Improved Auxiliary Particle Filters

Nicola Branchini (B159488)

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2020

Abstract

Sequential Monte Carlo methods (SMC) or Particle Filters are the *de facto* standard for approximate filtering in probabilistic state space models. Highly informative observations can cause classical SMC algorithms that resample naively using importance weights, such as the Bootstrap Particle Filter (BPF), to provide poor approximations of the filtering distribution: the Auxiliary Particle Filter (APF) generalizes the vanilla SMC framework to allow for observations to be incorporated into the resampling via the so called *simulation weights*. It is however not always clear when and why the APF performs better than the BPF: Elvira et al. [24] showed that when transition densities overlap significantly the APF can perform poorly and used this observation to motivate the use of a weighted mixture of transition kernels distribution as proposal, where the mixture weights are the simulation weights. In this work, we further exploit the structure of this weighted mixture proposal: these simulation weights should be chosen to satisfy our ultimate goal of minimizing the variance of the importance sampling weights, which is related to the discrepancy between proposal and posterior. We exploit this fact by designing a *novel*, *fully online* framework to build *convex* optimization strategies for adaptation of the simulation weights by minimizing the discrepancy between proposal and posterior evaluated at a deterministically chosen set of points, which crucially leverages the structure of SMC inference. This leads to a novel class of algorithms which we name Optimization Auxiliary Particle Filters (OAPF). While in its basic form the algorithm can be significantly more expensive than its competitors, we provide a simple strategy to reduce its cost and show empirically how it can at the same time improve estimation of the filtering distribution when using a large number of particles. This is most clear with Linear Gaussian systems where the underlying distribution is known; for models where this is not true, we show generally improved Effective Sample Size and sample weight variance. Finally, we suggest that there are several promising directions of future work for reducing the time complexity of the algorithms while maintaining performance, as well as smarter strategies for the choice of evaluation points.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Nicola Branchini (B159488)*)

Acknowledgements

I would like to wholeheartedly thanks my supervisor, Víctor Elvira, for his support and advice throughout the whole academic year, and not just during the official dissertation period. With the help of his guidance, I learnt a lot about particle filters but also and perhaps more importantly about research and academia. I hope that this work can be the starting point for a longer collaboration and partnership.

Table of Contents

Abbreviations

AMPF	Auxiliary Marginal Particle Filter
APF	Auxiliary Particle Filter
ESS	Effective Sample Size
IAPF	Improved Auxiliary Particle Filter
IHS	Iterative Hessian Sketching
IS	Importance Sampling
KL	Kullback-Leibler
LP	Linear Program
MCMC	Markov Chain Monte Carlo
ML	Machine Learning
NNLS	Non-Negative Least Squares
OAPF	Optimized Auxiliary Particle Filter
SFD	State Filtering Distribution
SIS	Sequential Importance Sampling
SMC	Sequential Monte Carlo Methods
TFD	Trajectory Filtering Distribution

List of Figures

List of Tables

Chapter 1

Introduction

The *optimal filtering* problem [3] is a classical task that has been extensively studied by different communities, from Signal Processing to Statistics: it consists of formulating a flexible latent variable probabilistic model for time series (called *state space model*) where the unknown quantity of interest can be sequentially estimated. Conducting Bayesian inference in these models allows to perform efficiently in a plethora of applications including target tracking, spread of infectious diseases, Global Positioning Systems, but also Machine Learning (ML) applications [68, 59, 49]. However, exact inference can only be performed in an extremely limited class of state space models, most notably in Linear Gaussian systems where one can exploit the *Kalman Filter* [40]. In this context, the most popular approximate inference techniques are Sequential Monte Carlo Methods (SMC), often referred to as Particle Filters [69, 26]. In this thesis, we retrace interpretations and derivations of *Auxiliary* Particle Filters, originally invented to tackle highly informative likelihoods, and propose a new class of algorithms that can provide better performance in many scenarios.

1.1 Structure

In Chapter 2 we introduce the necessary background to understand the methodology of SMC: to remain relevant to the work introduced in this dissertation, we discuss core methodological concepts with an algorithmic perspective, leaving convergence proofs to more theoretical works. In Chapter 3 the core reasoning and methodology behind our novel class of algorithms is developed: their performance is experimentally verified in Chapter 4. Finally, in Chapter 5 we discuss implications of our experimental findings, and propose directions for future work.

1.2 Contributions

The contributions of this work span the whole document and are specifically:

- The development of a novel class of online particle filtering algorithms, here named Optimized Auxiliary Particle Filter, for filtering in generic state space models, as a potential alternative to the classical and widely used algorithms such as the Bootstrap Particle Filter [28] and the Auxiliary Particle Filter [65]

- We propose simple strategies that reduce the computational complexity of the algorithm significantly *and* sometimes improve the accuracy of its estimates at the same time. A more in-depth treatment of such strategies is left as further work, but experiments suggest that there is promise in reducing the complexity of OAPF.

- An updated review, comparison and analysis of Auxiliary-like particle filters building on [27, 45, 72, 36, 21] (previous surveys/papers on improval strategies for PFs with a focus on APF), including discussion of more recent work by Elvira et al. [23, 24] and other more recent work [47, 2]

- In Chapters 2 and partially 3, we unify the the different perspectives behind the APF and explain in detail its relationship with Marginal Particle Filters and other SMC algorithms, providing the first (to the best of our knowledge) in-depth and accessible treatment of this widely used, but rarely explained, algorithm.

Chapter 2

Background & Related Work

In this Chapter we explain the most relevant concepts that need to be understood in order to implement and assess Particle Filtering algorithms. In Section 2.3 we provide an up-to-date literature review on state of the art methodology that tries to generalize and go beyond the original Auxiliary Particle Filter, partially updating the excellent survey by Whiteley et al. [72] but only focussing on inference.

2.1 Monte Carlo methods

Monte Carlo (MC) methods are a very broad class of methods in computational statistics that originated in physics [22, 54] . They can be informally described as attempting to infer unknown quantities that are related to a population using random samples from that population. A MC estimator $\widehat{I}$ of an expectation with respect to $p(\boldsymbol{x})$ of an integrable test function f can be obtained by generating a set of *independent and identically distributed* (i.i.d.) samples from $p(\boldsymbol{x})$ and taking an empirical average:

$$I = \mathbb{E}_{p(\boldsymbol{x})}[f(\boldsymbol{x})] = \int f(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \approx \widehat{I} = \frac{1}{N}\sum_{n=1}^{N} f\left(\boldsymbol{x}^{(n)}\right) \quad \boldsymbol{x}^{(n)} \sim p(\boldsymbol{x}) \tag{2.1}$$

Another class of MC methods known as Markov Chain Monte Carlo (MCMC) [66] do not require samples to be i.i.d., but these are not in the scope of this thesis. The standard MC estimator $\widehat{I}$ is unbiased, i.e. $\mathbb{E}_{p(\boldsymbol{x})}\left[\widehat{I}\right] = I$. Besides biasedness, the typical metric of evaluation for the performance of a MC estimator is its variance, as it represents the expected squared deviation from the true value of the desired quantity:

$$\mathbb{V}_q\left[\widehat{I}\right] = \mathbb{E}_q\left[\left(\widehat{I} - I\right)^2\right] = \frac{1}{N}\left(\int f^2(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - I^2\right) \tag{2.2}$$

It is well known that some MC estimators while biased can have lower variance than unbiased estimators. The main selling points for MC methods compared to other approximate inference techniques such as Variational methods [60, 38, 5] are their desirable asymptotic properties, since most estimators are consistent (i.e. asymptotically unbiased), as well as a variance of approximation error that decreases with $O(1/N)$ *regardless of the dimensionality* of the random vector $\boldsymbol{x}$ [21]. Consistency of the estimators is proved by standard law of large numbers arguments [66].

2.1.1 Importance Sampling

Importance Sampling (IS) is a MC method, but its origin go back to the 1950s in the context of rare event estimation in statistical physics [39]. Some of the appeal towards IS lies in its relative implementation simplicity and easier theoretical treatment compared to alternative MC methods such as MCMC. At its core, IS is a technique to approximate integrals with probabilistic inference being a special case. The main idea behind IS for inference is that MC can still be performed when it not possible to sample from the distribution of interest. Consider the task of forming an estimator for integral I which is an expectation of some integrable test function $f(\boldsymbol{x})$ under a posterior distribution $\pi(\boldsymbol{x}) = p(\boldsymbol{x}, \mathcal{D})/p(\mathcal{D})$. The expectation can be easily rewritten by introducing an artificial density q called ”proposal” in the following way:

$$
\begin{aligned}
I = \mathbb{E}_\pi[f(\boldsymbol{x})] &= \int f(\boldsymbol{x})\pi(\boldsymbol{x})\mathrm{d}\boldsymbol{x} && (2.3)\\
&= \frac{1}{\int p(\boldsymbol{x}, \mathcal{D})\mathrm{d}\boldsymbol{x}} \int f(\boldsymbol{x}) \frac{p(\boldsymbol{x}, \mathcal{D})}{q(\boldsymbol{x})} q(\boldsymbol{x})\mathrm{d}\boldsymbol{x} && (2.4)\\
&= \frac{1}{\mathbb{E}_q\left[\frac{p(\boldsymbol{x},\mathcal{D})}{q(\boldsymbol{x})}\right]} \cdot \mathbb{E}_q\left[f(\boldsymbol{x})\frac{p(\boldsymbol{x}, \mathcal{D})}{q(\boldsymbol{x})}\right] && (2.5)
\end{aligned}
$$

$$
\left\{\boldsymbol{x}^{(n)}\right\}_{n=1}^{N} \sim q(\boldsymbol{x}) \qquad \approx \frac{1}{\cancel{1/N}\sum_{n=1}^{N} \frac{p\left(\boldsymbol{x}^{(n)},\mathcal{D}\right)}{q\left(\boldsymbol{x}^{(n)}\right)}} \cdot \cancel{1/N} \sum_{n=1}^{N} f\left(\boldsymbol{x}^{(n)}\right) \frac{p\left(\boldsymbol{x}^{(n)}, \mathcal{D}\right)}{q\left(\boldsymbol{x}^{(n)}\right)} \overset{\text{def}}{=} \widehat{I}_{SN} \tag{2.6}
$$

where samples are drawn i.i.d. The ratio $\frac{p\left(\boldsymbol{x}^{(n)},\mathcal{D}\right)}{q\left(\boldsymbol{x}^{(n)}\right)}$ is a key quantity in IS and is called *importance weight* $w(\boldsymbol{x}^{(n)})$. While these weights are a function of the sample $\boldsymbol{x}^{(n)}$ this will often be implicit as common in the literature. The usual interpretation for the weight assigned to sample $\boldsymbol{x}^{(n)}$ is that of an importance factor which takes into account that $\boldsymbol{x}^{(n)}$ was sampled from the ”wrong” distribution. Notice that in this estimator a

MC estimate is formed concurrently, *with the same set of samples*, for both the normalizing constant of π and the unnormalized expectation. Thanks to this property the estimator $\widehat{I}_{SN}$ is often referred to as *self-normalized.* It is possible to show that this self-normalized estimator is not unbiased, but it is at least consistent, and it is easy to see that the estimate of the normalising constant in unbiased. The fact that a useful estimator can be formed without knowledge of the normalizing constant is very appealing for Bayesian inference. Finally, a fact that will be used throughout is that weighted IS samples can be used to form an empirical measure as an approximation of the target posterior:

$$\pi(\boldsymbol{x}) \approx \sum_{n=1}^{N} w^{(n)} \delta_{\boldsymbol{x}^{(n)}}(\boldsymbol{x}) \tag{2.7}$$

