SEQUENTIAL AND ADAPTIVE BAYESIAN COMPUTATION FOR INFERENCE AND OPTIMIZATION

Ömer Deniz Akyıldız

in partial fulfillment of the requirements for the degree of Doctor in Multimedia and Communications

Universidad Carlos III de Madrid
Department of Signal Theory and Communications

Advisor:
Joaquín Míguez

March 2019

On the mountains of truth, you can never climb in vain:
either you will reach a point higher up today,
or you will be training your powers
so that you will be able to climb higher tomorrow.
— Friedrich Nietzsche

PUBLISHED AND SUBMITTED CONTENT

The following works I authored or co-authored are used in this thesis entirely or partially.

1. Omer Deniz Akyildiz and Joaquin Miguez. Nudging the particle filter. *Under review*, 2017. Preprint can be accessed from: https://arxiv.org/abs/1708.07801.

    - The material in this preprint (which is currently under review) is used entirely in Chapter 3 of this thesis.

2. Omer Deniz Akyildiz, Victor Elvira, and Joaquin Miguez. The incremental proximal method: A probabilistic perspective. In Proceedings of *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4279–4283, 2018. The paper can be accessed from: https://arxiv.org/abs/1807.04594.

    - The material published in this paper is used in Chapter 4 of this thesis. A preliminary version of this paper appeared as a workshop contribution with the following details: Akyildiz, Ö. D., Elvira, V., Fernandez-Bes, J., Miguez, J. On the Relationship between Online Optimizers and Recursive Filters, NIPS 2016 Workshop on Optimizing the Optimizers. The pdf can be accessed from: http://www.probabilistic-numerics.org/assets/pdf/NIPS2016/Akyildiz_Elvira_Fernandez-Bes_Miguez.pdf.

3. Omer Deniz Akyildiz, Dan Crisan, and Joaquin Miguez. Parallel sequential Monte Carlo for stochastic optimization. *Preprint*. The preprint can be accessed from: https://arxiv.org/abs/1811.09469.

    - The material published in this paper is used in Chapter 4 of this thesis.

4. Omer Deniz Akyildiz, Joaquin Miguez. Dictionary filtering: A probabilistic approach to online matrix factorisation. *Signal, Image and Video Processing*, Springer, 2018. https://doi.org/10.1007/s11760-018-1403-9.

- The material published in this paper is used entirely in Chapter 5 of this thesis. An earlier version of this paper appears as a preprint with the title *Matrix Factorisation with Linear Filters*, which can be accessed from: https://arxiv.org/abs/1509.02088.

5. The following works are based on or related to this thesis, but are not entirely used.

   - Omer Deniz Akyildiz, Ines P. Marino, Joaquin Miguez, Adaptive noisy importance sampling for stochastic optimization, CAMSAP 2017, https://doi.org/10.1109/CAMSAP.2017.8313215.
   - Omer Deniz Akyildiz, Emilie Chouzenoux, Victor Elvira, Joaquín Míguez,. A probabilistic incremental proximal gradient method. arXiv preprint arXiv:1812.01655, 2018, https://arxiv.org/abs/1812.01655.

Abstract

With the advent of cheap and ubiquitous measurement devices, today more data is measured, recorded, and archived in a relatively short span of time than all data recorded throughout history. Moreover, advances in computation have made it possible to model much more complicated phenomena and to use the vast amounts of data to calibrate the resulting high-dimensional models. In this thesis, we are interested in two fundamental problems which are repeatedly being faced in practice as the dimension of the models and datasets are growing steadily: the problem of *inference* in high-dimensional models and the problem of *optimization* for problems when the number of data points is very large.

The inference problem gets difficult when the model one wants to calibrate and estimate is defined in a high-dimensional space. The behavior of computational algorithms in high-dimensional spaces is complicated and defies intuition. Computational methods which work accurately for inferring low-dimensional models, for example, may fail to generalize the same performance to high-dimensional models. In recent years, due to the significant interest in high-dimensional models, there has been a plethora of work in signal processing and machine learning to develop computational methods which are robust in high-dimensional spaces. In particular, the high-dimensional *stochastic filtering* problem has attracted significant attention as it arises in multiple fields which are of crucial importance such as geophysics, aerospace, control. In particular, a class of algorithms called *particle filters* has received attention and become a fruitful field of research because of their accuracy and robustness in low-dimensional systems. In short, these methods keep a cloud of particles (samples in a state space), which describe the empirical probability distribution over the state variable of interest. The particle filters use a model of the phenomenon of interest to propagate and predict the future states and use an observation model to assimilate the observations to correct the state estimates. The most common particle filter, called the bootstrap particle filter (BPF), consists of an iterative sampling-weighting-resampling scheme. However,

BPFs also largely fail at inferring high-dimensional dynamical systems due to a number of reasons.

