table 3.2, we find that using a Cauchy prior takes about *three times* as much runtime in a single run by the bootstrap filter as in the Gaussian case. The results suggest that simulating from the Cauchy distribution is much costlier than simulating from the Gaussian distribution.

Table 3.3: Percentages of cases where a single particle dominates (with normalized weight $> .5$) the entire particle ensemble in the bootstrap filter.

max $w_T > 0.5$		N_x		
		10^3	10^4	10^5
d_x	5	4.8%	1.6%	0.4%
	10	16.7%	7.3%	3.7%
	100	72.6%	62.2%	63.5%
	1000	95.3%	93.3%	92.2%

Table 3.4: Average runtime of a bootstrap particle filter in linear state space with Cauchy prior for different sets of state space dimension and number of particles.

Runtime (sec)		N		
		10^3	10^4	10^5
d_x	5	2.0×10^{-1}	7.6×10^{-1}	6.5×10^{0}
	10	3.9×10^{-1}	1.5×10^{0}	1.3×10^{1}
	50	1.8×10^{0}	7.3×10^{0}	6.6×10^{1}
	100	3.8×10^{0}	1.5×10^{1}	1.3×10^{2}
	1000	3.8×10^{1}	1.5×10^{2}	1.5×10^{3}

3.4 Discussion

Particle filters suffers from a fundamental problem, especially in a moderate to high dimensional state space, that particle weights collapse, with one of particle receiving a posterior weight close to unity while all other particles effectively getting a zero weight. We have illustrated this phenomena through simulations of the bootstrap particle-filter, but with resampling, in some of the simplest cases. Even in these simple cases, preventing weight collapse of the particles demand an ensemble size that must increase exponentially with the dimension size. This renders traditional particle filtering algorithms in the inference of high dimensional state space essentially useless.

CHAPTER 4

Particle Filters in High Dimension

We have noted in chapter 3 that bootstrap particle filter performs very poorly, experiencing weight collapse, when the dimension of the state space increases. The problem can be mitigated partly by increasing the number of particles at an exponential rate of the dimension of the state space. However, this approach quickly becomes impractical because of the exponential demand on the computational resources. For example, in weather forecasting applications, where the dimension may exceed 10^7, particle filter obviously becomes infeasible. Even an analytical method like the Kalman filter, assuming it is a plausible option, will prove to be too computationally expensive in such cases

For high dimensional state space models, it becomes crucial to devise guided particle filters. Specifically, the construction of proposal distributions that alleviate weight degeneracy is essential. Unfortunately, there is no universal recipe to date for generating such effective proposals (Chopin and Papaspiliopoulos (2020), ch. 17).

There have, however, recently been several interesting developments of particle filtering in the high dimensional setting. An accessible review of these algorithms can be found in Van Leeuwen et al. (2019). In this chapter, we study a few classes of algorithms that have emerged to deal specifically with inferences in high dimensional state space and have shown some promise of their capability, and in the next chapter, we compare their performance based on results from experimentation and simulation.

4.1 Localized Particle Filter

In many practical cases of state-space models, the high dimension of the system often stems from a large number of locations that each have a common set of state variables. In the SIR (Susceptible, Infectious, or Recovered) model of the transmission of a disease during a pandemic, for example, Covid, we might consider all the world's cities as locations and the number of S, I, andR individuals in each city as the state variables. Another case is weather forecasting for which we are interested in knowing the temperature, humidity, wind speed, precipitation, etc., of all the world cities. The idea of localization is based on the assumption that the properties at a specific location is limited to depend on the observations from a small area around the location.

Formally, let us discuss the problem at a particular time point t and consider that the locations are spread in a large network of grid points. We define

$\{x_{k,t}\}$ as the state at grid point k at time t. Each grid point has several model variables, so each $\{x_{k,t}\}$ is generally a vector. The key assumption that we make is that the posterior of the state at k depends only on a subset of the observations—observations in the grid points within a *neighborhood*—in which the neighborhood is defined by some criteria. Let us denote this *subset* of observations $y_{[\mathcal{K}],t} \in \{y_{k,t}\}$. We can then use the following approximation in the update step of the filtering model:

$$p(x_{k,t} \mid y_t) \approx p(x_{k,t} \mid y_{[\mathcal{K}],t}). \tag{4.1}$$

As a consequence of the assumption above, the observations $y_{[\mathcal{K}],t}$ depend not on the whole state vector but only on a part of it, which we denote by $x_{(\mathcal{K}),t} \in x_t$. We have then the following approximation for the prediction step of filtering:

$$p(y_{[\mathcal{K}],t} \mid x_t) \approx p(y_{[\mathcal{K}],t} \mid x_{(\mathcal{K}),t}) \tag{4.2}$$

The pdf $p(x_{k,t} \mid y_{[\mathcal{K}],t})$ can be rewritten as an integral over the joint pdf:

$$p(x_{k,t} \mid y_{[\mathcal{K}],t}) = \int p(x_{(\mathcal{K}),t} \mid y_{[\mathcal{K}],t}) dx_{(\mathcal{K})\setminus k}, \tag{4.3}$$

where $(\mathcal{K}) \setminus k$ is the set of grid points excluding the grid point k.

By the Bayes' rule and the particle approximation of the pdf as the weighted sum of Dirac delta function, we get

$$\begin{aligned} p(x_{(\mathcal{K}),t} \mid y_{[\mathcal{K}],t}) &= \frac{p(y_{[\mathcal{K}],t} \mid x_{(\mathcal{K}),t})}{p(y_{[\mathcal{K}],t})} p(x_{(\mathcal{K}),t}) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(y_{[\mathcal{K}],t} \mid x_{(\mathcal{K}),t,i})}{p(y_{[\mathcal{K}],t})} \delta(x_{(\mathcal{K}),t} - x_{(\mathcal{K}),t,i}) \\ &= \sum_{i=1}^{N} w_{(\mathcal{K}),t,i} \delta(x_{(\mathcal{K}),t} - x_{(\mathcal{K}),t,i}) \end{aligned} \tag{4.4}$$

Therefore, we have the following particle approximation of the posterior at k :

$$p(x_{k,t} \mid y_{[\mathcal{K}],t}) \approx \sum_{i=1}^{N} w_{(\mathcal{K}),t,i} \delta(x_{(\mathcal{K}),t} - x_{(\mathcal{K}),t,i}) \tag{4.5}$$

It is to be noted that $w_{(\mathcal{K}),t,i}$ depend only on the local observations $y_{[\mathcal{K}],t}$ and the local prior particles $x_{(\mathcal{K}),t,i}$, which are sampled from $p(x_{(\mathcal{K}),t} \mid x_{(\mathcal{K}),t-1})$. A consequence of this local sampling is that the variance of the weights will be much smaller.

By repeating the localization procedure for all grid points , we obtain all marginals of the posterior pdf. The challenge is that the we do not have a common ensemble of particles for all locations; the particles and the weights are rather location (neighborhood) dependent and so change from one grid point to the next. Attaining a consistent posterior for any pairs of state values $(x_{k,t}, x_{l,t})$, for any triplets of state values, and so on requires combining different local ensembles of particles effectively into a global ensemble that can propagate with model equations successfully has remained a great challenge until the recent advent of several ingenious methods, some of which are the focus of the next sections.

4.2 Block Particle Filter

The key idea of the block particle filter (BPF; Rebeschini and Van Handel (2015)) is to decompose the high-dimensional state space $\mathbb{R}^{d_x}$ into a set of lower dimensional blocks. At each time step t, multiple standard particle filter algorithms are run in parallel on the blocks, one algorithm on each block. The filtering distribution over the whole state space is then approximated as the product of the lower dimensional approximations on each block.

4.2.1 Core idea of BPF

Rebeschini and Van Handel (2015) sets the foundation of their algorithm with a trivial setting and demonstrates how the curse of dimensionality can be surmounted. Let $V = \{1, 2, \ldots, d\}$ be a finite index set and, for each element $v \in V$, $(X_{t,v}, Y_{t,v})_{v \in V}$ be a hidden Markov model that takes values in a measurable space $(E_v \times F_v, \mathcal{E}^v \bigotimes \mathcal{F}^v)$. Therefore, we have in this system a total of d Markov chains evolving simultaneously over the time. We further assume that each Markov chain is independent of all the other chains, i.e., $(\{X_{t,v}\}, \{Y_{t,v}\})_{v \in V}$ and $(\{X_{t,w}\}, \{Y_{t,w}\})_{w \in V}$ do not depend on each other for $v \neq w$.

This system thus represents a trivial yet high-dimensional model in which the dimension of the state space is d. This is one of the scenarios that we discussed in chapter 3 to demonstrate the limitation of applying a single sequential importance resampling (SIR) algorithm on the high dimensional case. A single SIR step, as we find from our filtering experiments in replicating some of the studies of Bengtsson, Bickel and Li (2008), results for the target distribution in an approximation that quickly deteriorates with increasing d even for this trivial setting. Intuitively, the underlying cause is that in the high dimensional models, the proposal distribution $p(X)$ and the desired sampling distribution become approximately mutually singular and essentially have disjoint support. Consequently, the density of the desired distribution at all points of the proposed ensemble of particles is small, with a very small fraction of density values predominating in relation to the others. Specifically, Bengtsson, Bickel and Li (2008) proved that the number of Monte Carlo samples N must increases at an exponential rate in dimension d of the state space to avoid weight collapse of particles.

Though this trivial model may have little relevance to the reality, it shows that it is possible to deal with such a high dimensional system because the state space is locally low-dimensional. If we are then interested in local errors, i.e., marginals of the filtering distribution on spatial regions of a fixed, smaller dimension, which can be as low as 1, rather than the global measure of error, such modeling approach will have some utility.

