Monte Carlo algorithms for performing Bayesian inference on Piecewise Deterministic Processes

by

Chiara Mignacco

to obtain the degree of Master of Science in Applied Mathematics
at the Delft University of Technology,
to be defended publicly on Thursday September 22, 2022 at 14:00.

Student number: 5395739
Project duration: November 15, 2021 – August 15, 2022
Thesis committee: Dr. ir. J. Bierkens, TU Delft, supervisor
Dr. ir. L. E. Meester, TU Delft
Dr. A. B. Duncan, Imperial College London, supervisor

TU Delft

Abstract

Since their introduction in 1993, particle filters are amongst the most popular algorithms for performing Bayesian inference on state space models that do not admit an analytical solution. In this thesis, we will present several particle filtering algorithms adapted to a class of models known as Piecewise Deterministic Markov Processes (PDMP), i.e. processes governed by one or more parameters that admit random jumps in their value at random times. Our work will focus on object tracking, the estimation of a target's kinematic state over time from a sequence of noisy or incomplete measurements. Moreover, we will combine these techniques with Markov Chain Monte Carlo methods in order to infer the model parameters. We will perform sequential inference on both parameters and states by introducing an adaptation of the SMC^2 to PDMPs. Finally, all algorithms will be tested both on simulated and real-world data (Piraeus AIS Dataset).

keywords: Particle Filters, Sequential Monte Carlo, Hidden Markov Model, Piecewise Deterministic Process, Object Tracking, Markov Chains Monte Carlo.

Table of Contents

List of Figures **iv**

List of Tables **vi**

Acronyms **vii**

1 Introduction **1**

2 Background on Piecewise Deterministic Processes **3**
2.1 Basic Definitions 4
2.1.1 Markov Processes 4
2.1.2 Markov chains 4
2.2 Piecewise Deterministic Processes 5
2.2.1 Notation 6
2.2.2 Model specification 6
2.2.3 Example 7
2.2.4 Object Tracking 8

3 Introduction to Particle Filtering **9**
3.1 The Filtering Problem 10
3.2 The PF Algorithm 11
3.3 SMC for Filtering and Smoothing 12
3.4 Particle Markov Chain Monte Carlo 13
3.4.1 Markov Chain Monte Carlo 13
3.4.2 Particle MCMC 14
3.5 Diversification of the Particles 16

4 Particle Filters: Methods and Models **17**
4.1 PDP Particle Filter Mixture Kernels 17
4.1.1 Model 17
4.1.2 Construction of the Algorithm 18
4.1.3 Specifics of the Model 20
4.2 Poisson Tree Particle Filter 21
4.2.1 Model 22
4.2.2 Construction of the Algorithm 22
4.2.3 Intensity parameter 24

5 Parameters and State Estimations **27**
5.1 Particle MCMC based on PTPF 27
5.1.1 Extended Probability Distribution and Conditional PTPF 27
5.1.2 Poisson Tree Gibbs sampler (PTGS) and Poisson Tree Metropolis Hastings (PTMH) 29
5.1.3 Some Results 30
5.1.4 Gibbs Sampler based on PF with Mixture Kernels 30
5.2 Sequential Parameters and State Estimations: the SMC^2 Algorithm 31
5.2.1 Iterated Batch Importance Sampling Algorithm 31
5.2.2 The SMC^2 Algorithm 32
5.2.3 Adaptation of SMC^2 to PDPs 33

6 Experiments **35**
6.1 Results and Comparison with simulated data 35
6.1.1 Particle filters 35
6.1.2 Particle MCMC Methods 44
6.2 Real World Data 44
6.2.1 The Dataset 45
6.2.2 Preprocessing 46
6.2.3 New Model's Assumptions 46
6.2.4 Results 47

7 Discussion and Conclusion **50**

A Feynman-Kac formulation **52**

B Algorithms **53**
B.1 PMCMC Algorithms 53
B.1.1 Particle Independent Metropolis Hastings 53
B.1.2 Particle Marginal Metropolis Hastings 54
B.1.3 Conditional SMC 54

C The Kalman Filter **55**

Bibliography **57**

List of Figures

1.1 Overview of implemented models and algorithms. 2

3.1 Example of ancestral line generated using a conditional SMC algorithm with 5 particles and 4 time steps. 16

4.1 Resampling scheme. 20
4.2 Distribution of particles around a trajectory using a PF with mixture kernels. 21
4.3 Distribution of particles around a trajectory using a PTPF. 24
4.4 Example of Poisson tree. Edge $x_4 \in \mathcal{G}^r\mathcal{F}^{r+1}$, $x_3 \in \mathcal{F}_0^r$, while $x_5 \in \mathcal{F}^r \setminus \mathcal{F}_0^r$ ([14], Figure 1, page 7). 25

6.1 Generated data: hidden trajectory (blue), observations (red crosses) and locations of jumps (orange dots). 36
6.2 Estimated trajectories (with 95% CI) generated by PTPF (left) and PF with mixture kernels (right). 36
6.3 Estimated trajectories (with 95% CI) generated by PTPF (left) and PF with mixture kernels (right) with outlier. 37
6.4 RMSE for different numbers of initial particles (100, 1000, 10000) on the two components using PF with mixture kernels. 37
6.5 RMSE for different numbers of initial particles (100, 1000, 10000) on the two components using the PTPF. 38
6.6 Number of unique particle resampled at each iteration (above) and ESS at each iteration (below) fo PF with mixture kernels. 40
6.7 Number of unique particles at the last iteration for PF with mixture kernels and PTPF. 40
6.8 Distribution of particles locations around $10^{th}, 20^{th}$ and 30^{th} observations on x axis for n=1000 initial number of particles. 41
6.9 Distribution of particles locations around $10^{th}, 20^{th}$ and 30^{th} observations on y axis for n=1000 initial number of particles. 42
6.10 Distribution of particles locations around the 20^{th} observation on x and y axes for n=500000 initial number of particles. 43
6.11 Distribution of last five jump times of PF with mixture kernels, real jump times (red lines) and magnitude of the acceleration. 43
6.12 Distribution of jump times of PTPF, real jump times (red lines) and magnitude of the acceleration. 44

6.13 Distribution of simulated inter-arrival times for PTGS, PF MK GS and SMC^2. 45
6.14 Observations and filtering estimate for the trajectories of three vessels in the Piraeus AIS Dataset. 47
6.15 Prediction of the next observation using PF with mixture kernels and PTPF for three different trajectories. 48
6.16 Distribution of the simulated inter-arrival times for PTGS, PF MK GS and SMC^2. 49

List of Tables

6.1 RMSE for different numbers of initial particles (100, 1000, 10000) on the two components using the PF with mixture kernels. 37
6.2 Time employed for different numbers of initial particles ((100, 1000, 10000)) on the two components using PF with mixture kernels. 38
6.3 RMSE for different numbers of initial particles (10000, 20000, 30000) on the two components using the PF with mixture kernels. 38
6.4 Time employed for different numbers of initial particles (10000, 20000, 30000) using the PF with mixture kernels. 38
6.5 RMSE for different numbers of initial particles (100, 1000, 10000) on the two components using PTPF. 39
6.6 Time employed for different numbers of initial particles (100, 1000, 10000) using PTPF. 39
6.7 RMSE and computational time using $N = 20000$ initial particles for PTPF. . 39

Acronyms

AIS Automatic Identification System ..... i, v, 44–47, 50

CI Confidence Interval ..... iv, 36, 37

ESS effective sample size ..... iv, 19, 21, 32, 34, 39, 40

GPS Global Positioning System 45

GS Gibbs sampler ..... v, 2, 15, 30, 45, 48–50

HHM hidden Markov model 9

IBIS Iterated Batch Importance Sampling ..... iii, 31

KDE Kernel Density Estimate 44

MCMC Markov Chain Monte Carlo ..... i–iii, 1, 13, 14, 16, 27, 29, 33, 44, 50

MH Metropolis Hastings 14, 15

MPP marked point process 8

pdf probability density function 9, 44, 48

PDMP Piecewise Deterministic Markov Processes i

PDP Piecewise Deterministic Processes 1–3, 8, 12, 16, 19, 21, 35, 46, 50

PF Particle Filter ..... iii–vi, 2, 9–11, 15, 17–19, 21, 22, 30–43, 45–50

PGS Particle Gibbs sampler 14, 15, 28, 29

PIMH Particle Independent Metropolis Hastings ..... iii, 14, 30, 53

PMCMC Particle Markov Chain Monte Carlo ..... iii, 1, 2, 14, 27, 30, 33, 44, 50, 51, 53

PMMH Particle Marginal Metropolis Hastings ..... iii, 15, 54

PTGS Poisson Tree Gibbs sampler ..... iii, v, 2, 15, 29, 30, 45, 48–50

PTMH Poisson Tree Metropolis Hastings ..... iii, 15, 29, 30

PTPF Poisson Tree Particle Filter ..... iii–vi, 2, 17, 21–24, 27–30, 33–44, 47, 48, 50

RMSE Root Meas Squared Error ..... iv, vi, 36–39

SMC Sequential Monte Carlo ..... iii, iv, 1, 9, 10, 14–17, 22, 31, 50, 53, 54

SSM state-space model 9

Chapter 1

Introduction

The task of filtering, within the Bayesian framework, is equivalent to estimating the posterior distribution of the trajectory of an hidden stochastic process over a time interval, given a sequence of its noisy observations. Many state space models admit a discrete-time Markov chain as the underlying hidden process. This kind of models are employed in a variety of fields, ranging from economics to physics. The state space model, in many cases of interest, is non-linear, non-Gaussian and does not typically admit analytic solutions. As a result, approximation methods must be used. Particle filtering methods, or Sequential Monte Carlo (SMC), is a widespread class of algorithms used for tackling in an online manner these types of estimation problems through a collection of weighted samples, referred to as particles.

Discrete-time models have been the focus of the majority of research on SMC techniques and their adaptations. However, since continuous-time models are generally very effective at modelling many physical processes, SMC methods for continuous-time models are a very active area of research. Performing discretisation, although possible, comes with the downsides of being computationally taxing and producing biased estimates that are difficult to reconcile with their unbiased counterpart. [93, 48] present a general theoretical framework for particle filter methods in a continuous setting, as well as stability results. Moreover, SMC is often used in combination with Markov Chain Monte Carlo with the goal of infering the parameters governing a process. This class of methods is called Particle Markov Chain Monte Carlo (PMCMC).

SMC algorithms are frequently applied to object tracking, i.e. the estimation of the motion features of a particular target (e.g. aircraft, ships, etc.). We will see that a moving object often follows trajectories that behave as piecewise deterministic processes (PDPs). PDPs present deterministic dynamics in continuous-time aside from a countable set of stopping times where they randomly jump to a new value and, therefore, may present a trajectory with discontinuities.

The aim of this thesis is to present Particle Filtering algorithms that can be adapted to these dynamics. The challenge lies in the fact that this type of process defined in continuous-time admits the overlapping of different generations of particles, making it difficult to establish a criterion for performing resampling. It is therefore necessary to introduce some kind of

discretisation of the algorithm in order to "synchronise" [14] particles generated at different times. We will present two algorithms that tackle this problem in different ways, namely the Particle Filter with mixture kernels (PF MK) [101] and the Poisson Tree Particle Filter (PTPF) [14], as well as the respective PMCMC adaptations: the Poisson Tree Gibbs sampler (PTGS) [14] and the Particle Filter Mixture Kernels Gibbs sampler (PF MK GS). Finally we will introduce a version of the SMC^2 algorithm from [21] for PDPs. These models and methods are summarised in Figure 1.1.

The thesis is structured as follows. In Chapter 2, we will introduce the concept of Piecewise Deterministic Processes and the particular model of interest on which we will operate the algorithms. We proceed with an overview of Particle filtering and PMCMC in Chapter 3, introducing the basic forms of the most significant algorithms. Chapters 4 and 5 provide a more in-depth explanation of some of the algorithms introduced in Chapter 2, respectively two different particle filter algorithms and their use in the context of parameters and state estimations. Finally, Chapters 6 and 7 are dedicated to the presentation and discussion of the results obtained from the use of the aforementioned algorithms, applied to the model described in Chapter 2. The code we used for the implementation of the results can be found in [85].

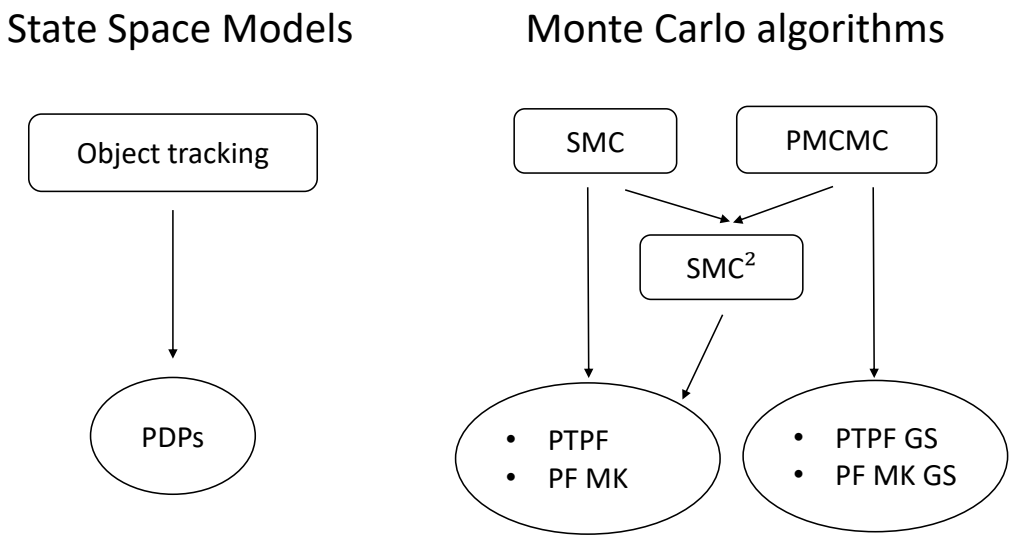

Figure 1.1: Overview of implemented models and algorithms.

Chapter 2

Background on Piecewise Deterministic Processes

Piecewise Deterministic Processes (PDP) were first introduced by [33] in 1984. The term refers to a class of non-diffusion stochastic processes, which evolves deterministically except for a countable set of random times, in which the process jumps to a new random value. [33] aimed at "providing a general family of stochastic models covering virtually all non-diffusion applications"[33].

The first attempts to build a general framework for non-diffusion processes were made by [25] and [60], who introduced a class of models called Piecewise Linear, with the goal of treating problems arising in queuing theory.

Mathematically, Piecewise Deterministic Processes are closely related to a class of stochastic jump processes, for which a stochastic calculus has been developed [32].

As discussed in [23], PDPs form a family of cadlag processes, which involve a deterministic motion interrupted by random jumps. The process' motion is characterised by two:

- The flow F;
- The transition measure Q, which determines the new location of the process at the jump time.

The process follows the (deterministic) flow $F(x,t)$ until the first jump time τ_1. Then, the new location of the jump is chosen according to the transition measure $Q(F(x,\tau_1),\cdot)$, and the process restarts from this new point. This fully describes a piecewise continuous trajectory. In 2006 [69] investigated the relation between PDP and marked point process.

Since their introduction, PDP have been largely studied in the literature, both from a theoretical and an applied point of view. Numerous problems in various fields involve PDP, such as biological population models, reliability, mathematical finance [30, 18] and object tracking problems [75, 95, 64]. The latter is the one we will focus on in the following sections.

2.1 Basic Definitions

In this section we will give the definition of Markov Process 2.1.1, Markov chain 2.1.2 and Piecewise deterministic process 2.2.

2.1.1 Markov Processes

For defining Markov Processes, we will use the definitions from [69]. Let $(\mathcal{F}_t)_{t\geq 0}$ be a filtration and let $X = (X_t)_{t\geq 0}$ be a measurable and adapted process defined on $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$, where:

- Ω is a set;
- $\mathcal{F}$ is a σ-algebra on Ω;
- $\mathbb{P}$ is a *probability measure*, i.e. $\mathbb{P}$ satisfies $\mathbb{P}(\Omega) = 1$.

Let X takes values in the state space $(G, \mathcal{G})$.

Definition 2.1.1. The process X is a Markov process with respect to the filtration $(\mathcal{F}_t)_{t\geq 0}$ if for every $s \leq t$ there exists a Markov Kernel $p_{st}(\cdot,\cdot)$ on G such that

$$P(X_t \in C|\mathcal{F}_s) = p_{st}(X_s, C) \qquad (C \in \mathcal{G})$$

X is a time-homogeneous process if, in addition, one may choose the Markov kernel depending on $(t-s)$ only. A time-homogeneous Markov process is also called a Markov process with stationary transition probabilities.

We call the Markov kernel p_{st} the transition probability from s to t. $p_{st}(\,\cdot\,)$ is not uniquely determined by 2.1.1 for all $x \in G$, however, since $\mathcal{G}$ is countably generated it holds that if p_{st} and p'_{st} are both transition probabilities, then $P(X_s \in C_{st}) = 1$, where $C_{st} = \{x \in G|p_{st}(x,\cdot) = p'_{st}(x,\cdot)\}$

2.1.2 Markov chains

For the definitions in this section, we will refer to [77].

Definition 2.1.2. A Markov chain is a discrete-time stochastic process $X = (X_n)_{n\geq 0}$ such that each random variable X_n takes values in a with values in a finite or countable set S (the state space) such that

$$P(X_{n+1} = j|X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_{n+1} = j|X_n = i)$$

(Markov property)

for every $n \geq 0$ and $j, i, i_{n-1}, \ldots, i_0 \in S$.

Moreover if we have that

$$P(X_{n+1} = j|X_n = i) = p_{ij}$$

does not depend on n, we call the Markov chain time-homogeneous.

The transition matrix of the chain is the matrix $P = (p_{ij})_{i,j \in S}$. It satisfies the following properties:

$$0 \leq p_{ij} \leq 1 \quad \forall i, j \in S \quad \text{and} \quad \sum_{j \in S} p_{ij} = \sum_{j \in S} P(X_{n+1} = j | X_n = i) = 1 \quad \forall i \in S$$

The transition graph of the chain is the oriented graph where vertices are states and an arrow from i to j exists if and only if $p_{ij} > 0$, taking value p_{ij} when it exists.

The distribution of the Markov chain at time $n \geq 0$ is given by

$$\pi_i^{(n)} = P(X_n = i) \quad i \in S$$

and its initial distribution is given by

$$\pi_i^{(0)} = P(X_0 = i) \quad i \in S$$

For every $n \geq 0$, we have $\sum_{i \in S} \pi_i^{(n)} = 1$.

Markov Chains on General State Spaces

We just defined Markov chains on a finite or countable set. We now generalise the notion to general state spaces, following the definitions from [92]. Let $\mathcal{X}$ be a general state space and $\mathcal{F}$ a σ-algebra with measurable subsets. The transition probabilities $\{P(x, A)\}_{x \in \mathcal{X}, A \in \mathcal{F}}$ are subject to the following assumptions:

- for each $x \in \mathcal{X}$, $P(x, \cdot)$ is a probability measure on $(\mathcal{X}, \mathcal{F})$;
- for each $A \in \mathcal{F}$, $P(x, A)$ is a measurable function of $x \in \mathcal{X}$.

Moreover ν is an initial distribution (any probability distribution on $(\mathcal{X}, \mathcal{F})$). Then the transition probabilities and the initial distribution define a Markov Chain $X_0, X_1, X_2, \ldots$ such that

$$P(X_0 \in A_0, X_1 \in A_1, \ldots, X_n \in A_n) = \int_{x_0 \in A_0} \nu(dx_0) \int_{x_1 \in A_1} P(x_0, dx_1) \ldots \\ \cdots \int_{x_{n-1} \in A_{n-1}} P(x_{n-1}, dx_n)$$

A stationary distribution for a Markov chain on general state space is a probability measure $\pi(\cdot)$ on $(\mathcal{X}, \mathcal{F})$ such that $\pi(A) = \int_{\mathcal{X}} \pi(dx) P(x, A)$ for all $A \in \mathcal{F}$. Markov chains on general state spaces may or may not have stationary distributions [92].

2.2 Piecewise Deterministic Processes

The following notation and definition are taken from [101].

2.2.1 Notation

We will assume that, with respect to the appropriate measure dx, the distributions admit a density. We will use the same symbol to refer to a measure and its associated probability density, then $\pi(x)dx = \pi(dx)$. The Dirac measure centered at x is indicated with $\delta_x(\cdot)$. For a Markov kernel $K(x, dy)$ acting from E_1 to E_2 and a probability measure $\mu(dx)$ on E_1, we will write the integral operation as $\mu K(\cdot) = \int_{E_1} K(x, \cdot)\mu(dx)$. The expression $\pi_n(x_k) = \int \pi_n(x_{1:n})dx_{1:k-1}dx_{k+1:n}$ describes the marginal of a joint density.

2.2.2 Model specification

Let's consider a Markov chain $(\tau_j, \theta_j)_{j\in\mathbb{N}}$, composed by pairs of non-decreasing times $\tau_j \in \mathbb{R}^+$ and parameters $\theta_j \in \Xi$. The transition kernel is

$$p(d(\tau_j, \theta_j)|\tau_{j-1}, \theta_{j-1}) = f(d\tau_j|\tau_{j-1})q(d\theta_j|\theta_{j-1}, \tau_{j-1}, \tau_j)$$

Consider the random continuous-time counting process $(\nu_t)_{t\geq 0}$

$$(\nu_t)_{t\geq 0} : \nu_t = \sum_{j=1}^{\infty} \mathbf{1}_{[0,t]}(\tau_j) = \max\{j : \tau_j \leq t\}$$

and the signal process $(\xi_t)_{t\geq 0}$ taking values in Ξ

$$\xi_t = F(t, \tau_{\nu_t}, \theta_{\nu_t})$$

with $\tau_0 = 0$, $\theta_0 = \xi_0$ and initial distribution $\xi_0 \sim q_0(\xi_0)$. Consider also the function $F : \mathbb{R}^+ \times \mathbb{R}^+ \times \Xi \to \Xi$ is deterministic with the condition $F(\tau_j, \tau_j, \theta_j) = \theta_j$.