2.1.2 Sequential Importance Sampling

In a context where the distribution of interest assumes some sort of sequential structure, applying IS "blindly" is not computationally efficient. In this thesis the focus is on sequential structure that arises from evolution over time, but the methods are more generally applicable. Consider the sequential estimation of a (posterior) probability distribution $\pi(\boldsymbol{x}_{1:t})$ where t is the time index, expressed as a function of its random variables $\boldsymbol{x}_{1:t}$. Without imposing any additional structure, applying IS implies using a distribution of the same form $q_t(\boldsymbol{x}_{1:t})$ which therefore proposes an entire trajectory of samples at each time step. Since computing importance weights in $\Theta(t)$ is undesirable, in practice one assumes an autoregressive structure on the proposal, which then takes the form:

$$q_t^{SIS}(\boldsymbol{x}_{1:t}) = q_{t-1}(\boldsymbol{x}_{1:t-1})\, q_t(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}) \tag{2.8}$$

so that proposing trajectory $\boldsymbol{x}_{1:t}^{(n)}$ amounts to sampling $\boldsymbol{x}_t^{(n)} \sim q_t(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})$ and simply append this to the previous trajectory assigned to the same index: $\boldsymbol{x}_{1:t}^{(n)} \leftarrow \boldsymbol{x}_{1:t-1}^{(n)} \cup \boldsymbol{x}_t^{(n)}$. The computation of importance weights also reduces in complexity, since they can now be easily expressed in terms of weights from the previous timestep and therefore computed incrementally:

$$\begin{aligned} w_t(\boldsymbol{x}_{1:t}) &= \frac{\pi_t(\boldsymbol{x}_{1:t})}{q_t(\boldsymbol{x}_{1:t})} && (2.9)\\ &= \frac{\pi_{t-1}(\boldsymbol{x}_{1:t-1})}{q_{t-1}(\boldsymbol{x}_{1:t-1})} \frac{\pi_t(\boldsymbol{x}_{1:t})}{\pi_{t-1}(\boldsymbol{x}_{1:t-1})\, q_t(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})} && (2.10)\\ &= w_{t-1}(\boldsymbol{x}_{1:t-1}) \cdot \frac{\pi_t(\boldsymbol{x}_{1:t})}{\pi_{t-1}(\boldsymbol{x}_{1:t-1})\, q_t(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})} && (2.11) \end{aligned}$$

where we can define the second term as $\varpi(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$, the *incremental importance weight* which can be computed in $\Theta(1)$. The method just described in what is known as Sequential Importance Sampling (SIS), and as we will see forms the basis of particle filtering.

2.2 State Space models and Filtering

State space models are probabilistic formulations that attempt to model the temporal evolution of a dynamical system. In this thesis, the focus will be on first-order Markov state space models [1]. In these models, a discrete-time multivariate stochastic process $\{\boldsymbol{x}_t\}_{t\geq 1}$ or *hidden state* is observed through a sequence of measurements $\{\boldsymbol{y}_t\}_{t\geq 1}$ and these stochastic processes are sampled from densities of the form:

$$\boldsymbol{x}_t \sim f_{\boldsymbol{\theta}}(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \tag{2.12}$$

$$\boldsymbol{y}_t \sim g_{\boldsymbol{\phi}}(\boldsymbol{y}_t \mid \boldsymbol{x}_t) \tag{2.13}$$

often referred to as *transition* density f and *observation* density g, with corresponding parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$. This thesis is concerned with inference, and it is assumed that parameters are fixed known quantities and they will therefore be omitted. For recent reviews of parameter estimation in state space models, see [42, 52]. There is also an implicit assumption of model correctness; for recent work on particle filtering in misspecified models, see [6].

Typical inference tasks that arise in state space models of this form are *filtering* and *smoothing*, respectively the sequential estimation of $p(\boldsymbol{x}_{1:t} \mid \boldsymbol{y}_{1:t})$ or $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$ and $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:T})$ with $T > t$. This thesis is focussed on filtering problems; a recent survey on smoothing techniques is [51].

The following are the two defining equations for the filtering task:

$$p(\boldsymbol{x}_{1:t} \mid \boldsymbol{y}_{1:t}) = p\left(\boldsymbol{x}_{1:t-1} \mid \boldsymbol{y}_{1:t-1}\right) \frac{f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})\, g(\boldsymbol{y}_t \mid \boldsymbol{x}_t)}{p\left(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}\right)} \tag{2.14}$$

$$p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) = \frac{g(\boldsymbol{y}_t \mid \boldsymbol{x}_t)\, p\left(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1}\right)}{p\left(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}\right)} \tag{2.15}$$

The latter distribution can be easily obtained from the former by marginalization of $\boldsymbol{x}_{1:t-1}$. While one is typically interested in inferring 2.15 which we name State Filtering

[1] Also known as *Hidden Markov Models*

Distribution (SFD), we will see that traditional particle filtering algorithms perform IS with 2.14 or Trajectory Filtering Distribution (TFD) and form estimates exploiting the MC marginalization property.

2.2.1 Particle Filtering

The MC methods used to perform approximate filtering in state space models are based on sequential importance sampling, and are generally known as Sequential Monte Carlo (SMC) or Particle Filters (PFs). These can be defined as a set of algorithms used to sequentially approximate a distribution of interested via weighted samples or "particles". A first attempt at such an algorithm could be using standard SIS to approximate filtering distributions, setting $\pi_t(\boldsymbol{x}_{1:t})$ to be $p(\boldsymbol{x}_{1:t} \mid \boldsymbol{y}_{1:t})$ or $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$; the former is the standard SMC choice. However, performing IS on a space of increasing dimension t has additional consequences that unlike computational complexity are not fixed by vanilla SIS. Indeed, it can be easily shown that the variance of importance weights increases exponentially with t for simple models where the filtering distribution (as well as the IS proposal) factorize [21, 59]. This fundamental problem manifests itself, among other ways, through a phenomenon known as *weight degeneracy*: after a few steps, all but one importance weight will be ≈ 0, which implies the posterior is approximated using a single sample. To alleviate this issue one can introduce a *resampling* step in the SIS algorithm: before proposing new particles at time t, the previous (weighted) set of particles gets replaced by a new set of unweighted particles, where these have been sampled (usually with replacement) with probability proportional to w_{t-1}. Intuitively, this allows to eliminate particles with very low weight with high probability, which would bring little contribution to posterior estimates. Additionally, resampling can also nicely be interpreted as a smart choice of proposal, where rather than the vanilla SIS proposal 2.8, one uses:

$$q_t^{SMC}(\boldsymbol{x}_{1:t}) = \widehat{p}_{t-1}\left(\boldsymbol{x}_{1:t-1} \mid \boldsymbol{y}_{1:t-1}\right) q_t\left(\boldsymbol{x}_t \middle| \boldsymbol{x}_{1:t-1}, \boldsymbol{y}_{1:t}\right) \tag{2.16}$$

(where the added conditioning on on $\boldsymbol{y}_{1:t-1}$ and $\boldsymbol{y}_{1:t}$ respectively comes from the context of state space models). In 2.16, the first term is the particle approximation of the target distribution at $t-1$, namely $\sum_{m=1}^{M} w_{t-1}^{(m)} \delta_{\boldsymbol{x}_{t-1}^{(m)}}(\boldsymbol{x}_{t-1})$. Sampling from q_t^{SMC} can be achieved by first sampling from $\widehat{p}_{t-1}\left(\boldsymbol{x}_{1:t-1} \mid \boldsymbol{y}_{1:t-1}\right)$ and since this is a mixture it can be achieved by resampling $\boldsymbol{x}_{t-1}^{(m)}$ with probabilities $w_{t-1}^{(m)}$; the resulting samples from this operation can be used then to sample from the second term in 2.16.

Evaluating PFs can be challenging: in Linear Gaussian systems one can directly compare the approximate distributions with that given by the KF; for other models this is not possible, and so one has to resort to approximate means. Generally, one would be interested in minimizing the theoretical (conditional on previous particles and current measurement) variance of the importance weights. It is easy to show that the proposal which minimizes this objective is $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{y}_t)$, often referred to as the ”optimal SMC proposal” or ”optimal SMC kernel” [20]. Of course, computing this proposal and sampling from are intractable tasks. The Effective Sample Size (ESS) has been established in IS and SMC as a measure of the theoretical number of independent samples that have been generated from the target distribution [46]. Since this is also intractable, it is most commonly approximated by $1/\boldsymbol{w}^\top \boldsymbol{w}$ where $\boldsymbol{w}$ are the *normalized* IS weights. Issues with this metric have been highlighted in [53].

2.2.2 The Auxiliary Particle Filter

The Auxiliary Particle Filter (APF) that was originally presented in [65] is a variation of the standard BPF. In this Section, we try to merge several perspectives from the literature, including the original one by Pitt and Shepard.

More generally, the APF can be thought of a class of PF algorithms which attempt to incorporate measurement information into the resampling step. Indeed, a standard PF algorithm such as the BPF performs resampling at the end of iteration $t-1$, and thus resampling strategies use importance weights from $t-1$. However, one can equivalently resample at the beginning of iteration t. Since it is reasonable to assume that iteration t starts as the t-th measurement becomes available, this information could be used to perform resampling of the particles from the previous iteration. This allows the set of particles to move into a region of higher posterior probability earlier in the algorithm. It is then intuitive that APF algorithms will generally tend to perform more accurate approximations with highly informative likelihoods.

Another way to see what the APF does, is to interpret it as a change to the SMC proposal, where now the resampling part of the proposal is not the previous particle approximation, but some arbitrary distribution that incorporates the measurement:

$$q_t^{APF}(\boldsymbol{x}_{1:t}) = q_t(\boldsymbol{x}_{1:t-1} \mid \boldsymbol{y}_{1:t}) \cdot q_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{1:t-1}, \boldsymbol{y}_{1:t}) \tag{2.17}$$

This equation mirrors the SMC proposal in 2.16, but allows the left part to depend on $\boldsymbol{y}_t$, not restricting it to be the previous particle approximation. Algorithm 1 below shows details for the procedure for a generic particle filter.

Notice the ratio w_{t-1}/λ_t that is multiplied by the vanilla SMC IS ratio. When $\lambda_t = w_{t-1}$ (i.e. particles are resampled according to w_{t-1}) we recover vanilla SMC algorithms; the more general case represents the class of APF algorithms. The role of this ratio was already explored in e.g. [27, 11]. A perhaps subtle point is that one should not use the resampled particles $\boldsymbol{x}_{t-1}^{i^{(m)}}$ in the computation of the IS weights, but simply $\boldsymbol{x}_{t-1}^{(m)}$ in order to reduce the variance of the weights [12] [2]. There are several free parameters that one can select in order to obtain concrete algorithms: the choice of λ_t, the choice of proposal $q_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{y}_{1:t})$, the choice of resampling strategy (multinomial, strafitied, etc.) . For example, by choosing the proposal for $\boldsymbol{x}_t$ to be $f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$, and setting $\lambda_t = w_{t-1}$ one obtains the *Bootstrap* PF [28]; the original APF chooses $\lambda_t = w_{t-1} \cdot p(\boldsymbol{y}_t \mid \boldsymbol{x}_{t-1})$ and $f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1})$. This is motivated by an ”optimal” choice which simply sets simulation weights equal to their true value under the probabilistic dynamics of the state space model, namely $\lambda_t = p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{1:t})$. If one were to select the optimal simulation weights, and at the same time be able to propagate particles using the optimal SMC kernel, the resulting algorithm is often referred to as fully adapted APF [36, 21, 72]. Interestingly, it is not always the case that this theoretical algorithm *always* performs better than even the BPF [36].