For example, if we have a state space whose large dimension arises from its spatial structure, such as geographical sites in case of meteorology or oceanography, and if the neighboring sites are correlated with each other while regions sufficiently apart experience decayed correlations to point of being practically non-existent, we can make an 'approximation' and apply the local filtering model. Large-scale interacting systems can often exhibit an approximate version of this property, referred to as the "decay of correlations" or, "the rate of approach of some initial distribution to an invariant one", has

been particularly studied in many fields, particularly in statistical mechanics and stochastic differential equations (Liverani (1995)). To express informally, this phenomena implies that the states $(x_{t,v}, y_{t,v})$ and $(x_{t,w}, y_{t,w})$ at any two sites $v, w \in V$ are perhaps strongly correlated when v and w are close to each other; however, these states can be expected to be nearly independent when v and w are located far apart as measured with respect to the natural distance d in the graph G (that is, $d(v, w)$ is the length of the shortest path in G between $v, w \in V$). The idea then follows that the decay of correlations phenomena also makes, in practice, many non-trivial state space models locally low-dimensional, in the sense that the conditional distribution at each site only needs to be updated by observations in a neighborhood whose size is independent of the overall dimension. That is, for some large enough distance b,

$$p(x_{t,v} \in \cdot \mid y_0, \ldots, y_t) \approx p(x_{t,v} \in \cdot \mid y_{w,1}, \ldots, y_{t,w}), d(v, w) \leq b), \tag{4.6}$$

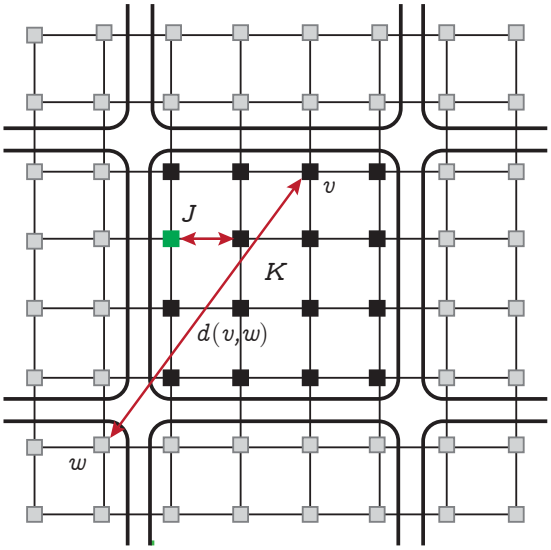


Figure 4.1: Schematic diagram of a block particle filter that decomposes the high dimensional state space into lower dimensional blocks exploiting the decay of correlations phenomena.

In brief, the hidden Markov model $(X_t, Y_t)_{t\geq 0}$ at each time t is assumed to be a random field $(\{X_{t,v}\}, \{Y_{t,v}\})_{v\in V}$ indexed by a finite undirected graph $G = (V, E)$, in which V stands for the set of vertices and E for the set of edges. The graph describes the location relationship of data, evaluated by the decay of correlations, and the spatial degree of freedom of the model.

The state spaces $\mathbb{X}$ and $\mathbb{Y}$ of X_t and Y_t, respectively, are of the following product forms:

$$\mathbb{X} = \prod_{v\in V} \mathbb{X}_v, \qquad \mathbb{Y} = \prod_{v\in V} \mathbb{Y}_v,$$

The transition and observation densities are given by

$$f(x_t \mid x_{t-1}, \theta_t) = \prod_{v\in V} f_v(x_{t,v} \mid x_{t-1}, \theta_{t,v}), \quad g(y_t \mid x_t, \theta_t) = \prod_{v\in V} g_v(y_{t,v} \mid x_{t,v}, \theta_{t,v}),$$

where $f_v : \mathbb{X} \times \mathbb{X}_v \to \mathbb{R}_+$ and $g_v : \mathbb{X}_v \times \mathbb{Y}_v \to \mathbb{R}_+$.

To formally define the block particle filtering algorithm, the vertex set V is partitioned into K non-overlapping blocks, so that

$$V = \bigcup_{K \in \mathcal{K}} K, \qquad K \cap K' = \varnothing \text{ for } K \neq K', K, K' \in \mathcal{K}.$$

A blocking operator, B, is introduced to deal with the marginals of product distributions at the block level:

$$\mathsf{B}(\rho) \coloneqq \bigotimes_{K \in \mathcal{K}} \mathsf{B}_K(\rho),$$

where $\mathsf{B}_J(\rho)$ denotes the marginal of ρ on $\prod_{v \in J} \mathbb{X}_v$ for any measure ρ on $\mathbb{X} = \prod_{v \in V} \mathbb{X}_v$ and $J \subseteq V$. The random field described by the measure $\mathsf{B}(\rho)$ on $\mathbb{X}$ is independent, following the decay of correlation phenomena, across different blocks defined by the partition $\mathcal{K}$, while the marginal on each block agrees with the original measure ρ. The block particle filter inserts an additional blocking step into the bootstrap particle filter recursion, that is,

$$\hat{\pi}_0^\mu = \mu, \qquad \hat{\pi}_t^\mu = \hat{\mathsf{F}}_t \hat{\pi}_{t-1}^\mu \qquad (t \geq 1),$$

where $\hat{\mathsf{F}}_t \coloneqq \mathsf{C}_t \mathsf{B} \mathsf{S}^N \mathsf{P}$ consists of four steps in a sequence:

$$\hat{\pi}_{t-1}^\mu \xrightarrow[\text{sampling}]{\text{prediction}} \hat{\pi}_{t|t-1}^\mu = \mathsf{S}^N \mathsf{P} \hat{\pi}_{t-1}^\mu \xrightarrow[\text{correction}]{\text{blocking}} \hat{\pi}_t^\mu = \mathsf{C}_t \mathsf{B} \hat{\pi}_{t|t-1}^\mu$$

The resulting algorithm is presented below:

Algorithm 4.1 Block Particle Filter

Data: Fix the number of time-steps $T \geq 1$ and the number of particles $N \gg 1$. Observations $y_1, \ldots, y_T$ are given.
Initialize $\hat{\pi}_0^\mu = \mu$;
Sample $\hat{x}_{0,i} \sim \hat{\pi}_0^\mu, i = 1, \ldots, N$;
for $t = 1, \cdots, T$ **do**
  Resample i.i.d. $\hat{x}_{t-1,i} \sim \hat{\pi}_{t-1}^\mu, i = 1, \ldots, N$;
  for each $v \in V$ **do**
    Sample $x_{t,v,i} \sim f_v(\hat{x}_{t-1,i}, \cdot), i = 1, \ldots, N$;
    Compute
    $$w_{t,K,i} = \frac{\prod_{v \in K} g_v(x_{t,v,i}, y_{t,v})}{\sum_{l=1}^N \prod_{v \in K} g_v(x_{t,v,l}, y_{t,v})}, i = 1, \ldots, N, K \in \mathcal{K}$$
  Let $\hat{\pi}_t^\mu = \bigotimes_{K \in \mathcal{K}} \sum_{i=1}^N w_{t,K,i} \delta_{x_{t,K,i}}$;
Compute the approximate filter $\pi_t^\mu f \approx \hat{\pi}_t^\mu f$.

To summarize, at each time step t, a standard particle filter is applied to each block K. The approximation of the filtering distribution over the whole the state space is then obtained by multiplying the approximations of the distribution for each block. In simpler terms, Rebeschini and Van Handel (2015) replaces the global resampling step of the bootstrap particle filter with a local resampling step. This new approach involves independently resampling

neighborhoods to generate new particles. In their theoretical discussion, they provide an overarching perspective on the problem. They specifically focus on the diminishing correlation between local values as spatial distance increases. They argue that this phenomenon is analogous to the well-understood temporal stability property of nonlinear filters.

4.2.2 Key results

Let r be stipulated as the maximum distance (l_1 norm) between any two vertices (v, v') in the undirected graph G for the vertices to be considered in the same neighborhood. Neighborhood implies that the state of a vertex v at time t depends only on the states of the vertices located in the same neighborhood at the previous time step $(t-1)$. The neighborhood of v is thus defined as

$$N(v) \coloneqq \{v' \in V, d(v, v') \leq r\}$$

Let the following be defined

$$\Delta \coloneqq \max_{v \in V} \operatorname{card}\{v' \in V, d(v, v') \leq r\}$$
$$\Delta_{\mathcal{K}} \coloneqq \max_{K \in \mathcal{K}} \operatorname{card}\{K' \in \mathcal{K}, d(K, K') \leq r\}$$

Here, the block distance $d(K, K')$ is defined as the minimum of the distances between all the pairs of vertices that can be formed by taking a vertex from each of the two blocks, K, K'.

Suppose that $\epsilon \leq f_v(x_{t,v} \mid x_{t-1}) \leq \epsilon^{-1}$ for all v. Then for any $J \subseteq K \in \mathcal{K}$,

$$\|\pi_t - \hat{\pi}_t\| \leq \underbrace{c_1 e^{-\gamma_1 d(J, K^c)}}_{\text{bias}} + \underbrace{c_2 \frac{e^{\gamma_2 \operatorname{card}(K)}}{\sqrt{N}}}_{\text{variance}},$$

provided $\epsilon \geq \epsilon_0$, where $\epsilon_0, c_1, c_2, \gamma_1, \gamma_2$ do not depend on $\operatorname{card}(V)$ or n. The result suggests that increasing the block size to include more vertices (dimensions) in the block increases the variance while reducing the bias, and vice versa. Optimizing block size is expected to yield consistent algorithm uniformly in time and dimension.

4.3 Space-time Particle Filter

The Space-time Particle Filter proposed by Beskos et al. (2017) is an improvement of the Location Bootstrap Filter of Briggs, Dowd and Meyer (2013). The word location implies that the particle filter is applied over locations that form the spatial domain of the state space. A state space system that is manifested over a set of distinct locations at each time step is referred to as a spatio-temporal system. To apply the Location Particle Filter of Briggs, Dowd and Meyer (2013),

- first, an ordered sequence of locations, $(1, \ldots, L)$, must be defined such that location l and location $l+1$ are spatial neighbors for each $l \in 1, \ldots, L$. If the L locations are situated in a straight line, the order arises naturally and can be recognized easily. In the case of two-dimensional or, more realistically, the three-dimensional arrangement of the locations, the

locations can be thought of situated in $2D$ or $3D$ grids, where there are many possible ways to define the order.

- To compute the observation update at time step t, a sample must be drawn from the filtering density. In the standard state space case without distinguishing the locations, this density as given by equation (2.7) is

$$p(x_t \mid y_{0:t}) = \frac{g(y_t \mid x_t)\, p(x_t \mid y_{0:t-1})}{\int g(y_t \mid x_t)\, p(x_t \mid y_{0:t-1})\, dx_t}.$$

  In case of the location particle filter, the filtering density takes the following form:

$$p(x_{t,1:L} \mid y_{0:t,1:L}) = \frac{g(y_{t,1:L} \mid x_{t,1:L})\, p(x_{t,1:L} \mid y_{0:t-1,1:L})}{\int g(y_{t,1:L} \mid x_{t,1:L})\, p(x_{t,1:L} \mid y_{0:t-1,1:L})\, dx_t}.$$

- The location-domain state-space model is initialised using the time-domain prediction pdf $p(x_{t,1:L} \mid y_{0:t-1,1:L})$. An ensemble of N particles $\{x_{t,1:L,i} \mid y_{0:t-1,1:L}\}_{i\in 1,\dots,N}$ from this pdf is available from the time-domain prediction step. A standard particle filter is initialized by specifying the pdf at location $l = 1$ and calculating the weight $p(y_{t,1} \mid x_{t,1,i})$ for each prior particle i, and performing resampling using these weights over the *whole spatial domain*. This implies that these resampled particles are now samples of $p(x_{t,1:L} \mid y_{t,1:1})$. A small amount of noise, whose choice is a matter of active research, is added to avoid identical particles.