Starting from ξ_0 a realization of the process $(\xi_t)_{t\geq 0}$ evolves deterministically according to F until the first jump time, which occurs at τ_1. At this time the process "jumps" to the value θ_1. The signal process evolves deterministically from the new value θ_1 until τ_2, at which it takes the value θ_2, and again for the next jumps.

We are interested in the number of jumps that occur in a finite time interval. If we consider the interval $[0, t_n]$ the number of jumps in that time-frame is $k_n = \nu_{t_n}$. We can write the joint probability distribution of the number of jumps and their locations as:

$$p_n(k_n, d\tau_{1:k_n}) = S(t_n, \tau_{k_n}) \prod_{j=1}^{k_n} f(d\tau_j|\tau_{j-1})$$

where S is the survivor function[1] associated with the transition kernel $f(d\tau_j|\tau_{j-1})$:

$$S(t, \tau) = 1 - \int_{\tau}^{t} f(ds|\tau)$$

The joint probability p has support in $\bigcup_{k=0}^{\infty}\{k\} \times \mathbb{T}_{n,k}$, where $\mathbb{R}^k \supset \mathbb{T}_{n,k} = \{\tau_{1:k} : 0 < \tau_1 < \tau_2 < \cdots < \tau_k \leq t_n\}$.

[1] The probability of no jumps in the interval (t_n, τ_{k_n})

We also have the random variables Y_n, from which the observation of the signal process in the n^{th} window $(t_{n-1}, t_n]$ are sampled. The observations in Y_n are conditionally independent of the past, given the signal in $(t_{n-1}, t_n]$.

Given a function F (deterministic with the conditions described above), the process $(\xi_t)_{t\in[0,t_n]}$ is uniquely identified by its skeleton $(k_n, \tau_{1:k_n}, \theta_{1:k_n})$. For each n, we can define a collection of random variables $X(k_n, \xi_{n,0}, \theta_{n,1:k_n}, \tau_{n,1:k_n})$ with values in

$$E_n = \bigcup_{k=0}^{\infty} \{k\} \times \Xi^{k-1} \times \mathbb{T}_{n,k}$$

Then we have that $E_n \subset E_{n+1}$.

In the following chapters, we will be interested in approximating the posterior distribution of $(\xi_t)_{t\in[0,t_n]}$ given the observation process $y_{1:n}$. Since the the signal process is a deterministic function of the jump times and parameters, in order to find the posterior distribution of $(\xi_t)_{t\in[0,t_n]}$ given $y_{1:n}$, it suffices to find the posterior distribution of X_n, denoted as $\pi_n(x_n)$, which has the form

$$\pi_n(x_n) \propto p_n(k_n, \tau_{n,1:k_n}) q_0(\xi_{n,0}) \prod_{j=1}^{k_n} q(\theta_{n,j}|\theta_{n,j-1}, \tau_{n,j}, \tau_{n,j-1}) \prod_{p=1}^{n} g(y_p|\xi_{n,t_{p-1},t_p})$$

where g is the likelihood function. In the subsequent chapters, we will describe how to approximate this distribution using Monte Carlo methods.

2.2.3 Example

In this section, we will consider the planar motion of a vehicle following piecewise constant acceleration dynamics.

The parameter θ has two components x and y, corresponding to the two coordinates of the plane. Each component contains three values, position s, velocity v, and acceleration a. In this setting Ξ takes values in $\mathbb{R}^6$. We have that

$$\theta_j = \begin{bmatrix} \theta_j^x \\ \theta_j^y \end{bmatrix}, \quad \theta_j^x = \begin{bmatrix} s_j^x \\ v_j^x \\ a_j^x \end{bmatrix}, \quad \theta_j^y = \begin{bmatrix} s_j^y \\ v_j^y \\ a_j^y \end{bmatrix}$$

and that

$$F(t, \tau_{\nu_t}, \theta_{\nu_t}) = \begin{bmatrix} F^x(t, \tau_{\nu_t}, \theta_{\nu_t}) \\ F^y(t, \tau_{\nu_t}, \theta_{\nu_t}) \end{bmatrix}$$

where

$$F^x(t, \tau_{\nu_t}, \theta_{\nu_t}) = \begin{bmatrix} 1 & (t-\tau_{\nu_t}) & \frac{1}{2}(t-\tau_{\nu_t})^2 \\ 0 & 1 & (t-\tau_{\nu_t}) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{\nu_t}^x \\ v_{\nu_t}^x \\ a_{\nu_t}^x \end{bmatrix}$$

and F^y behaves equivalently.

The inter arrival times follow some given distribution $f(d\tau_j|\tau_{j-1})$, while the transition kernel $q(d\theta_j|\theta_{j-1}, \tau_{j-1}, \tau_j)$ which governs the random jump of the acceleration components at the random time τ_j has the form

$$\begin{aligned} q(d\theta_j^x|\theta_{j-1}, \tau_{j-1}, \tau_j) &= \delta_{s_j^{x,-}}(ds_j^x)\delta_{v_j^{x,-}}(dv_j^x)q(da_j^x) \\ s_j^{x,-} &= s_{j-1}^x + v_{j-1}^x(\tau_j - \tau_{j-1}) + \frac{a_{j-1}^x}{2}(\tau_j - \tau_{j-1})^2 \\ v_j^{x,-} &= v_{j-1}^x + a_{j-1}^x(\tau_j - \tau_{j-1}) \end{aligned}$$

At time $t = 0$ the vehicle has position, velocity and acceleration equal to a given ξ_0.

This model could be suitable for tracking highly maneuvering targets as in [9].

2.2.4 Object Tracking

The aim of tracking is to estimate a target's kinematic state (position, velocity, acceleration, ...) over time from a sequence of noisy or incomplete measurements. The target's state is a continuous process and usually presents a specific underlying structure. Nonetheless, a discrete-time Markov process is used in various tracking systems to represent target dynamics, and its state sampling rate is determined by the frequency at which measurements arrive [61]. By discretising the state at observation time, we create a relatively simple setting in which Kalman filtering [1] or particle filtering [15, 53] and smoothing methods may be used. In real datasets, however, target trajectories are characterised by long periods of smoothness with only a few sharp changes. For instance, aircraft and marine vehicles mostly transit in a straight line with occasional turns to intercept new headings. The same principle applies in many other situations involving unpredictably moving objects. This means that if the state sampling rate is adjusted to the nature of the data we could, theoretically, obtain a much more accurate representation of target trajectories. This would mean allocating more particles in regions with sharp changes in trajectory and fewer in smoother regions [78].

[61, 64, 101] introduced models that consider the kinematic state as a continuous, deterministic process dependent on dynamics parameters and an underlying sequence of changepoint times. It can be thought of as a realised marked point process (MPP), and the resulting state trajectory as a PDP [69]. Through variable rate particle filters, this sequence of changepoints can be numerically estimated, yielding an approximation of the posterior distribution of the current state. In tracking problems, the start and end of target manoeuvres are represented by these changepoints, each of which is controlled by a parameter vector [12].

[61] provides an overview of the most relevant work in modelling and estimating continuous-time jump models for application in tracking scenarios.

Chapter 3

Introduction to Particle Filtering

When solving real-world problems, we are often asked to estimate an unknown quantity given certain observations of it. If we have prior knowledge about the phenomenon we want to approximate, we can make inferences about it using a Bayesian model, based on the well-known formula

$$\underbrace{p(x|y)}_{\text{posterior distribution of x given y}} \propto \underbrace{p(y|x)}_{\text{likelihood}} \cdot \underbrace{p(y)}_{\text{prior on } y} \tag{3.1}$$

for some hidden process x and its observation y.

In many cases, the observations in y are collected sequentially. Therefore, we need to update the posterior distribution as more data becomes available (online inference). This type of algorithm is prevalent in the field of inference on parameters and states of state-space models (SSM). An example could be tracking an aircraft given some radar measurements, as in [9].

In a few cases, it is possible to derive an exact analytical expression to compute the sequence of posterior distributions. For instance, when the SSM that models the data is Gaussian and linear, we can use the recursion given by the well-known Kalman filter. If, on the other hand, the data belong to a partially observed, finite-state Markov chain model, it is possible to find a solution using the hidden Markov model (HHM) filter. While these two filters are the most widespread, there are configurations which admit other finite-dimensional filters ([99, 100]). However, most real-world application use complex data which cannot be solved in closed form since they could be non-linear, high-dimensional or non-Gaussian.

Particle Filters (PFs), also called Sequential Monte Carlo (SMC) methods, were introduced in 1993 by [86] as a new algorithm applicable in general settings. They are a general class of Monte Carlo methods that sample sequentially from a sequence of probability density functions (pdfs). PFs have a lot of advantages: they are parallelisable, flexible and easy to implement. Unlike some other algorithms, such as the extended Kalman filter [98, 51], PFs do not make use of linearisation techniques or functional approximation. Nevertheless, PF are computationally expensive. Fortunately, given the increasing availability of computational

power, they can be successfully employed in various fields, such as computer vision [94, 87], robotics [96], economics and mathematical finance [98, 31, 26], computational physics and bioinformatics [66].

Particle filters can be viewed in statistics and probability as mean-field particle interpretations of Feynman-Kac probability measures. Molecular chemistry and computational physics gave origin to particle integration techniques [91, 67]. Path particle integration techniques of the Feynman-Kac type are also employed in Quantum Monte Carlo, notably Diffusion Monte Carlo methods [47, 5, 13], in computational physics. Furthermore, Feynman-Kac interacting particle techniques are linked to mutation-selection genetic algorithms, which are utilised in evolutionary computing to tackle complicated optimisation problems.

Given their increasing popularity, since their introduction PFs have been largely addressed by the literature, both from a theoretical and applied point of view. A lot of tutorials have been published on the subject, such as [83, 15, 55, 54]. The more recent paper [53], provides a simple and unified framework including all the basic and advanced SMC methods.

The study of particle filter convergence began in 1996 [35, 36] and culminated in 2000 with the publication of the book [46] and a series of articles [27, 42, 43, 41, 84, 49, 44]. The more recent books [38, 37] also address this topic.

In the following section, we present a general probabilistic model and its Bayesian inference objectives, as well as a basic form of the PF algorithm.

3.1 The Filtering Problem

We now introduce the problem of estimating the posterior distribution of a hidden process, i.e. a general version of the piecewise deterministic model described in section 2.2.2.

Let's consider a Markov process $(x_t)_{t\in\mathbb{N}}$ with $x_t \in \mathcal{X}$, initial distribution $\mu(x_1)$ and transition probability $p(x_t|x_{t-1})$. The process x is hidden, but we have some observations of it $(y_t)_{t\in\mathbb{N}}$ and $y_t \in \mathcal{Y}$. We assume the observations to be conditionally independent, given the hidden process.

We write $x_{1:t}$ and $y_{1:t}$ to indicate the signal process and its observation up to time t. Our goal is to estimate recursively the posterior distribution $p(x_{1:t}|y_{1:t})$. In order to do that, we use Bayes formula for conditional probabilities (as in (3.1))

$$p(x_1, x_2, \dots, x_t|y_1, y_2, \dots, y_t) = \frac{p(y_1, y_2, \dots, y_t|x_1, x_2, \dots, x_t)p(x_1, x_2, \dots, x_t)}{p(y_1, y_2, \dots, y_t)}$$

where

$$p(y_1, y_2, \ldots, y_t) = \int p(y_1, y_2, \ldots, y_t | x_1, x_2, \ldots, x_t) p(x_1, x_2, \ldots, x_t) dx_1 d_2 \ldots dx_t$$

$$p(y_1, y_2, \ldots, y_t | x_1, x_2, \ldots, x_t) = \prod_{i=1}^{t} p(y_i | x_i)$$

$$p(x_1, x_2, \ldots, x_t) = p_1(x_1) \prod_{i=1}^{t} p(x_i | x_{i-1})$$

Particle filters provide us with an approximation of this probability density. As the number of particles we employ in the algorithm grows, the result becomes more accurate [35, 36, 37, 42, 43]. The nonlinear filtering problem consists of computing sequentially the distributions given by the nonlinear filtering equation

$$p(x_t | y_1, y_2, \ldots, y_{t-1}) \longrightarrow p(x_t | y_1, y_2, \ldots, y_t) = \frac{p(y_t | x_t) p(x_t | y_1, y_2, \ldots, y_{t-1})}{p(y_t | x'_t) p(x'_t | y_1, y_2, \ldots, y_t) dx'_t} \quad (update)$$

$$\longrightarrow p(x_{t+1} | y_1, y_2, \ldots, y_t) = \int p(x_{t+1} | x_t) p(x_t | y_1, y_2, \ldots, y_t) dx_t \quad (prediction)$$

where $p(x_1 | y_1, y_2, \ldots, y_{t-1}) = p(x_1)$ for $t = 1$.

In Appendix A we provide a Feynman-Kac formulation of the filtering problem.

3.2 The PF Algorithm

We now describe the Particle Filter algorithm that we will use to achieve the goal we set in the previous section, namely the recursive approximation of the sequence of densities $\pi_t(x_t|\theta) = p(x_t|y_{1:t}, \theta)$ (where θ is a fixed parameter).

First, we define a fixed number N_x of independent random variables $(x_1^n)_{1 \leq n \leq N_x}$, with common prior distribution $\mu_\theta(x_1)$. The algorithm consists mainly of three steps, a *selection*, an *update* and a *weighting* step:

- Selection: N_x conditionally independent random variables are sampled from the previous iteration (this step is skipped in the first iteration of the algorithm)
- Update: we update each of the selected particles according to a transition probability

  $$x_{t-1}^n \longrightarrow x_t^n \sim q_{t,\theta}(\cdot | x_{t-1}^n)$$

  for $n \in N_x$
- Weighting: we assign a weight to each particle (which will be used in the selection step of the following iteration)

We now provide a pseudo-code version of the PF algorithm just described (Algorithm 1). The algorithm we propose is from [21], however other schemes also exists [16, 72, 81].

Algorithm 1 Particle Filter

Step 1: at iteration $t = 1$:

1: sample x_1 from $p_{1,\theta}(x_1)$

2: Compute and normalise the weights:

$$w_{1,\theta}(x_1^n) = \frac{\mu_\theta(x_1^n) g_\theta(y_1|x_1^n)}{q_{1,\theta(x_1^n)}}, \qquad W_{1,\theta}^n = \frac{w_{1,\theta}(x_1^n)}{\sum_{i=1}^{N_x} w_{1,\theta}(x_1^i)}$$

Step 2: for $t = 2 : T$:

1: sample the index $a_{t-1}^n \sim \mathcal{M}(W_{t-1,\theta}^{1:N_x})$ of the ancestor particle n

2: sample x_t^n from $q_{t,\theta}(\cdot|x_{t-1}^{a_{t-1}^n})$

3: Compute and normalise the weights:

$$w_{t,\theta}(x_{t-1}^{a_{t-1}^n}, x_t^n) = \frac{p_\theta(x_t^n|x_t^{a_{t-1}^n}) g_\theta(y_t|x_t^n)}{q_{t,\theta}(x_t^n|x_{t-1}^{a_{t-1}^n})}, \qquad W_{t,\theta}^n = \frac{w_{t,\theta}(x_{t-1}^{a_{t-1}^n}, x_t^n)}{\sum_{i=1}^{N_x} w_{t,\theta}(x_{t-1}^{a_{t-1}^i}, x_t^i)}$$

$\mathcal{M}(W_{t-1,\theta}^{1:N_x})$ stands for the multinomial distribution which assigns probability $W_{t-1,\theta}^{1:N_x}$ to outcome $n \in N_x$ and $(q_{t,\theta})_{t\in T}$ stands for a sequence of conditional proposal distributions depending on θ. q is often chosen equal to the prior $q_{1,\theta}(x_1) = \mu_\theta(x_1)$ and $q_{t,\theta}(x_t|x_{t-1}) = \mu_\theta(x_t|x_{t-1})$ for $t \geq 2$. For this choice, the weights have the form $w_{t,\theta}(x_{t-1}^{a_{t-1}^n}, x_t^n) = g_\theta(y_t|x_t^n)$, where g_θ is the likelihood function.

The one we just presented is the basic form of a Particle Filter algorithm. In Chapter 4 we will present some algorithms for the approximation of the posterior distribution of a PDP given its observations.

3.3 SMC for Filtering and Smoothing

Consider again the following Markov model [100]:

$$\begin{aligned} x_{t+1} &\sim p(x_{t+1}|x_t) && \text{transition density} \\ y_{t+1} &\sim g(y_{t+1}|x_{t+1}) && \text{observation density} \end{aligned}$$

where $\{x_t\}$ are the (hidden) states of the system and $\{y_t\}$ its observation over a given time interval $t \in \{1, 2, \ldots, T\}$. Note that $p(\cdot|\cdot)$ and $g(\cdot|\cdot)$ may be non-Gaussian and involve non-linearity.

As we have already anticipated in the previous section, the problem we will deal with is the estimation of the filtering distribution, i.e. the posterior $p(x_t|y_{1:t})$.

On the other hand, in the smoothing problem, all sample observations are used, therefore also those from the future. Hence, the objective will be to approximate $p(x_t|y_{1:T})$.

Smoothing can be performed recursively backwards in time using the smoothing formula [63]

$$p(x_t|y_{1:T}) = \int p(x_{t+1}|y_{1:T}) \frac{p(x_t|y_{1:t})p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})} dx_{t+1}$$

Smoothing features are less well established than particle filtering, which is now a very well-known theory and practice. In order to approximate the individual marginal smoothing densities, existing methods for smoothing with particle filters have either used the two-filter formula [71] or the two-filter method [51, 71]. Since analyses of historical states typically focus on trajectories and therefore call for the analysis of collections of states together, these marginal distributions are of little interest in many applications. [62] provides a sequential procedure for Maximum a Posteriori (MAP) sequence estimation based on the Viterbi algorithm and dynamic programming if a single "best" estimate for the smoothed trajectory is required. In the Bayesian inference context, a single best estimate is uncommonly appropriate, particularly when distributions are multimodal, therefore, in this case, we aim for random state sequence generation.

By allowing the random creation of whole historical trajectories derived from the joint smoothing density $p(x_{1:t}|y_{1:t})$, the new methods complete particle filtering methodology. First, a particle-based approximation of the filtering density at each time step is created and stored using a forward filtering pass. The next step is to perform a backwards "simulation smoothing" pass in order to produce sampled realisations from the smoothing density.

We can make a parallelism with a technique used in linear Gaussian models and hidden Markov models, in particular, the nonlinear/non-Gaussian equivalent of forward filtering/backwards sampling methods [17, 57, 34].

There is a notable difference between the suggested method and the MAP estimation discussed in [62]. The latter only uses the forward particle filter to create a grid of potential state values at each time point and employs the Viterbi algorithm to determine the most likely state trajectory across that grid of state values.

In this thesis, we will not address the smoothing problem but only the filtering problem. For further reading see [73, 40, 10, 56, 65, 50].

3.4 Particle Markov Chain Monte Carlo

3.4.1 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods, as says the name, combine the properties of Markov chains and Monte Carlo methods:

- Monte Carlo consists in estimating the properties of a distribution by analysing random samples from that very same distribution.
- The Markov chain side of this method refers to the concept an underlying sequential process is involved in the sampling, and each random sample is used as a waypoint for

generating the next one. Each new sample depends just on the one before it (Markov property).

MCMC is of great use in Bayesian inference as the latter focuses on posterior distributions, which are often difficult to work with via analytic examination. These algorithms target the distribution $p_\theta(x_{1:T}|y_{1:T})$ (or $p(x_{1:T}, \theta|y_{1:T})$).

One of the most popular MCMC algorithm is the Metropolis Hastings (2).

Algorithm 2 Metropolis Hastings

1: sample $X'_{1:T} \sim q_\theta(x_{1:T}|y_{1:T})$
2: given a current state $s_n = X_{1:T}$ we compute the acceptance probability:

$$p_a = \min(1, \frac{p_\theta(X'_{1:T}|y_{1:t})q_\theta(X_{1:T}|y_{1:t})}{p_\theta(X_{1:T}|y_{1:t})q_\theta(X'_{1:T}|y_{1:t})})$$

3: set $s_{n+1} = X'_{1:T}$ with probability p_a, and $s_{n+1} = X_{1:T} = s_n$ with probability $1 - p_a$

where $q_\theta(x_{1:t}|y_{1:t})$ is a given proposal density.

If we are targeting $p(x_{1:T}, \theta|y_{1:T})$, it is common to sample alternatively from $p(x_{1:T}|\theta, y_{1:T}) = p_\theta(x_{1:T}|y_{1:T})$ and $p(\theta|x_{1:T}, y_{1:T})$. However, sampling exactly from $p_\theta(x_{1:T}|y_{1:T})$ is feasible only if the model is linear Gaussian or if it is a finite state space hidden Markov model ([17, 58]). Otherwise, we require the design of the proposal densities making this type of algorithm often impossible to implement.

3.4.2 Particle MCMC

The idea behind Particle MCMC Particle Markov Chain Monte Carlo (PMCMC) methods, introduced by [2, 3], consists of approximating the standard MCMC algorithms by taking the proposal distribution to be the output of an SMC algorithm using $N_x \geq 1$ particles and targeting $p_\theta(x_{1:t}|y_{1:t})$, for instance in a MH update [29].

PMCMC are "exact approximations" of the standard MCMC algorithms targeting either $p_\theta(x_{1:T}|y_{1:T})$ or $p(x_{1:T}, \theta|y_{1:T})$, i.e. "for any fixed number $N_x \geq 1$ of particles their transition kernels leave the target density of interest invariant"[3]. Moreover, they can be seen as standard MCMC updates that, under mild standard assumptions (see [3] for more details), lead to convergence.