Algorithm 1: Generic Particle Filter

1. Draw M samples from prior: $\boldsymbol{x}_1^{(m)} \sim p(\boldsymbol{x}_1)$ and set $w_1^{(m)} = 1/M$ for all m
2. For each $t = 2 \dots T$:
   - *Resampling*: From set of weighted particles $\left\{\boldsymbol{x}_{t-1}^{(m)}, w_{t-1}^{(m)}\right\}$, resample using simulation weights $q_t(\boldsymbol{x}_{t-1}^{(m)} \mid \boldsymbol{y}_{1:t}) \overset{\text{def}}{=} \lambda_t^{(m)}$ to obtain resampled indices $i^{(m)}$ for $m = 1 \dots M$
   - Draw new particles from the proposal: $\boldsymbol{x}_t^{(m)} \sim q_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{i^{(m)}}, \boldsymbol{y}_{1:t})$ using resampled indices
   - Calculate new importance weights:

$$\boldsymbol{w}_t^{(m)} = \frac{p(\boldsymbol{x}_{1:t}^{(m)} \mid \boldsymbol{y}_{1:t})}{q_t(\boldsymbol{x}_{1:t}^{(m)} \mid \boldsymbol{y}_{1:t})} \propto \frac{w_{t-1}^{(m)}}{\lambda_t^{(m)}} \cdot \frac{g(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(m)}) f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(m)})}{q_t(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(m)}, \boldsymbol{y}_{1:t})} \tag{2.18}$$

[2] resampled particles are then just used for propagation

2.2.3 M^2 Particle Filtering

We have already hinted in Section 2.2.1 that performing IS with target $p(\boldsymbol{x}_{1:t} \mid \boldsymbol{y}_{1:t})$ causes problems. A more theoretically elegant solution than explicitly introduce resampling was proposed in Klaas et al. [45]: in their framework of Marginal Particle Filters (MPF), the SFD is used as target distribution, and the marginalization of $\boldsymbol{x}_{t-1}$ in the IS numerator is used to motivate the use of a mixture proposal to be used in the IS denominator, where the weights are the APF simulation weights (forming what they call Auxiliary Marginal Particle Filter (AMPF)), since sampling from a mixture is equivalent to resampling followed by propagation from the mixture kernels. This obviously requires a $O(M^2)$ cost for the computation of the IS weights, but Klaas et al. point out that standard N-body methods can be used to reduce the cost to $O(M \log M)$ for a controllable approximation error. The same proposal was reintroduced from the perspective of Multiple Importance Sampling (MIS) by Elvira et al. [23, 24]: when simulating from multiple kernels, it is better in terms of variance of estimators to use the full mixture in the IS weight (see [25] for more precise results). This perspective allowed to make a better selection for the simulation weights which takes into account whether transition kernels (used for propagation of particles in [23, 24]) overlap significantly: when they do, APF can perform poorly, and their novel choice of λ improves the situation.

2.3 Related Work

There have been several attempts at improving the accuracy of the estimates from vanilla SMC algorithms beyond the original APF. Early work by Godsill et al. [27] discusses the role of the ratio w_{t-1}/λ_{t-1} (with different notation) , as well using MCMC to sample from the optimal SMC proposal. The work by Klaas et al. [45] already described in the previous Section showed promise in using a mixture proposal and makes it clear what a reasonable choice for both simulation weights and kernels should be when targeting the SFD (we expand on this in Section 3.1). An interesting paper by Kronander and Schön [47] follows the idea that likelihood information should be used to propagate the particle, and form a linear time PF by combining this insight with MIS guarantees for the choice of importance weight. Guarniero et al. develop offline methods for the estimation of λ's via a parameterized approach [29, 30] in a similar manner to this work (the main difference being their offline focus). They propose a generic framework called *twisted-auxiliary* SMC: the simulation weights in APF can be seen as a special case of a positive function named *twisting potential* within the framework. Twisting potentials are seen as free parameters that should be optimized according to some global objective function that represents the ultimate goal of inference. Their work builds on a line of research that to the best of our knowledge originated in [36, 21], where the APF is interpreted as a standard SMC algorithm that estimates a different target distribution. For an accessible treatment of this line of research, see this survey by Naesseth et al. [59]. These works have been recently extended by Heng et al. [31].

Other interesting works that optimize a proposal in an online fashion include Cornebise et al. [16], who adapt a mixture of experts proposal in order to approximate the optimal SMC kernel. Akyildiz et al. [2] take an approach that could be complementary to APF: they move the particles in regions of high likelihood preemptively via gradient methods.

The excellent report by Whiteley and Johansen [72] describes and summarizes a larger range of improval strategies for the APF covering parameter estimation and use of MCMC moves, as well as providing an insightful treatment of its asymptotic variance and comparisons with vanilla SMC.

Chapter 3

Methodology

In this Section, we go through the reasoning behind the development of OAPF, a novel, online class of PF algorithms that find simulation weights by solving an optimization problem, and uses the same weighting strategy as M^2 algorithms such as Improved Auxiliary Particle Filter (IAPF) [23] and AMPF [45]. Furthermore, we investigate the potential for highly reducing the computational cost of the optimization problem with a concrete example, and suggest a simple strategy that will prove accurate in the experiments (Chapter 4).

We start by motivating the use of a mixture-type proposal, building on the concepts explained in Sections 2.2.3: the use of a mixture of kernels proposal, which is a linear combination of the simulation weights, will be the key ingredient that enables the development of a *convex* optimization problem. Two distinct concrete optimization strategies are proposed in Sections 3.2.2 and 3.2.1, and for both a thorough review of the existing algorithms is provided. Finally, in Section 3.2.3 we suggest a simple way to control the computational cost and, as will be shown in the experiments, increase the estimation accuracy as well when using a large number of particles. A note on analysing runtime complexity: in this work, we ignore complexity terms that come from the dimensionality of the state space [1].

[1]For example, evaluation of a Gaussian probability density function takes D^3 where D is the dimensionality of the random vector, because of the inversion of the covariance matrix. Cholesky decompositions and forward-backward substitutions are used in practice, but these don't change the asymptotic complexity.

3.1 Selecting a mixture proposal

We have briefly mentioned in the previous Chapter how already in [45] clear benefits were shown from giving up on the linear-time (in the number of particles) cost for an online SMC algorithm that is interested in estimating $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$. This paper also suggests between the lines what the form of a ”optimal” mixture proposal could look like. Consider the usual PF approximation of the SFD:

$$p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) \approx g(\boldsymbol{y}_t \mid \boldsymbol{x}_t) \sum_{k=1}^{M} w_{t-1}^{(k)} f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(k)}) \tag{3.1}$$

which makes use of the M particle approximation of the same distribution at $t-1$. Now by considering the form of the one step optimal SMC proposal $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{y}_t) = \frac{g(\boldsymbol{y}_t|\boldsymbol{x}_t) f(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})}{p(\boldsymbol{y}_t|\boldsymbol{x}_{t-1})}$, one can equivalently express 3.1 as:

$$p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) \approx \sum_{k=1}^{M} w_{t-1}^{(k)} p(\boldsymbol{y}_t \mid \boldsymbol{x}_{t-1}^{(k)}) p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(k)}, \boldsymbol{y}_t) \tag{3.2}$$

This suggests that if our proposal is a weighted mixture of kernels, in order to match the posterior as closely as possible, then it should satisfy:

- The weights should be as close as possible to $w_{t-1}^{(k)} p(\boldsymbol{y}_t \mid \boldsymbol{x}_{t-1}^{(k)})$. These correspond to the optimal choice for the simulation weights λ of the APF.
- The kernels used to propagate the particles should be as close as possible to $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(k)}, \boldsymbol{y}_t)$

These are the same conditions for full adaptation of the APF. However, as mentioned in Chapter 2, Johansen et al. [36, 72] show that somewhat counter-intuitively [2] even a fully adapted APF does not necessarily outperform BPF in all situations. For the APF case, some light had been shed in [24] from the perspective of kernels overlap. It does not seem clear whether a fully adapted version of the AMPF (Auxiliary Marginal Particle Filter), which essentially just uses full mixtures in the IS weight computations, would outperform other PFs in all scenarios [3]. As we will see soon, in this work we fix the kernels used to propagate the particles to be the transition kernels, and all the work into matching the mixture proposal closely will go into efficient amplification via the mixture weights.
Elvira et al. showed that using a full mixture is beneficial not only for the IS weight

[2]Especially considering the previous line of reasoning of matching the mixture posterior
[3]although it would likely not

computation, but also for the selection of simulation weights: this led to an improved APF (IAPF), with a better choice of λ [23]. The main question is then what could be an "optimal" choice for these "simulation" weights λ, so that the resulting distribution achieves minimum variance of the importance weights. The APF and IAPF make explicit, analytic choices for this weight. Previously in this Section, we have seen that the APF choice can be interpreted as trying to optimally match the weights of the approximate mixture posterior. APF's choice can also be intepreted as a direct approximation of the (smoothing) distribution $p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{1:t})$ [4] evaluated at each particle, i.e. $p(\boldsymbol{x}_{t-1}^{(m)} \mid \boldsymbol{y}_{1:t})$, where the m-th kernel is approximated with a Diract delta mass at its center :

$$p(\boldsymbol{x}_{t-1}^{(m)} \mid \boldsymbol{y}_{1:t}) = \overbrace{p(\boldsymbol{x}_{t-1}^{(m)} \mid \boldsymbol{y}_{1:t-1})}^{w_{t-1}^{(m)}} \int \frac{f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(m)})p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})}{p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1})} \mathrm{d}\boldsymbol{x}_t \tag{3.3}$$

$$\approx \ \propto w_{t-1}^{(m)} \cdot \int \delta_{\boldsymbol{\mu}_t^{(m)}}(\boldsymbol{x}_t) g(\boldsymbol{y}_t \mid \boldsymbol{x}_t) \mathrm{d}\boldsymbol{x}_t \tag{3.4}$$

$$= w_{t-1}^{(m)} \cdot g(\boldsymbol{y}_t \mid \boldsymbol{\mu}_t^{(m)}) \tag{3.5}$$

where notice that 3.4 performs both an approximation (by replacing $f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{(m)})$ with $\delta_{\boldsymbol{\mu}_t^{(m)}}$) as well as the exclusion of a proportionality constant (by replacing $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})/p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1})$ with $g(\boldsymbol{y}_t \mid \boldsymbol{x}_t)$). Moreover, it is also interesting to observe that, if particles were resampled exactly from $p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{1:t})$, then the resulting new set of particles would be automatically distibuted according to $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$.
On the other hand, IAPF chooses λ_t by looking at transition kernels overlap. It is difficult to intepret the IAPF choice as some approximation of $p(\boldsymbol{x}_{t-1}^{(m)} \mid \boldsymbol{y}_{1:t})$, but such an intepretation would be appealing since it would preserve a derivation of the simulation weights follows by purely applying the rules of probability while making some approximation.
In this work, we observe that it would be desirable to make a choice of simulation weights that is more flexible than choosing an analytic form: we will see how to do so in the next Section.