- The filtering algorithm then moves to the next grid point $l = 2$, computes the weights of $p(y_{t,2} \mid x_{t,2,i})$, and samples the full ensemble adding a small noise to prevent ensemble collapse and finally generating samples from the posterior $p(x_{t,1:L} \mid y_{t,1:2})$. This procedure is repeated for all grid points until we have samples from $p(x_{t,1:L} \mid y_{t,1:L})$.

While the location bootstrap filter uses jitter density to avoid identical particles, Beskos et al. (2017) exploit the spatial transition density $p(x_{t,l} \mid x_{t,l-1}, x_{t-1,1:L})$, in which t is the time index and l the spatial index. So they exploit the pdf of the state $x_{t,l}$ at time t and grid point l, conditioned on all previous grid points $x_{t,1:l-1}$ at the same time t, and conditioned on all grid points at time $t-1$, denoted $x_{t-1,1:L}$. They do this by introducing a set of M local particles j, for each global particle i, with $i \in 1, \dots, N$.

For each of the global particles i they run the following algorithm over the whole grid:

1. Starting from location $l = 1$, the M local particle ensembles grow in dimension as they move over the grid toward the final position L. At the first grid point, the prior particles at that grid point are weighted with the local likelihood $p(y_1 \mid x_1)$ and resampled. Let us designate these particles as $\hat{x}_{j,1}$, where j is the local particle index, and 1 is the grid point index.

2. The mean $\bar{w}_1$ of the unnormalized weights is calculated. Here, the subscript 1 refers to the grid point.

3. For the next grid point, each of these M resampled particles is propagated to that grid point by drawing from $p(x_{t,2} \mid \hat{x}_{t,j,1}, x_{t-1,j,1:L})$. Since each of the M particles is drawn independently, they will differ and no noise needs to be added.

4. Then the unnormalized weights $p(y_{t,2} \mid x_{t,2})$ are calculated, and their mean $\bar{w}_t$, followed by a resampling step.

5. This process is repeated till $l = L$, i.e., till the whole space is covered.

6. Finally, the total weight $w_{t,1} = \prod_{l=1}^{L} \bar{w_t}^l$ is calculated, which is the unnormalized weight of the first global particle.

The resulting algorithm is presented below:

Algorithm 4.2 Space-Time Particle Filter

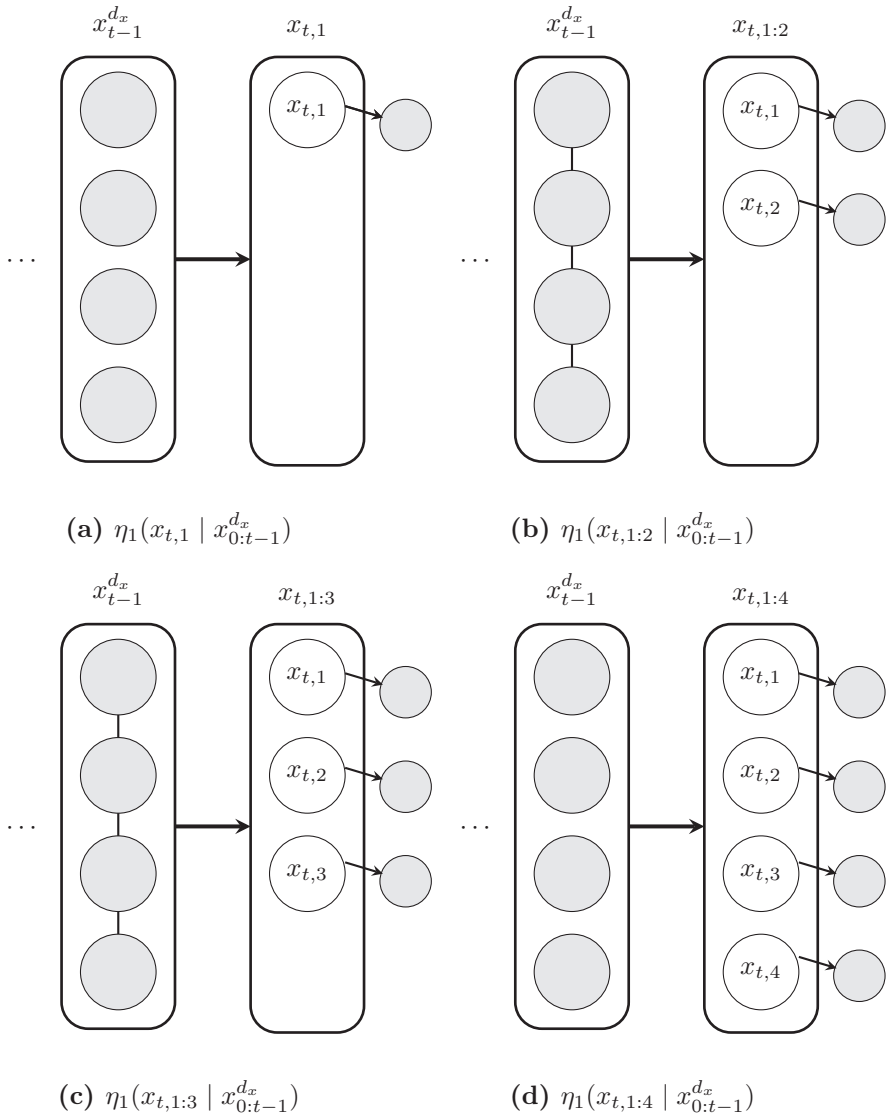
for $i = 1, \cdots, N$ **do**
  for Each grid point *j,* and local grid points k **do**
    for $m = 1, \cdots, M$ **do**
      $x^n_{m,j} \sim p(x^n_j | x^n_{1:l-1}, x^{n-1} | x^n_{1:L})$;
      $\widetilde{w}_m \leftarrow p(y_k | x_{i,k})$
    $\bar{w}_{i,j} \leftarrow \frac{1}{M} \sum_{m=1}^{M} \widetilde{w}_m$
  $w_i \leftarrow \prod_{j=1}^{L} \bar{w}_{i,j}$
$\mathbf{w} \leftarrow \mathbf{w}/\mathbf{w^T 1}$; Resample

While the filter can still uffer from degeneracy, Beskos et al. (2017) discuss the challenge and suggest potential solutions.

STPF has the underlying theoretical proof that it converges to the correct posterior for an increasing number of particles. Furthermore, the authors show that degeneracy can be avoided if the number of particles grows as the square of the dimension of the system – much faster convergence than e.g. the optimal proposal density.

4.4 Nested Sequential Monte Carlo

The goal of Nested Sequential Monte Carlo (NSMC) is to approximate the locally optimal proposal distribution by running a separate internal, or nested, SMC algorithm over the components of $\{x_t^{d_x}\} = (x_{t,1}, \ldots, x_{t,d_x})$ for each particle we require. Running a nested SMC sampler with M internal particles for each particle in the outer algorithm, however, results in $\mathcal{O}(NM)$ operations, instead of N operations in the standard particle filter algorithm. A nested sampler within another sampler thus might appear wasteful of computational resources. However, it can be demonstrated that for many models such an approach might result in a more accurate estimate than, for example, a corresponding standard SMC algorithm with NM particles and the prior (bootstrap) proposal. The key idea is to construct an SMC approximation to the locally optimal proposal $q_t^*(x_t \mid x_{0:t-1})$. For each particle $x_{t,i}$ that we want to simulate, we construct an SMC approximation to q^*:



(a) $\eta_1(x_{t,1} \mid x_{0:t-1}^{d_x})$ **(b)** $\eta_1(x_{t,1:2} \mid x_{0:t-1}^{d_x})$

(c) $\eta_1(x_{t,1:3} \mid x_{0:t-1}^{d_x})$ **(d)** $\eta_1(x_{t,1:4} \mid x_{0:t-1}^{d_x})$

Figure 4.2: Schematic diagram of a nested sequential Monte Carlo filter (NSMC) that uses a nested SMC sampler for each particle in the main algorithm.

$$\widehat{q}_t^*(x_t \mid x_{0:t-1}) = \sum_{j=1}^{N} \frac{\tilde{v}_{d_x,j,i}}{\sum_l \tilde{v}_{d_x,l,i}} \delta_{\bar{x}_{1:d_x,j,i}}(x_t) \tag{4.7}$$

where $(\tilde{v}_{d_x,j,i}, \bar{x}_{1:d_x,j,i})$ are properly weighted for $q_t^*(x_t \mid x_{0:t-1,i})$.

(4.7) is constructed using a nested SMC sampler on the components of x_t; first targeting $x_{t,1}$, then $x_{t,1:2}$, etc, where the final target is $q_t^*(x_{t,1:d_x} \mid x_{0:t-1,i})$. With this approximation NSMC replaces the sampling step and the weighting

step by

$$\widetilde{w}_{t,i} = \prod_{k=1}^{d_x} \frac{1}{M} \sum_{j=1}^{M} \widetilde{v}_{k,j,i} \tag{4.8}$$

where $\widetilde{v}_{k,j,i}$ are the corresponding weights in the nested SMC sampler. The algorithm is summarized below:

Algorithm 4.3 Nested Sequential Monte Carlo.

input : Unnormalized target distributions $\tilde{\gamma}_t$, nested SMC sampler targeting $q_t^*(x_t \mid x_{0:t-1})$, number of samples N and nested samples M.

output : Samples and weights $\{x_{0:t-1,i}, w_{t,i}\}$ approximating γ_t for $t \geq 0$.