The two most popular PMCMC algorithms are Particle Gibbs sampler (PGS) [29, 28] and Particle Independent Metropolis Hastings (PIMH) [22, 45].

In Appendix B.1.1 we provide a pseudo code for the PIMH algorithm. This is the same as the standard MH algorithm described in Algorithm 2, but, instead of sampling from a probability distribution $q_\theta(x_{1:T}|y_{1:T})$, we use an SMC algorithm targeting the actual posterior

distribution $p(x_{1:T}|y_{1:T})$.

If we want to sample from $p(\theta, x_{1:T}|y_{1:T})$ instead, we have to use another version of Metropolis-Hastings, namely the Particle Marginal Metropolis Hastings (PMMH) [3] (see Appendix B.1.2). We already know that

$$p(\theta, x_{1:T}|y_{1:T}) \propto p(\theta|y_{1:T})p(x_{1:T}|y_{1:T})$$

This version of the algorithm, also used in [7, 4], uses as MH update

$$q(\theta', x'_{1:T}|\theta, x_{1:T}) = q(\theta'|\theta)p_{\theta'}(x'_{1:T}|y_{1:T}) \tag{3.2}$$

where $p_{\theta'}(x'_{1:T}|y_{1:T})$ is "adapted" to θ and it can be substituted by an SMC approximation of it. From (3.2) followed that the acceptance probability is computed as

$$\frac{p(\theta', x'_{1:T}|y_{1:T})q(\theta, x_{1:T}|\theta', x'_{1:T})}{p(\theta, x_{1:T}|y_{1:T})q(\theta', x'_{1:T}|\theta, x_{1:T})} = \frac{p_{\theta'}(y_{1:T})p(\theta')q(\theta|\theta')}{p_{\theta}(y_{1:T})p(\theta)q(\theta'|\theta)}$$

For more details see [3]

PGS is a different algorithm used to sample from the posterior density $p(\theta, x_{1:T}|y_{1:T})$. The difference from the previous algorithms is that in the step using an SMC algorithm, we must instead use a conditional SMC update. This type of update is similar to that produced by its standard counterpart, except that one of the trajectories is fixed at the beginning. This means that this trajectory will "survive" and be present among the final trajectories given by the output of the algorithm. In Appendix B.1.3 is given a general version of the conditional SMC. Figure 3.1 shows an example of ancestral lineages generated by a conditional SMC algorithm.

Now we can present a pseudo-code for the PGS (3).

Algorithm 3 Particle Gibbs sampler

1: **Initialization**: set $n = 0$ and $\theta(0)$, $X_{1:T}$ and $B_{1:T}(0)$ arbitrarily.
2: for $n \geq 0$:

- sample $\theta_i \sim p(\cdot|y_{1:T}, X_{1:T}(n-1))$
- run a conditional SMC algorithm targeting $p(x_{1:T}|y_{1:T})$ conditional on $X_{1:T}(n-1)$
- sample $X_{1:T} \sim \hat{p}_{\theta(i)}(\cdot|y_{1:T})$

In section 5.1 we will introduce three versions of this class of methods, namely the Poisson Tree Metropolis Hastings, the Poisson Tree Gibbs sampler [14] and the Particle Filter with Mixture Kernels Gibbs sampler.

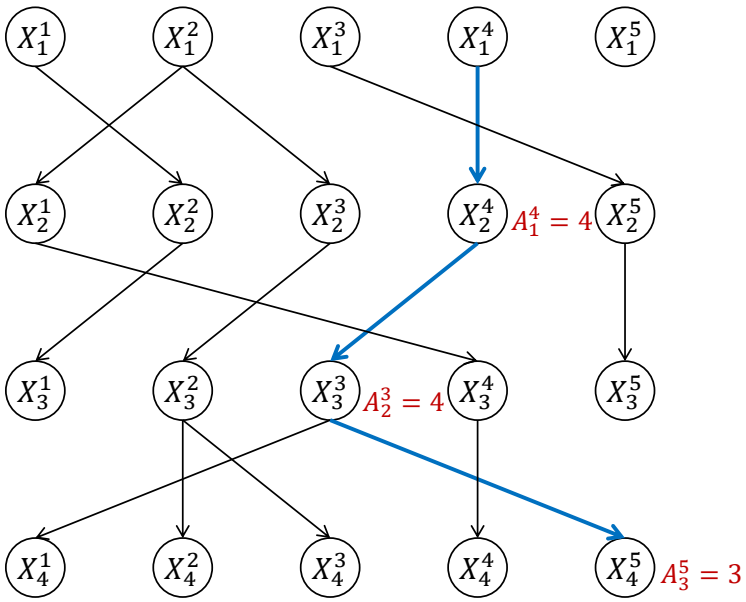

Figure 3.1: Example of ancestral line generated using a conditional SMC algorithm with 5 particles and 4 time steps.

3.5 Diversification of the Particles

Resampling means producing several copies of the same particle, and is a practice used in SMC algorithms to decrease variance in importance weights (Figure 4.1). In the context of PDPs, we may incur the case where no jump occurs between two or more consecutive observations. Thus, if we sample from the prior distribution of the model under consideration, it is possible that no diversification in particles occurs for multiple iterations, which could result in the propagation of errors and instability of the algorithm [101].

The problem of diversification and repeated calculations was faced in [61], which, however, does not consider the situation in which new particles are proposed without taking into account new observations. Other authors [20] suggest the use of MCMC diversification moves [59] are crucial for the correct and efficient functioning of the SMC algorithm. In chapter 4 we will present an algorithm from [101] that makes use of an *adjustment move* to tackle this problem.

Chapter 4

Particle Filters: Methods and Models

In this chapter we will present, analyse and compare two existent PF algorithm, namely the PF with mixture kernels [101] and the Poisson Tree Particle Filter [14].

4.1 PDP Particle Filter Mixture Kernels

In [39] a general framework for SMC samplers is presented. Let's consider the model presented in Section 2.2.2 from [101]. We wish to approximate the sequence of probability measure $(\pi_n)_{n\in\mathbb{N}}$, which is defined on the sequence of spaces $(\prod_{p=1}^{n} E_p)_{n\in\mathbb{N}}$ (where $(E_n)_{n\in\mathbb{N}}$ is a collection of spaces). Note that the space sequence has the target at time index n as a marginal distribution at that time index.

We wish to approximate the target distribution by the sequence $(\tilde{\pi}_n)_{n\in\mathbb{N}}$, which has the form

$$\tilde{\pi}_n(x_{1:n}) = \pi_n(x_n) \prod_{p=1}^{n-1} L_p(x_{p+1}, x_p)$$

where L_n is a "backward" kernel acting from E_{n+1} into E_n. This structure leads to the incremental weights

$$w_n(x_{n-1}, x_n) \propto \frac{\pi_n(x_n)\beta_{n-1,m}(x_n)L_{n-1}(x_n, x_{n-1})}{\pi_{n-1}(x_{n-1})\alpha_{n,m}(x_{n-1})K_n(x_{n-1}, x_n)} \tag{4.1}$$

where K_n is the Markov kernel extending $\tilde{\pi}_{n-1}$ to $\tilde{\pi}_n$. At time n the importance weights depend only on x_n and x_{n-1}, and not on the history of the process.

4.1.1 Model

The model we will refer to in the following sections is the one described in section 2.2.2 and 2.2.3.

4.1.2 Construction of the Algorithm

The objective, as already mentioned, is to approximate the distribution of the hidden process given its observations. Recall that $\{(k_n, \tau_{k_n}, \theta_{k_n})^{(i), w_n^{(i)}}\}_{i=1}^N$ is the set of N particles generated by us, which are used to approximate the signal process in the following way

$$p^N(d\xi_{t_n}|y_{1:n}) = \sum_{i=1}^{N} w_n^{(i)} \delta_{\xi_{t_n}^{(i)}}(d\xi_{t_n}), \qquad \delta_{\xi_{t_n}^{(i)}} = F(t_n, \tau_{k_n^{(i)}}^{(i)}, \theta_{k_n^{(i)}}^{(i)})$$

In order to build this approximation we need to compute the weights as specified in formula (4.1). To do that, we have to define two fundamental elements:

- The "forward" (proposal) kernels K_n;
- The "backward" kernels L_n.

It is also possible to use kernels which are mixtures, i.e. formed by many components. In the proposed algorithm, we will use a mixture kernel in which each component applies a different increment to the parameter k_n. The proposal kernel will have the general form

$$K_n(x_{n-1}, x_n) = \sum_{m=1}^{M} \alpha_{n,m}(x_{n-1}) K_{n,m}(x_{n-1}, x_n) \qquad \forall x_n, \ \sum_{m=1}^{M} \alpha_{n,m} = 1 \tag{4.2}$$

which means it is a function of the current state. Note that M represents the total number of components in the kernel.

When we apply a proposal kernel in this form, there exists an optimal[1] backward kernel (provided by [39]), which leads to the following incremental weights

$$w_n(x_{n-1}, x_n) \propto \frac{\pi_n(x_n)}{\int_{E_{n-1}} \pi_{n-1}(x_{n-1})[\sum_{m=1}^{M} \alpha_{n,m}(x_{n-1} K_{n,m}(x_{n-1}, x_n))] dx_{n-1}} \tag{4.3}$$

However, often the denominator in equation (4.3) is intractable. In this cases we have to use an approximation of the optimal backward kernel, which takes the form

$$L_{n-1}(x_n, x_{n-1}) = \sum_{m=1}^{M} \beta_{n-1,m}(x_n) L_{n-1,m}(x_n, x_{n-1})$$

If we employ kernels in this form, the incremental importance weights will be given by

$$w_n(x_{n-1}, x_n, m) \propto \frac{\tilde{\pi}_n(x_{1:n})}{\tilde{\pi}_{n-1}(x_{1:n-1}) K_{x_{n-1}, x_n}} = \frac{\pi_n(x_n) L_{n-1,m}(x_n, x_{n-1})}{\pi_{n-1}(x_{n-1}) K_{n,m}(x_{n-1}, x_n)} \tag{4.4}$$

The choice of kernels is crucial for the correct functioning of the algorithm. We will now give an overview of the possible kernel choices and the type of incremental weights they lead to, together with a pseudo-code version of the PF with mixture kernels algorithm from [101]. In the next section, we will give the precise form of the kernels dependent on the model assumptions.

We will consider two different types of moves: the birth move and the adjustment move.

[1] Optimal in this case means that, by performing resampling at every step, the variance of the importance weights is minimized.

- **Birth move**: We call the kernel associated with this move $K_{n,b}$. This move consists in adding one or more jump times and associated parameters. In the simple case (single birth move) the dimensionality is increased by 1, $k_n = k_{n-1} + 1$ and a new jump τ_{n,k_n} and parameter θ_{n,k_n} are sampled respectively from the distributions $h_n(\cdot|\tau_{n-1,k_{n-1}})$ (on $(\tau_{n-1,k_{n-1}}, t_n]$) and ${}_n(\cdot|x_n \backslash \theta_{k_n})$. The birth kernel will have the form

$$K_{n,b}(x_{n-1}, dx_n) = \delta_{k_{n-1}+1}(k_n)\delta_{\tau_{n-1,1:k_{n-1}}}(d\tau_{n,1:k_{n-1}}) \times \delta_{\theta_{n-1,1:k_{n-1}}}(d\theta_{n,1:k_n-1})h_n(d\tau_{n,k_n}|\tau_{n-1,k_{n-1}})\eta_n(d\theta_{n,k_n}|x_n \backslash \theta_{n,k_n})$$

- **Adjustment move**: We call the kernel associated with this move $K_{n,a}$. This move preserves the number of jumps, but changes the location of the most recent jump by sampling another one in the interval $(\tau_{n-1,k_{n-1}-1}, t_n]$ from the distribution $h_n(\cdot|x_{n-1})$. The adjustment move does not change the parameter associated with the jump. This move leads to a kernel in the form

$$K_{n,a} = \delta_{k_{n-1}}(k_n)\delta_{\tau_{n-1,1:k_{n-1}-1}}(d\tau_{n,1:k_{n-1}}) \times \delta_{\theta_{n-1,1:k_{n-1}}}(d\theta_{n,1:k_n})h_n(d\tau_{n,k_n}|x_{n-1})$$

In Algorithm4 we present a version of the PF algorithm for PDPs.

Algorithm 4 PDP Particle Filter with mixture kernels

1: n=1
2: **for** $i = 1$ to N **do**
3: $\quad X_1^{(i)} \sim \eta(\cdot)$
4: $\quad w_1^{(i)} \propto \frac{\pi_1(X_1^{(i)})}{\eta(X_1^{(i)}))}$
5: **end for**
6: $n \leftarrow n + 1$
7: **for** $i = 1$ to N **do**
8: $\quad X_n^{(i)} \sim K_n(X_{n-1}^{(i)}, \cdot)$
9: $\quad w_n^{(i)} \propto w_{n-1}^{(i)} \frac{\pi_n(X_n^{(i)})L_{n-1}(X_n^{(i)}, X_{n-1}^{(i)})}{\pi_{n-1}(X_{n-1}^{(i)})K_n(X_{n-1}^{(i)}, X_n^{(i)})}$
10: **end for**
11: Resampling can be conducted at this stage.
12: Optionally, move each $X^{(i)}$ according to a π_n-invariant Markov kernel.
13: **Go to** 6

Figure 4.2 shows how a potential output of 4 looks like.

In order to avoid the problem described in section 3.5, we usually resample when the effective sample size (ESS), defined as [74]

$$\text{ESS} = \frac{\left(\sum_{i=1}^N w_i\right)^2}{\sum_{i=1}^N (w_i)^2} \tag{4.5}$$

falls below a given threshold, usually set to be half of the total number of particles. From the definition of ESS, this means that we resample whenever a few particles take on a very high weight relative to the rest, leading to a degenerate model. Since the weights are normalised at each step (so they sum to 1), the numerator in (4.5) is equal to 1. Each time resampling is performed, all particles are assigned an equal weight (see resampling scheme Figure 4.1).

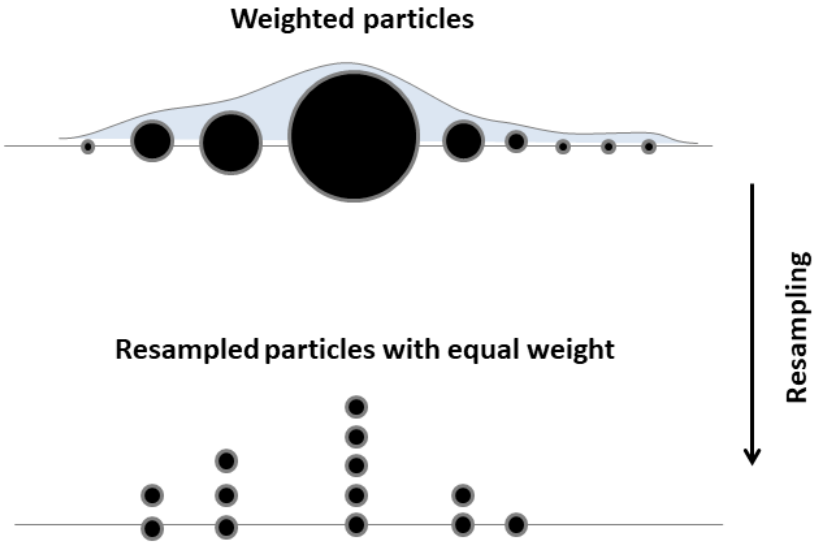

Figure 4.1: Resampling scheme.

4.1.3 Specifics of the Model

We will apply the algorithm just described to the model presented in section 2.2.3. In particular, this model will be used in the context of object tracking. To do this, we will apply a forward mixture kernel consisting of two elements.

- A birth move: this move adds a single jump uniformly in the interval $(\tau_{n,k_n-1}, t_n]$. The forward kernel has the form

$$K_{n,1}(x_{n-1}, dx_n) = \delta_{k_{n-1}+1}(k_{n-1})\delta_{\tau_{n-1,1:k_{n-1}}}(d\tau_{n,1:k_{n-1}}) \times \frac{d\tau_{n,k_n}}{t_n - \tau_{n,k_n-1}} 1_{(\tau_n,k_n-1,t_n]}(\tau_{n,k_n})$$

  while the correspondent backward kernel is

$$L_{n-1,1}(x_n, dx_{n-1}) = \delta_{k_n-1}(k_{n-1})\delta_{\tau_n,1:k_n-1}(d\tau_{n-1,1:k_{n-1}})$$

- An adjustment move: this move applies a Gaussian random walk kernel to the last jump time. The kernel is restricted to $(\tau_{n,k_n-1}, t_n]$ and we can write it as

$$K_{n,2}(x_{n-1}, dx_n) \propto 1_{(\tau_{n-1,k_{n-1}-1},t_n]}(\tau_{n,k_n})\mathcal{N}(\tau_{n-1,k_{n-1}}, \sigma^2_{adjust})d\tau_{n,k_n} \times \delta_{k_{n-1}}(k_n)\delta_{\tau_{n-1},1:k_{n-1}-1}(d\tau_{n,1:k_n-1})$$

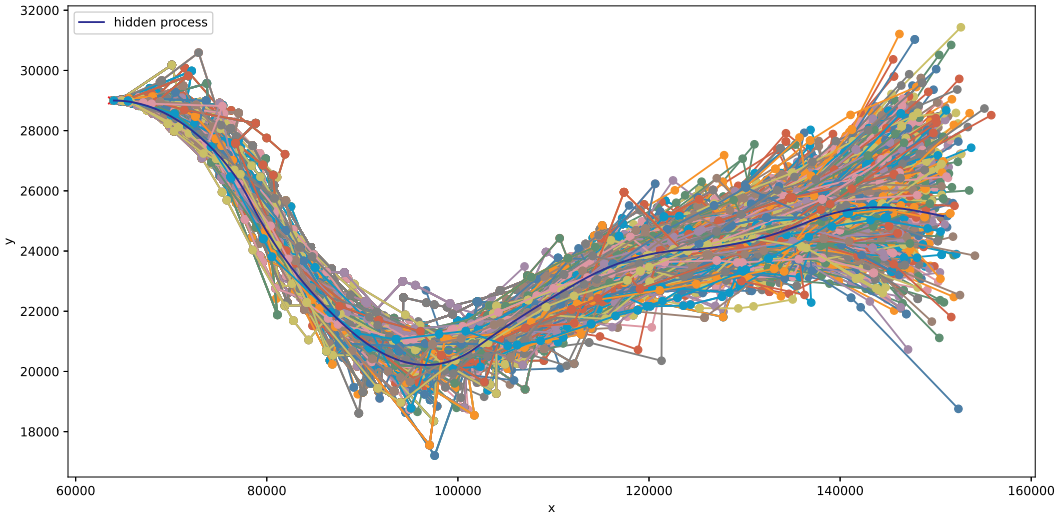

Figure 4.2: Distribution of particles around a trajectory using a PF with mixture kernels.

In this case the backward kernel is given by a Gaussian kernel of the same variance, restricted to $(\tau_{n-1,k_{n-1}-1}, t_{n-1}]$

$$L_{n-1,2}(x_n, d_x n-1) \propto 1_{(\tau_{n-1,k_{n-1}-1},t_n]}(\tau_{n-1,k_{n-1}})\mathcal{N}(\tau_{n,k_n}, \sigma^2_{adjust})d\tau_{n-1,k_{n-1}} \\ \times\delta_{k_n}(k_{n-1})\delta_{\tau_{n,1:k_n-1}(d\tau_{n-1,1:k_{n-1}-1})}$$

Moreover, we resample every time the ESS falls under 50%.

Choice of the Kernel

At each step, it is determined which of the two moves to use. To do this, we will use a distribution of Bernoulli whose parameter is the survivor function $S(\cdot,\cdot)$ (which we defined in section 2.2.2) in the interval $(\tau_{n,k_n-1}, t_n]$. In case of outcome 1, an adjustment move will be made, or otherwise, a birth move. This is equivalent in choosing $\alpha_{n,a}(x_{n-1})$ (from expression (4.2)) equal to

$$\alpha_{n,a}(x_{n-1}) \begin{cases} 0 & \text{if } t_n - \tau_{n-1,k_{n-1}-1} > t_{max} \\ S(t_n, \tau_{n-1,k_{n-1}}) & \text{o/w} \end{cases}$$

4.2 Poisson Tree Particle Filter

In this section we will discuss another approach to particle filtering for PDPs, namely the Poisson Tree Particle Filter (PTPF) [14].

As in the model in [89], the PTPF produces a branching process. What is new in this case is the introduction of a Poisson resampling scheme. Unlike the standard PF model presented in Algorithm 1, in which the "children" chose the parent particle from which to generate themselves, in this model, each particle gives rise to a random number of children. This characteristic allows us to easily generate an SMC algorithm for continuous time models. Furthermore, this algorithm can be used in the construction of a Gibbs particle sampler (as in [80]), which we will describe in chapter 5, and prove to be uniformly ergodic under the standard assumptions [79].

4.2.1 Model

The underlying model for the construction of PTPF is the same as that described in sections 2.2.2 and 2.2.3, with one slight modification.

The trajectory of the signal process $(\xi_t)_{t\geq 0}$, which we previously defined, in the interval between two jumps $[\tau_{k_n}, \tau_{k_{n+1}}[$ can be identified by its initial location as well as by its end location. While for the construction of the PF with mixture kernels we used the former method, in this case, we will use the latter. Moreover, we will refer to the model's propagation kernel as K.

4.2.2 Construction of the Algorithm

We will now introduce the main elements for the construction of the PTPF from [14].