In this work, we propose a novel particle filter, named *the nudged particle filter* (NuPF), which specifically aims at improving the performance of particle filters in high-dimensional systems. The algorithm relies on the idea of nudging, which has been widely used in the geophysics literature to tackle high-dimensional inference problems. In particular, in addition to standard sampling-weighting-resampling steps of the particle filter, we define a general nudging step based on the gradient of the likelihoods, which generalize some of the nudging schemes proposed in the literature. This step is based on modifying the particles, generated in the sampling step, using the gradients of the likelihoods. In particular, the nudging step moves a fraction of the particles to the regions under which they have high-likelihoods. This scheme results in significantly improved behavior in high-dimensional models. Unlike the proposed nudging schemes in the literature, the NuPF does not rely on Gaussianity assumptions and can be defined for a general likelihood. We analytically prove that, because we only move a fraction of the particles and not all of them, the algorithm has a convergence rate that matches standard Monte Carlo algorithms. More precisely, the NuPF has the same asymptotic convergence guarantees as the bootstrap particle filter. As a byproduct, we also show that the nudging step improves the robustness of the particle filter against model misspecification. In particular, model misspecification occurs when the true data-generating system and the model posed by the user of the algorithm differ significantly. In this case, a majority of computational inference methods fail due to the discrepancy between the modeling assumptions and the observed data. We show that the nudging step increases the robustness of particle filters against model misspecification. Finally, we demonstrate the empirical behavior of the NuPF with several examples. In particular, we show results on high-dimensional linear state-space models, a misspecified Lorenz 63 model, a high-dimensional Lorenz 96 model, and a misspecified object tracking model. In all examples, the NuPF infers the states successfully.

The second problem, the so-called *scability* problem in optimization, occurs because of the large number of data points in modern datasets. With the increasing abundance of data, many problems in signal processing, statistical inference, and machine learning turn into a large-scale optimization problems. For example, in signal processing, one might be interested in estimating a sparse signal given a large number of corrupted observations. Similarly, maximum-likelihood inference problems in statistics result in large-scale optimization problems. Another significant application domain is machine learning, where all important training

methods are defined as optimization problems. To tackle these problems, computational optimization methods developed over the past decades are inefficient since they need to compute function evaluations or gradients over all the data for a single iteration. Because of this reason, a class of optimization methods, termed stochastic optimization methods, have emerged. The algorithms of this class are designed to tackle problems which are defined over a big number of data points. In short, these methods utilize a subsample of the dataset in order to update the parameter estimate and do so iteratively until some convergence criterion is met. However, there is a major difficulty that has to be addressed: Although the convergence theory for these algorithms is understood, they can have unstable behavior in practice. In particular, the most commonly used stochastic optimization method, namely the stochastic gradient descent, can diverge easily if its step-size is poorly set. Over the years, practitioners have developed a number of rules of thumb to alleviate stability issues.

We argue in this thesis that one way to develop robust stochastic optimization methods is to frame them as inference methods. In particular, we show that stochastic optimization schemes can be recast as inference methods and can be understood as inference algorithms. Framing the problem as an inference problem opens the way to compare these methods to the optimal inference algorithms and understand why they might be failing or producing unstable behavior. In this vein, we show that there is an intrinsic relationship between a class of stochastic optimization methods, called incremental proximal methods, and Kalman (and extended Kalman) filters. The filtering approach to stochastic optimization results in an automatic calibration of the step-size, which removes the instability problems depending on the step-sizes.

The probabilistic interpretation of stochastic optimization problems also paves the way to develop new optimization methods based on strategies which are popular in the inference literature. In particular, one can use a set of sampling methods in order to solve the inference problem and hence obtain the global minimum. In this manner, we propose a parallel sequential Monte Carlo optimizer (PSMCO), which is aiming at solving stochastic optimization problems. The PSMCO is designed as a zeroth order method which does not use gradients. It only uses subsets of the data points in order to move at each iteration. The PSMCO obtains an estimate of a global minimum at each iteration by utilizing a cheap kernel density estimator. We prove that the resulting estimator converges to a global minimum almost surely as the number of Monte Carlo samples tends to infinity. We also empirically demonstrate that the algorithm is able to reconstruct multiple global minima and solve difficult global optimization problems.