3.2 Optimization of simulation weights

Recent work by Zhao et al. [74] has started to attempt to fit a distribution to the true filtering posterior using optimization in an online fashion, in this case with variational

[4]This fact was already explicited in [27]

methods. However, Zhao et al. use the *inclusive* [5] Kullback-Leibler (KL) divergence typical of VI for machine learning applications: this direction of KL is well known to strongly underestimate the variance of the true posterior (a phenomenon sometimes referred to as *zero-forcing* [57]), which is undesirable in a PF setting. In this thesis, we will propose a different optimization procedure that adapts a proposal to the target posterior $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$, which also addresses the desiderata expressed in the previous Section. More specifically, while we fix the functional form of the proposal to be a weighted mixture of kernels, we optimize the simulation weights by minimizing a convex objective derived by imposing that the proposal should match the posterior at a set of locations. These locations are not randomly chosen : they should exploit knowledge given by the particle approximation at $t-1$, $\left\{w_{t-1}^{(m)}, \boldsymbol{x}_{t-1}^{(m)}\right\}_{m=1}^{M}$.
With this spirit, we can think of imposing a constraint for each particle, forcing the proposal to closely match the posterior at some representative point associated to that particle. For example, we could impose strict equality between proposal and posterior at locations $\left\{\boldsymbol{z}^{(m)}\right\}_{m=1}^{M}$:

$$\sum_{k=1}^{M} \lambda_t^{(k)} f(\boldsymbol{z}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(k)}) = g(\boldsymbol{y}_t \mid \boldsymbol{z}_t^{(m)}) \sum_{k=1}^{M} w_{t-1}^{(k)} f(\boldsymbol{z}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(k)}) \qquad \text{for } m = 1, \ldots, M \quad (3.6)$$

These points could be chosen to be the means of the transition kernels. In matrix notation, letting $\boldsymbol{\lambda} = \left(\lambda_t^{(1)}, \lambda_t^{(2)}, \ldots, \lambda_t^{(M)}\right)^\top$, $\boldsymbol{w} = \left(w_{t-1}^{(1)}, w_{t-1}^{(2)}, \ldots w_{t-1}^{(M)}\right)^\top$, and $\boldsymbol{f}^{(m)} = \left(f(\boldsymbol{z}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(1)}), f(\boldsymbol{z}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(2)}), \ldots, f(\boldsymbol{z}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(M)})\right)^\top$, the constraints become :

$$\boldsymbol{\lambda}^\top \boldsymbol{f}^{(m)} = \underbrace{g(\boldsymbol{y}_t \mid \boldsymbol{z}_t^{(m)}) \odot \boldsymbol{w}^\top \boldsymbol{f}^{(m)}}_{\tilde{\boldsymbol{\pi}}^{(m)}} \qquad \text{for } m = 1, \ldots, M \qquad (3.7)$$

Where $\odot$ is elementwise multiplication, in this case between a scalar and a vector. More compactly, we can express 3.7 as :

$$\overbrace{\begin{bmatrix} — & \boldsymbol{f}^{(1)^\top} & — \\ \vdots & \vdots & \vdots \\ — & \boldsymbol{f}^{(M)^\top} & — \end{bmatrix}}^{\boldsymbol{F}} \begin{bmatrix} | \\ \boldsymbol{\lambda} \\ | \end{bmatrix} = \begin{bmatrix} — & g(\boldsymbol{y}_t \mid \boldsymbol{z}_t^{(1)}) \odot \boldsymbol{f}^{(1)^\top} & — \\ \vdots & \vdots & \vdots \\ — & g(\boldsymbol{y}_t \mid \boldsymbol{z}_t^{(M)}) \odot \boldsymbol{f}^{(M)^\top} & — \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{w} \\ | \end{bmatrix}$$

so that :

$$\boldsymbol{F}\boldsymbol{\lambda} = \boldsymbol{g} \odot \boldsymbol{F}\boldsymbol{w} \qquad (3.8)$$

[5]That is, $\mathrm{KL}(q \mid\mid p)$ where q is the proposal, p the true posterior.

A solution for $\boldsymbol{\lambda}$ could be found by (letting $\boldsymbol{g} \odot \boldsymbol{F}\boldsymbol{w} := \widetilde{\boldsymbol{\pi}}$):

$$\boldsymbol{\lambda} = \boldsymbol{F}^{-1}\widetilde{\boldsymbol{\pi}} \tag{3.9}$$

Of course, this linear system of M equations does not need to have a unique solution (or one at all) in general. Moreover, $\boldsymbol{\lambda}$ need to be used for the ”delayed” resampling of particles, and therefore need to be nonnegative [6]. We can thus turn this into a constrained optimization problem. Two distinct strategies will be presented in the following subsections.

3.2.1 Optimization via Linear Programming

One possible approach is to encode this problem as the following Linear Program (LP):

$$\begin{array}{cc} \text{Minimize } \sum_m \xi_m & \text{w.r.t } \boldsymbol{\lambda}, \boldsymbol{\xi} \\ \boldsymbol{\lambda}^\top \boldsymbol{f}^{(m)} = \widetilde{\boldsymbol{\pi}}^{(m)} + \xi_m & \text{for } m = 1, \ldots, M \\ \multicolumn{2}{c}{\text{Subject to: } \boldsymbol{\lambda}, \boldsymbol{\xi} \geq 0} \end{array}$$

with $2M$ variables and $3M$ constraints. We have turned each of the constraints from 3.7 into a relaxed version with added residuals $\boldsymbol{\xi}$ of the same dimension of $\boldsymbol{\lambda}$. This LP jointly optimizes for $\boldsymbol{\lambda}$ and for residuals $\boldsymbol{\xi}$. Linear Programming problems are extremely well studied, and there are two main class of algorithms that are used to solve them: interior point methods and Simplex variants [15, 7].
The worst case case complexity of the Simplex algorithm is exponential in the number of constraints [15], however this almost never arises in practice, where the complexity is approximately $O(n_{\text{var}} \cdot n^2_{\text{constr}})$ [7] (that is, in our case, $O(M^3)$). Interior point methods may generally be less accurate than Simplex, but their worst case runtime is approximately $O(M^3)$[7].

3.2.2 Optimization via Non-Negative Least Squares (NNLS)

This constrained optimization problem can be approached as a Non-Negative Least Squares (NNLS) problem:

$$\begin{array}{cc} \text{Minimize } \left\|\boldsymbol{F}\boldsymbol{\lambda} - \widetilde{\boldsymbol{\pi}}\right\|_2 & \text{w.r.t } \boldsymbol{\lambda} \\ \multicolumn{2}{c}{\text{Subject to: } \boldsymbol{\lambda} \geq 0} \end{array}$$

[6]One can normalize them afterwards. Of course, they could not *all* be 0 either.

of the typical form $||\boldsymbol{Ax}-\boldsymbol{b}||_2^2$, which can be more explicitly simplified and reformulated as the following Quadratic Program (QP):

$$\boxed{\begin{array}{c}\text{Minimize } \frac{1}{2}\boldsymbol{\lambda}^\top \boldsymbol{F}^\top \boldsymbol{F}\boldsymbol{\lambda} - \boldsymbol{F}^\top \widetilde{\boldsymbol{\pi}} \\ \text{Subject to: } \boldsymbol{\lambda} \geq 0\end{array}}$$

This problem is convex so that a unique solution exists, and can be found by a plethora of constrained convex optimization routines. Firstly, note that a widely used *ad-hoc* algorithm based on active-set methods exists by Lawson et al. [48]. This algorithm has been extended into a faster version [8]. Note also that there exist fast randomized and approximate algorithms for constrained least squares problems based on Iterative Hessian Sketching (IHS), for which a survey is given in [64]. These algorithms iteratively refine a solution by sampling random matrices [7] $\boldsymbol{S} \in \mathbb{R}^{K\times M}$ (with $K << M$, M being the number of constraints in our setting) and following the update rule:

$$\boldsymbol{\lambda}^{(l+1)} = \arg\min_{\boldsymbol{\lambda}\in\mathcal{C}} \frac{1}{2}\left\|\boldsymbol{S}^{(l+1)}\boldsymbol{A}(\boldsymbol{\lambda}-\boldsymbol{\lambda}^{(l)})\right\|_2^2 - \left(\boldsymbol{A}^\top(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{\lambda}^{(l)})\right)^\top \left(\boldsymbol{\lambda}-\boldsymbol{\lambda}^{(l)}\right) \tag{3.10}$$

where l denotes the iteration index [8], $\mathcal{C}$ the convex constraint region (in our case, it would be the space of positive real vectors) and for our setting $\boldsymbol{A} = \boldsymbol{F}^\top\boldsymbol{F}, \boldsymbol{b} = \widetilde{\boldsymbol{\pi}}$. IHS algorithms are thus relatively simple to implement, have strong theoretical guarantees of approximation, and importantly reduce the runtime to $O(\log 1/\varepsilon)\cdot O(KM^2)$ where ε is related to the closeness in approximation, so that their application to our optimization problem could be interesting further work. Finally, note that very recent work on exact NNLS [58] provides fast and sparse solutions, which are useful for reducing the complexity in the weighting step. The usefulness of a sparse solution will become more apparent once we discuss strategies to reduce the runtime of the optimization problem significantly.

The optimization problem we presented defines a class of algorithms (since Optimization strategy , selection of evaluation points and other choices such as kernels used to propagate the particles remain as free parameters) which we name Optimized Auxiliary Particle Filter (OAPF). This is described in full detail below in Algorithm 2. Compared with the generic PF Algorithm 1, it has the MPF or equivalently IAPF $O(M^2)$ importance weight computation.