1 **for** $t = 1$ *to* T **do**
2 **for** $i = 1$ *to* N **do**
3 Sample $x_{0:t,i} \sim \widehat{\gamma}_{t-1}(x_{0:t-1}) q_t^*(x_t \mid x_{0:t-1})$ with (4.7).
4 Set $\widetilde{w}_{t,i} = \prod_{k=1}^{d_x} \sum_{j=1}^{M} \widetilde{v}_{k,j,i}$ by (4.8).
5 Normalize weights $w_{t,i} = \dfrac{\widetilde{w}_{t,i}}{\sum_j \widetilde{w}_{t,i}}$.

To construct $\widehat{q}_t^*(x_t \mid x_{0:t-1,i})$ internal SMC sampler with M particles is run that targets

$$\eta_l(x_{t,1:l} \mid x_{0:t-1}) \propto \tilde{\eta}_l(x_{t,1:l} \mid x_{0:t-1}) \coloneqq \prod_{k=1}^{l} f(x_{t,k} \mid x_{t-1,k}) g(y_{t,k} \mid x_{0:t,k}), \tag{4.9}$$

for $l = 1, \ldots, d_x$. We denote the particles and weights for the nested SMC by $\bar{x}_{1:l,j,i}$ and $\tilde{v}_{l,j,i}$, respectively. It can be shown that a simple proposal for the above target distribution is $\bar{x}_{1:l,j,i} \sim \widehat{\eta}_{l-1}(x_{t,1:l-1} \mid x_{1:t-1,i}) f(x_{t,l} \mid x_{t-1,l,i})$, i.e. using the prior proposal. We have then $\widehat{\eta}_l$ as

$$\widehat{\eta}_l(x_{t,1:l} \mid x_{1:t-1,i}) = \sum_{j=1}^{M} \frac{\widetilde{v}_{l,j,i}}{\sum_m \widetilde{v}_{l,m,i}} \delta_{\bar{x}_{1:l,j,i}}(x_{t,1:l}), \tag{4.10}$$

where the weights are given by

$$\widetilde{v}_{l,j,i} = g(y_{t,l} \mid (x_{0:t-1,k,i}, \bar{x}_{l,j,i})) = \mathcal{N}\left(y_{t,l} \mid \bar{x}_{l,j,i} + \sum_{m=1}^{t-1} \beta^{t-m} x_{m,l}^i, r\right) \tag{4.11}$$

4.5 Divide and Conquer Monte Carlo

Divide and Conquer SMC (DaC-SMC) (Crucinio and Johansen (2022)) is an extension of the standard SMC algorithm. It views the high dimensional state space as a rooted tree $\mathbb{T}$, in which a collection of target distributions $\{\gamma_u\}_{u \in \mathbb{T}}$ is indexed by the nodes of $\mathbb{T}$, and particles evolve from the leaves of the tree to its root, $\mathcal{R}$, at any time t.

4.5.1 Core Idea

In the standard SMC, we have a sequence of unnormalized target densities $\{\gamma_t\}_{t\geq 0}$, with

$$\gamma_t(x_{0:t}) = p_0(x_0;\theta)\prod_{t=0}^{T} g_t(y_t \mid x_t;\theta)\prod_{t=1}^{T} f_t(x_t \mid x_{t-1};\theta). \tag{4.12}$$

Particle filtering algorithms proceed in iterations and at time t approximates the normalized density

$$\pi_t(x_{0:t}) = \frac{\gamma_t(x_{0:t})}{\int \gamma_t(x_{0:t})dx_{0:t}}$$

with an ensemble of particles $\{x_{0:t,i}\}_{i=1,\ldots,N}$. In the case of DaC-SMC, multiple target distributions are defined on spaces whose dimensions grow as we progress up the tree. At each time t and for each node u, we have the density $\pi_{t,u} \propto \gamma_{t,u}$ over $\mathbb{R}^{|\mathbb{T}_u|}$ where $\mathbb{T}_u$ denotes the sub-tree of $\mathbb{T}$ rooted at node u and includes only node u and all its descendants. $|\mathbb{T}_u|$ denotes the cardinality of this sub-tree.

As in the standard SMC, each distribution $\gamma_{t,u}$ is approximated by a particle ensemble $\{x_{t,u,i}\}_{i=1,\ldots,N}$. These distributions are gradually merged whenever the corresponding branches of $\mathbb{T}$ merge. For simplicity, suppose $\mathbb{T}$ is a perfect binary tree with each non-leaf node u having two children: a left child $l(u)$ and a right child $r(u)$. If u is a leaf node, a simple importance sampling step is performed with proposal $q_{t,u}$ and the importance weight $w_{t,u} \coloneqq \gamma_{t,u}/q_{t,u}$ are computed to obtain a weighted particle ensemble $\{x_{t,u,i}, w_{t,u,i}\}_{i=1,\ldots,N}$ to approximate $\gamma_{t,u}$. If u is a non-leaf node, we collect the particle ensemble associated with each of u's children:

$$\widehat{\gamma}_{t,l(u)} = \frac{1}{N}\sum_{i_1=1}^{N} w_{t,l(u)}\delta(x_{t,l(u)} - x_{t,l(u),i_1}), \tag{4.13}$$

$$\widehat{\gamma}_{t,r(u)} = \frac{1}{N}\sum_{i_2=1}^{N} w_{t,r(u)}\delta(x_{t,r(u)} - x_{t,r(u),i_2}). \tag{4.14}$$

The product $\gamma_{t,\mathcal{C}_u}$ of the marginal distributions $\gamma_{t,l(u)}, \gamma_{t,r(u)}$ at the children nodes is the then estimated as follows: $\gamma_{t,\mathcal{C}_u} \coloneqq \gamma_{t,l(u)} \times \gamma_{t,r(u)} \approx \widehat{\gamma}_{t,l(u)} \times \widehat{\gamma}_{t,r(u)}$.

To compensate for the mismatch between $\gamma_{t,u}$ and $\gamma_{t,l(u)} \times \gamma_{t,r(u)}$, the product-form approximation, i.e., $\widehat{\gamma}_{t,l(u)} \times \widehat{\gamma}_{t,r(u)}$, is reweighed, resulting in the following mixture importance weights:

$$m_{t,u}(x_{t,l(u)}, x_{t,r(u)}) \coloneqq \frac{\gamma_{t,u}(x_{t,l(u)}, x_{t,r(u)})}{\gamma_{t,l(u)}(x_{t,l(u)}) \times \gamma_{t,r(u)}(x_{t,r(u)})} \tag{4.15}$$

Before resampling, these mixture weights are incorporated with the product of the weights obtained at the children nodes, leading to weights of the following form for each node u :

$$\widetilde{w}_{t,u}(x_{t,l(u)}, x_{t,r(u)}) = w_{t,l(u)}(x_{t,l(u)})\, w_{t,r(u)}(x_{t,r(u)})\, m_{t,u}(x_{t,l(u)}, x_{t,r(u)}). \tag{4.16}$$

Resampling N times from $\widetilde{w}_{t,u}\widehat{\gamma}_{t,l(u)} \times \widehat{\gamma}_{t,r(u)}$, using any unbiased resampling scheme, such as multinomial, stratified, systematic, or residual resampling, we then an equally weighted particle ensemble $\{\widetilde{x}_{t,u,i}, w_{t,u,i} = 1\}_{i=1,\ldots,N}$ approximating $\gamma_{t,u}$. Algorithm 4.4, which is applied to the root node to carry out the sampling process, summarizes this.

Algorithm 4.4 dac-smc(u) for u in $\mathbb{T}$.

if *u is a leaf node* **then**
  Initialize: draw $x_{t,u,i} \sim q_{t,u}$ and compute $w_{t,u,i} = \gamma_{t,u}/q_{t,u}$ for $i = 1, \ldots, N$.
else
  Recurse: set $(\{x_{t,v,i}, w_{t,v,i}\})_{i=1,\ldots,N} \coloneqq$ DaC-SMC(v) for $v \in \{l(u), r(u)\}$ and obtain $\widehat{\gamma}_{t,l(u)} \times \widehat{\gamma}_{t,r(u)}$ by (4.13) and (4.14).
  Merge: compute weights $\widetilde{w}_{t,u,(i_1,i_2)}$ in (4.16) for all $i_1, i_2 = 1, \ldots, N$.
  Resample: draw $\{\tilde{x}_{t,u,i}\}_{i=1,\ldots,N}$ using weights $\tilde{w}_{t,u,(i_1,i_2)}$ and set $w_{t,u,i} = 1$ for $i = 1, \ldots, N$.

4.5.2 DaC within Marginal SMC for Filtering

We have so far addressed the idea of DAC-SMC for a particular time t. To apply it to the filtering problem, a suitable collection of unnormalized target densities $\{\widetilde{\gamma}_{t,u}\}_{u\in\mathbb{T}}$, indexed by the nodes of the tree $\mathbb{T}$, needs to be identified for each time t. Graphically, this corresponds to a time-dependent path graph in which each node has, associated with it, a copy of the tree $\mathbb{T}$, corresponding to space, and this copy of $\mathbb{T}$ evolves over t. With this consideration,Algorithm 4.4 takes as input at the leaves particle ensemble approximating the filtering distribution at time $t-1$ and delivers as the output at the root a particle population approximating the filtering distribution at time t. At a given time t, to build the collection $\{\widetilde{\gamma}_{t,u}\}_{u\in\mathbb{T}}$, $x_t^{d_x}$ is decomposed spatially into low dimensional elements. The dimension of a decomposed element can be as low as 1, but the computational cost of merging many components successively along the tree toward the root will then increase. On the other hand, keeping the cardinality of the dimension of an element too high will undermine the DaC principle and any benefits derived from it.
To illustrate the whole process, a simple decomposition obtained by one-to-one mapping between the d_x components $(x_t^1, \ldots, x_t^{d_x})$ and the leaves of $\mathbb{T}$ is now considered. As we move up the tree, the components are merged pairwise until $x_t = x_t^{1:d_x}$ is recovered at the root node $\mathcal{R}$.

We denote the set of components associated with node u by $\mathcal{V}_u$. The cardinality $|\mathcal{V}_u|$ of $\mathcal{V}_u$ increases from 1 at the leaf level to d_x at the root. Figure 4.3 shows the space decomposition for $d_x = 8$.