The random structure $\mathbb{A} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$ originates from PTPF, in particular:

- $(\mathcal{V}, \mathcal{E})$ is a directed graph. If $(i, j) \in \mathcal{E}$, where $i, j \in \mathcal{V}$ we use the notations $i \to j$, $i = pa(j)$ and $j \in ch(i)$;
- $\mathbf{X} = \{X_i : i \in \mathcal{V}\}$ is a collection of random variables taking values in $\mathcal{X}$;
- $\mathbf{T} = \{T_i : i \in \mathcal{V}\}$ is a collection of random variables taking values ranging in $[t_{min}, \infty[$;
- $S \in \mathcal{V}$ is a (random) node in the graph that identifies a selected path.

Additionally, we take into account two collections of random variables:

- The intensity parameter $\Lambda = \{\Lambda_i : i \in \mathcal{V}\}$;
- The weights $W = \{W_i : i \in \mathcal{V}\}$.

Both being functions of $\mathbb{A}$ and of the fixed observation $\Upsilon = y$. Furthermore, we will employ a "propagation" transition kernel $R(X_i, T_i, \cdot, \cdot)$, such that the model's kernel K is absolutely continuous with respect to R.

Let i identify the particle born at time $T_{pa(i)}$ $(pa(i) \to i)$. We assign to this particle a jump time T_i, until which i evolves deterministically. The location right before the jump occurring at T_i, also called the end location, is denoted by $X_i = \xi(T_{i^-})$. If $T_i \geq t_{max}$ we i is

called a terminal node and $i \in \mathcal{V}_{end}$. To each particle i we assign an intensity parameter Λ_i, which carries information on the "history" of the particle. The way we compute it is described in section 4.2.3. Moreover, particle i is also assigned a weight $W_i = \frac{dK}{dR} l(X_i, T_i, X_j, T_j)$, where $\frac{dK}{dR}(x,t,x',t') = \frac{K(x,t,dx',dt')}{R(x,t,dx',dt')}$ is the Radon-Nikodym derivative (note that we choose the propagation kernel R such that the model kernel K is absolutely continuous with respect to it), while $l(X_i, T_i, X_j, T_j)$ is the likelihood in the interval $[T_{pa(i)}, T_i[$. At the jump time τ_i the particle i "gives birth"to a number $N_i \sim \text{Poiss}(\Lambda_i W_i)$ of children particles, which evolve independently according to the kernel $R(X_i, T_i, \cdot, \cdot)$. Note that it may happen that for a particle i $N_i = 0$. In that case, that branch of the tree "dies".

A particle is called **active** if it still has not been considered to give birth and if it is not a terminal node. When there are no more active particles left, the estimate $\hat{Z}$ of the norming constant z is computed as

$$\hat{Z} = \sum_{j:\tau_j \geq t_{max}} \frac{W_j}{C_{pa(j)}}$$

Finally we select a particle, S, existing at the moment t_{max} with probability $W_S / C_{pa(S)}$. The ancestry line of this particle identifies a unique sample path of the hidden process. In Algorithm 5 we give the pseudo-code for the PTPF algorithm from [14].

Algorithm 5 Poisson Tree Particle Filter

Initialization:
$\mathcal{V} := \mathcal{V}_{act} := \{0\}$; $\mathcal{E} := \emptyset$; $\mathcal{V}_{end} = \emptyset$; $X_0 := x_0$; $T_0 := t_0$;
$C_0 := \Lambda_0 := \lambda_0$; $W_0 = 1$
Main Loop:
while $\mathcal{V}_{act} \neq \emptyset$ **do**
  Choose $i \in \mathcal{V}_{act}$
  if $i \neq 0$ **then**
    Compute $\Lambda_i = \mathbb{L}(\mathcal{H}(T_i))$
    $C_i = C_{pa(i)} \Lambda_i$
  end if
  Sample $N_i \sim \text{Poiss}(\Lambda_i W_i)$
  if $N_i \geq 0$ **then**
    Create set $\text{ch}(i)$ of cardinality N_i
    $\mathcal{V} := \mathcal{V} \bigcup \text{ch}(i)$, $\mathcal{E} := \mathcal{E} \bigcup \{i \to j : j \in \text{ch}(i)\}$
    for all $j \in \text{ch}(i)$ **do**
      Sample $(X_j, T_j) \sim R(X_i, T_i \cdot, \cdot)$ (propagation step)
      Compute $W_j = \frac{dK}{dR} l(X_i, T_i, X_j, T_j)$ (weighting step)
      if $T_j \geq t_{max}$ **then**
        $\mathcal{V}_{end} = \mathcal{V}_{end} \bigcup \{j\}$
        $\mathcal{V}_{act} = \mathcal{V}_{act} \bigcup \{j\}$
      end if
    end for
  end if

$\mathcal{V}_{act} = \mathcal{V}_{act} \setminus \{i\}$
end while
if $\mathcal{V}_{end} \neq \emptyset$ **then**
$\hat{Z} := \sum_{i \in \mathcal{V}_{end}} \frac{W_i}{C_{pa(i)}}$
Select $s \in \mathcal{V}_{end}$ with probability $P(S = s) \propto \frac{W_s}{C_{pa(s)}}$
$\hat{Z} := 0$
end if
Output: $\hat{Z}, (X_{an(s)}, T_{an(s)})$

$\mathbb{L}(\mathcal{H}(T_i))$ is a function depending on the history of the particle i; [14] provides a way to compute it which we will describe in the following section. Figure 4.3 shows the set of trajectories identified by the nodes existing at a time $t \geq t_{max}$.

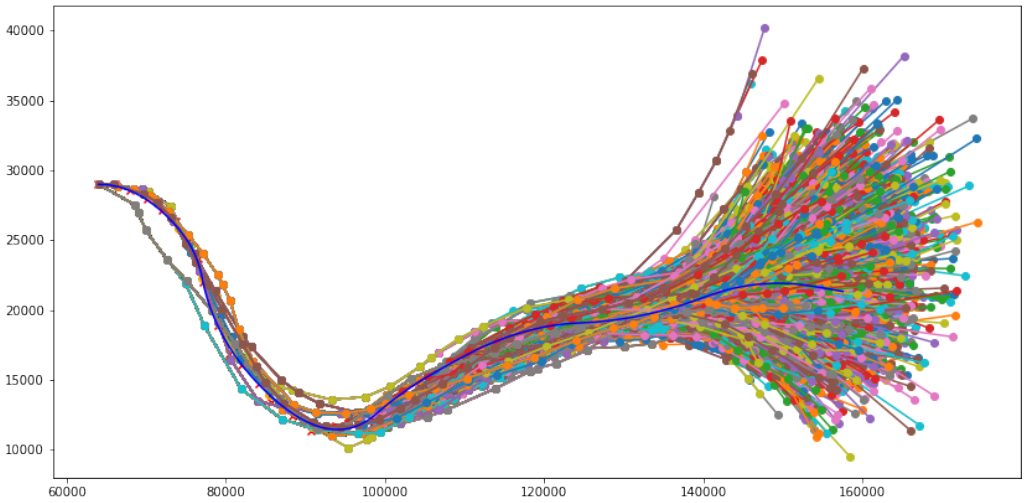

Figure 4.3: Distribution of particles around a trajectory using a PTPF.

4.2.3 Intensity parameter

In this section, we first provide a way for computing the intensity parameter Λ we mentioned in section 4.2.2, then we discuss why this approach works.

We partition the interval $[t_{min}, t_{max}]$ into smaller sub-intervals in the following way.

$$t_{min} = t^0_{syn} < t^1_{syn} < \cdots < t^r_{syn} < \cdots < t^q_{syn} = t_{max}$$

where t_{syn} stand for "synchronisation time".

Now we define

$$\mathcal{F}^r = \{i : t^r_{syn} \leq T_i \leq t^{r+1}_{syn}\}$$

if $i \in \mathcal{F}^r$ (i.e. i is in the r^{th} strip), we call the subset of $\mathcal{F}^r$ of the particle that propagate in r^{th} the strip but whose parents were in a previous strip $\mathcal{F}_0^r$:

$$\mathcal{F}_0^r = \{i : T_{pa(i)} < t_{syn}^r \leq T_i \leq t_{syn}^{r+1}\}$$

$\mathcal{F}^r \setminus \mathcal{F}_0^r$ is the set of nodes in $\mathcal{F}^r$ whose parents are also in $\mathcal{F}^r$. $\mathcal{G}^r$ is the subset of the particles that "skip" the r^{th} strip, i.e. they were born before t_{syn}^r and propagate after t_{syn}^{r+1}:

$$\mathcal{G} = \{i : T_{pa(i)} < t_{syn}^r, T_i \geq t_{syn}^{r+1}\}$$

Figure 4.4 illustrates different configurations. Let's call L_0^r the partial likelihood corresponding

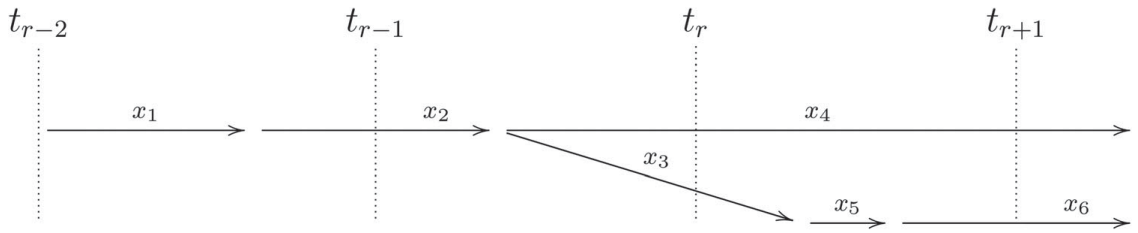

Figure 4.4: Example of Poisson tree. Edge $x_4 \in \mathcal{G}^r \mathcal{F}^{r+1}$, $x_3 \in \mathcal{F}_0^r$, while $x_5 \in \mathcal{F}^r \setminus \mathcal{F}_0^r$ ([14], Figure 1, page 7).

to a path in the previous strip. For every $i \in \mathcal{F}_0^r$,

$$L_0^i = l(\xi_{i[t_{syn}^{r-1}, t_{syn}^r[})$$

and define

$$L_0^r = \sum_{i \in \mathcal{F}_0^r} L_0^i$$

Now that we have defined the essential elements, we will propose a method for computing the intensity parameter as in [14]. Assuming that $\mathcal{F}_0^r \neq 0$, we set

$$\Lambda_i = \begin{cases} \frac{1}{W_i} \frac{L_i^r}{L_0^r} b(\lambda_0 - |\mathcal{G}^r|) & \text{for } i \in \mathcal{F}_0^r \\ \frac{1}{W_i} & \text{for } i \in \mathcal{F}^r \setminus \mathcal{F}_0^r \end{cases} \tag{4.6}$$

where b is a non-decreasing function $b :]-\infty, \infty[\to [0, \infty[$. And λ_0 is the expected initial number of particles.

The idea behind this is that we want to keep the number of particles generated at each step constant on average, so close to λ_0. We have that, under (4.6), the expected number of children of particles in $\mathcal{F}_0^r$

$$\sum_{i \in \mathcal{F}_0^r} \Lambda_i W_i = b(\lambda_0 - |\mathcal{G}^r|)$$

while the expected number of children for the nodes in $\mathcal{F}^r \setminus \mathcal{F}_0^r$ is 1. Particles corresponding to $\mathcal{G}^r$ pass through the strip $[t_{syn}^r, t_{syn}^{r+1}[$ unchanged. The expected number of particles that exist just before t_{syn}^{r+1} is obtained by combining these results

$$b(\lambda_0 - |\mathcal{G}^r|) + |\mathcal{G}^r|$$

We choose b to be $b(l) = \max(l, b_0)$, where b_0 is a small given constant. Note that if we choose $b(l) = \max(l, 0)$, then the expected number of particles immediately before t_{syn}^{r+1} would be equal to $\max(\lambda_0, |\mathcal{G}^r|)$. Hence if $\lambda_0 \leq |\mathcal{G}^r|$, the particles in $\mathcal{F}^r$ would have zero chance to propagate.

It is important to note that we need to "count" how many T_i fall within a certain strip, but we do not need to sort them, which is very important for an efficient implementation of the algorithm.

Chapter 5

Parameters and State Estimations

As mentioned in section 2, MCMC methods aim to construct a Markov chain whose stationary distribution is equal to the target distribution. In the case of the model considered in the previous chapters, our target distribution is $p_\theta(x_{1:T}|y_{1:T})$, where θ is known. If we additionally want to make inferences on the parameters of the model (θ is no longer known), the target distribution becomes $p(\theta, x_{1:T}|y_{1:T})$.

In [21], a method is proposed for performing states and parameters estimation sequentially. This means that while PMCMC algorithms provide samples from $p(\theta, x_{1:T}|y_{1:T})$, the new algorithm, namely the SMC^2 algorithm, provides samples from

$$p(\theta, x_{1:t}|y_{1:t}) \qquad \forall t \in [1, T]$$

In Section 5.2 we will try to adapt this algorithm to a piecewise deterministic model.

5.1 Particle MCMC based on PTPF

As we mentioned in section 3.4, we are going to describe two MCMC methods for performing parameters and states estimations based on PTPFs [14].

Before giving a pseudo-code version of the two algorithms, we define the extended probability distributions of the PTPF and the conditional PTPF. These will lead to some interesting results in terms of the convergence of the two PMCMC algorithms [14].

5.1.1 Extended Probability Distribution and Conditional PTPF

We define two types of extended probabilities:

- The **extended proposal**, the joint probability distribution of $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$, denoted by $\psi(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$
- The **extended target**, the joint probability distribution of $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$ focused on the trees with $\mathcal{V}_{end} \neq \emptyset$, i.e. marginalised to the target. The extended target is denoted by $\phi(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$

In order to provide a full expression for these two probability distributions, we first need to introduce the concept of equivalence class.

When we consider a tree, the way we label nodes and edges is not relevant to the correct functioning of the algorithm. Two trees are said to be equivalent if there is a one-to-one correspondence between the node sets (such that the edges are preserved). Thus, two structures $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$ and $(\mathcal{V}', \mathcal{E}', \mathbf{X}', \mathbf{T}', S')$ are equivalent if they differ only in the way they are labelled. If we consider the propagation step of a node i, we know that the probability that it will give birth to n_i children is

$$e^{-\lambda_i w_i} \frac{(\lambda_i w_i)^{n_i}}{n_i!} \tag{5.1}$$

Each of these children is assigned a pair (X_i, T_i). There are $n_i!$ ways in which these can be assigned. So by multiplying (5.1) by $n_i!$ we obtain the probability distribution of the equivalence class, which we will denote by $[\mathbb{A}] = [(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)]$.

We can now proceed to give a precise form (from [14]) to the extended probability distributions introduced above

$$\psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}) = \prod_{i \in \mathcal{V} \setminus \mathcal{V}_{end}} e^{-\lambda_i w_i} (\lambda_i w_i)^{|chi(i)|} \prod_{j \in chi(i)} R(x_i, t_i, dx_j, dt_j) \tag{5.2}$$

$$\psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s) = \psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}) \frac{w_s}{c_{pa(s)} \hat{z}} \tag{5.3}$$

where

$$\hat{z} = \sum_{i \in \mathcal{V}_{end}} \frac{w_i}{c_{pa(i)}}$$

Note that 5.2 is the marginal distribution of all the variables except S, and 5.3 is the conditional distribution of S given the rest (with abuse of notation, we call both of them $\psi(\cdot)$). We write the extended target as

$$\begin{aligned} \phi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s) &= \psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s) \frac{\hat{z}}{z} \\ &= \pi(dx_{an(s)}, dt_{an(s)}) \cdot \psi_{cond}((\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s | x_{an(s)}, t_{an(s)}) \end{aligned} \tag{5.4}$$

where ψ_{cond} is given by

$$\begin{aligned} \psi_{cond}((\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s | x_{an(s)}, t_{an(s)}) = & \prod_{i \in \mathcal{V} \setminus \mathcal{V}_{end} \setminus an(s)} e^{-\lambda_i w_i} (\lambda_i w_i)^{|chi(i)|} \prod_{j \in chi(i)} R(x_i, t_i, dx_j, dt_j) \\ & \times \prod_{i \in an(s) \setminus \{s\}} e^{-\lambda_i w_i} (\lambda_i w_i)^{|chi(i)|} \prod_{j \in chi(i) \setminus an(s)} R(x_i, t_i, dx_j, dt_j) \end{aligned}$$

(see [14] for further details).

Now, we proceed with giving a definition for the conditional PTPF from [14]. This algorithm will be used for the construction of the Particle Gibbs sampler. It is almost the same

as the standard version of the PTPF, the only difference being that the node S (and thus the path identified by it) is given as input.

Algorithm 6 Conditional PTPF

Input: $(\tilde{X}_{1:M}, \tilde{T}_{1:M})$ (conditioning path)
Initialization:
$\mathcal{V} := \mathcal{V}_{act} := \{0\}$; $\mathcal{E} := \emptyset$; $\mathcal{V}_{end} = \emptyset$; $X_0 := x_0$; $T_0 := t_0$;
$C_0 := \Lambda_0 := \lambda_0$; $W_0 = 1$
1: **for** $k := 1$ to M **do**
2: $\mathcal{V} := \mathcal{V} \bigcup \{k\}$, $\mathcal{E} := \mathcal{E} \bigcup \{k-1 \to k\}$
3: $(X_k, T_k) := (\tilde{X}_k, \tilde{T}_k)$
4: **end for**
5: $S := M$ (singled out vertex S is set to be equal to the endpoint M of the fixed path $(\tilde{X}_{1:M}, \tilde{T}_{1:M})$)
6: $\mathcal{V}_{act} = \mathcal{V} \setminus M$, $\mathcal{V}_{end} := \{M\}$
7: **Main loop**: same as the main loop in standard PTPF
Output: $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$

Note that the output of the conditional PTPF is a tree with distribution ψ_{cond}.

5.1.2 Poisson Tree Gibbs sampler (PTGS) and Poisson Tree Metropolis Hastings (PTMH)

We can now finally define the MCMC algorithms based on the PTPF. We recall that we aim to simulate a Markov chain $\Xi^0, \Xi^1, \ldots, \Xi^n, \ldots$, (with $\Xi_n = \{\Xi^n(t) | t_{min} \leq t \leq t_{max}\}$) whose stationary distribution is the posterior distribution of the hidden process $\Xi = x_{1:T}$ given its observations $y_{1:T}$, i.e. $p(x_{1:T}|y_{1:T})$.

Firstly, we will introduce the PTGS algorithm, namely the version using the PTPF of the more general PGS introduced in Section 3.4. In Algorithm 7, we provide a pseudo code for this algorithm. Note that $(X_{1:M}, T_{1:n})$ indicates the space-time skeleton, which uniquely identifies the piecewise deterministic path.

Algorithm 7 One step of PTGS

Input: $\hat{Z}$, $(\tilde{X}_{1:M}, \tilde{T}_{1:M})$ (Output of previous step)
1: Run PTPF and obtain $\hat{Z}^*$, $(X^*_{1:M}, T^*_{1:M})$
2: Sample $U \sim$U(0,1)
3: **if** $U \leq \frac{\hat{Z}^*}{\hat{Z}}$ **then**
4: $(X'_{1:M'}, T'_{1:M'}) := (X^*_{1:M'}, T^*_{1:M'})$ $\hat{Z}' := \hat{Z}^*$ (Accept proposal)
5: **else**
6: $(X'_{1:M'}, T'_{1:M'}) := (\tilde{X}_{1:M'}, \tilde{T}_{1:M'})$ $\hat{Z}' := \hat{Z}$ (Reject proposal)
7: **end if**
Output: $(X'_{1:M'}, T'_{1:M'})$

On the other hand, we call PTMH the Poisson resampling version of the PIMH (see section 3.4) (8).

Algorithm 8 One step of PTMH

Input: $\hat{Z}$, $(\tilde{X}_{1:M}, \tilde{T}_{1:M})$ (Output of previous step)
1: Run cPTPF and obtain $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$
2: Forget S and sample a new S': select $S' \in \mathcal{V}_{end}$ with probability $P(S' = s') \propto W_{s'}/C_{pa(s')}$
Output: $(X'_{1:M'}, T'_{1:M'}) := (X_{an(S')}, T_{an(S')})$

Both the PTGS and the PTMH produce Markov chains on the space of trees: $\mathbb{T}_0, \mathbb{T}_1, \ldots, \mathbb{T}_n, \ldots$.

5.1.3 Some Results

Now that we have defined the extended probability distributions and the PMCMC algorithms, we present some results from [14].

Proposition 5.1.1. Let f be a non-negative function on the space of skeletons $(x_{1:m}, t_{1:m})$. If the structure $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T})$ is produced by PTPF then the following estimator of $z\pi(f)$ is unbiased

$$\widehat{z\pi(f)} = \begin{cases} \sum_{i \in \mathcal{V}_{end}} \frac{W_i}{C_{pa(i)}} f(X_{an(i)}, T_{an(i)}) & \text{if } \mathcal{V}_{end} \neq \emptyset \\ 0 & \text{if } V_{end} = \emptyset \end{cases}$$

In particular, $\hat{Z}$ is an unbiased estimator of z.

Theorem 5.1.1. Markov chains generated by algorithms PTMH and PTGS have the equilibrium distribution equal to the target $\pi = \pi_{post}$ given by (5.4).

The proof of Proposition 5.1.1 and Theorem 5.1.1 are also provided by [14].

5.1.4 Gibbs Sampler based on PF with Mixture Kernels

To describe this algorithm, we will refer to algorithm 3 presented in section 3.4. The only element we are missing in order to construct such an algorithm is the conditional version of the PF with mixture kernels. However, in the latter, the trajectories are built independently of each other, so it is sufficient to add the fixed trajectory to the set of trajectories produced by the algorithm.

Thus, with this modification of PF with mixture kernels, we can use algorithm 3 to obtain an approximation of the target distribution. In the following chapters we will refer to this algorithm as Particle Filter Mixture Kernels Gibbs sampler (PF MK GS).