[7] for details on how these need to be chosen, see [64]

[8] note that this is the iteration index within the same SMC iteration, to optimize the simulation weights

Algorithm 2: Optimized Auxiliary Particle Filter

```
1. Draw $M$ samples from prior: $\boldsymbol{x}_1^{(m)} \sim p(\boldsymbol{x}_1)$ and set $w_1^{(m)} = 1/M$ for all $m$
2. For each $t = 2 \dots T$:
    • *Selection*: select points where to evaluate the posterior $\left\{\boldsymbol{z}^{(m)}\right\}_{m=1}^{M}$
    • ***Optimization***: compute simulation weights $\boldsymbol{\lambda}$ by optimizing proposal $\boldsymbol{\lambda}^\top \boldsymbol{f}^{(m)}$ to be close to $\widetilde{\boldsymbol{\pi}}^{(m)}$ for all $m$, subject to $\boldsymbol{\lambda} \geq 0$
    • *Propagation*: sample $\boldsymbol{x}_t^{(m)}$ from proposal $\boldsymbol{\lambda}^\top \boldsymbol{f}$ in two steps:
        – *Resampling*: From set of weighted particles $\left\{\boldsymbol{x}_{t-1}^{(m)}, w_{t-1}^{(m)}\right\}$, resample using $\lambda_t^{(m)}$ to obtain resampled indices $i^{(m)}$ for $m = 1 \dots M$
        – Draw new particles from the proposal: $\boldsymbol{x}_t^{(m)} \sim q_t(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}^{i^{(m)}}, \boldsymbol{y}_{1:t})$ using resampled indices
    • *Weighting*: calculate new importance weights:
        $$\boldsymbol{w}_t^{(m)} = \frac{p(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{y}_{1:t})}{q_t(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{y}_{1:t})} \propto \frac{g(\boldsymbol{y}_t \mid \boldsymbol{x}_t^{(m)}) \sum_{k=1}^{K} w_{t-1}^{(k)} f(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(k)})}{\sum_{k=1}^{K} \boldsymbol{\lambda}_t^{(k)} f(\boldsymbol{x}_t^{(m)} \mid \boldsymbol{x}_{t-1}^{(k)})} \quad (3.11)$$
```

3.2.3 Reducing time complexity

The bottleneck in time complexity lies in the *optimization* stage of OAPF: both strategies previously described take approximately $O(M^3)$ steps. For large M ($10^3, 10^4$ etc.), not only will the algorithm be slow, but as experiments confirm (more on this in Chapter 4), numerical optimization issues arise that negatively impact performance. In Figure 3.1, a one dimensional example shows how it is likely unnecessary to include *all* evaluation points in the optimization problem. By considering only a subset $K << M$, the matrix $\boldsymbol{F}$ gets reduced from $M \times M$ to $K \times K$, and the target vector $\widetilde{\boldsymbol{\pi}}$ from $M \times 1$ to $K \times 1$, giving a runtime of $O(K^3)$. For small enough K (i.e. such that $K^3 < M^2$), the bottleneck in the algorithm becomes the weighting step, which as discussed in Section 2.2.3 can be reduced to $O(M \log M)$ with accurate and controllable approximations. Furthermore, note that by only solving the optimization problem for K simulation weights, the other are automatically set to 0: when $K << M$, then $\boldsymbol{\lambda}$ will

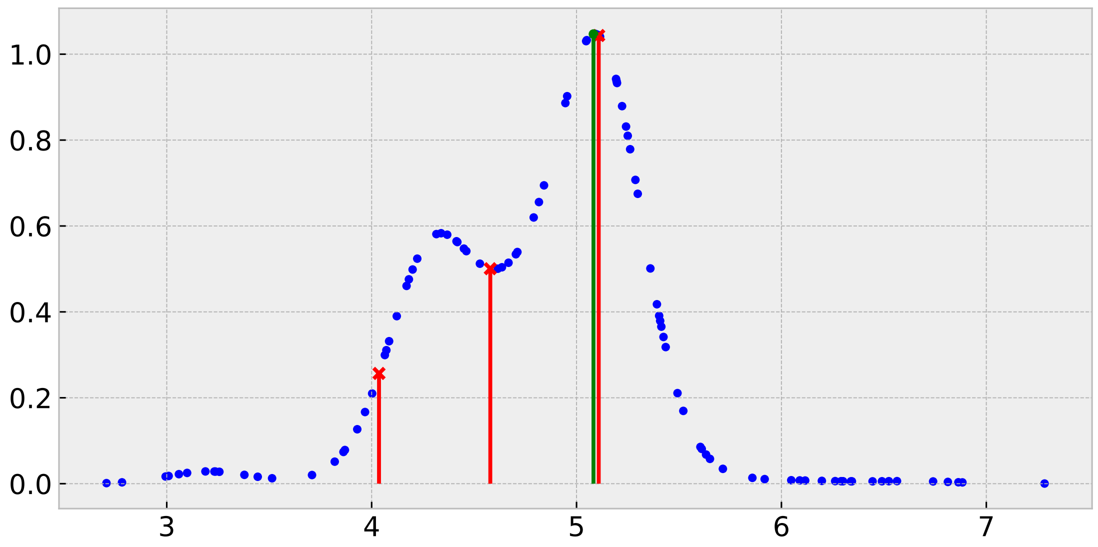


Figure 3.1: In blue, evaluations of a posterior generated by multiplying a mixture of 100 Gaussian kernels with a Gaussian likelihood. Building the OAPF proposal would take approximately 100^3 computation steps. However, one can see that it seems highly wasteful to adapt the mixture of Gaussian kernels proposal by evaluating at all blue points: e.g., if the proposal matches the posterior at the rightmost red point, it will probably go through the green point as well, so we don't need to include that point in the optimization. Moreover, it may also be wasteful to match proposal and target at points where the target has little probability mass. From these considerations, one may consider that trying to only match the three highlighted red points (and perhaps a couple more), would likely result in a proposal that is closely as good as the one we would get by using all blue points.

be very sparse by construction, and efficient matrix-vector operations can be employed in the weighting step, as an alternative to N-body approximations used in [45].
The main question to be asked is how to select the K evaluation points. For simplicity, consider only points that form a subset of the M evaluation points. Following the reasoning given in Figure 3.1, we could look at entries in $\widetilde{\boldsymbol{\pi}}$ and only keep the K highest entries, then eliminating the corresponding rows and columns from $\boldsymbol{F}$. This extremely simple strategy works surprisingly well even with a very small K, as will be discussed in Chapter 4. The potential for reduction of size in the optimization problem was experimentally noticed by solving instances using 1000 or more particles with OAPF: these return extremely sparse solution vectors $\boldsymbol{\lambda}$ (about 90 percent sparsity), which indeed suggests that most of them are unnecessary.

Chapter 4

Evaluation & Experiments

In this Chapter, the purpose is to evaluate the performance of OAPF compared to traditional algorithms (BPF,APF) but also the recent IAPF [23]. We design three sets of experiments, each with its own Section. All experiments are coded in Python, making use of the popular libraries NumPy [61], SciPy [37], Matplotlib [34] as well as PyTorch [63] for more efficient vectorization [1]. For Kalman Filter implementation, we use the open source library PyKalman [2]. The latest version of project code is available at https://github.com/nicola144/auxiliary-particle-filters. Note that some Figures may refer to OAPF as ”NPF”[3] as when they were produced the algorithm was still unnamed.

4.1 Matching a Mixture of Gaussians

In this first experiment, the main goal is to visually inspect how the proposal that results from the optimizatoin problem we have developed looks like. This is directly based on the experiment in Elvira et al. (2019) [24].
Consider examining a single iteration of a PF algorithm; for the sake of this experiment, suppose then we have $M = 4$ particles available from the previous iteration, for which we will arbitrarily select importance weights and values of the particles. These 4 particles provide 4 transition kernels which we will select to be univariate Gaussians with equal variance σ^2_{kern}, and means $\{\boldsymbol{x}_{t-1}\}_{m=1}^{M=4}$. The likelihood will also be Gaussian with variance σ^2_{lik}. With these parameters, the mixture of 4 kernels multiplied by the

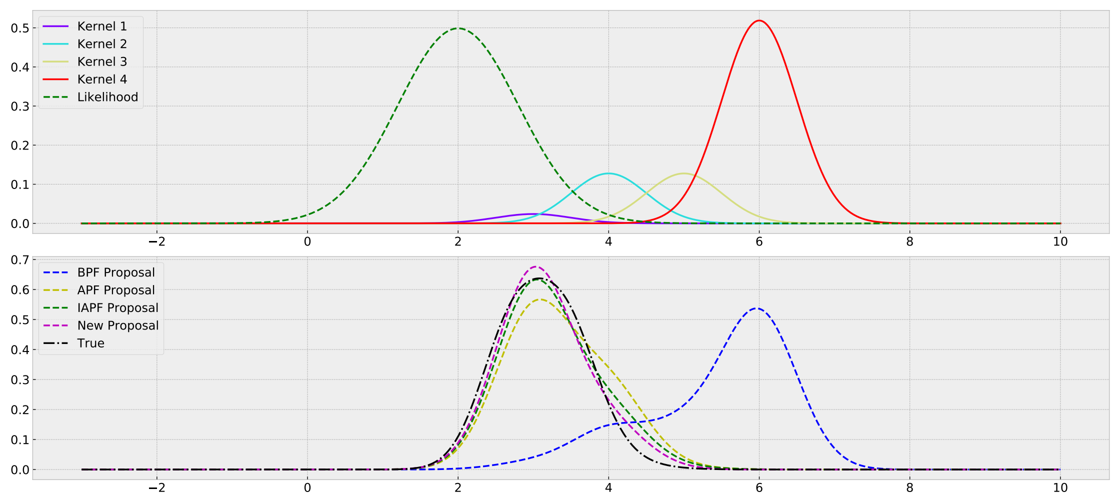
[1] we found that using PyTorch reduced computation times for building $\boldsymbol{F}$ from minutes to seconds compared to vectorization in NumPy.
[2] https://pykalman.github.io/
[3] New Particle Filter

likelihood can be obtained forming the true posterior, and it can be normalized with high precision with numerical integration methods such as Simpson's rule. We will compare the mixture proposals given by BPF, APF, IAPF [23], and our novel OAPF. This can be done visually, but also calculating the χ^2 distance from each proposal to the posterior[4], since this measure is proportional to the asymptotic variance of the (normalized) IS estimator of the normalizing constant [5].

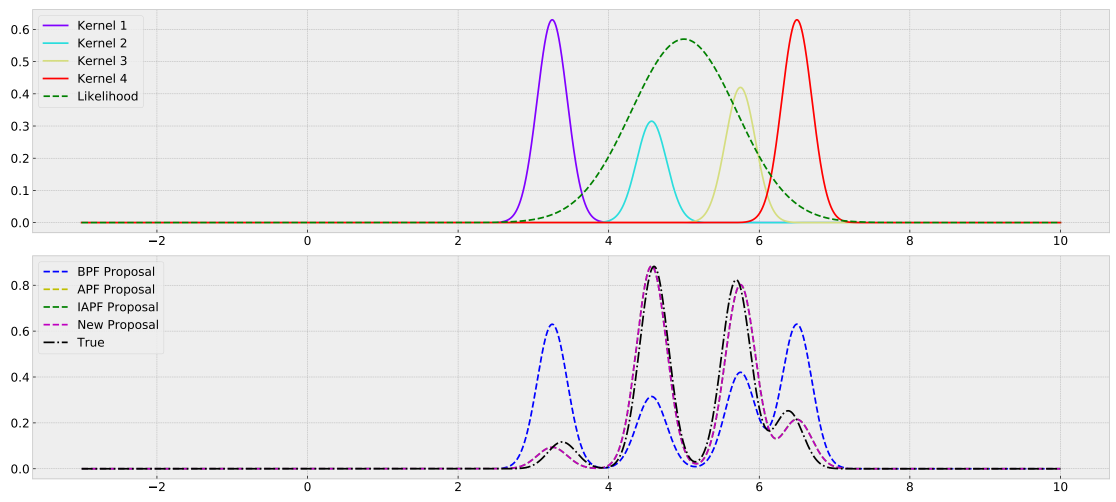
Results from two experiments with the above setting are shown in Figure 4.2. For all experiments we are reporting, simulation weights are optimized via NNLS; in one-dimensional settings considered in this Section, we did not find meaningful differences between the solutions given by the two different optimization strategies proposed in 3.2. Moreover, we select transition kernel means as evaluation points $\left\{z^{(m)}\right\}_{m=1}^{M}$. In Figure 4.1a , the same parameters are used as in Elvira et al. [24]: in this example, the original paper demonstrated the features of IAPF, which improves over BPF and APF in a scenario with informative likelihood *and* overlapping transition kernels. In this scenario we found, testing additional similar settings, that our novel PF is always at least as good as IAPF, and very often at least slightly better. In Figure 4.1b we complement 4.1a with a setting where there is still an informative likelihood (thus favourable for APF) , but now kernels have little overlap. As one would expect, the APF performs greatly in this scenario, and in fact all algorithms except BPF perform practically the same. Indeed, the proposals from APF, IAPF anad OAPF in this case are indistinguishable from the Figure given how close they are. We found this behaviour to be consistent across similar examples.