Because of the inherent sequential structure of the state space, the collection $\{\widetilde{\gamma}_{t,u}\}_{u\in\mathbb{T}}$ at time t is specified in terms of the filtering distribution $\{\widetilde{\gamma}_{t-1,\mathcal{R}}\}$ at the previous time $t-1$, as shown below in (4.17). Functions $f_{t,u} : \mathbb{R}^{d_x} \times \mathbb{R}^{|\mathcal{V}_u|} \to \mathbb{R}$ and $g_{t,u} : \mathbb{R}^{d_y} \times \mathbb{R}^{|\mathcal{V}_u|} \to \mathbb{R}$, for $t \geq 1$ and $u \in \mathbb{T}$, are auxiliary functions that act as proxies for the marginals of the transition density and observation likelihood, respectively, at the node u.

$$\widetilde{\gamma}_{t,u}(z_{t,u}) = g_{t,u}(z_{t,u}, (y_t(d))_{d\in\mathcal{V}_u}) \int f_{t,u}(x_{t-1}, z_{t,u})\widetilde{\gamma}_{t-1,\mathcal{R}}(x_{t-1})dx_{t-1}. \quad (4.17)$$

Here, $z_{t,u} = (x_t(d))_{d\in\mathcal{V}_u}$ are the components of x_t associated with node u and x_{t-1} denotes the previous state of the system.

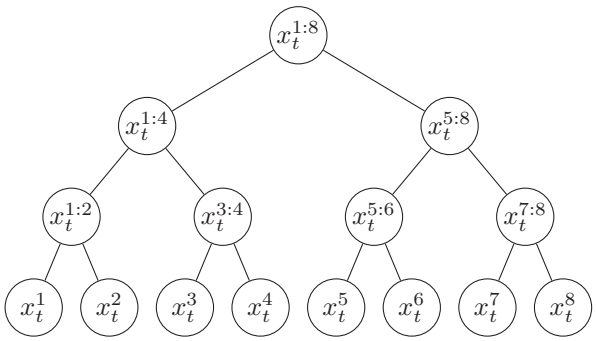


Figure 4.3: Space decomposition by the divide and conquer sequential Monte Carlo (DaC-SMC) algorithm for a state space with the cardinality of its dimension $d_x = 8$.

For each time t, once the the space decomposition over $\mathbb{T}$ and the collection of distributions $\{\widetilde{\gamma}_{t,u})_{u\in\mathbb{T}}$ have been identified, Algorithm 4.4 can be applied the root $\mathcal{R}$ of $\mathbb{T}$. However, the integral w.r.t. $\gamma_{t-1,\mathcal{R}}$ in (4.17) cannot be computed analytically, requiring an approximation of $\widetilde{\gamma}_{t-1,\mathcal{R}}$ by the particle population at the $\mathcal{R}$ corresponding to the particle approximation obtained at the previous time step, as is done in the case of the standard particle filter:

$$\gamma_{t,u}(z_{t,u}) := g_{t,u}(z_{t,u}, (y_t(d))_{d\in\mathcal{V}_u})\frac{1}{N}\sum_{i=1}^{N} f_{t,u}(z_{t-1,\mathcal{R},i}, z_{t,u}). \tag{4.18}$$

Given $\{z_{t-1,\mathcal{R},i}\}_{i=1,\ldots,N}$ at each leaf node of the tree, we sample one component of x_t per node from $N^{-1}\sum_{i=1}^{N} q_{t,u}(z_{t-1,\mathcal{R},i}, \cdot)$, the importance weights are then given by

$$w_{t,u}(z_{t,u}, x_{0:t-1,u}) = \frac{g_{t,u}(z_{t,u}, (y_t(d))_{d\in\mathcal{V}_u})\sum_{i=1}^{N} f_{t,u}(z_{t-1,\mathcal{R},i}, z_{t,u})}{\sum_{i=1}^{N} q_{t,u}(z_{t-1,\mathcal{R},i}, z_{t,u})} \tag{4.19}$$

As in the bootstrap particle filter, if we choose $q_{t,u} = f_{t,u}$, the equations simplifies to $w_{t,y}(z_{t,u}, x_{1:t-1,u}) = g_{t,u}(z_{t,u}, (y_t(d))_{d\in\mathcal{V}_u})$, considerably reducing the cost of evaluating the weights at the leaves.

For any non-leaf node u, the particle ensembles $\{z_{t,l(u),i}\}_{i=1,\ldots,N}$ and $\{z_{t,r(u),i}\}_{i=1,\ldots,N}$ on its left and right child, respectively, are collected, and $\gamma_{t,\mathcal{C}_u} := \gamma_{t,l(u)} \times \gamma_{t,r(u)}$ is estimated according to the weighted estimator in (4.13) and (4.14).

$$\widehat{\gamma}_{t,l(u)} = \frac{1}{N}\sum_{i_1=1}^{N} w_{t,l(u)}\delta(z_{t,l(u)} - z_{t,l(u),i_1}), \tag{4.20}$$

$$\widehat{\gamma}_{t,r(u)} = \frac{1}{N}\sum_{i_2=1}^{N} w_{t,r(u)}\delta(z_{t,r(u)} - z_{t,r(u),i_2}). \tag{4.21}$$

As discussed earlier, we reweight the particle approximation of $\gamma_{t,\mathcal{C}_u}$ to target $\gamma_{t,u}$. In this case, the mixture weights are given by

$$
\begin{aligned}
m_{t,u}(z_{t,\mathcal{C}_u}) =& \frac{\gamma_{t,u}(z_{t,\mathcal{C}_u})}{\gamma_{t,\mathcal{C}_u}(z_{t,\mathcal{C}_u})} \qquad (4.22)\\
=& \frac{g_{t,u}(z_{t,\mathcal{C}_u}, (y_t(d))_{d\in\mathcal{V}_u})}{g_{t,l(u)}(z_{t,l(u)}, (y_t(d))_{d\in\mathcal{V}_{l(u)}}) g_{t,r(u)}(z_{t,r(u)}, (y_l(d))_{d\in\mathcal{V}_{r(u)}})} \times \\
& \frac{N^{-1}\sum_{i=1}^N f_{t,u}(z_{t-1,\mathcal{R},i}, z_{t,\mathcal{C}_u})}{N^{-1}\sum_{i=1}^N f_{t,l(u)}(z_{t-1,\mathcal{R},i}, z_{t,l(u)}) N^{-1}\sum_{i=1}^N f_{t,r(u)}(z^n_{t-1,\mathcal{R}}, z_{t,r(u)})},
\end{aligned}
$$

where we defined $z_{t,\mathcal{C}_u} := (z_{t,l(u)}, z_{t,r(u)})$ the vector obtained by merging the components on the left and on the right child of u.

For each pair in (4.20)–(4.21), we obtain the incremental mixture weights in (4.22):

$$m_{t,u,i_1,i_2} := m_{t,u}(z_{t,l(u),i_1}, z_{t,r(u),i_2})$$

and the updated weights

$$
\begin{aligned}
\tilde{w}_{t,u,i_1,i_2} &= \tilde{w}_{t,u}(z_{t,l(u),i_1}, z_{t,r(u),i_2}) \\
&:= w_{t,l(u)}(z_{t,l(u),i_1}) w_{t,r(u)}(z_{t,r(u),i_2}) m_{t,u}(z_{t,l(u),i_1}, z_{t,r(u),i_2}),
\end{aligned}
$$

for $i_1, i_2 = 1, \ldots, N$.

The algorithm below depicts the complete process for Dac-SMC in the filtering context.

Algorithm 4.5 dac-smc(t) for $t \geq 1$.

Given $(\{z^n_{t-1,\mathcal{R}}\}_{n=1}^N) :=$ dac smc $(t-1)$.
for u leaf node **do**
  Initialize: draw $z_{t,u,i} \sim N^{-1}\sum_{i=1}^N q_{t,u}(z_{t-1,\mathcal{R},i}, \cdot)$ and compute $w_{t,u,i}$ as in (4.19) for all $i = 1, \ldots, N$.
for u non-leaf node **do**
  Recurse: set $(\{z_{t,v,i}, w_{t,v,i}\}_{i=1}^N) :=$ dac-smc (t, v) for v in $\{l(u), r(u)\}$ and obtain the product of $\widehat{\gamma}_{t,l(u)}$ and $\widehat{\gamma}_{t,r(u)}$ in (4.20)–(4.21).
  Merge: compute the mixture weights m_{t,u,i_1,i_2} in (4.22) and $\tilde{w}_{t,u,i_1,i_2}$ for all $i_1, i_2 = 1, \ldots, N$.
  Resample: draw $\{\widetilde{z}_{t,u,i}\}_{i=1}^N$ using weights $\tilde{w}_{t,u,i_1,i_2}$ and set $w^n_u = 1$ for $i = 1, \ldots, N$.
  Update: set $z_{t,u,i} = \widetilde{z}_{t,u,i}$ for all $i = 1, \ldots, N \leq N$.
Output $(\{z_{t,\mathcal{R},i}\}_{i=1}^N)$.

CHAPTER 5

Experimental Results

In this chapter, we compare the results obtained with three algorithms: NSMC of C. Naesseth, Lindsten and Schon (2015), STPF of Beskos et al. (2017), and DaC of (Crucinio and Johansen, 2022). We apply the three filtering algorithms on a linear Gaussian SSM and compare their results against the exact filtering distribution given by the Kalman filter on the same SSM. We also evaluate the performance of a standard bootstrap particle filter in our comparison. All the experiments have been run in serial using the Windows Statistics server at the university. Each experiment was repeated multiple times in parallel using the computing nodes of the server.

5.1 Linear Gaussian SSM

We start by considering a simple yet high-dimensional, linear Gaussian SSM studied in C. Naesseth, Lindsten and Schon (2015) and Crucinio and Johansen (2022) for which we have the exact solution for the filtering distributions from the Kalman filter. The model is given by

$$\begin{aligned} X_0 &\sim \mathcal{N}_{d_x}(0, I_{d_x}) \\ X_t &= 1.0AX_{t-1} + U_t, \quad U_t \sim \mathcal{N}_{d_x}(0, \Sigma) \\ Y_t &= X_t + V_t \qquad V_t \sim \mathcal{N}_{d_y}(0, \sigma_y^2 I_{d_y}) \end{aligned}$$

Here, $A = A_1 A_2^{-1}$ where

$$A_1 = \begin{bmatrix} \tau+\lambda & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & \tau & 0 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \tau & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \tau \end{bmatrix},$$

$$A_2 = \begin{bmatrix} \tau+\lambda & 0 & 0 & \cdots & \cdots & 0 & 0 \\ -\lambda & \tau+\lambda & 0 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & -\lambda & \tau+\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & -\lambda & \tau+\lambda \end{bmatrix}^{-1},$$

and

$$\Sigma^{-1} = A_2^\top \begin{bmatrix} \tau & 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & \tau+\lambda & 0 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \tau+\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \tau+\lambda \end{bmatrix} A_2$$

5.2 Performance Metrics

To evaluate the performances of the algorithms, we compare several measures of performance.