5.2 Sequential Parameters and State Estimations: the SMC^2 Algorithm

As we have already mentioned, in this section, we will introduce an algorithm for inference on parameters and states in a sequential manner. This algorithm was introduced by [21], and is proposed as a union of a PF (see 1) and the Iterated Batch Importance Sampling (IBIS) algorithm [19]. We proceed to give a background on SMC^2, its objectives and a pseudo-code version of the algorithm.

5.2.1 Iterated Batch Importance Sampling Algorithm

We have already seen in the previous sections how PFs work. The algorithm introduced by [19] aims at exploring consistently a sequence of distributions of interest, in our case the posterior distribution of the parameter θ, $p(\theta|y_{1:y})$.

The algorithm is the following

Algorithm 9 Iterated Batch Importance Sampling [19]

Sample θ^m from $p(\theta)$ and set $\omega^m \leftarrow 1$. For times $t = 1 \dots T$:

1: Calculate the incremental weights and their weighted average

$$\psi_t(\theta^m) = p(y_1|y_{1:t-1}, \theta^m), \quad L_t = \frac{1}{\sum_{m=1}^{N_\theta} w^m} \sum_{m=1}^{N_\theta} w^m \psi_t(\theta^m)$$

where $p(y_1|y_{1:0}, \theta^m) = p(y_1|\theta^m)$ for t=1.

2: Update of the importance weights

$$w^m \leftarrow w^m \psi_t(\theta^m)$$

3: If a chosen degeneracy criterion is fulfilled, sample θ^m independently from the mixture distribution

$$\frac{1}{\sum_{m=1}^{N_\theta} \omega^m} \sum_{m=1}^{N^\theta} \omega^m G_t(\theta^m, \cdot)$$

replace the current particle system with the set of new unweighted particles

$$(\theta^m, \omega^m) \leftarrow (\tilde{\theta}^m, 1)$$

Where

$$\frac{\sum_{m=1}^{N_\theta} \omega^m \psi(\theta^m)}{\sum_{m=1}^{N_\theta} \omega^m}$$

when $N_\theta \to \infty$, for any integrable ψ, is a consistent[1] normal estimator of

$$\mathbb{E}[\psi(\theta)|y_{1:t}] = \int \psi(\theta) p(\theta|y_{1:t}) d\theta$$

[1] In statistics, a consistent estimator or asymptotically consistent estimator is an estimator having the property that as the number of data points used increases indefinitely, the resulting sequence of estimates converges in probability [88]

A proof for this is provided by [20]. Moreover, L_t is itself a consistent, asymptotically normal estimator of the likelihood.

Finally, the algorithm includes a resampling and a mutation step. The resampling is performed by means of a multinomial distribution: particles are selected in proportion to their weight (see Figure 4.1). As a degeneracy criterion the ESS (defined as in 4.5) can be used. In the mutation phase, new particles are proposed according to a Gaussian random walk

$$\tilde{\theta}^m | \theta^m \sim N(\theta^m, c\hat{\Sigma}) \tag{5.5}$$

where

$$\hat{\Sigma} = \frac{1}{\sum_{m=1}^{N_\theta} \omega^m} \sum_{m=1}^{N_\theta} \omega^m (\theta^m - \hat{\mu})(\theta^m - \hat{\mu})^T$$

and

$$\hat{\mu} = \frac{1}{\sum_{m=1}^{N_\theta} \omega^m} \sum_{m=1}^{N_\theta} \omega^m \theta^m$$

5.2.2 The SMC^2 Algorithm

We now have all the elements to define the SMC^2 algorithm (Algorithm 10).

Algorithm 10 SMC^2 [19]

Sample θ^m from $p(\theta)$ and set $\omega^m \leftarrow 1$. For times $t = 1 \dots T$:

1: For each particle θ^m, perform iteration t of the PF as described in1.

- If t = 1, sample independently $x^{1:N_x,m_1}$ from ψ_{1,θ^m}, then compute

$$\hat{p}(y_1|\theta^m) = \frac{1}{N_x} \sum_{n=1}^{N_x} \omega_{1,\theta}(x_1^{n,m})$$

- If $t \geq 1$, sample $\left(x_{1:t}^{1:N_x,m}, a_{1:t-1}^{1:N_x,m}\right)$ conditional on $\left(x_{1:t-1}^{1:N_x,m}, a_{1:t-2}^{1:N_x,m}\right)$ from ψ_{t,θ^m}, then compute

$$\hat{p}(y_t|y_{1:t-1}, \theta^m) = \frac{1}{N_x} \sum_{n=1}^{N_x} \omega_{t,\theta} \left(x_{t-1}^{a_{t-1}^{n,m},m}, x_t^{n,m}\right)$$

2: Update of the importance weights

$$w^m \leftarrow w^m \hat{p}(y_t|y_{1:t-1})$$

3: If a chosen degeneracy criterion is fulfilled, sample $\left(\tilde{\theta}^m, \tilde{x}_{1:t}^{1:N_x,m}, \tilde{a}_{1:t-1}^{1:N_x,m}\right)$ independently from the mixture distribution

$$\frac{1}{\sum_{m=1}^{N_\theta} \omega^m} \sum_{m=1}^{N^\theta} \omega^m K_t \left(\left(\theta^m, x_{1:t}^{1:N_x,m}, a_{1:t-1}^{1:N_x,m}\right), \cdot \right)$$

replace the current particle system with the set of new unweighted particles

$$\left(\theta^m, x_{1:t}^{1:N_x,m}, a_{1:t-1}^{1:N_x,m}, \omega^m\right) \leftarrow \left(\tilde{\theta}^m, \tilde{x}_{1:t}^{1:N_x,m}, \tilde{a}_{1:t-1}^{1:N_x,m}, 1\right)$$

[21] also provides a formal justification of the SMC^2 algorithm.

Step 3 in the algorithm involves the use of a PMCMC kernel K_t, which we describe in the following section.

The MCMC Rejuvenation Step

This step corresponds to a scheme defined as

Algorithm 11 rejuvenation step

1: Sample $\tilde{\theta}$ from the proposal kernel $T\left(\theta, d\tilde{\theta}\right)$
2: Run a PF with the new to obtain the new paths $\left(\tilde{x}_{1:t}^{1:N_x}, \tilde{a}_{1:t-1}^{1:N_x}\right)$ and compute $\hat{Z}_t\left(\tilde{\theta}, \tilde{x}_{1:t}^{1:N_x}, \tilde{a}_{1:t-1}^{1:N_x}\right)$
3: Compute the acceptance probability

$$p_a = \frac{p(\tilde{\theta})\hat{Z}_t\left(\tilde{\theta}, \tilde{x}_{1:t}^{1:N_x}, \tilde{a}_{1:t-1}^{1:N_x}\right) T\left(\tilde{\theta}, \theta\right)}{p(\theta)\hat{Z}_t\left(\theta, x_{1:t}^{1:N_x}, a_{1:t-1}^{1:N_x}\right) T\left(\theta, \tilde{\theta}\right)}$$

and accept the move with probability $\min(p_a, 1)$.

with proposal distribution

$$q_\theta\left(\tilde{\theta}, \tilde{x}_{1:t}^{1:N_x}, \tilde{a}_{1:t-1}^{1:N_x}\right) = T\left(\theta, \tilde{\theta}\right)\psi_{t,\tilde{\theta}}\left(\tilde{x}_{1:t}^{1:N_x}, \tilde{a}_{1:t-1}^{1:N_x}\right)$$

which admits as invariant distribution the extended distribution $\pi_t(\theta, x_{1:t}^{1:N_x}, a_{1:t}^{1:N_x})$. $\hat{Z}_t$ is an unbiased estimator of $p(y_{1:t}|\theta)$ (proved in [37], Proposition 7.4.1) and takes the form

$$\hat{Z}_t\left(\theta, x_{1:t}^{1:N_x}, a_{1:t-1}^{1:N_x}\right) = \left(\frac{1}{N_x}\right)^t \left(\sum_{n=1}^{N_x} \omega_{1,\theta}(x_1^n)\right) \prod_{s=2}^{t} \left(\sum_{n=1}^{N_x} \omega_{s,\theta}(x_{s-1}^{a_{s-1}^n} x_s^n)\right)$$

The proposal kernel T is the Gaussian kernel (5.5).

5.2.3 Adaptation of SMC^2 to PDPs

We must now readjust the algorithm presented to the context of the piecewise deterministic model presented in section 2.2.2.

What changes from its standard counterpart is the PF component. This will therefore be replaced by one of the two SMC algorithms presented in chapter 4, namely the PF with mixture kernels and the PTPF

The Model's Parameters

As we have mentioned, the model whose parameters we wish to infer is that described in section 2.2.2, i.e. an object moving in a plane and whose acceleration component is subject to random changes.

In the model's assumptions, the jump times, i.e. the time instants at which changes in acceleration occur, are random, and the inter-arrival times are distributed as a Gamma with given scale and shape parameters. However, the model only observes the position of the object at regular time intervals. In a real scenario, hence with data not generated by us, we do not know the actual value of the scale and shape parameters. Therefore, maintaining the assumption that jump times are Gamma distributed, we will use the SMC^2 algorithm to perform Bayesian inference on them.

SMC^2 with Mixture Kernels PF

This type of algorithm, as it is defined, already discretises the piecewise deterministic process. In fact, a move (birth or adjust step) is chosen at each step for each of the particles, and it is, therefore, possible to have a complete picture of the particles in the system at a given time step. This allows the ESS to be calculated and thus resampling to take place. In this last step, it is, therefore, sufficient to replace the resampling from the multinomial distribution of the standard version with a rejuvenation step (as described in 5.2.2).

SMC^2 with PTPF

Adapting the SMC^2 algorithm to PTPF is less straightforward, however, as resampling in the latter is incorporated into every particle propagation move. Furthermore, jump times are sampled freely by using a distribution and are not forced into a precise interval.

In order to be able to determine how many particles survive at a given time t, we will run the algorithm setting $t_{max} = t$. At this point all particles in $\mathcal{V}_{end}$, i.e. active particles for which he jump time $T_i \geq t_{max}$ will be used for calculating the ESS. The rejuvenation step is obtained by running another particle filter with the new parameters for scale and shape of the gamma distribution and with $t_{max} = t$. To calculate the acceptance probability, we must reformulate how we calculate $\hat{Z}_t$. We will use the partitions introduced by [14] for particle synchronisation in calculating the intensity parameter Λ. $\hat{Z}$ will therefore be computed as

$$\hat{Z}_t\left(\theta, x_{1:t}\right) = \left(\frac{1}{N_{syn}^1}\right)\left(\frac{1}{N_{syn}^2}\right)\cdots\left(\frac{1}{N_{syn}^q}\right)\prod_{s=1}^{q}\left(\sum_{n=1}^{N_{syn}^s}\omega_{s,\theta}(x_s^n)\right)$$

where $1 = t_{syn}^1 \leq t_{syn}^2 \leq \cdots \leq t_{syn}^q = t$ are the partitions mentioned above, and N_{syn}^k is the number of particles that falls in the k^{th} partition.

Chapter 6

Experiments

6.1 Results and Comparison with simulated data

6.1.1 Particle filters

In this section, we will analyze the results obtained with the two different models by comparing them. In particular, we will consider the average trajectory produced by the model versus the actual trajectory (and thus in model bias), the number of unique particles employed at each step, and the distribution around a single point.

Initial Settings

The model is the same we presented in section 2.2.3. First, we generated the data on which to build the algorithms. The process was observed for a total duration of 200s, and we generated (independent) Gaussian observations, with standard deviation of 200m every 5s ($\Delta_t = t^{n+1} - t^n = 5$s). The acceleration components were independently sampled from an isotropic Gaussian distribution with mean zero and standard deviation $10m/s^2$. The inter-jump times were sampled from a Gamma distribution with 10 as shape and 2.5 scale parameters as in [101]. Under these assumptions, we generated the data on which used for implementing the models (Figure 6.1). Note that for this model (which is linear and conditionally Gaussian), it is analytically impossible to integrate out the parameters $\theta_{1:k_n}$, and only jump times need to be sampled. This can be seen as the equivalent of the Rao-Blackwellised SMC for discrete-time filtering [52, 20].

For the PDP PF with mixture kernels we used 1000 initial number of particles, shape and scale parameters for the Gamma distribution respectively 5 and 1.25 (half of the parameters used to generate the data) and $\sigma_{adjust} = \frac{\Delta_t}{1000} = 5.0 \cdot 10^{-3}$ (weight degeneracy can be avoided by choosing σ_{adjust} orders of magnitudes smaller than the inter-observation time [101]).

For the PTPF we used 1000 initial particles, $\lambda_0 = 50$, $b_0 = 1$, shape and scale parameters for the Gamma distribution respectively 5 and 1.25 (as for the PDP PF with mixture kernels). Moreover we set $\Delta_{t_{syn}} = t^{r+1}_{syn} - t^r_{syn} = \Delta_t \cdot 4$.

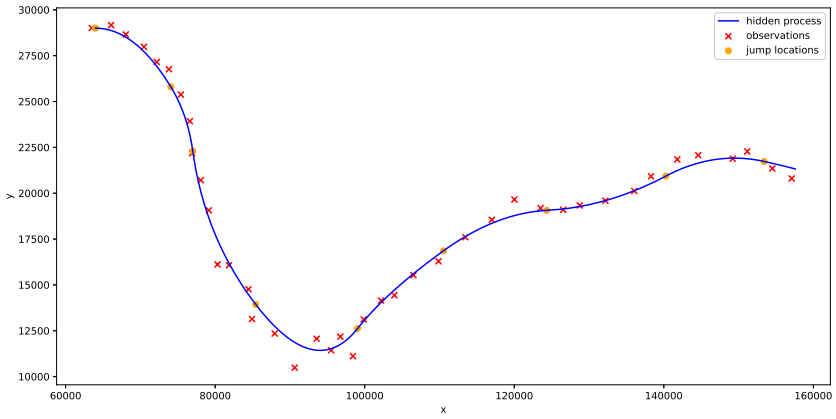

Figure 6.1: Generated data: hidden trajectory (blue), observations (red crosses) and locations of jumps (orange dots).

Average Trajectory and Bias

In this section, we will compare how the weighted average of the trajectories produced by the two algorithms approximates the hidden process for $N = 1000$. Then we calculate the RMSE between the hidden trajectory and the mean trajectory of the two algorithms with particle numbers equal to $N = 100, 1000, 10000$.

Firstly, we note from Figure 6.2 that both algorithms approximate the hidden trajectory relatively well. However, the PF with mixture kernels appears to be more sensitive to deviations on observations. We repeat the experiment by inserting a few observations that are much

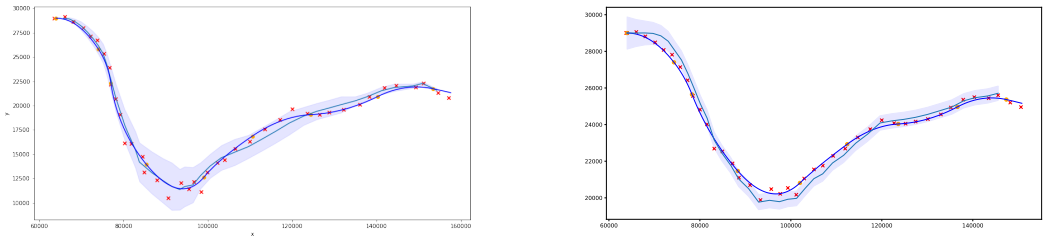

Figure 6.2: Estimated trajectories (with 95% CI) generated by PTPF (left) and PF with mixture kernels (right).

noisier than the others and observe how the two algorithms approximate the trajectory. The modified observation were the $5^{th}, 15^{th}, 25^{th}$ and 35^{th}. We can see from Figure 6.3 that in fact the PF with mixture kernel tends to change its average trajectory more if noisier observations are present, compared to the PTPF, which is less affected. Moreover, it should be noted that deviations in the PF with mixture kernels actually only occur if for most particles a birth move has taken place in that range. In the case of a majority of adjustment moves, in fact, the deviation is of less interest, as is the case for the observation in position 5 and 15.

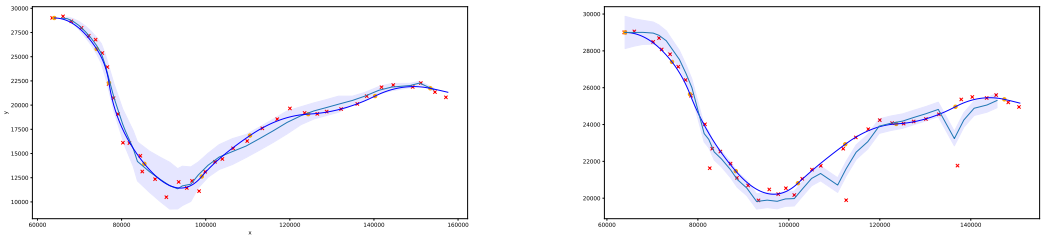

Figure 6.3: Estimated trajectories (with 95% CI) generated by PTPF (left) and PF with mixture kernels (right) with outlier.

We now quantify the error between the trajectory used and the ground truth for different initial particle numbers using RMSE. To obtain the results we will present, we repeated the experiment 10 times for each number of initial particles and averaged the output results.

We begin by reporting the results of the PF with mixture kernel. We can see from Table 6.1 and Figure 6.4 that the error drops considerably from using 100 to 1000 initial particles. However, the difference is not considerable when comparing the results obtained using 1000 and 10000 initial particles. However, the time taken to complete a cycle of iterations increases linearly with the number of particles (Table 6.2).

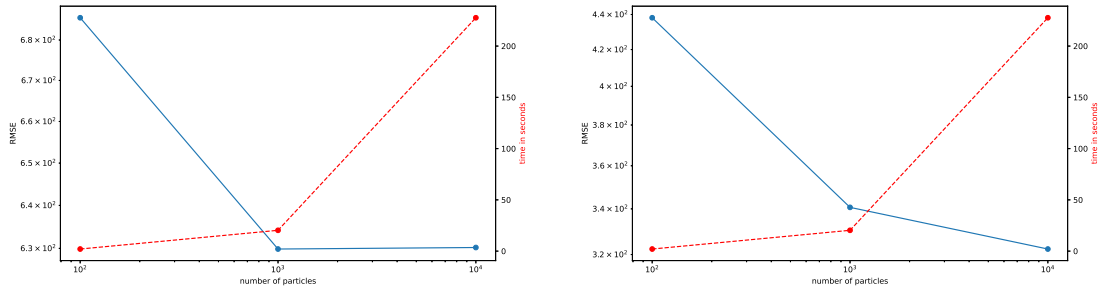

Figure 6.4: RMSE for different numbers of initial particles (100, 1000, 10000) on the two components using PF with mixture kernels.

	x			y		
Number of initial particles	100	1000	10000	100	1000	10000
RMSE	685.57	629.85	630.20	438.10	340.67	322.35

Table 6.1: RMSE for different numbers of initial particles (100, 1000, 10000) on the two components using the PF with mixture kernels.

If we repeat the same experiment this time with $N = 10000, 20000$ and 30000 initial particles (Table 6.3), we notice that there is in fact no substantial change in the RMSE. We can conclude that using a much larger number of initial particles than a thousand (in PF with

	time		
Number of initial particles	100	1000	10000
seconds	2.05	20.31	227.63

Table 6.2: Time employed for different numbers of initial particles ((100, 1000, 10000)) on the two components using PF with mixture kernels.

mixture kernels) does not lead to significantly better results in terms of algorithm bias but nevertheless takes much longer (Table 6.4).

Now, we do the same analysis for the PTPF. From Figure 6.5 and Tables 6.5 and 6.6

	x			y		
Number of initial particles	10000	20000	30000	10000	20000	30000
RMSE	628.34	627.89	626.06	320.79	325.58	324.31

Table 6.3: RMSE for different numbers of initial particles (10000, 20000, 30000) on the two components using the PF with mixture kernels.

	time		
Number of initial particles	10000	20000	30000
seconds	191.05	632.50	3679.06

Table 6.4: Time employed for different numbers of initial particles (10000, 20000, 30000) using the PF with mixture kernels.

we can see how time also grows linearly with the number of particles and the error decreases steadily, in particular on the y axis.

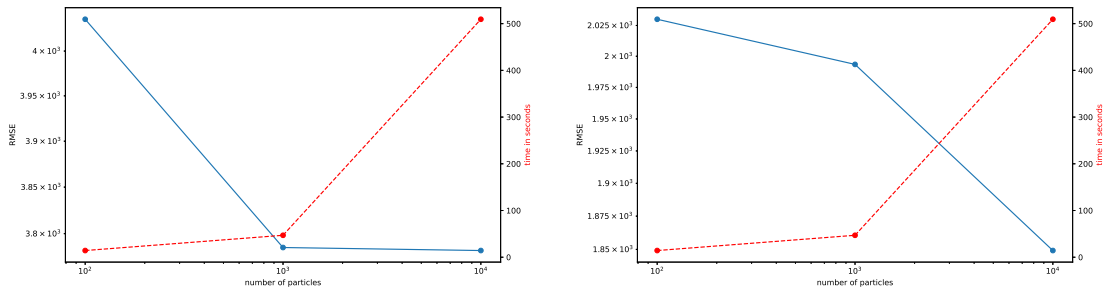

Figure 6.5: RMSE for different numbers of initial particles (100, 1000, 10000) on the two components using the PTPF.

	x			y		
Number of initial particles	100	1000	10000	100	1000	10000
RMSE	4035.91	3785.23	3782.02	2030.08	1993.43	1849.14

Table 6.5: RMSE for different numbers of initial particles (100, 1000, 10000) on the two components using PTPF.

	time		
Number of initial particles	100	1000	10000
seconds	14.31	46.95	509.40

Table 6.6: Time employed for different numbers of initial particles (100, 1000, 10000) using PTPF.

We run again the algorithm with $N = 20000$ particles, obtaining the results shown in Table 6.7. We can see that again the bias of the algorithm do not change significantly.