[4]this can also be calculated using efficient numerical integration

[5]In SMC we use Self-Normalized estimators so this is not exactly optimal, but it is still a good indicator



(a) This example is reproduced exactly from Elvira et al. [24]. Here, transition kernels are scaled by their importance weight w_{t-1}. In the second plot, proposals and true posterior integrate to 1. OAPF (slightly) improves on IAPF by putting more probability mass on the highest region of posterior probability. In this example, $\sigma_{\text{lik}} = 0.8, \sigma_{\text{kern}} = 0.5$, $\{\boldsymbol{x}_{t-1}\}_{m=1}^{M=4} = \{3,4,5,6\}$, with corresponding weights $\{0.03, 0.16, 0.16, 0.65\}$



(b) In this example, we take well separated transition kernels, as well as an informative likelihood. This is the regime where Elvira et al. [24] found that the APF operates on best. Indeed one can see that APF, IAPF and OAPF all recover the same solution in this case. In this example, $\sigma_{\text{lik}} = 0.7, \sigma_{\text{kern}} = 0.2$, $\{\boldsymbol{x}_{t-1}\}_{m=1}^{M=4} = \{3.25, 4.57, 5.75, 6.5\}$, with corresponding weights $\{0.316, 0.158, 0.210, 0.316\}$

Figure 4.1: Mixture of Gaussians posterior reconstruction examples. OAPF posteriors obtained by NNLS or Simplex have negligible differences (the ones plotted are obtained by NNLS).

Proposal	χ^2
BPF	13.639
APF	0.130
IAPF	0.060
OAPF	**0.030**

Table 4.1: χ^2 from proposals to true posterior from Figure 4.1a

Proposal	χ^2
BPF	1.064
APF	0.085
IAPF	0.085
OAPF	0.085

Table 4.2: χ^2 from proposals to true posterior from Figure 4.1b

4.2 Sequential estimation

In this Section we move onto more concrete PF experiments. Here, we explain settings that will be common to all subsequent experiments. It is worth keeping in mind that we focus on a setting where resampling is performed at each iteration, therefore estimates are quickly affected by path degeneracy. Moreover, the optimization for OAPF is always done using NNLS; more exploration is needed to investigate the differences between the solutions given by NNLS and Simplex in general. The matrix of transition kernel evaluations $\boldsymbol{F}$ used by OAPF has been conditioned by adding $0.1 \cdot \boldsymbol{I}$ in all runs. Moreover, the set of evaluation points $\left\{\boldsymbol{z}^{(m)}\right\}_{m=1}^{M}$ is always chosen to be equal to the set of transition kernel means $\left\{\boldsymbol{\mu}^{(m)}\right\}_{m=1}^{M}$, with $\boldsymbol{\mu}^{(m)} \overset{\text{def}}{=} \mathbb{E}_{f(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{(m)})}[\boldsymbol{x}_t]$. All sequences will be of length $T = 100$ timesteps. Experiments are reproducible with global random seed 5.

4.2.1 Linear Gaussian state space model

We start evaluations by considering Linear Gaussian state space models. These are appealing for evaluation purposes, because we can directly compare approximate posteriors from PFs to the true posterior as calculated by the Kalman Filter (KF). More precisely, consider a state space model with the following prior, transition and observation densities:

$$p(\boldsymbol{x}_1) = \mathcal{N}_{\boldsymbol{x}_1}(\boldsymbol{0}, \boldsymbol{I}) \tag{4.1}$$

$$f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}_{\boldsymbol{x}_t}(\boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{c}, \boldsymbol{R}) \tag{4.2}$$

$$g(\boldsymbol{y}_t \mid \boldsymbol{x}_t) = \mathcal{N}_{\boldsymbol{y}_t}(\boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{g}, \boldsymbol{Q}) \tag{4.3}$$

Algorithm	BPF	APF	IAPF	OAPF	OAPF (**Reduced $\boldsymbol{F}, \widetilde{\boldsymbol{\pi}}$**)
MSE with KF mean : $M = 5$	5.368	3.306	2.401	**2.145**	–
$M = 10$	1.852	1.012	0.689	**0.603**	–
$M = 100$	0.094	0.101	0.020	0.021	**0.018 (2 kernels)**
$M = 1000$	0.0063	0.0871	0.0075	0.0078	**0.0065 (20 kernels)**

Table 4.3: MSE between the estimated means of a 2 dimensional Linear Gaussian system from PFs and the mean given by KF. The fourth column shows the MSE given by OAPF when reducing the size of the optimization problem, by only selecting certain rows and columns of $\boldsymbol{F}$ based on entries in $\widetilde{\boldsymbol{\pi}}$. When reducing the system, the number of kernels retained is $M/50$. For 5 and 10 particles this is not done, as the size is already small enough.

The posterior SFD is calculated in closed form with the Kalman filter and is given by:

$$p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t}) = \mathcal{N}_{\boldsymbol{x}_t}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \tag{4.4}$$

$$\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + \boldsymbol{K}(\boldsymbol{y}_t - \boldsymbol{C}\overline{\boldsymbol{\mu}}_t) \tag{4.5}$$

$$\boldsymbol{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C})\overline{\boldsymbol{\Sigma}}_t \tag{4.6}$$

$$\overline{\boldsymbol{\mu}}_t = \boldsymbol{A}\boldsymbol{\mu}_{t-1} + \boldsymbol{c} \tag{4.7}$$

$$\overline{\boldsymbol{\Sigma}}_t = \boldsymbol{A}\boldsymbol{\Sigma}_{t-1}\boldsymbol{A}^\top + \boldsymbol{R} \tag{4.8}$$

$$\boldsymbol{K} = \overline{\boldsymbol{\Sigma}}_t\boldsymbol{C}^\top\left(\boldsymbol{C}\overline{\boldsymbol{\Sigma}}_t\boldsymbol{C}^\top + \boldsymbol{Q}\right)^{-1} \tag{4.9}$$

We start by applying BPF, APF, IAPF and OAPF to the task of tracking the distribution of a $D = 2$ dimensional hidden state. For this first example, the parameters will be: $\boldsymbol{A} = \boldsymbol{C} = \boldsymbol{I}; \boldsymbol{c} = \boldsymbol{g} = \boldsymbol{0}; \boldsymbol{R} = 5 \cdot \boldsymbol{I}, \boldsymbol{Q} = 0.2 \cdot \boldsymbol{I}$. In this example we choose uncorrelated dimensions for simplicity, as well as a larger transition covariance compared to a small observation covariance. This is to resemble the situation explained in the previous Section, where the APF has troubles because of the potentially overlapping kernels. Table 4.3 shows the results of estimating the mean of the SFD over time: note that the MSE is averaged across timesteps *and* across dimensions.
Finally, we track a 5 dimensional hidden state with the same parameters as previously: MSE results can be found in Table 4.4.

Algorithm	BPF	APF	IAPF	OAPF	OAPF (**Reduced** $\boldsymbol{F}, \widetilde{\boldsymbol{\pi}}$)
$M = 1000$	0.4421	0.2364	**0.0862**	0.0917	0.0896 (20 kernels)

Table 4.4: MSE between the estimated means of a 5 dimensional Linear Gaussian system.

4.2.2 Stochastic Volatility

We will also perform inference in a multivariate *stochastic volatility* model (SVM), a type of stochastic process where the variance is a latent variable that follows itself a stochastic process. These are extremely useful models to apply for many tasks in mathematical finance and are often used to evaluate PFs [65, 45]. We employ the state space model known as *basic SVM* [13] also used in related work by Guarniero et al. [30, 29]. It is defined by the following prior, transition and observation densities:

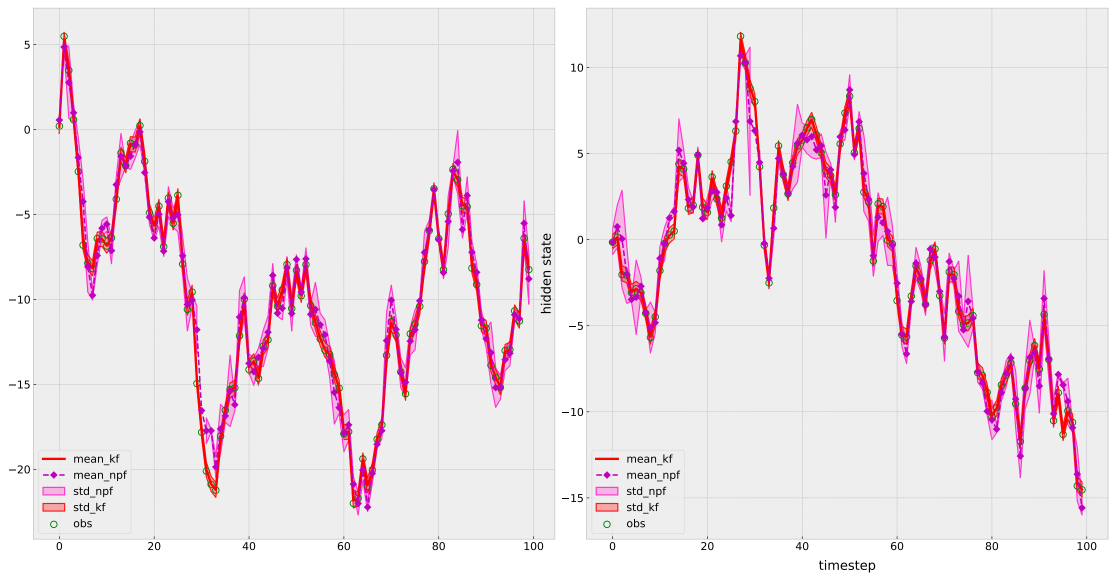
$$p(\boldsymbol{x}_1) = \mathcal{N}_{\boldsymbol{x}_1}(\boldsymbol{m}, \boldsymbol{U}_0) \tag{4.10}$$

$$f(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}_{\boldsymbol{x}_t}(\boldsymbol{m} + \operatorname{diag}(\boldsymbol{\phi})(\boldsymbol{x}_{t-1} - \boldsymbol{m}), \boldsymbol{U}) \qquad t = 2, \ldots, T \tag{4.11}$$

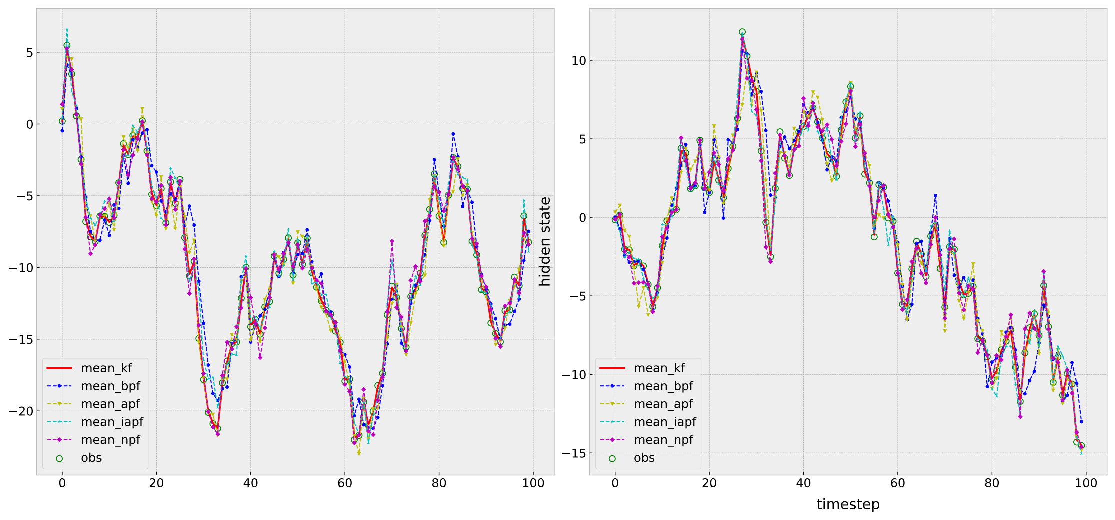
$$g(\boldsymbol{y}_t \mid \boldsymbol{x}_t) = \mathcal{N}_{\boldsymbol{y}_t}(\boldsymbol{0}, \exp(\operatorname{diag}(\boldsymbol{x}_t))) \qquad t = 1, \ldots, T \tag{4.12}$$

Unlike for Linear Gaussian systems, here there is no ground truth on the posterior distribution: this makes evaluation signficantly more challenging. We follow partially [45] and [65], showing the estimated ESS and the particle weight variance over time for each of the PF algorithms.