Algorithm runtime

We have the run the algorithms for $T = 50$ time steps. For each algorithm, we have recorded the runtime taken in each time step $t \in T$ and summed up the 50 runtimes to get the time for one complete run. These runs were made for several sets of dimension size and particle number (d_x, N). Each complete algorithmic run was repeated multiple times, 100 in the low dimensional cases and 20 in the high dimensional cases, to monitor any variability in the execution time from one run to the next.

Wasserstein-1 distance

The Wasserstein distance (see, e.g., Panaretos and Zemel (2019)), also known as the Earth Mover's Distance (EMD) or optimal transport distance, is a measure of the distance between two probability distributions over a metric space. It quantifies the minimum work or 'cost' of transforming one distribution into the other, by solving the optimal transport problem, where the cost is associated with the amount of mass that needs to be moved and the distance it is moved. The Wasserstein distance of order 1, or Wasserstein-1 distance, for each of the d_x marginals, is given by

$$D_{\mathrm{W1}_{t,k}} \coloneqq \int \left| F_{t,k}(x_t) - \widehat{F}_{t,k}(x_t) \right| dx_t, \tag{5.1}$$

where $D_{\mathrm{W1}_{t,k}}$ denotes Wasserstein-1 distance, $F_{t,k}(x_t)$ the 1-dimensional cumulative distribution function of marginal k at time t from the Kalman filter and $\widehat{F}_{t,k}(x_t)$ its particle approximation from the algorithms we are comparing.

Kolmogorov-Smirnov distance

The Kolmogorov-Smirnov (KS) distance, also referred to as the Kolmogorov-Smirnov statistic, is a measure of the dissimilarity between two probability distributions. It is based on the maximum vertical difference between the empirical cumulative distribution functions (ECDFs) of the two distributions being compared and is given by

$$D_{\mathrm{KS}_k} \coloneqq \max_x \left| F_{t,k}(x_t) - \widehat{F}_{t,k}(x_t) \right| dx_t, \tag{5.2}$$

where $F_{t,k}(x_t)$ and $\widehat{F}_{t,k}(x_t)$ denote, respectively, the 1-dimensional cumulative distribution function of marginal k at time t and its particle approximation.

Relative mean squared error

Finally, we compute the relative mean squared error (ReMSE) for component k at time t as

$$\mathrm{ReMSE}_{t,k} \coloneqq \frac{\mathbb{E}[(\widehat{x}_{t,k} - \mu_{t,k})^2]}{(\sigma_{t,k})^2} \tag{5.3}$$

where $\widehat{x}_{k,t}$ denotes the estimate of the mean of component k at time t and $\mu_{t,k}, \sigma_{t,k}$ denote the true mean and true variance of $x_t^k \mid y_{0:t}$ obtained from the Kalman filter. We approximate the ReMSE using an empirical average over all corresponding runs of each experiment .

5.3 Experimental results

5.3.1 Linear Gaussian SSM with uncorrelated dimensions

We first set $\tau = 1, \lambda = 0,$ and $\sigma_y^2 = \sqrt{2}$. This makes the model identical to the linear Gaussian model that we studied in chapter 3.

ReMSE

We present the ReMSE's of the algorithms in Figure 5.1 for four values of d_x and three different numbers of particles N for each d_x. Each box plot is based on 20 runs for each set of (d_x, N). First we note, expectantly, that the ReMSE decreases with the increasing ensemble size in all cases for all the three algorithms. Overall, STPF outperforms the other algorithms for any combination of (d_x, N). However, it has the highest variability in its performance among the algorithms.

W1 distance and KS distance

We present the W1 distances of the algorithms in Figure 5.2. We see that STPF has the highest performance in this metric for $d_x = 16$ and 32. For $d_x = 64$ and 256, NSMC emerges as the best performer. STPF has also the highest variability in its performance,

We present the KS distances of the algorithms in Figure 5.3. Looking into the scale of KS distance, all the three algorithms show somewhat comparable performance on this metric, with STPF slightly outperforming the others, while DAC-SMC and NSMC are close to each other.

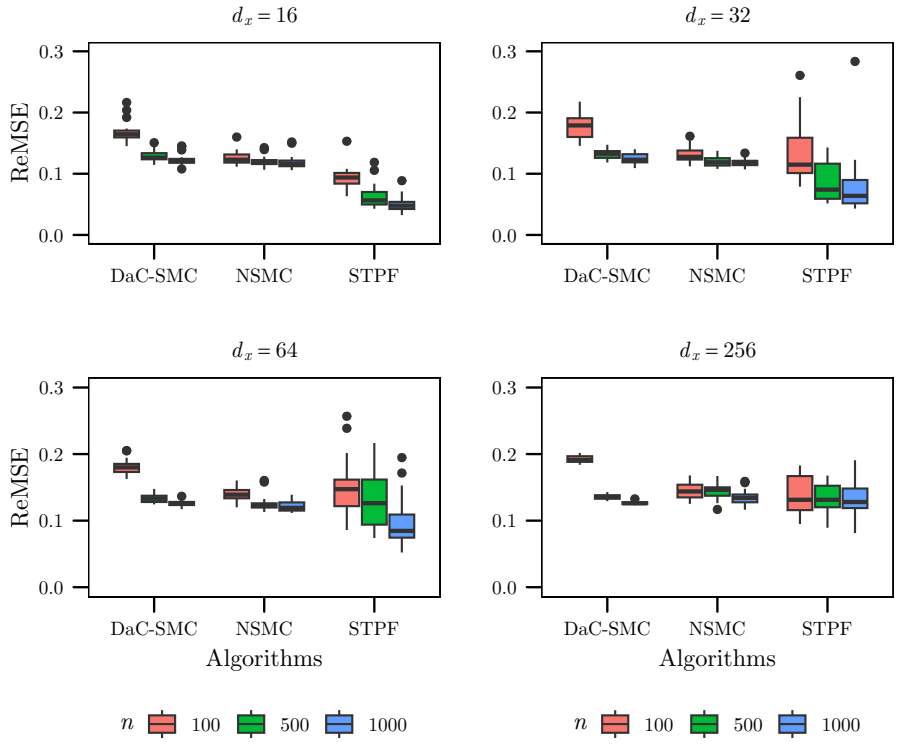


Figure 5.1: **ReMSE with $\lambda = 0$.** Comparison of the relative mean squared errors, first averaged over the dimension d_x and then averaged for the number of runs (20 in each case) of the algorithms.

5.3.2 Linear Gaussian SSM with moderately correlated dimensions

We now repeat the experiments setting $\lambda = 1$, while keeping all the other parameters unchanged and presen the results.

ReMSE

We present the ReMSE's of the algorithms in Figure 5.4 for four values of d_x and three different numbers of particles N for each d_x. Each box plot is based on 20 runs for each set of (d_x, N). We again note that the ReMSE decreases with the increasing ensemble size in all cases. Overall, STPF outperforms the other algorithms.

Looking at scale of the vertical axis and comparing with the case for $\lambda = 0$, we discover that ReMSEs have now (with $\lambda = 1$) deteriorated significantly for DAC-SMC, while performances of the other two algorithms have remained relatively stable.

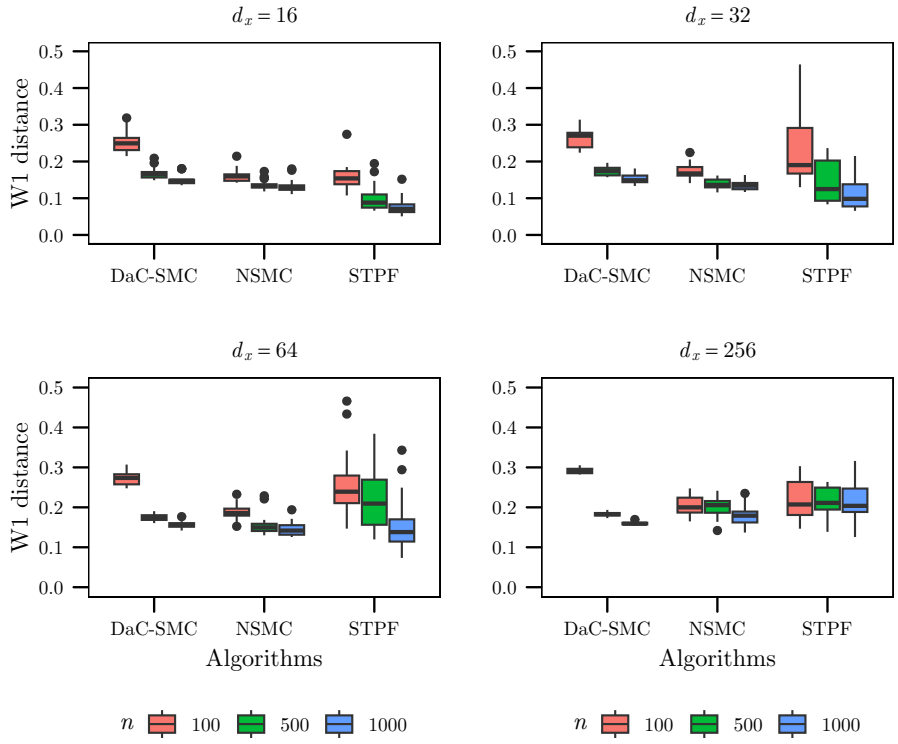


Figure 5.2: **Wasserstein-1 distance with $\lambda = 0$.** Comparison of Wasserstein-1 distance between the marginal distribution produced by the Kalman Filter and those produced by the three algorithms over 20 runs.

W1 distance and KS distance

We present the W1 distances of the algorithms in Figure 5.5. We see that Dac-SMC has the lowest performance in this metric as well. Its performance is also the most variable from one algorithmic run to the next, except for $d_x = 256$, suggesting its variability might stabilize in the high dimensional setting. NSMC and STPF are comparable in their performances here, with the latter having a lower mean ReMSE, but with a higher variance, over the runs. NSMC shows somewhat stable performance. Again, comparing with the $\lambda = 0$ case, we find that performance of DAC-SMC has significantly been affected by the correlation among the components of state variable.

We present the KS distances of the algorithms in Figure 5.6. All the three algorithms show comparable performance on this metric, with STPF outperforming the others, followed by DAC-SMC.