We can conclude that, in terms of bias, the performance of both algorithms do not improve notably after with more that 10000 initial particles.

	x	y
RMSE	3769.43	1849.14

Table 6.7: RMSE and computational time using $N = 20000$ initial particles for PTPF.

Number of Unique Particles

Figure 6.6 shows the number of unique particles that survive each resampling step and the ESS for the PF with mixture kernels. During the resampling step usually about half of the particles is selected to "survive" to the next step. These numbers, however, do not give us an accurate indication of how many particles actually survive all the resempling steps, i.e., looking at the final configuration, how many unique particles there are for each step.

Figure 6.7 depicts the number of distinct particles at each iteration of the algorithm. In order to create these plots, at the algorithm's last iteration, the particle histories were saved, and the number of distinct particles at each observation time was then counted and plotted versus t.

The variety of particle positions by themselves cannot fully inform us of the algorithm's effectiveness. Despite the possibility of a large number of distinct particles, substantial variance in the importance weights would lower the quality of the particle approximation. The variance of estimates obtained from the particle set is increased through resampling, hence we plot the same number of unique particles after resampling. These numbers exhibit the degeneracy of the significance weights and the particle positions to varying degrees.

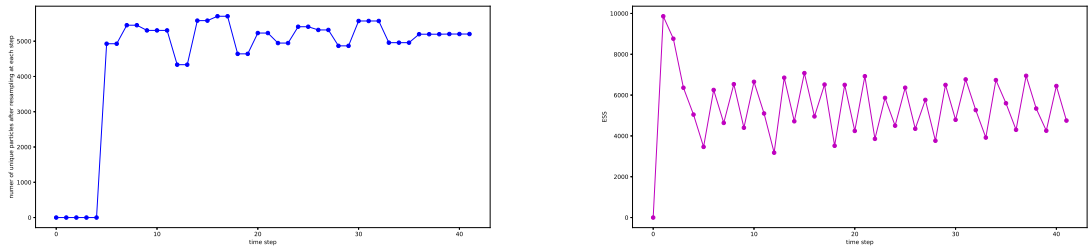

Figure 6.6: Number of unique particle resampled at each iteration (above) and ESS at each iteration (below) fo PF with mixture kernels.

Variations in particle locations are present at earlier stages in the PF with mixture kernels, moreover, in recent history all particles are distinct. Nonetheless, some particles share the same state, hence the number of distinct particles is less than their total number. This is an example of the phenomenon described in section 4.1.2. Because the importance weights are degenerate, resampling reduces diversity. Still, the PF with mixture kernels shows more diversity that the PTPF (so the one with Poisson resampling) , which present heterogeneity in the particles only in the very recent history.

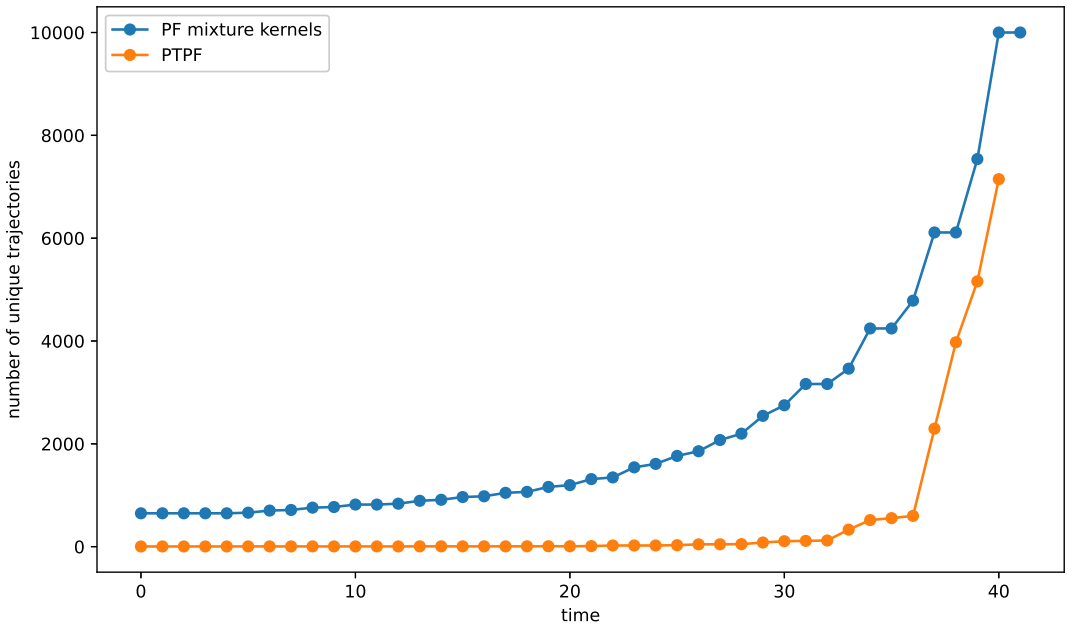

Figure 6.7: Number of unique particles at the last iteration for PF with mixture kernels and PTPF.

Distribution of the Particles around a single point

We now analyse how the particles are distributed around a single observation for the two different algorithms. Note that these plots refer to the final configuration after resampling.

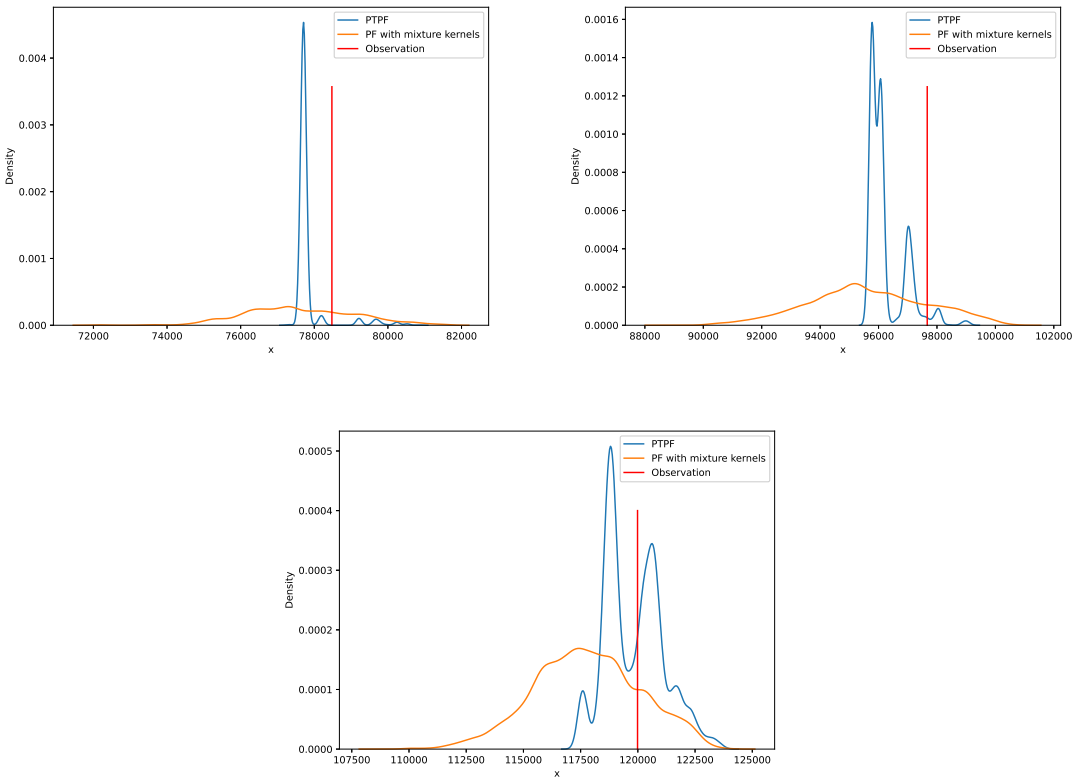

Figure 6.8: Distribution of particles locations around $10^{th}, 20^{th}$ and 30^{th} observations on x axis for n=1000 initial number of particles.

Figures 6.8 and 6.9 shows the spatial distributions of the particles at three different observation times for the two algorithms considered and for x and y respectively.

We note that the particles produced by the PF with mixture kernels have a much broader distribution around the location of the observation, while the particle distribution of the PTPF occurs in peaks in the vicinity of the observation. This phenomenon can be related to what we saw in the previous section. In fact, Figure 6.7 shows how the PF with mixture kernels exhibits diversity in particles already from the first steps, while the PTPF shows diversity only in the recent past from the last iteration. This leads the particles of the PTPF to form peaks with small variance in contrast to the distribution of PF with mixture kernels which, as we have already observed, has a larger variance. This aspect plays in favour of the PF with mixture kernel, which manages to avoid the degenerate case more easily in which we have a few very heavy particles.

It should also be noted that the variance in the distribution of particles around the ob-

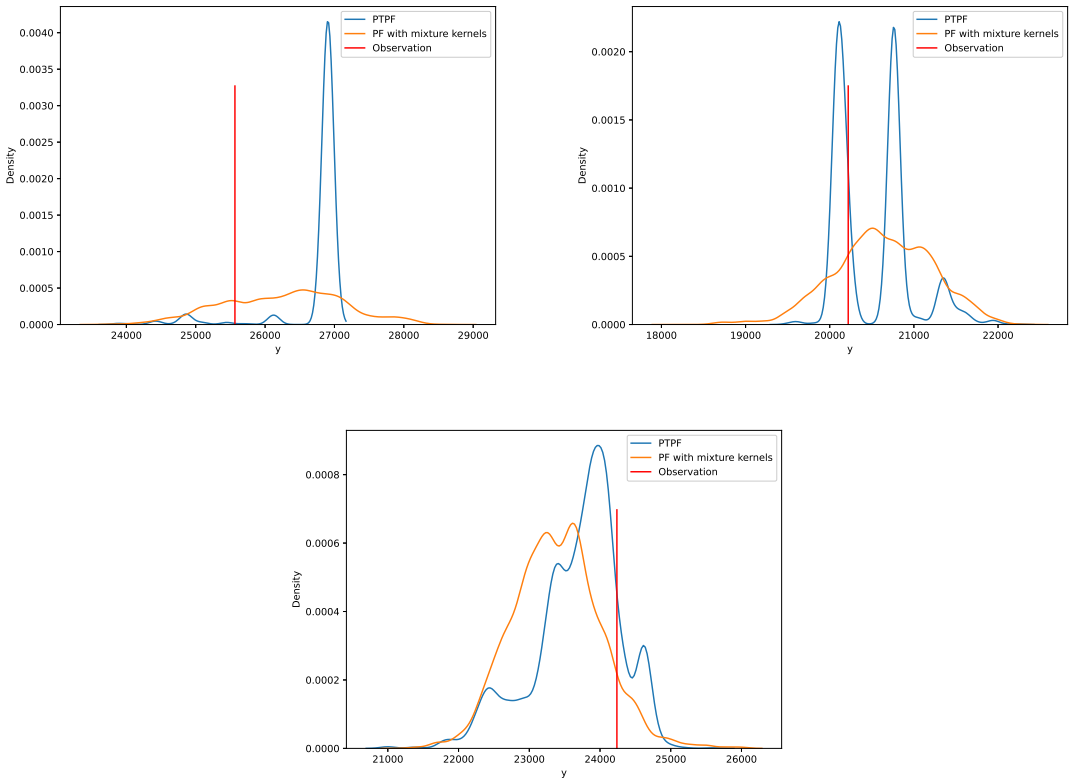

Figure 6.9: Distribution of particles locations around $10^{th}, 20^{th}$ and 30^{th} observations on y axis for n=1000 initial number of particles.

servation grows with increasing time in the PTPF and has the opposite trend in the pf with mixture kernels. This could indicate that with advancing time steps, the PTPF exhibits more diversity, and the pf with mixture kernels is instead more precise in identifying the trajectory. The latter fact could, however, be peculiar to the particular input data.

If we increase the number of initial particles, however, both of the distributions will eventually converge to the real one. Figure 6.10 shows the distribution of the particles around the 20^{th} observation time. In order to understand whether the algorithms were actually converging towards the right distribution, we used a Kalman Filter as comparison. In we C have included some background on the latter.

Distribution of Jump Locations

Figures 6.11 and 6.12 compare algorithm performance in terms of estimating jump locations. It should be noted that in this setting, even though acceleration jumps occur, only the vehicle's position is observed. As a result, even optimal estimates of jump times will be highly variable in recent history.

PTPF, given the way jumps are generated, has a much higher variance than PF with

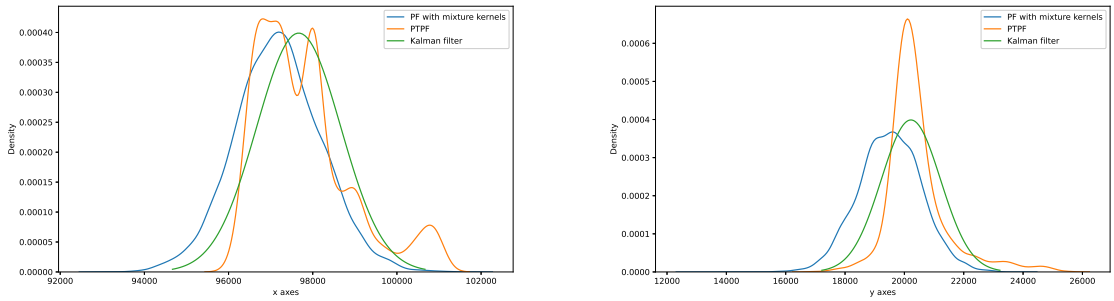

Figure 6.10: Distribution of particles locations around the 20^{th} observation on x and y axes for n=500000 initial number of particles.

mixture kernels. In addition, it should be noted that the final jump times of the PTPF all occur at a time greater than t_{max} (hence the distributions starting from the penultimate jump are shown in the figure). Both algorithms also overestimate the timing of jumps. This, as mentioned earlier, is due to the fact that only changes in position are observed. It is, therefore, more accurate to compare jumps to changes in acceleration magnitude (also shown in Figures 6.11 and 6.12). The acceleration magnitude is obtained by differentiating two times the space vector.

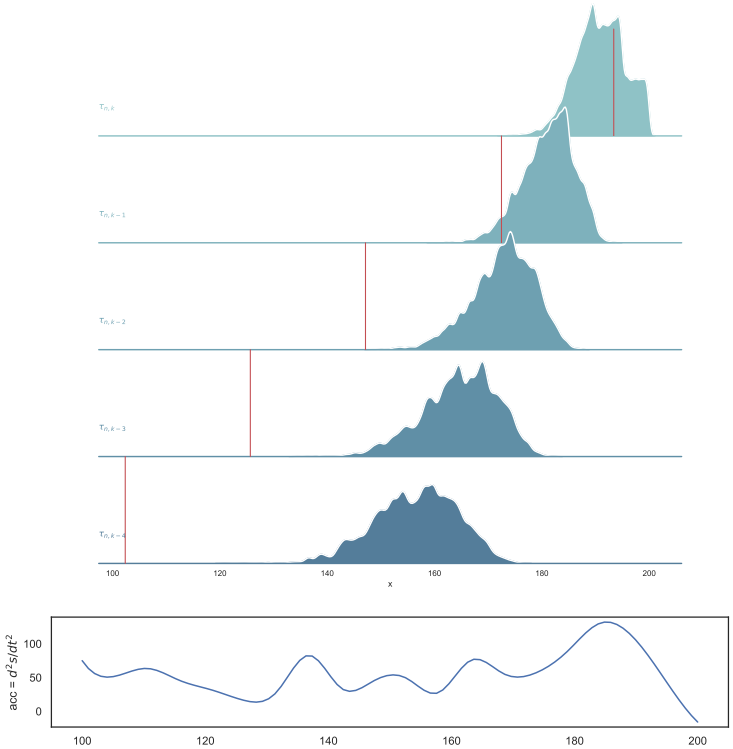

Figure 6.11: Distribution of last five jump times of PF with mixture kernels, real jump times (red lines) and magnitude of the acceleration.

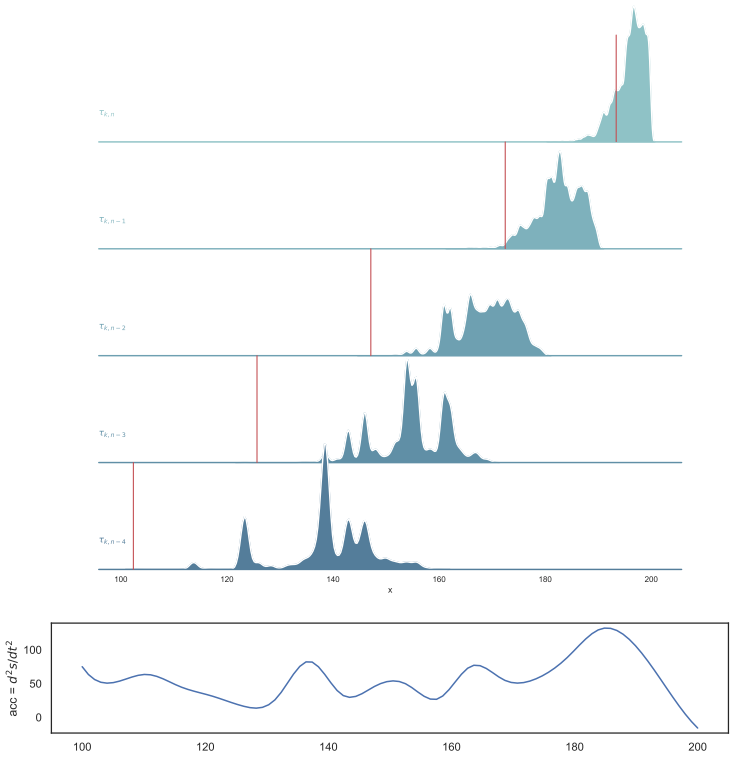

Figure 6.12: Distribution of jump times of PTPF, real jump times (red lines) and magnitude of the acceleration.

6.1.2 Particle MCMC Methods

In this section, we will compare the results obtained using the different PMCMC methods introduced in chapter 5. In particular, we will compare the results obtained in estimating the shape and scale parameters of the Gamma distribution from which they are sampled in jump times from a random guess.

In order to do that we will keep track of the time difference between two consecutive jump times ($\Delta\tau = \tau_i - \tau_{i-1}$) and plotting their distribution (obtained thorough KDE). This is an approximation of the distribution of the inter-arrival times. Figure 6.13 shows the results for the four algorithms we are considering for 1000 iterations, together with the pdf of a Gamma distribution with parameters shape equal to 10 and scale equal to 2.5 (which correspond to the one used to generate the data). Note that all algorithms seem to converge to real distribution. However, the SMC^2 appear to converge faster than the others.

6.2 Real World Data

In this section, we will test the performance of the algorithms introduced so far on a real-world dataset. We will use the Piraeus AIS Dataset [97].

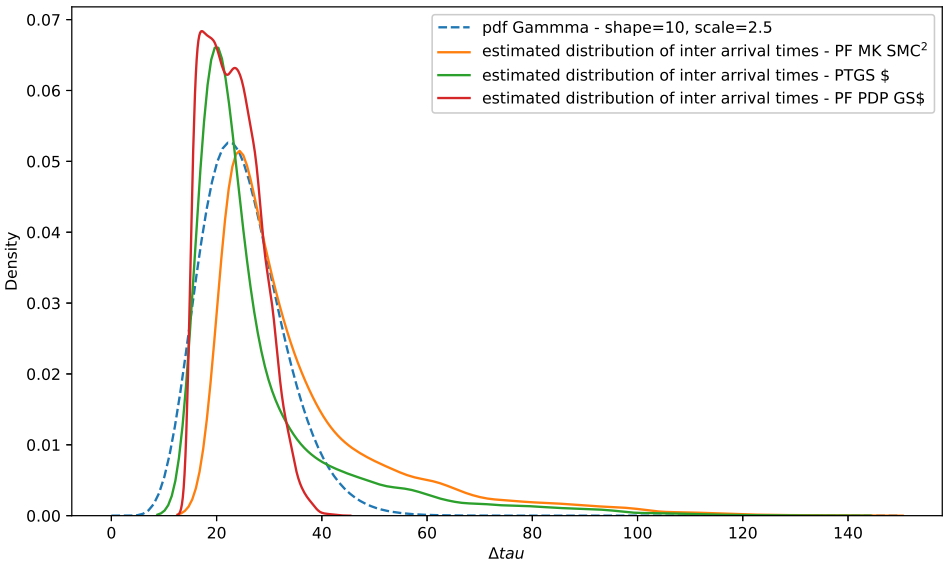

Figure 6.13: Distribution of simulated inter-arrival times for PTGS, PF MK GS and SMC^2.

6.2.1 The Dataset

The Piraeus AIS Dataset contains vessel location data transmitted by several types of boats and gathered by the AIS. The data are geographically and temporally connected, providing information on the vessels and the area of interest, as well as meteorological data within the larger region of the port of Piraeus (Greece), one of Europe's and the world's biggest ports. The data covers over two and a half years, with a time span going from May 9^{th} 2017 to December 26^{th} 2019.

Vessel Movement and AIS

Marine mobility data includes a wide range of information that can be quite useful in the context of the maritime sector. The effective use of positional (tracking) messages is critical in this approach. A wide range of marine monitoring applications can utilise the acquired data, offering valuable information for collision avoidance, traffic management, intelligent navigation, maritime situational awareness, and other real-world applications.

The Automatic Identification System (AIS) has been declared an obligatory requirement for the most popular vessel types among the several surveillance systems used to detect and locate boats at sea in real-time (cargo, passenger, fishing ships, etc.). The AIS is a surveillance technique that uses both GPS and shipboard sensors.

A vessel sends AIS signals that comprise kinematic information (position, speed, etc.) as well as static and voyage-related data on a regular basis. Among the information transmitted, the one of interest is the vessel's identifier, declared by its Maritime Mobile Service Identity

number, position, course, heading, and speed [97].