We start by evaluating on a $D = 2$ dimensional hidden state. The parameters are set to $\boldsymbol{m} = \boldsymbol{0}, \boldsymbol{U}_0 = \boldsymbol{I}, \boldsymbol{U} = 0.1 \cdot \boldsymbol{I}, \boldsymbol{\phi} = \boldsymbol{1}$. We plot ESS and weight variance over time for 100 and 1000 particles in Figure 4.3. In two dimensions the best performing PFs appear to be the APF (in terms of weight variance) and the OAPF (in terms of ESS). The BPF clearly comes out as the worst, having extremely large variance of weights at many iterations. Finally, we perform the same experiment, that is with the same exact parameters, but in 5 dimensions (as we did for Linear Gaussian systems). As Figure 4.3c shows, the OAPF starts performing worse compared to lower dimensions, while still being the top choice along with APF. Interestingly, IAPF seems to perform worse than one would expect for this model. Indeed, in almost all one dimensional additional examples tried for the experiments in Section 4.1, the IAPF proposal is always at least as good as that of APF or BPF. Of course, one should remember that ESS and sample weight variance are approximate measures of performance.

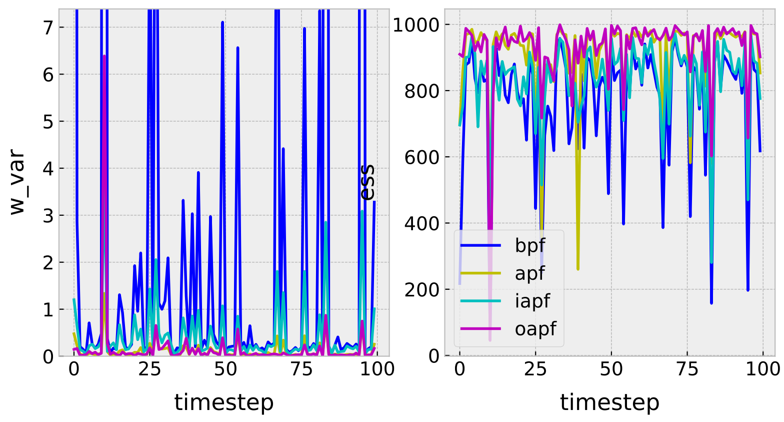


(a) Evolution of the two-dimensional posteriors given by KF and OAPF over time (on the left the first dimension of the state is plotted, on the right the second dimension). Notice visually how the mean is already fairly well tracked, whereas the posterior uncertainty of OAPF often overestimates that given by KF . This is likely due to the still relatively low number of particles used.

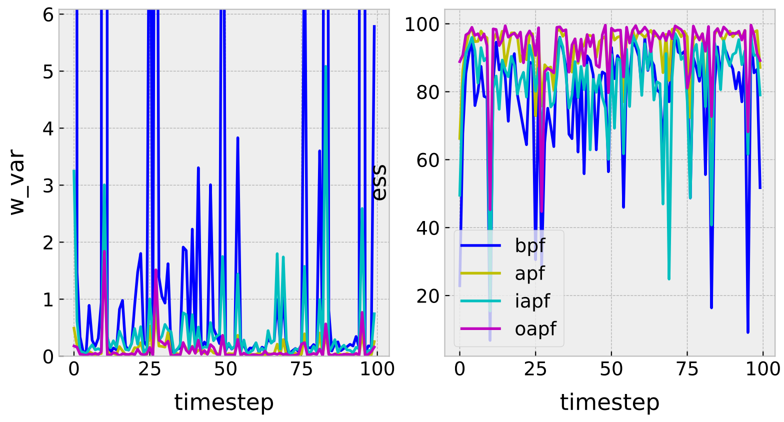


(b) The OAPF mean comes out to have the lowest MSE among the particle filters; however as in timestep around 70 on the left, it can still happen that it makes large mistakes when other filters won't.

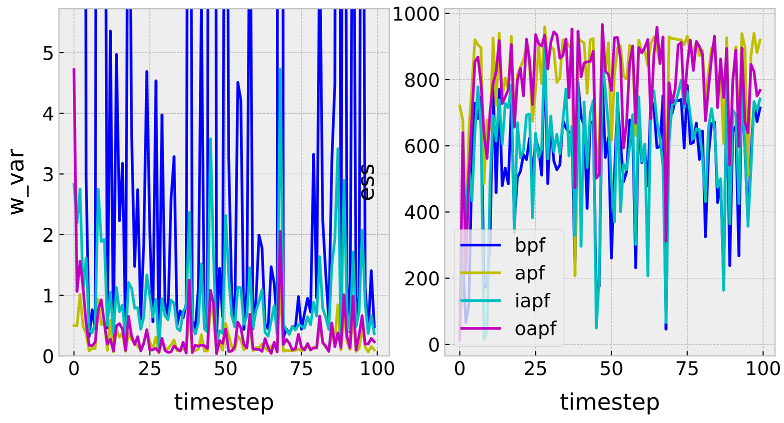
Figure 4.2: This Figure complements the results from Table 4.3 (second line): Part 4.2a shows the full posterior by KF and OAPF (mean $\pm$ 1 standard deviation), whereas part 4.2b shows means only, but for BPF,APF,IAPF and OAPF. Recall that 10 particles are used in this experiment.



(a) Weight variance (left) and ESS (right) over time using 1000 particles. Importantly, to produce this plot, we reduced the size of $\boldsymbol{F}, \widetilde{\boldsymbol{\pi}}$ 50-fold to improve performance, obtaining a 20×20 matrix and 20×1 vector for the optimization of OAPF



(b) Same as above, with 100 particles. $\boldsymbol{F}, \widetilde{\boldsymbol{\pi}}$ are respectively reduced to 2×2 and 2×1.



(c) These are results for a 5 dimensional SVM exactly equal to the 2 dimensional above. Here 1000 particles where used, with the same reduction as Figure 4.3a

Figure 4.3: Evaluation of the basic SVM with a 2-dimensional hidden state (Figure 4.3a and 4.3b) and 5 dimensional (4.3c). Legend is valid for both left and right plots. Note that weight variance is plotted up to a range so that curves are visible for algorithms other than BPF, since this has extremely high variance for some iterations.

Chapter 5

Conclusions

5.1 Summary & Discussion

In this thesis, we reviewed basic relevant concepts behind Particle Filters such as Monte Carlo methods, Importance Sampling, and state space models. Then, we reviewed relevant past work that tries to improve the accuracy of estimates of traditional PF algorithms, most notably: Marginal Particle Filters by Klaas et al. [45], Improved Auxiliary Particle Filters by Elvira et al. [23, 24], and Iterated Auxiliary Particle Filters by Guarniero et al. [29, 30]. Inspired by ideas from all these works, we proposed a novel class of PF algorithms named OAPF (Optimized Auxiliary Particle Filters). To design this algorithm, we exploited the use of a proposal which takes the form of a linear combination of the APF's simulation weights, so that they can be optimized with respect to the target posterior evaluated at a set of points. Effectively, OAPF matches pointwise proposal and target at locations that incorporate knowledge from the particle approximation at $t-1$. We also have seen that naively using as many transition kernels as there are particles can be extremely wasteful of computational resources, as well as harmful for accuracy of estimates, since OAPF optimization involves dealing with a matrix of size $K \times K$ with K being the number of transition kernels one decides to deal with.

We executed 3 sets of experiments to evaluate the performance of OAPF:

- Firstly, we simulated a single iteration by providing an artificial set of weights and particles to represent the filtering distribution at $t-1$. This setting allows for a clear analysis of when transition kernels have significant/negligible overlap, and the proposal can be visually compared to the posterior. The OAPF essentially performs as a better version of the IAPF: its estimation benefits appear larger in the large overlap situation, whereas it performs similarly to APF as the kernels overlap decreases.

- Then we proceded to track the SFD given by a Linear Gaussian state space model. The OAPF generally tracks the mean of the KF more accurately than other algorithms for both the 2 dimensional and 5 dimensional examples; however, larger number of particles seem to require the reduction in size of the optimization problem to achieve competitive performance with IAPF. This is especially true as the dimensionality increases, since the number of particles needed to estimate the state increases exponentially with its dimension. We showed visually (for the 2 dimensional state) that after ≈ 10 particles, the posterior uncertainty is also converging to that given by the KF.

- Finally, we evaluated on a version of the multivariate basic Stochastic Volatility model, also using 2 and 5 dimensional hidden states. This is a challenging nonlinear state space model so that a closed form solution for the SFD is not available: to evaluate performance, we analysed ESS and weight variance over timesteps.

5.2 Scope

The practical application of OAPF can present some numerical optimization issues, which seem to become worse as the dimensionality of the hidden state increases. Firstly, since NNLS involves the inversion of large matrices, these could be ill-conditioned, thus we added a diagonal term to alleviate this problem. However, for a dimensionality of state space $D > 5$ approximately, the algorithm's estimates seem to become worse than its competitors. It is worth keeping in mind that SMC methods provide poor estimates of the posterior distributions in high dimensions in general, as they suffer from the curse of dimensionality. Furthermore, one may notice that the same approach of optimizing the weights of a mixture proposal could be used in plain IS as well. However, here is where the selection of evaluation points comes into play: in SMC, we can choose (e.g.) the centers of the transition kernels, or any set of representative points linked to the particles at the previous iteration. These points already provide a reasonable location where the posterior should be evaluated. This advantage is not generally present in plain IS: knowing where there is significant probability mass in the posterior is intractable. Finally, note that we have ignored runtime complexity that comes from

the dimensionality of the state space. This is reasonable since we work in relatively low dimensions; however, it needs to be stated that, due to the building of the matrix of transition kernels evaluations $\boldsymbol{F}$, our algorithm performs significantly more probability density evaluations than its competitors, and this could be prohibitive for some applications.
Despite the presented issues, our novel algorithm presents several advantages:

- *Simplicity of implementation*: The algorithm requires only modifying a function that computes the simulation weights. For the optimization strategies we have proposed, there are convenient subroutines from the SciPy and Numpy libraries that perform the optimization efficiently e.g. `scipy.optimize.nnls` and `scipy.optimize.linprog`. Our code shows that once one has implemented a modular framework for PF algorithms, it only requires a few lines of code to change from IAPF to OAPF.