Runtime

In Table 5.1, we haave computed the runtimes for a single time step, taking the average over 20 repetitions, of the algorithms for the all sets of (d_x, N). Below we present the case for $d_x = 256$. We see that DAC-SMC is quite expensive, while NSMC has the lowest runtimes.

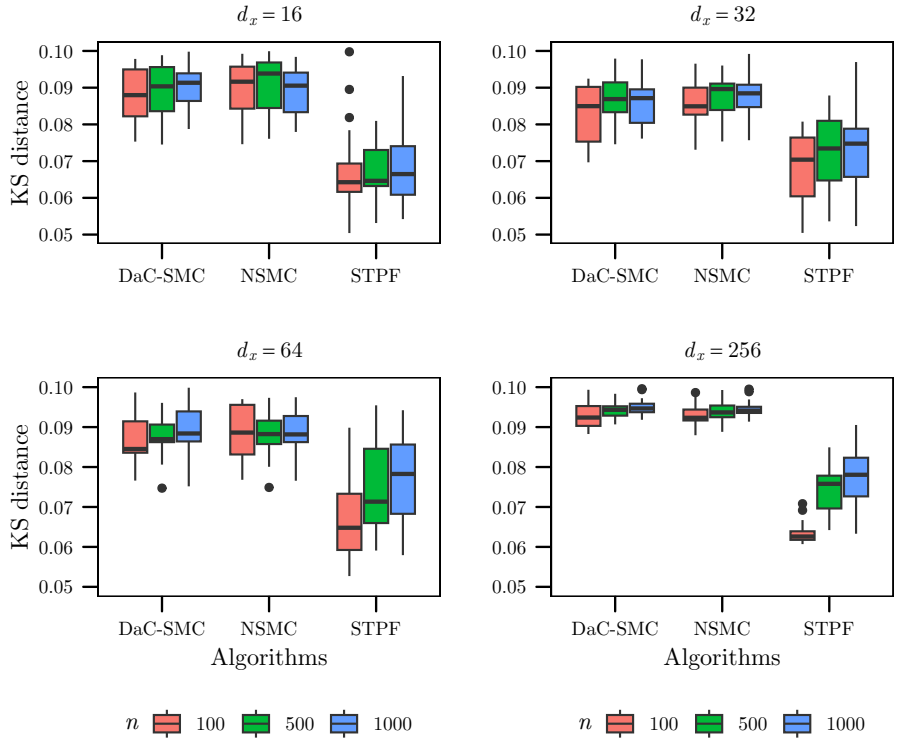


Figure 5.3: **KS distance with $\lambda = 0$.** Comparison of KS distance between the marginal distribution produced by the Kalman Filter and those produced by the three algorithms over 20 runs

Table 5.1: Comparison of the runtimes in seconds of the algorithms for $d_x = 256$

Runtime (sec)		N		
		10^2	5×10^2	10^3
$d_x = 256$	DAC-SMC	1.4×10^1	7.5×10^2	3.7×10^3
	NSMC	6.5×10^0	3.5×10^1	6.3×10^1
	STPF	2.8×10^1	1.5×10^2	3.1×10^2

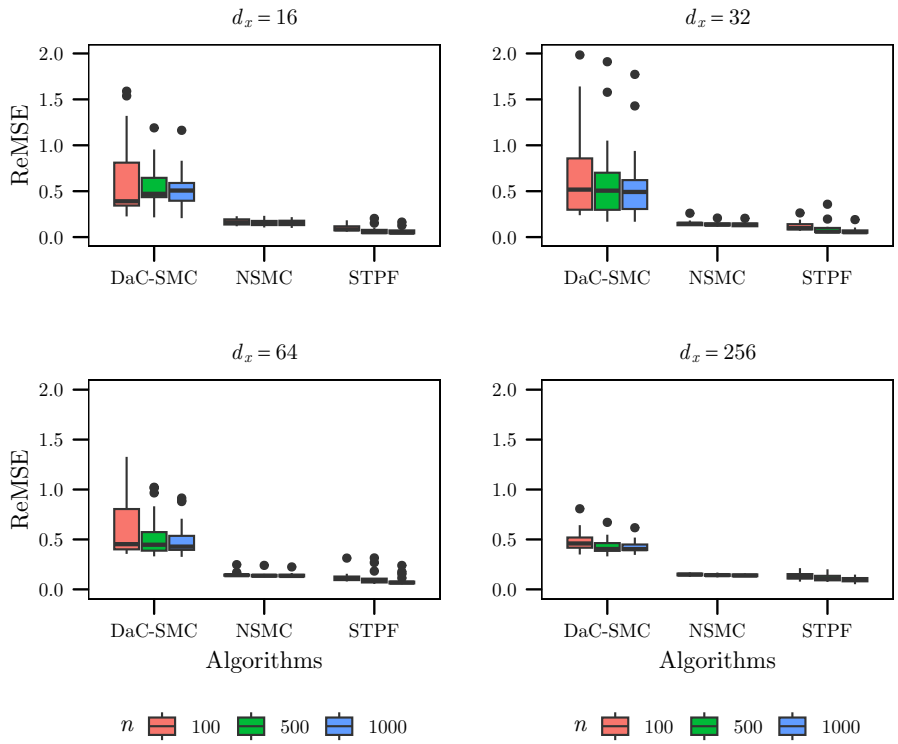


Figure 5.4: **ReMSE with $\lambda = 1$**. Comparison of the relative mean squared errors, first averaged over the dimension d_x and then averaged for the number of runs (20 in each case) of the algorithms.

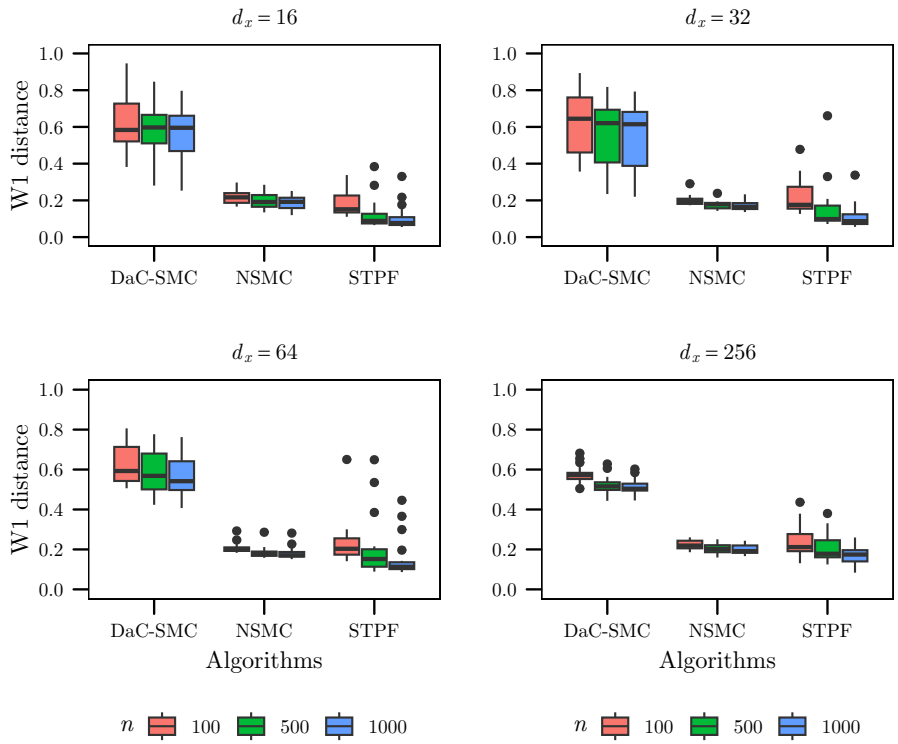


Figure 5.5: **Wasserstein-1 distance with $\lambda = 1$.** Comparison of Wasserstein-1 distance between the marginal distribution produced by the Kalman Filter and those produced by the three algorithms over 20 runs.

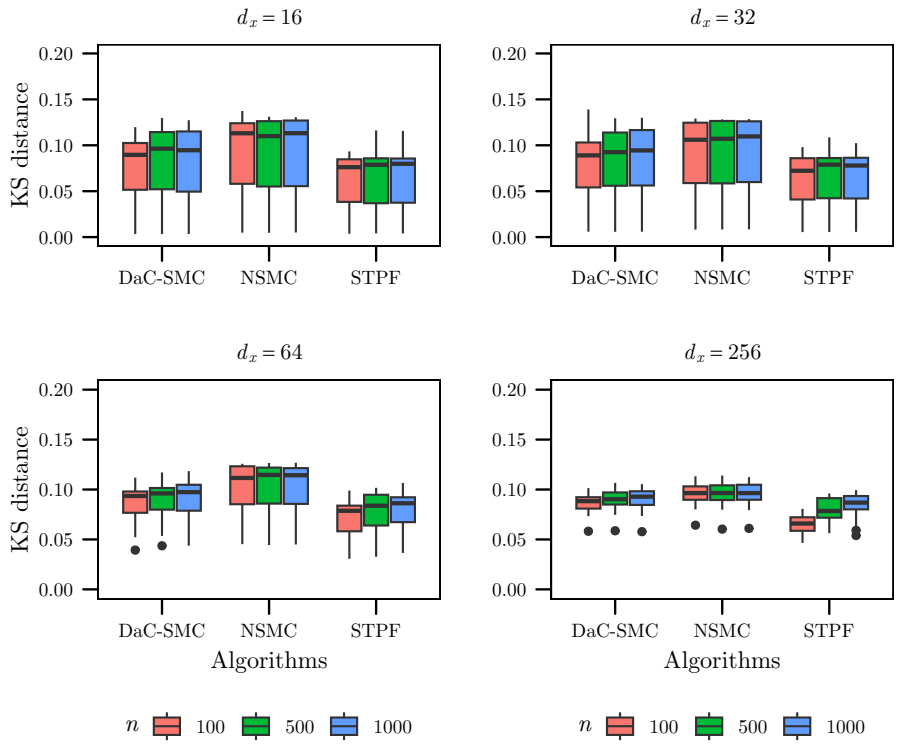


Figure 5.6: **KS distance with $\lambda = 1$.** Comparison of KS distance between the marginal distribution produced by the Kalman Filter and those produced by the three algorithms over 20 runs.