Data Description

The AIS-related data are grouped in kinematic and static. The ones we will use for our analysis are the former. They consist of 32 files in CSV format, each row containing a decoded AIS message, with the following information:

- *timestamp*: UNIX timestamp of the received AIS message (ms.);
- *vessel_id*: (artificial) vessel's identifier
- *lon*: longitude of vessel's position (angular units);
- *heading*: vessel's heading relative to true north (degrees [0-360));
- *course*: vessel's course over ground (degrees [0-360));
- *speed*: vessel's speed (knots).

6.2.2 Preprocessing

Before using the Piraeus AIS Dataset, we must modify the data so that it is usable. First, after removing rows containing null or invalid values from the dataset, we reordered it chronologically. Then we subtracted the minimum value of time so that t_0=0. Finally, after realising that, on average, a vessel reports its position every 1000 seconds, we divided all values in the column containing the timestamp by 1000. In this way, we will work with a Δ_t close to 1.

6.2.3 New Model's Assumptions

This type of problem can be modelled as described in section 2.2.2, taking into account that, in this case, the velocities of the moving object are also observed.

So we will assume that the motion of the vessel is determined by a PDP subject to jumps in the acceleration component. The inter arrivals times will follow a gamma distribution (whose parameters we will infer), and the accelerations will be sampled independently of a normal distribution. We will also assume the noise on the observed positions and velocities to be Gaussian.

The only modification we will make to the PFs will therefore be to include a velocity component in the weight calculation.

It should also be noted that the time intervals at which information on the vessel's motion is collected are no longer regular. Instead of a fixed time delta, it is sufficient to use a vector containing the amount of time elapsed between every two consecutive observations.

6.2.4 Results

Filtering

We start with the filtering problem, i.e. estimating the posterior distribution given the observations.

We selected three vessels id and considered their position for 500 consecutive time steps. We used both PTPF and the PF with mixture kernels to estimate the trajectories (Figure 6.14).

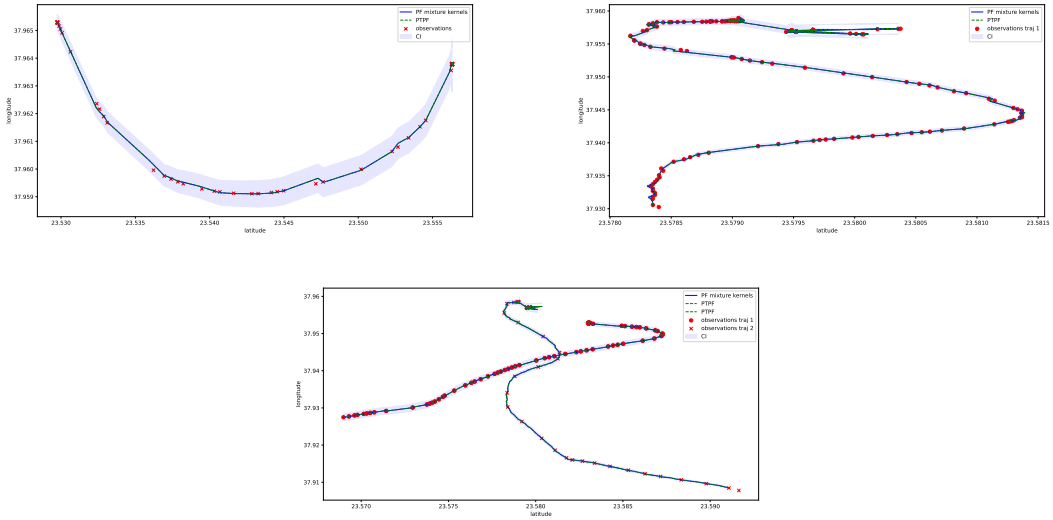

Figure 6.14: Observations and filtering estimate for the trajectories of three vessels in the Piraeus AIS Dataset.

Prediction

If we know the position of a vessel up to a certain point, we might want to predict its position in the near future. To do this, we need to introduce the formula for the prediction step

$$p(x_{t+1}|y_{1:t}) = \int f(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t \tag{6.1}$$

In our case, this means to carry out one propagation step for each of the particles at that time step e taking their weighted sum (the weights being the likelihood of the parent particles).

In the case of the PF with mixture kernels, this means at time step t choosing a move (birth or adjust) for each of the particles and computing the position of the trajectory at time $t+1$. For PTPF, we should instead run the algorithm by imposing t as t_{max}. We will then proceed to calculate the position of all particles at time t_{max} and their respective weights. We can then continue by running all particles (starting at time t) and this time assigning the value of $t+1$ to t_{max}.

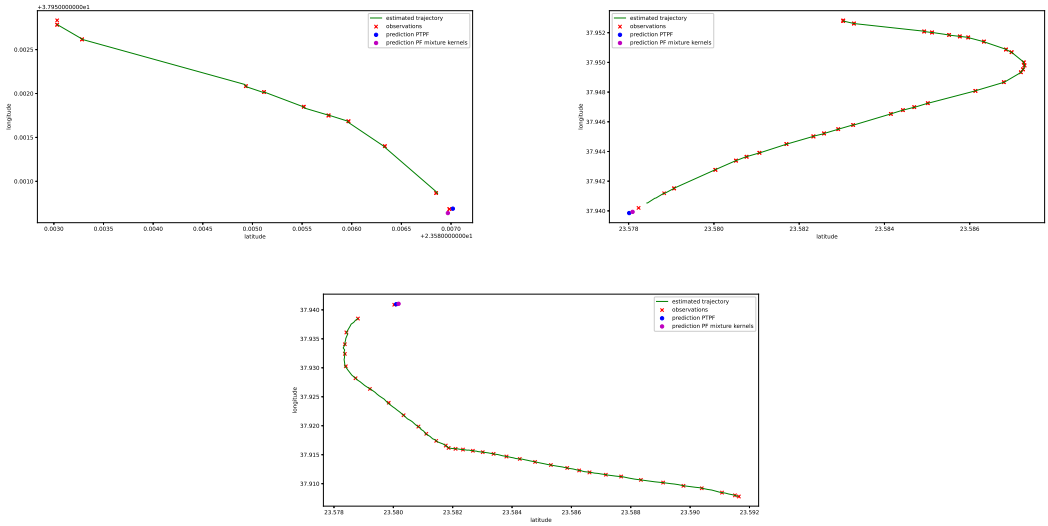

Figure 6.15: Prediction of the next observation using PF with mixture kernels and PTPF for three different trajectories.

Figure 6.15 shows the results obtained using formula (6.1) for both PF algorithms we are considering. Both predictions come very close to the actual (unseen) observations that we are targeting.

Inference on the Parameters

So far, we have seen how to estimate the posterior distribution and how to predict the position of the vessel at a later time, given certain model assumptions. However, the parameters used are only a guess and may not correspond to the actual parameters of the model. In particular, we have assumed that the inter-arrival times are distributed as a Gamma, with given shape and scale. We will therefore use the methods introduced in the previous chapters.

We ran the PTGS, PF MK GS and SMC^2 for 1000 iterations. Figure 6.16 shows the results obtained. The distributions of simulated inter-arrival times are plotted together with the pdf of a range with shape equal to 3 and scale equal to 5, which is the distribution to which all methods seem to converge.

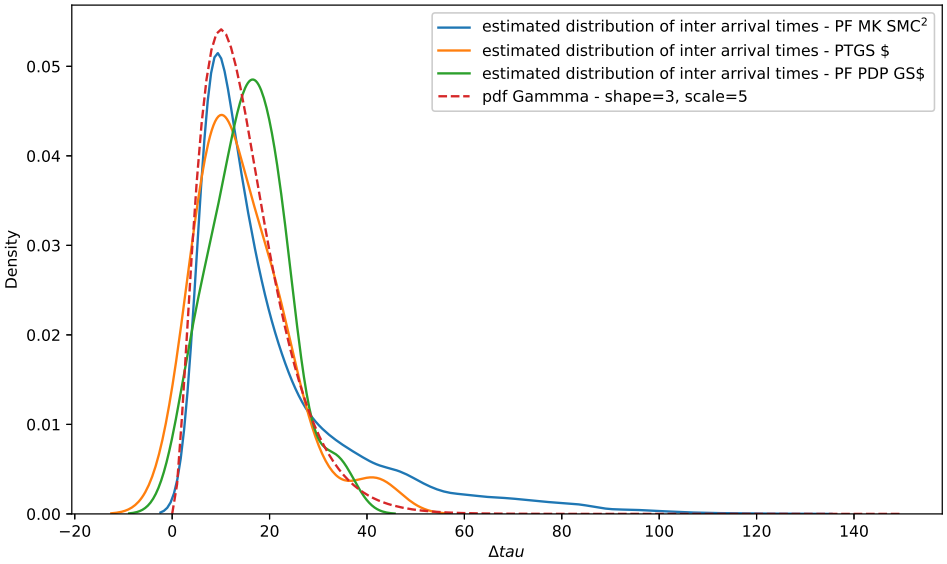

Figure 6.16: Distribution of the simulated inter-arrival times for PTGS, PF MK GS and SMC^2.

Chapter 7

Discussion and Conclusion

In this thesis, we presented several algorithms for performing inference on the states and parameters of piecewise deterministic Markov models in the object tracking framework.

Initially, we introduced a class of algorithms called Particle Filters in order to obtain an approximation of the posterior distribution of the hidden process given some of its noisy observations. In particular, we discussed in detail the implementation and properties of the PF with mixture kernels, introduced by [101], and the PTPF, presented in [14]. These two algorithms were specifically designed to be applied to piecewise deterministic processes and to introduce some kind of discretisation to deal with such processes. We compared the two algorithms in terms of bias, diversification of trajectories, and general ability to return an accurate estimate of the posterior distribution by testing them on both generated and real-world data (Piraeus AIS dataset). We concluded that Mixture kernels outperforms PTPF in estimating trajectories and diversity of the generated particles. However, PTPF comes closest in recognising when the random jumps occurred.

These particle filters were then employed in the context of MCMC models to construct PMCMC algorithms. In particular we implemented the PTGS [14] and the PF PDP GS, i.e. GS adapted to the PF with mixture kernels. We then introduced the SMC^2 model from [21] to allow us to estimate the parameters and states sequentially. This algorithm, already present in the discrete version, was modified to be used in conjunction with the implemented particle filters and thus be adapted to the PDPs approximation. The models were then tested to estimate the parameters governing the distribution of inter-arrival times. All algorithms converged to the same distribution, known in the case of generated data and unknown in the case of real ones, with SMC^2 having the most accurate result for the same number of iterations. This, in fact, can be attributed to the construction of the algorithm, which not only involves multiple iterations but also inferring parameters sequentially within a single iteration.

Note that, to test each of the implemented algorithms, we made certain assumptions about the type of underlying model we were dealing with. However, different assumptions are possible, such as the type of distribution governing inter-arrival times or acceleration. We can formulate the model itself differently; some examples are in [61, 90, 24].

Finally, PMCMC algorithms were used in this thesis to infer the parameters governing the distribution of inter-arrival times. Still, they could be applied to estimating other parameters as well, such as those governing the random jump of the acceleration value.

Appendix A

Feynman-Kac formulation

Having a sequence of observations $Y_1 = y_1, \dots Y_t = y_t, Y_T = y_T$ for $t = 1, \dots T$, we can set $G_t(x_t) = p(y_t|x_t)$. We can now write the Feynman-Kac formula

$$\begin{aligned} \int F(x_1, \dots, x_T) p(x_1, \dots, x_T | y_1, \dots, y_T) dx_1, \dots, dx_T = \\ = \frac{\int F(x_1, \dots, x_T) \{\prod_{t=1}^{T} p(y_t|x_t)\} p(x_1, \dots, x_T) dx_1, \dots, dx_T}{\int \{\prod_{t=1}^{T} p(y_t|x_t)\} p(x_1, \dots, x_T) dx_1, \dots, dx_T} \\ = \frac{E(F(X_1, \dots, T)) \prod_{t=1}^{T} G_t(X_t)}{E(\prod_{t=1}^{T} G_t(X_t))} \end{aligned} \tag{A.1}$$

where F is a bounded function on the set of trajectories of X_t for $t = 1, \dots T$.

Computational physics, biology, information theory, and computer sciences are just some of the fields in which Feynman-Kac path integration models appear with different interpretations [46, 38, 37].

Appendix B

Algorithms

B.1 PMCMC Algorithms

B.1.1 Particle Independent Metropolis Hastings

In Algorithm 12 we give a pseudo code version of the PIMH [3], used to sample from $p_\theta(x_{1:T}|y_{1:T})$.

Algorithm 12 Particle Independent Metropolis Hastings

1: **Initialization**: set $n = 0$ and run an SMC algorithm to targeting $p_\theta(x_{1:T}|y_{1:T})$ and sample $X_{1:T} \sim \hat{p}_\theta(\cdot|y_{1:T})$. Let $\hat{p}_\theta(y_{1:T})$ denote the corresponding marginal likelihood estimate.

2: for $n > 0$:

- Run an SMC algorithm to target $p_\theta(x_{1:T}|y_{1:T})$ and sample $X'_{1:T}(0) \sim \hat{p}_\theta(\cdot|y_{1:T})$. Let $\hat{p}'_\theta(y_{1:T})$ denote the corresponding marginal likelihood estimate.
- given the current state $X_{1:T}(n-1)$ we compute the acceptance probability:

$$p_a = \min(1, \frac{p'_\theta(y_{1:t})}{p_\theta(y_{1:t})(n-1)})$$

- set $X_{1:T}(n) = X'_{1:T}$ and $p_\theta(y_{1:t})(n) = p'_\theta(y_{1:t})$ with probability p_a, and $X_{1:T}(n) = X_{1:T}(n-1)$ and $p_\theta(y_{1:t})(n) = p_\theta(y_{1:t})(n-1)$ with probability $1 - p_a$

where

$$\hat{p}_\theta(y_{1:T}) := \hat{p}_\theta(t_1) \prod_{n=2}^{T} \hat{p}_\theta(y_n|y_{1:n-1}) \tag{B.1}$$

and $\hat{p}_\theta(y_n|y_{1:n-1}) = \frac{1}{N}\sum_{k=1}^{N} w_n(X^n_{1:n})$ is an estimate at time n of

$$p_\theta(y_n|y_{1:n-1}) = \int w_n(x_{1:n}) q_\theta(x_n|y_n, x_{n-1}) p_\theta(x_{1:n-1}|y_{1:n-1}) dx_{1:n}$$

(from [3]).

B.1.2 Particle Marginal Metropolis Hastings

In Algorithm 13 we give a pseudo code version of the PMMH algorithm [3], used to sample from $p(\theta, x_{1:T}|y_{1:T})$.

Algorithm 13 Particle Marginal Metropolis Hastings

1: **Initialization**: n=0

- set $\theta(0)$ arbitrarily

run an SMC algorithm to targeting $p_{\theta(0)}(x_{1:T}|y_{1:T})$ and sample $X_{1:T} \sim \hat{p}_{\theta(0)}(\cdot|y_{1:T})$. Let $\hat{p}_{\theta(0)}(y_{1:T})$ denote the corresponding marginal likelihood estimate (see (B.1)).

2: for $n > 0$:

- sample $\theta' \sim q(\cdot|\theta(n-1))$
- run an SMC algorithm to targeting $p_{\theta'}(x_{1:T}|y_{1:T})$ and sample $X'_{1:T}(0) \sim \hat{p}_{\theta'}(\cdot|y_{1:T})$. Let $\hat{p}'_{\theta'}(y_{1:T})$ denote the corresponding marginal likelihood estimate.
- given the current state $X_{1:T}(n-1)$ we compute the acceptance probability:

$$\min(1, \frac{p_{\theta'}(y_{1:T})p(\theta')q(\theta(n-1)|\theta')}{p_{\theta(n-1)}(y_{1:T})p(\theta(n-1))q(\theta'|\theta(n-1))}$$

and$\theta(n) = \theta'$, $p_{\theta(n-1)}(y_{1:t})(n) = p_{\theta'}(y_{1:t})$ with probability p_a, and $\theta(n) = \theta(n-1)$, $X_{1:T}(n) = X_{1:T}(n-1)$ and $p_\theta(y_{1:t})(n) = p_\theta(y_{1:t})(n-1))$ with probability $1 - p_a$

B.1.3 Conditional SMC

In Algorithm 14 we give a pseudo code version of the Conditional SMC algorithm.

Algorithm 14 Conditional SMC

1: let $B_{1:T}$ be an ancestral line and $X_{1:T}$ its associated path.

2: for $n = 1$

- for $k \neq B_1$, sample $X_1^k \sim q(\cdot|y_1)$
- compute its associated weight $w_1(X_1^k)$ and normalise it to W_1^k

3: for $n = 2, 3, \ldots, T$:

- for $k \neq B_n$, sample $A_{n-1}^k \sim \mathcal{F}(\cdot|\mathbf{W}_{n-1})$
- for $k \neq B_n$, sample $X_n \sim q(\cdot|y_n, X_{n-1}^{A_{n-1}^k})$
- compute the associated weights $w_1(X_{1:n}^k)$ and normalise it to W_n^k

Appendix C

The Kalman Filter

The Kalman filter is the most extensively used method for dynamic estimation problems in which the state and measurements are influenced by stochastic noise with given statistical properties. Its core idea consists in estimating a joint probability distribution over the variables of the system for each time-frame by performing a minimisation of the variance of the error between the estimated and true state of the system. A similar algorithm was devised by P. Swerling and T.N. Thiele [76] before Rudolf E. Kálmán developed the procedure that is now named after him. The Kalman filter algorithm is widely used in modern applications including computer vision, object tracking, electronics, economics, finance and physics [68, 70, 6, 11, 8]

To implement the optimal filter, reliable statistical characterisations of the system noise and measurement error in terms of mean and variance are needed.

Consider the linear time-varying system:

$$\begin{aligned}\dot{\mathbf{x}} &= F(t)\mathbf{x} + B(t)\mathbf{u} + G(t)\mathbf{w}\\ \mathbf{y}_k &= H_k\mathbf{x}_k + \mathbf{v}_k\end{aligned}$$

with $\mathbf{x}$ being the state vector, $\mathbf{u}$ the input vector, $\mathbf{Y}$ the observation vector, $\mathbf{w}$ a mean-zero white noise with the spectral density Q e density matrix G associated with the system's states, and $\mathbf{v}$ a mean-zero white noise with covariance matrix R associated with the observations.

Considering the first equation, in writing differential equation describing the propagation of the optimal state estimate, the white noise term is omitted as it can not be predicted

$$\dot{\hat{\mathbf{x}}} = F(t)\hat{\mathbf{x}} + B(t)\mathbf{u} \tag{C.1}$$

Defining the estimate error as

$$\varepsilon \equiv \mathrm{x} - \hat{\mathrm{x}}$$

its propagation equation is

$$\dot{\varepsilon} = F(t)\varepsilon + G(t)\mathbf{w}$$

The variance of the error can be defined as

$$P \equiv \mathbb{E}\left[\varepsilon\varepsilon^T\right]$$

and its propagation equation, solvable for the initial condition $P(t_0) = P_0$, is

$$\dot{P} = \mathbb{E}\left[\varepsilon\dot{\varepsilon}^T\right] + \mathbb{E}\left[\dot{\varepsilon}\varepsilon^T\right] = FP + PF^T + GQG^T,$$

as derived in [82]. The time dependency is omitted for notation clarity.

Define the discredited state estimate correction

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k\left[\mathbf{y}_k - H_k\hat{\mathbf{x}}_k^-\right]$$

where the subscript k denotes the time t_k and the superscripts $+$ and $-$ are used for the instants right after and just before an observation respectively. $\hat{\mathbf{x}}_k^-$ comes from the differential equation describing the estimate propagation C.1.

Then the error variance can be written as

$$P_k^+ \equiv \mathbb{E}\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^+\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^+\right)^T\right]$$

Plugging in the second equation in the linear time-varying system, the discrete time update of the matrix P is obtained

$$P_k^+ = \left[I - K_kH_k\right]P_k^-\left[I - K_kH_k\right]^T + K_kR_kK_k^T,$$

The optimal gain K_k is derived by minimizing the cost $J_k = \operatorname{tr} P_k^+$, thus

$$K_k = P_k^-H_k^T\left[H_kP_k^-H_k^T + R_k\right]^{-1}$$

and P_k^- from the the propagation equation of P.
Ultimately, substituting the solution for the optimal gain K_k, the discrete time propgation equation for P becomes

$$P_k^+ = \left[I - K_kH_k\right]P_k^-.$$

Summing up, one has

Model	$\dot{\mathbf{x}}(t) = F(t)\mathbf{x}(t) + B(t)\mathbf{u}(t) + G(t)\mathbf{w}(t),\ \text{con}\ \mathbf{w} \sim WN(\mathbf{0}, Q)$ $\mathbf{y}_k = H_k\mathbf{x}_k + \mathbf{v}_k,\ \text{con}\ \mathbf{v}_k \sim WN\left(\mathbf{0}, R_k\right)$
Initialization	$\hat{\mathbf{x}}\left(t_0\right) = \mathbf{x}_0, P\left(t_0\right) = P_0$
Propagation	$\dot{\hat{\mathbf{x}}}(t) = F(t)\hat{\mathbf{x}}(t) + B(t)\mathbf{u}(t)$ $\dot{P}(t) = F(t)P(t) + P(t)F^T(t) + G(t)Q(t)G^T(t)$
Gain	$K_k = P_k^-H_k^T\left[H_kP_k^-H_k^T + R_k\right]^{-1}$
Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k\left[\mathbf{y}_k - H_k\hat{\mathbf{x}}_k^-\right]$ $P_k^+ = \left[I - K_kH_k\right]P_k^-$

Bibliography

[1] Brian DO Anderson and John B Moore. *Optimal filtering*. Courier Corporation, 2012 (cit. on p. 8).