- For low dimensional problems (1 to 5 approximately), our results suggest it can provide the most accurate posterior estimates, although a wider range of nonlinear non-Gaussian models could be tested.

- When using large number of particles in order to seek high accuracy, the algorithm benefits both in accuracy and runtime by reducing the size of the optimization problem. Several strategies can be thought of in order to achieve this: we presented an extremely simple one that requires a couple of lines of code, but still improves results in our experiments and greatly reduces the computational cost. Furthermore, even without addressing the specific setting of our optimization, NNLS can be performed very efficiently and accurately with Iterative Hessian Sketching as explained in Section 3.2.2, which reduce the runtime to $O(\log 1/\varepsilon) \cdot O(KM^2)$ where K and ε are user chosen (see that Section for more details).

- Finally, empirically we found that for large number of particles (≥ 1000) the solutions returned by NNLS in particular are extremely sparse ($\geq 90\%$): this fact, combined with the above findings, further suggest the potential computational savings both in the optimization stage of Algorithm 2 as well as in the computation of IS weights. It has been suggested in the literature that enforcing nonnegativity in least squares problems naturally promotes sparsity.

5.3 Further work

There are several promising directions for future work. In particular, more techniques could be developed in order to reduce the size of the optimization problem of OAPF in a principled way, exploiting the structure of the SMC problem. For example, one could try to reduce kernels by merging those that share a common ancestor in the PF ancestry tree. There could also be smarter ways to select evaluation points, rather than simply select the means of the transition kernels. Indeed, if the likelihood is ”far away” from the BPF proposal [1], then it may be difficult to adapt the mixture of weighted kernels so that it matches the posterior well, since the latter would be highly skewed towards the likelihood. In this case, selecting at least some evaluation points that are in regions of high likelihood could help the proposal match the posterior more closely.

Finally, since OAPF iteratively adapts a parameterized proposal to the posterior, one could perform Variational Inference (VI). As mentioned at the beginning of Chapter 3, performing VI online with the inclusive KL divergence is likely a bad idea: it would be interesting to adapt recent work on VI with the χ^2 divergence on general probabilistic models [18] to the PF setting, perhaps deriving specific Variational Bounds by using a mixture of kernels Variational distribution.

Moreover, there is room for work to adapt OAPF to make use of future observations when these are available. In this case, having access to L further observations, one could use kernels such as $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{y}_{t:t+L})$ and modified simulation weights $p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{1:t+L})$.

[1] which is a proxy for where the kernels sit in input space.

Bibliography

[1] Forman S Acton. *Numerical Methods that Work*. MAA, 1990.

[2] Ömer Deniz Akyildiz and Joaquín Míguez. Nudging the particle filter. *Statistics and Computing*, 30(2):305–330, 2020.

[3] Brian DO Anderson and John B Moore. *Optimal filtering*. Courier Corporation, 2012.

[4] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[5] Matthew J Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, UCL (University College London), 2003.

[6] Ayman Boustati, Ömer Deniz Akyildiz, Theodoros Damoulas, and Adam Johansen. Generalized bayesian filtering via sequential monte carlo. *arXiv preprint arXiv:2002.09998*, 2020.

[7] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Rasmus Bro and Sijmen De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 11(5):393–401, 1997.

[9] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[10] James V Candy. *Bayesian signal processing: classical, modern, and particle filtering methods*, volume 54. John Wiley & Sons, 2016.

[11] Olivier Cappé, Simon J Godsill, and Eric Moulines. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[12] James Carpenter, Peter Clifford, and Paul Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation*, 146(1):2–7, 1999.

[13] Siddhartha Chib, Yasuhiro Omori, and Manabu Asai. Multivariate stochastic volatility. In *Handbook of Financial Time Series*, pages 365–400. Springer, 2009.

[14] Nicolas Chopin et al. Central limit theorem for sequential monte carlo methods and its application to bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.

[15] Vasek Chvatal. *Linear programming*. Macmillan, 1983.

[16] Julien Cornebise, Eric Moulines, and Jimmy Olsson. Adaptive sequential monte carlo by means of mixture of experts. *Statistics and Computing*, 24(3):317–337, 2014.

[17] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[18] Adji Bousso Dieng, Dustin Tran, Rajesh Ranganath, John Paisley, and David Blei. Variational inference via χ upper bound minimization. In *Advances in Neural Information Processing Systems*, pages 2732–2741, 2017.

[19] Arnaud Doucet. *On Sequential Simulation-based Methods for Bayesian Filtering*. Department of Engineering, University of Cambridge, 1998.

[20] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[21] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[22] Roger Eckhardt, Stan Ulam, and Jon Von Neumann. the monte carlo method. *Los Alamos Science*, (15):131, 1987.

[23] Víctor Elvira, Luca Martino, Mónica F Bugallo, and Petar M Djurić. In search for improved auxiliary particle filters. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1637–1641. IEEE, 2018.

[24] Victor Elvira, Luca Martino, Monica F Bugallo, and Petar M Djuric. Elucidating the auxiliary particle filter via multiple importance sampling [lecture notes]. *IEEE Signal Processing Magazine*, 36(6):145–152, 2019.

[25] Víctor Elvira, Luca Martino, David Luengo, Mónica F Bugallo, et al. Generalized multiple importance sampling. *Statistical Science*, 34(1):129–155, 2019.

[26] Simon Godsill. Particle filtering: the first 25 years and beyond. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7760–7764. IEEE, 2019.

[27] Simon Godsill and Tim Clapp. Improvement strategies for monte carlo particle filters. In *Sequential Monte Carlo methods in practice*, pages 139–158. Springer, 2001.

[28] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

[29] Pieralberto Guarniero. *The Iterated Auxiliary Particle Filter and Applications to State Space Models and Diffusion Processes*. PhD thesis, University of Warwick, 2017.

[30] Pieralberto Guarniero, Adam M Johansen, and Anthony Lee. The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647, 2017.

[31] Jeremy Heng, Adrian N Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential monte carlo. *arXiv preprint arXiv:1708.08396*, 2017.

[32] Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, 1995.

[33] Jeroen D Hol, Thomas B Schon, and Fredrik Gustafsson. On resampling algorithms for particle filters. In *2006 IEEE nonlinear statistical signal processing workshop*, pages 79–82. IEEE, 2006.

[34] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[35] Adam M Johansen and Arnaud Doucet. Auxiliary variable sequential monte carlo methods. *Statistics group technical report*, 7(09), 2007.

[36] Adam M Johansen and Arnaud Doucet. A note on auxiliary particle filters. *Statistics & Probability Letters*, 78(12):1498–1504, 2008.

[37] Eric Jones, Travis Oliphant, Pearu Peterson, et al. Scipy: Open source scientific tools for python. 2001.

[38] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[39] Herman Kahn. Random sampling (monte carlo) techniques in neutron attenuation problems–i. *Nucleonics*, 6(5):27–passim, 1950.

[40] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[41] Keiji Kanazawa, Daphne Koller, and Stuart Russell. Stochastic simulation algorithms for dynamic probabilistic networks. *arXiv preprint arXiv:1302.4965*, 2013.

[42] Nikolas Kantas, Arnaud Doucet, Sumeetpal S Singh, Jan Maciejowski, Nicolas Chopin, et al. On particle methods for parameter estimation in state-space models. *Statistical science*, 30(3):328–351, 2015.

[43] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[44] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[45] Mike Klaas, Nando de Freitas, and Arnaud Doucet. Toward practical n2 monte carlo: the marginal particle filter. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 308–315, Arlington, Virginia, 2005. AUAI Press.

[46] Augustine Kong, Jun S Liu, and Wing Hung Wong. Sequential imputations and bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288, 1994.

[47] Joel Kronander and Thomas B Schön. Robust auxiliary particle filters using multiple importance sampling. In *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, pages 268–271. IEEE, 2014.

[48] Charles L Lawson and Richard J Hanson. *Solving least squares problems*. SIAM, 1995.

[49] Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential monte carlo. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[50] Tiancheng Li, Miodrag Bolic, and Petar M Djuric. Resampling methods for particle filtering: classification, implementation, and strategies. *IEEE Signal processing magazine*, 32(3):70–86, 2015.

[51] Fredrik Lindsten and Thomas B Schön. Backward simulation methods for monte carlo statistical inference. *Foundations and Trends® in Machine Learning*, 6(1):1–143, 2013.

[52] David Luengo, Luca Martino, Mónica Bugallo, Víctor Elvira, and Simo Särkkä. A survey of monte carlo methods for parameter estimation. *EURASIP Journal on Advances in Signal Processing*, 2020(1):1–62, 2020.

[53] Luca Martino, Víctor Elvira, and Francisco Louzada. Effective sample size for importance sampling based on discrepancy measures. *Signal Processing*, 131:386–401, 2017.

[54] N Metropolis. The beginning. *Los Alamos Science*, 15:125–130, 1987.

[55] Lyudmila Mihaylova, Avishy Y Carmi, François Septier, Amadou Gning, Sze Kim Pang, and Simon Godsill. Overview of bayesian sequential monte carlo methods for group and extended object tracking. *Digital Signal Processing*, 25:1–16, 2014.

[56] Isaac Miller, Mark Campbell, and Dan Huttenlocher. Map-aided localization in sparse global positioning system environments using vision and particle filtering. *Journal of Field Robotics*, 28(5):619–643, 2011.

[57] Tom Minka et al. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

[58] Nicolas Nadisic, Arnaud Vandaele, Nicolas Gillis, and Jeremy E Cohen. Exact sparse nonnegative least squares. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5395–5399. IEEE, 2020.

[59] Christian A Naesseth, Fredrik Lindsten, Thomas B Schön, et al. Elements of sequential monte carlo. *Foundations and Trends® in Machine Learning*, 12(3):307–392, 2019.

[60] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[61] Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[62] Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000.

[63] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, High-Performance deep learning library. In H Wallach, H Larochelle, A Beygelzimer, F d Alché-Buc, E Fox,

and R Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019.

[64] Mert Pilanci and Martin J Wainwright. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016.

[65] Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

[66] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[67] Donald B Rubin. Using the sir algorithm to simulate posterior distributions. *Bayesian statistics*, 3:395–402, 1988.

[68] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[69] Adrian Smith. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[70] Sebastian Thrun, Dieter Fox, Wolfram Burgard, et al. Monte carlo localization with mixture proposal distribution. In *AAAI/IAAI*, pages 859–865, 2000.

[71] Eric Veach and Leonidas J Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428, 1995.

[72] Nick Whiteley and Adam M Johansen. Auxiliary particle filtering: recent developments. *Bayesian time series models. Cambridge University Press, Cambridge*, 2011.

[73] VS Zaritskii, VB Svetnik, and LI Šimelevič. Monte-carlo technique in problems of optimal information processing. *Avtomatika i telemekhanika*, (12):95–103, 1975.

[74] Yuan Zhao, Josue Nassar, Ian Jordan, Mónica Bugallo, and Il Memming Park. Streaming variational monte carlo. *arXiv preprint arXiv:1906.01549*, 2019.