CHAPTER 6

Conclusions

6.1 Summary

We started the thesis with aim of studying the well-known phenomenon of weight collapse of the sequential Monte Carlo (SMC) methods, especially in the moderate to high dimensional state space setting. In chapter 2, we discussed the foundational concepts of SMC and their application in state and parameter estimation in non-linear, non-Gaussian state space models for which no analytical solution exists. It is also known that the resampling step is the most computationally-expensive part of the SMC methods, and there are four common resampling strategies that are employed by these methods. We compared the runtimes of these resampling strategies on a simple $1D$ state space model with a varying number of particles and found that systematic resampling is the fastest among them. We however did not notice any appreciable difference in the accuracy of the strategies in estimating the state.

In chapter 3, we studied, through experiments and simulations, the fundamental problems, such as high computational cost, particle weight collapse, high estimation error, and lack of diversity in particle genealogy, encountered by the particle filter in the high dimensional setting, even when the state space is linear and Gaussian. We in particular note here that there is currently no known methodology to successfully overcome any of these problems, and implementing particle filter remains infeasible for high dimensional problems and applications, e.g., in weather forecasting, in which the dimension routinely achieves the order of 10^7.

In chapter 4, we discussed a few classes of emerging algorithms that have shown some promise in making inferences on the high dimensional state spaces, namely the Block Particle Filter (Rebeschini and Van Handel (2015), Space-Time Particle Filter (Beskos et al. (2017), Nested Sequential Monte Carlo (C. Naesseth, Lindsten and Schon (2015), and Divide and Conquer Sequential Monte Carlo (Crucinio and Johansen (2022). We noted that a few of these methods have got some theoretical support, while studies are being carried out to establish the theory behind the others. Within these algorithms, there are also certain variations or modularities that might be adopted to push their performance limits.

In chapter 5, we presented the results from simulations by algorithms on a linear Gaussian state space model with different degrees of correlations among its dimensions. We compare the results based on several metrics, such as runtime, Wasserstein distance and Kolmogorov–Smirnov distance between distributions

produced by the Kalman filter and those produced by these algorithms, and their relative mean square error compared to the estimates made by the Kalman filter. We noted that STPF has the highest performance in all the metrics, except in runtime, in which NSMC outperforms. We also noted that performance of all the algorithms deteriorate when there is a certain degree of correlation among the components of state space, compared to the case when each dimension is independent of the other dimensions.

6.2 Challenges we faced

We have spent a substantial amount of time in experimentation and simulation. SMC in high dimension is in itself an emerging branch. After initially studying the fundamental problems of SMC in the high dimension, we started our experimentation in our local machine on small dimensional problems. However, as we were increasing the size of the dimension, we started to encounter problems with the computational resources. Running a particle filtering algorithm on a problem of moderate dimension was taking weeks for a single run. In addition, long periods of continuous running of the local machine were heavily taxing the power and operating system stability of the machine.

We then transferred the routines of the experiments to our university servers. To conduct the simulations in chapter 3, we resorted to the machine learning nodes (ML) at high performance computing (HPC) clusters of the university, acoording to the availability. While it resolved the constraint of the computational resources, using Jupyter Notebook at the ML nodes required that our local machine also remain continuously on operation in synchronization with the ML node, and interruption of either of the machines resulted in data loss over several days in multiple occasions.

For simulations in chapter 5, we resorted to Windows Statistics Server at the university. This server was by comparison more stable than, but not as computationally powerful as, the ML nodes. In addition, the virtual machine at the Statistics server had 23 cores, which might be deemed enough for other simulation works but proved to a limiting factor in conducing many parallel runs of the algorithms.

6.3 Future works

We have greatly enjoyed our works throughout the thesis. Because of our prior, limited familiarity with Bash/Shell scripting, Monte Carlo simulations, high computing tasks, and parallel computing, we had to spend more time on repeated experimentation and simulation. Now that we have got some familiarity with these methods and can set up these experiments comparatively quickly, we could first investigate the problems with even higher dimensions. Then we could explore in detail the theoretical properties of the emerging algorithms that we addressed. We have, among other areas, have also a particular interest in online Bayesian parameter estimation. During our experiments in chapter 2, we did some parameter estimation tasks using particle marginal metropolis hastings (PMMH) on a population ecology model of $1D$. We could undertake similar challenges in the high dimensional setting. Overall, SMC in high dimension is

nascent branch with a lot of promises and challenges, which certainly elicits our strong passion.

Appendices

APPENDIX A

Derivations and Calculations

A.1 Bayesian Filtering Equations

$$p(x_t \mid y_{0:t-1}) = \int f(x_t \mid x_{t-1})\, p(x_{t-1} \mid y_{0:t-1})\, dx_{t-1}. \tag{A.1}$$

$$p(x_t \mid y_{0:t}) = \frac{g(y_t \mid x_t)\, p(x_t \mid y_{0:t-1})}{\int g(y_t \mid x_t)\, p(x_t \mid y_{0:t-1}) dx_t}. \tag{A.2}$$

Assume that at the time-step $t-1$, we have the posterior $p(x_{t-1} \mid y_{0:t-1})$ available. The joint-distribution of x_t and x_{t-1} given $y_{0:t-1}$ can now be calculated by the Chapman–Kolmogorov equation:

$$p(x_t, x_{t-1} \mid y_{0:t-1}) = \int p(x_t \mid x_{t-1})\, p(x_{t-1} \mid y_{0:t-1})\, dx_{t-1}$$

By the Chain rule of probability and Markov property, we can rewrite the left side of the equation above as

$$\begin{aligned} p(x_t, x_{t-1} \mid y_{0:t-1}) &= p(x_t \mid x_{t-1}, y_{0:t-1})\, p(x_{t-1} \mid y_{0:t-1}) \\ &= p(x_t \mid x_{t-1})\, p(x_{t-1} \mid y_{0:t-1}). \end{aligned}$$

Hence,

$$p(x_t \mid y_{0:t-1}) = \int f(x_t \mid x_{t-1})\, p(x_{t-1} \mid y_{0:t-1})\, dx_{t-1}.$$

The distribution of x_t given $y_{0:t-1}$ can be computed by Bayes' rule:

$$\begin{aligned} p(x_t \mid y_{0:t}) &= p(x_t \mid y_t, y_{0:t-1}) \\ &= \frac{1}{Z_t}\, p(y_t \mid x_t, y_{0:t-1})\, p(x_t \mid y_{0:t-1}) \\ &= \frac{1}{Z_t}\, p(y_t \mid x_t)\, p(x_t \mid y_{0:t-1}) \end{aligned}$$

Bibliography

Bengtsson, T., Bickel, P. and Li, B. (2008). 'Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems'. In: *Probability and statistics: Essays in honor of David A. Freedman*. Vol. 2. Institute of Mathematical Statistics, pp. 316–335.

Beskos, A. et al. (2017). 'A stable particle filter for a class of high-dimensional state-space models'. In: *Advances in Applied Probability* vol. 49, no. 1, pp. 24–48.

Briggs, J., Dowd, M. and Meyer, R. (2013). 'Data assimilation for large-scale spatio-temporal systems using a location particle smoother'. In: *Environmetrics* vol. 24, no. 2, pp. 81–97.

Cappé, O., Moulines, E. and Rydén, T. (2009). 'Inference in hidden markov models'. In: *Proceedings of EUSFLAT conference*, pp. 14–16.

Chopin, N. and Papaspiliopoulos, O. (2020). *An introduction to sequential Monte Carlo*. Vol. 4. Springer.

Crucinio, F. R. and Johansen, A. M. (2022). 'A divide and conquer sequential Monte Carlo approach to high dimensional filtering'. In: *arXiv preprint arXiv:2211.14201*.

Doucet, A., De Freitas, N. and Gordon, N. (2001). 'An introduction to sequential Monte Carlo methods'. In: *Sequential Monte Carlo methods in practice*, pp. 3–14.

Godsill, S. (2019). 'Particle filtering: the first 25 years and beyond'. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 7760–7764.

Gordon, N. J., Salmond, D. J. and Smith, A. F. (1993). 'Novel approach to nonlinear/non-Gaussian Bayesian state estimation'. In: *IEE proceedings F (radar and signal processing)*. Vol. 140. 2. IET, pp. 107–113.

Iambartsev, A. (2018). 'Importance sampling'. In: https://www.ime.usp.br/~yambar/MAE5704/Aula4MonteCarlo-II/aula4slidesT-2018.pdf.

Kalman, R. E. (1960). 'A new approach to linear filtering and prediction problems'. In.

Kong, A., Liu, J. S. and Wong, W. H. (1994). 'Sequential imputations and Bayesian missing data problems'. In: *Journal of the American statistical association* vol. 89, no. 425, pp. 278–288.

Liverani, C. (1995). 'Decay of correlations'. In: *Annals of Mathematics* vol. 142, no. 2, pp. 239–301.

Naesseth, C., Lindsten, F. and Schon, T. (2015). 'Nested sequential monte carlo methods'. In: *International Conference on Machine Learning*. PMLR, pp. 1292–1301.

Naesseth, C. A., Lindsten, F., Schön, T. B. et al. (2019). 'Elements of sequential monte carlo'. In: *Foundations and Trends® in Machine Learning* vol. 12, no. 3, pp. 307–392.

Panaretos, V. M. and Zemel, Y. (2019). 'Statistical aspects of Wasserstein distances'. In: *Annual review of statistics and its application* vol. 6, pp. 405–431.

Pavliotis, G. A. (2016). *Stochastic processes and applications*. Springer.

Pitt, M. K. and Shephard, N. (1999). 'Filtering via simulation: Auxiliary particle filters'. In: *Journal of the American statistical association* vol. 94, no. 446, pp. 590–599.

Rebeschini, P. and Van Handel, R. (2015). 'Can local particle filters beat the curse of dimensionality?' In.

Robert, C. P. and Casella, G. (2004). *Monte Carlo statistical methods*. Springer.

Snyder, C. et al. (2008). 'Obstacles to high-dimensional particle filtering'. In: *Monthly Weather Review* vol. 136, no. 12, pp. 4629–4640.

Storvik, G. (2017). 'Sequential Monte Carlo for state space models'. In: https://www.uio.no/studier/emner/matnat/math/STK4051/h17/pensumliste/smc.pdf.

Van Leeuwen, P. J. et al. (2019). 'Particle filters for high-dimensional geoscience applications: A review'. In: *Quarterly Journal of the Royal Meteorological Society* vol. 145, no. 723, pp. 2335–2365.

Wan, E. A. and Van Der Merwe, R. (2000). 'The unscented Kalman filter for nonlinear estimation'. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee, pp. 153–158.