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. "Particle Markov chain Monte Carlo for efficient numerical simulation". In: *Monte Carlo and quasi-Monte Carlo methods 2008*. Springer, 2009, pp. 45–60 (cit. on p. 14).

[3] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. "Particle markov chain monte carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3 (2010), pp. 269–342 (cit. on pp. 14, 15, 53, 54).

[4] Christophe Andrieu and Gareth O. Roberts. "The pseudo-marginal approach for efficient computation". In: *Annals Statist* (2007), pp. 27–44 (cit. on p. 15).

[5] Roland Assaraf, Michel Caffarel, and Anatole Khelif. "Diffusion Monte Carlo methods with a fixed number of walkers". In: *Physical Review E* 61.4 (2000), p. 4566 (cit. on p. 10).

[6] Mohsen Bahmani-Oskooee and Ford Brown. "Kalman filter approach to estimate the demand for international reserves". In: *Applied Economics* 36.15 (2004), pp. 1655–1668 (cit. on p. 55).

[7] Mark A Beaumont. "Estimation of population growth or decline in genetically monitored populations". In: *Genetics* 164.3 (2003), pp. 1139–1160 (cit. on p. 15).

[8] Pierre Billoir, Michel De Cian, Paul André Günther, and Simon Stemmle. "A parametrized Kalman filter for fast track fitting at LHCb". In: *Computer Physics Communications* 265 (2021), p. 108026 (cit. on p. 55).

[9] W.D. Blair, G.A. Watson, T. Kirubarajan, and Y. Bar-Shalom. "Benchmark for radar allocation and tracking in ECM". In: *IEEE Transactions on Aerospace and Electronic Systems* 34.4 (1998), pp. 1097–1114 (cit. on pp. 8, 9).

[10] Mark Briers, Arnaud Doucet, and Simon Maskell. "Smoothing algorithms for state–space models". In: *Annals of the Institute of Statistical Mathematics* 62.1 (2010), pp. 61–89 (cit. on p. 13).

[11] Carmen Broto and Esther Ruiz. "Estimation methods for stochastic volatility models: a survey". In: *Journal of Economic surveys* 18.5 (2004), pp. 613–649 (cit. on p. 55).

[12] Pete Bunch and Simon Godsill. "Dynamical models for tracking with the variable rate particle filter". In: *2012 15th International Conference on Information Fusion*. IEEE. 2012, pp. 1769–1775 (cit. on p. 8).

[13] M Caffarel, DM Ceperley, and MH Kalos. "Comment on "Feynman-Kac path-integral calculation of the ground-state energies of atoms"". In: *Physical review letters* 71.13 (1993), p. 2159 (cit. on p. 10).

[14] Tomasz Cakała, Błażej Miasojedow, and Wojciech Niemiro. "Particle MCMC With Poisson Resampling: Parallelization and Continuous Time Models". In: *Journal of Computational and Graphical Statistics* 30.3 (2021), pp. 671–684 (cit. on pp. 2, 15, 17, 21–25, 27, 28, 30, 34, 50).

[15] Olivier Cappé, Simon Godsill, and Eric Moulines. "An overview of existing methods and recent advances in sequential Monte Carlo". In: *Proceedings of the IEEE* 95.5 (2007), pp. 899–924 (cit. on pp. 8, 10).

[16] James Carpenter, Peter Clifford, and Paul Fearnhead. "Improved particle filter for nonlinear problems". In: *IEE Proceedings-Radar, Sonar and Navigation* 146.1 (1999), pp. 2–7 (cit. on p. 11).

[17] Chris K Carter and Robert Kohn. "On Gibbs sampling for state space models". In: *Biometrika* 81.3 (1994), pp. 541–553 (cit. on pp. 13, 14).

[18] S Centanni and M Minozzo. "Estimation and filtering by reversible jump MCMC for a doubly stochastic Poisson model for ultra-high-frequency financial data". In: *Statistical Modelling* 6.2 (2006), pp. 97–118 (cit. on p. 3).

[19] Nicolas Chopin. "A sequential particle filter method for static models". In: *Biometrika* 89.3 (2002), pp. 539–552 (cit. on pp. 31, 32).

[20] Nicolas Chopin. "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference". In: *The Annals of Statistics* 32.6 (2004), pp. 2385–2411 (cit. on pp. 16, 32, 35).

[21] Nicolas Chopin, Pierre E Jacob, and Omiros Papaspiliopoulos. "SMC2: an efficient algorithm for sequential analysis of state space models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75.3 (2013), pp. 397–426 (cit. on pp. 2, 11, 27, 31, 33, 50).

[22] Nicolas Chopin and Sumeetpal S Singh. "On particle Gibbs sampling". In: *Bernoulli* 21.3 (2015), pp. 1855–1883 (cit. on p. 14).

[23] Bertrand Cloez, Renaud Dessalles, Alexandre Genadot, Florent Malrieu, Aline Marguet, and Romain Yvinec. "Probabilistic and piecewise deterministic models in biology". In: *ESAIM: Proceedings and Surveys* 60 (2017), pp. 225–245 (cit. on p. 3).

[24] Alessandro Corbetta, Chung-min Lee, Roberto Benzi, Adrian Muntean, and Federico Toschi. "Fluctuations around mean walking behaviors in diluted pedestrian flows". In: *Physical Review E* 95.3 (2017), p. 032316 (cit. on p. 50).

[25] David R Cox. "The analysis of non-Markovian stochastic processes by the inclusion of supplementary variables". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 51. 3. Cambridge University Press. 1955, pp. 433–441 (cit. on p. 3).

[26] Drew Creal. "A survey of sequential Monte Carlo methods for economics and finance". In: *Econometric reviews* 31.3 (2012), pp. 245–296 (cit. on p. 10).

[27] Dan Crisan, Pierre Del Moral, and Terry Lyons. *Discrete filtering using branching and interacting particle systems*. Citeseer, 1998 (cit. on p. 10).

[28] Johan Dahlin, Fredrik Lindsten, and Thomas B Schön. "Particle Metropolis–Hastings using gradient and Hessian information". In: *Statistics and computing* 25.1 (2015), pp. 81–92 (cit. on p. 14).

[29] Johan Dahlin and Thomas B Schön. "Getting started with particle Metropolis-Hastings for inference in nonlinear dynamical models". In: *arXiv preprint arXiv:1511.01707* (2015) (cit. on p. 14).

[30] Angelos Dassios and Ji-Wook Jang. "Kalman-Bucy filtering for linear systems driven by the Cox process with shot noise intensity and its application to the pricing of reinsurance contracts". In: *Journal of applied probability* 42.1 (2005), pp. 93–107 (cit. on p. 3).

[31] Paresh Date and Ksenia Ponomareva. "Linear and non-linear filtering in mathematical finance: a review". In: *IMA Journal of Management Mathematics* 22.3 (2011), pp. 195–211 (cit. on p. 10).

[32] Mark HA Davis. "The representation of martingales of jump processes". In: *SIAM Journal on control and optimization* 14.4 (1976), pp. 623–638 (cit. on p. 3).

[33] Mark HA Davis. "Piecewise-deterministic Markov processes: A general class of non-diffusion stochastic models". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 46.3 (1984), pp. 353–376 (cit. on p. 3).

[34] Piet De Jong and Neil Shephard. "The simulation smoother for time series models". In: *Biometrika* 82.2 (1995), pp. 339–350 (cit. on p. 13).

[35] Pierre Del Moral. "Nonlinear filtering: Interacting particle resolution". In: *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics* 325.6 (1997), pp. 653–658 (cit. on pp. 10, 11).

[36] Pierre Del Moral. "Measure-valued processes and interacting particle systems. Application to nonlinear filtering problems". In: *The Annals of Applied Probability* 8.2 (1998), pp. 438–495 (cit. on pp. 10, 11).

[37] Pierre Del Moral. *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. Vol. 88. Springer, 2004 (cit. on pp. 10, 11, 33, 52).

[38] Pierre Del Moral. "Mean field simulation for Monte Carlo integration". In: *Monographs on Statistics and Applied Probability* 126 (2013), p. 26 (cit. on pp. 10, 52).

[39] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. "Sequential monte carlo samplers". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3 (2006), pp. 411–436 (cit. on pp. 17, 18).

[40] Pierre Del Moral, Arnaud Doucet, and Sumeetpal Singh. "Forward smoothing using sequential Monte Carlo". In: *arXiv preprint arXiv:1012.5390* (2010) (cit. on p. 13).

[41] Pierre Del Moral and Alice Guionnet. "Central limit theorem for nonlinear filtering and interacting particle systems". In: *Annals of Applied Probability* (1999), pp. 275–297 (cit. on p. 10).

[42] Pierre Del Moral and Alice Guionnet. "On the stability of measure valued processes with applications to filtering". In: *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics* 329.5 (1999), pp. 429–434 (cit. on pp. 10, 11).

[43] Pierre Del Moral and Alice Guionnet. "On the stability of interacting processes with applications to filtering and genetic algorithms". In: *Annales de l'Institut Henri Poincaré (B) Probability and Statistics*. Vol. 37. 2. Elsevier. 2001, pp. 155–194 (cit. on pp. 10, 11).

[44] Pierre Del Moral, Peng Hu, Liming Wu, et al. "On the concentration properties of interacting particle processes". In: *Foundations and Trends® in Machine Learning* 3.3–4 (2012), pp. 225–389 (cit. on p. 10).

[45] Pierre Del Moral, Robert Kohn, and Frédéric Patras. "On particle Gibbs samplers". In: *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*. Vol. 52. 4. Institut Henri Poincaré. 2016, pp. 1687–1733 (cit. on p. 14).

[46] Pierre Del Moral and Laurent Miclo. "Branching and interacting particle systems approximations of Feynman-Kac formulae with applications to non-linear filtering". In: *Seminaire de probabilites XXXIV* (2000), pp. 1–145 (cit. on pp. 10, 52).

[47] Pierre Del Moral and Laurent Miclo. "Particle approximations of Lyapunov exponents connected to Schrödinger operators and Feynman–Kac semigroups". In: *ESAIM: Probability and Statistics* 7 (2003), pp. 171–208 (cit. on p. 10).

[48] Pierre Del Moral and Spiridon Penev. *Stochastic Processes: From Applications to Theory*. Chapman and Hall/CRC, 2017 (cit. on p. 1).

[49] Pierre Del Moral and Emmanuel Rio. "Concentration inequalities for mean field particle models". In: *The Annals of Applied Probability* 21.3 (2011), pp. 1017–1052 (cit. on p. 10).

[50] Randal Douc, Aurélien Garivier, Eric Moulines, and Jimmy Olsson. "Sequential Monte Carlo smoothing for general state space hidden Markov models". In: *The Annals of Applied Probability* 21.6 (2011), pp. 2109–2145 (cit. on p. 13).

[51] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering". In: *Statistics and computing* 10.3 (2000), pp. 197–208 (cit. on pp. 9, 13).

[52] Arnaud Doucet, Neil J Gordon, and Vikram Krishnamurthy. "Particle filters for state estimation of jump Markov linear systems". In: *IEEE Transactions on signal processing* 49.3 (2001), pp. 613–624 (cit. on p. 35).

[53] Arnaud Doucet, Adam M Johansen, et al. "A tutorial on particle filtering and smoothing: Fifteen years later". In: *Handbook of nonlinear filtering* 12.656-704 (2009), p. 3 (cit. on pp. 8, 10).

[54] Ludwig Fahrmeir and Gerhard Tutz. "State space and hidden markov models". In: *Multivariate Statistical Modelling Based on Generalized Linear Models* (2001), pp. 331–383 (cit. on p. 10).

[55] Paul Fearnhead. "Computational methods for complex stochastic systems: a review of some alternatives to MCMC". In: *Statistics and Computing* 18.2 (2008), pp. 151–171 (cit. on p. 10).

[56] Paul Fearnhead, David Wyncoll, and Jonathan Tawn. "A sequential smoothing algorithm with linear computational cost". In: *Biometrika* 97.2 (2010), pp. 447–464 (cit. on p. 13).

[57] Sylvia Frühwirth-Schnatter. "Data augmentation and dynamic linear models". In: *Journal of time series analysis* 15.2 (1994), pp. 183–202 (cit. on p. 13).

[58] Sylvia Frühwirth-Schnatter and Leopold Sögner. "Bayesian estimation of stochastic volatility models based on OU processes with marginal Gamma law". In: *Annals of the Institute of Statistical Mathematics* 61.1 (2009), pp. 159–179 (cit. on p. 14).

[59] Walter R Gilks and Carlo Berzuini. "Following a moving target—Monte Carlo inference for dynamic Bayesian models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.1 (2001), pp. 127–146 (cit. on p. 16).

[60] BV Gnedenko and IN Kovalenko. "Introduction to the theory of mass service". In: *Nau (trad, ingles: 1968, Jerusalem)* (1966) (cit. on p. 3).

[61] Simon Godsill. "Particle filters for continuous-time jump models in tracking applications". In: *ESAIM: Proceedings*. Vol. 19. EDP Sciences. 2007, pp. 39–52 (cit. on pp. 8, 16, 50).

[62] Simon Godsill, Arnaud Doucet, and Mike West. "Maximum a posteriori sequence estimation using Monte Carlo particle filters". In: *Annals of the Institute of Statistical Mathematics* 53.1 (2001), pp. 82–96 (cit. on p. 13).

[63] Simon Godsill, Arnaud Doucet, and Mike West. "Monte Carlo smoothing for nonlinear time series". In: *Journal of the american statistical association* 99.465 (2004), pp. 156–168 (cit. on p. 13).

[64] Simon Godsill, Jaco Vermaak, William Ng, and Jack F Li. "Models and algorithms for tracking of maneuvering objects using variable rate particle filters". In: *Proceedings of the IEEE* 95.5 (2007), pp. 925–952 (cit. on pp. 3, 8).

[65] Pieralberto Guarniero, Adam M Johansen, and Anthony Lee. "The iterated auxiliary particle filter". In: *Journal of the American Statistical Association* 112.520 (2017), pp. 1636–1647 (cit. on p. 13).

[66] Ehsan Hajiramezanali, Mahdi Imani, Ulisses Braga-Neto, Xiaoning Qian, and Edward R Dougherty. "Scalable optimal Bayesian classification of single-cell trajectories under regulatory model uncertainty". In: *BMC genomics* 20.6 (2019), pp. 1–11 (cit. on p. 10).

[67] JH Hetherington. "Observations on the statistical iteration of matrices". In: *Physical Review A* 30.5 (1984), p. 2713 (cit. on p. 10).

[68] Lim Chot Hun, Ong Lee Yeng, Lim Tien Sze, Koo Voon Chet, and K Jian. "Kalman filtering and its real-time applications". In: *Real-time Systems* (2016) (cit. on p. 55).

[69] Martin Jacobsen and Joseph Gani. "Point process theory and applications: marked point and piecewise deterministic processes". In: (2006) (cit. on pp. 3, 4, 8).

[70] Youngjoo Kim and Hyochoong Bang. "Introduction to Kalman filter and its applications". In: *Introduction and Implementations of the Kalman Filter* 1 (2018), pp. 1–16 (cit. on p. 55).

[71] Genshiro Kitagawa. "Monte Carlo filter and smoother for nonlinear non Gaussian state models". In: *Journal of Computation and Graphical Statistics* 5 (1996), pp. 1–25 (cit. on p. 13).

[72] Genshiro Kitagawa. "A self-organizing state-space model". In: *Journal of the American Statistical Association* (1998), pp. 1203–1215 (cit. on p. 11).

[73] Genshiro Kitagawa and Seisho Sato. "Monte Carlo smoothing and self-organising state-space model". In: *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 177–195 (cit. on p. 13).

[74] Augustine Kong, Jun S Liu, and Wing Hung Wong. "Sequential imputations and Bayesian missing data problems". In: *Journal of the American statistical association* 89.425 (1994), pp. 278–288 (cit. on p. 19).

[75] Vassilios Lasdas and MHA Davis. "A piecewise deterministic process approach to target motion analysis (sonar)". In: *Proceedings of the 28th IEEE Conference on Decision and Control,* IEEE. 1989, pp. 1395–1396 (cit. on p. 3).

[76] Steffen L Lauritzen. "Time series analysis in 1880: A discussion of contributions made by TN Thiele". In: *International Statistical Review/Revue Internationale de Statistique* (1981), pp. 319–331 (cit. on p. 55).

[77] Olivier Lévque. *Lecture notes on Markov chains*. Aug. 2011 (cit. on p. 4).

[78] X Rong Li and Vesselin P Jilkov. "Survey of maneuvering target tracking. Part I. Dynamic models". In: *IEEE Transactions on aerospace and electronic systems* 39.4 (2003), pp. 1333–1364 (cit. on p. 8).

[79] Fredrik Lindsten, Randal Douc, and Eric Moulines. "Uniform ergodicity of the particle Gibbs sampler". In: *Scandinavian Journal of Statistics* 42.3 (2015), pp. 775–797 (cit. on p. 22).

[80] Fredrik Lindsten, Michael I Jordan, and Thomas B Schon. "Particle Gibbs with ancestor sampling". In: *Journal of Machine Learning Research* 15 (2014), pp. 2145–2184 (cit. on p. 22).

[81] Jun S Liu and Rong Chen. "Sequential Monte Carlo methods for dynamic systems". In: *Journal of the American statistical association* 93.443 (1998), pp. 1032–1044 (cit. on p. 11).

[82] F Landis Markley and John L Crassidis. *Fundamentals of spacecraft attitude determination and control*. Vol. 1286. Springer, 2014 (cit. on p. 56).

[83] Simon Maskell and Neil Gordon. "A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking". In: *IEE Target Tracking: Algorithms and Applications (Ref. No. 2001/174)* (2002), pp. 2–1 (cit. on p. 10).

[84] Laurent Miclo and Pierre Del Moral. "Genealogies and increasing propagation of chaos for Feynman-Kac and genetic models". In: *The Annals of Applied Probability* 11.4 (2001), pp. 1166–1198 (cit. on p. 10).

[85] Chiara Mignacco. *Particle Filter Methods*. Version 2.0.4. Sept. 2022. URL: https://github.com/chiaramignacco/particle-filter-methods.git (cit. on p. 2).

[86] Daniel J. Salmond Neil J. Gordon Adrian F.M. Smith. In: *IEE Proceedings F (Radar and Signal Processing)* 140 (2 Apr. 1993), 107–113(6). ISSN: 0956-375X (cit. on p. 9).

[87] Katja Nummiaro, Esther Koller-Meier, Luc Van Gool, et al. "A color-based particle filter". In: *First International Workshop on Generative-Model-Based Vision*. Vol. 2002. Denmark, Kopenhagen: Datalogistik Institut, Kobenhavns Universitet. 2002, p. 01 (cit. on p. 10).

[88] Antonio Pacifico. "Robust open Bayesian analysis: Overfitting, model uncertainty, and endogeneity issues in multiple regression models". In: *Econometric Reviews* 40.2 (2021), pp. 148–176 (cit. on p. 31).

[89] Brooks Paige, Frank Wood, Arnaud Doucet, and Yee Whye Teh. "Asynchronous anytime sequential monte carlo". In: *Advances in neural information processing systems* 27 (2014) (cit. on p. 22).

[90] Guillaume Ravel, Michel Bergmann, Alain Trubuil, Julien Deschamps, Romain Briandet, and Simon Labarthe. "Inferring characteristics of bacterial swimming in biofilm matrix from time-lapse confocal laser scanning microscopy". In: *arXiv preprint arXiv:2201.04371* (2022) (cit. on p. 50).

[91] Marshall N. Rosenbluth and Arianna W. Rosenbluth. "Monte Carlo calculation of the average extension of molecular chains". In: *Journal of Chemical Physics* 23 (1955), pp. 356–359 (cit. on p. 10).

[92] Jeffrey S Rosenthal. *A First Look At Rigorous Probability Theory*. World Scientific Publishing Company, 2006 (cit. on p. 5).

[93] Mathias Rousset. "On the control of an interacting particle estimation of Schrödinger ground states". In: *SIAM journal on mathematical analysis* 38.3 (2006), pp. 824–844 (cit. on p. 1).

[94] Sankalita Saha, Neal K Bambha, and Shuvra S Bhattacharyya. "Design and implementation of embedded computer vision systems based on particle filters". In: *Computer Vision and Image Understanding* 114.11 (2010), pp. 1203–1214 (cit. on p. 10).

[95] David D Sworder, Micheal Kent, Robert Vojak, and RG Hutchins. "Renewal models for maneuvering targets". In: *IEEE Transactions on Aerospace and Electronic Systems* 31.1 (1995), pp. 138–150 (cit. on p. 3).

[96] Sebastian Thrun. "Particle Filters in Robotics." In: *UAI*. Vol. 2. Citeseer. 2002, pp. 511–518 (cit. on p. 10).

[97] Andreas Tritsarolis, Yannis Kontoulis, and Yannis Theodoridis. *The Piraeus AIS Dataset for Large-scale Maritime Data Analytics*. Zenodo, Oct. 2021. URL: https://doi.org/10.5281/zenodo.6323416 (cit. on pp. 44, 46).

[98] Rudolph Van Der Merwe, Arnaud Doucet, Nando De Freitas, and Eric Wan. "The unscented particle filter". In: *Advances in neural information processing systems* 13 (2000) (cit. on pp. 9, 10).

[99] Paolo Vidoni. "Exponential Family State Space Models Based on a Conjugate Latent Process". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 61.1 (1999), pp. 213–221 (cit. on p. 9).

[100] Mike West and Jeff Harrison. *Bayesian forecasting and dynamic models*. Springer Science & Business Media, 2006 (cit. on pp. 9, 12).

[101] Nick Whiteley, Adam M Johansen, and Simon Godsill. "Monte Carlo filtering of piecewise deterministic processes". In: *Journal of Computational and Graphical Statistics* 20.1 (2011), pp. 119–139 (cit. on pp. 2, 5, 8, 16–18, 35, 50).