By further exploiting the relationship between inference and optimization, we also propose a probabilistic and online matrix factorization method, termed the *dictionary filter* to solve large-scale matrix factorization problems. Matrix factorization methods have received significant interest from the machine learning community due to their expressive representations of high-dimensional data and interpretability of their estimates. As the majority of the matrix factorization methods are defined as optimization problems, they suffer from the same issues as stochastic optimization methods. In particular, when using stochastic gradient descent, one might need to try and err many times before deciding to use a step-size. To alleviate these problems, we introduce a matrix-variate probabilistic model for which inference results in a matrix factorization scheme. The scheme is online, in the sense that it only uses a single data point at a time to update the factors. The algorithm bears relationship with optimization schemes, namely with the incremental proximal method defined over a matrix-variate cost function. By way of intuition we developed for the optimization-inference relationship, we devise a model which results in similar update rules for matrix factorization as for the incremental proximal method. However, the probabilistic updates are more stable and efficient. Moreover, the algorithm does not have a step-size parameter to tune, as its role is played by the posterior covariance matrix. We demonstrate the utility of the algorithm on a missing data problem and a video processing problem. We show that the algorithm can be successfully used in machine learning problems and several promising extensions of the method can be constructed easily.

Contents

List of Acronyms

APF	auxiliary particle filter
BPF	bootstrap particle filter
DF	dictionary filter
EKF	extended Kalman filter
GD	gradient descent
INMF	incremental nonnegative matrix factorization
IPM	incremental proximal method
IS	importance sampling
KDE	kernel density estimator
KF	Kalman filter
LGSSM	linear-Gaussian state-space model
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
NMF	nonnegative matrix factorization
NMSE	normalized mean squared error
NPF	nested particle filter
NuPF	nudged particle filter
ODE	ordinary differential equation

pdf	probability density function
PF	particle filter
pMH	particle Metropolis-Hastings
PPI	proximal point iteration
PSMCO	parallel sequential Monte Carlo optimizer
SGD	stochastic gradient descent
SGDMF	stochastic gradient descent matrix factorization
SMC	sequential Monte Carlo
SNIS	self-normalized importance sampling
SSM	state-space model
SVGD	Stein variational gradient descent

1 Introduction

1.1 Introduction

Given a scientific problem, every computational scientist collects *observed data* and then develops a *model* of a phenomenon of interest in order to process the data. The scientist then will seek ways to calibrate the free variables of the model with respect to the observed data. In order to do so, the scientist can develop a *probabilistic model* and recast the problem as an *inference* problem. Similarly, but with a quite different philosophy, he or she can also formulate a *cost function*, thus framing the problem as an *optimization* problem, where the free variables would satisfy a certain cost criterion and a set of constraints when calibrated. In this thesis, we are interested in developing computational schemes for some instances of these two general computational problems of inference and optimization.

Inference, (or more specifically, *Bayesian inference* [1]), starts with formulating a probabilistic model, in order to capture characteristics of a real world phenomenon, in which the conditional dependence and independence between observations (observed data) and hidden variables of interest are defined via probability distributions. The task of an inference scheme is then to obtain the posterior probability distribution, which is the distribution of the variables of interest conditioned on observations. The main computations which need to be carried out given a probability model are *conditioning* on the observed variables and *marginaliza-*

tion of the latent variables to obtain the posterior distribution of the variables of interest. Conditioning requires the application of the Bayes' rule while marginalization requires a possibly high-dimensional integration, neither of which can be done in closed form in general. The computational problem here is then to approximate the processes of conditioning and marginalization using computational statistical methods. Moreover, the need of computing expectations with respect to the posterior probability distributions again results in the computation of possibly high-dimensional integrals. In this thesis, we will be interested in such computational schemes for high-dimensional problems, e.g., *sequential Bayesian computation* schemes, such as numerical methods for *stochastic filtering* [2].

Similarly, the procedure of *optimization* starts from a model which is, in contrast to probabilistic models, described in the form of a cost function[1], rather than as a collection of probability distributions. This cost function is usually defined by taking the characteristics of the specific problem into account. In a similar way to probabilistic modeling, the cost function explicitly defines the observed data and the parameters of interest. The task is then to find the parameter setting which minimizes the cost function by building a computational optimization method which guarantees some criterion of descent. Usually, derivative information of the cost function, e.g., gradients or Hessians, is used to move within the parameter space. Our interest in optimization in this thesis is twofold. First, we are interested in stochastic optimization schemes and their connections with sequential Bayesian computational methods. Secondly, as a byproduct, we will be developing stochastic zeroth order [3] optimization schemes.

In the next two subsections, we provide a brief description of our main interests and potential problems we address in this work.

1.1.1 Stochastic filtering

Our primary interest is the problem of *stochastic filtering* [4, 5, 2]. Briefly put, stochastic filtering problem refers to the problem of inferring the states of a dynamical system from a sequence of observations, which are usually corrupted by noise. In particular, consider a dynamical system whose state variables are defined over time as $x_0, \ldots, x_t, \ldots$, which is described compactly as $(x_t)_{t\geq 0}$. Given observations $(y_t)_{t\geq 1}$ from this dynamical system, the computational goal is to build and sequentially update conditional probability distributions of the states $(x_t)_{t\geq 0}$ over time while assimilating observations one by one. The evolution of the states is

[1] A cost function qualifies the goodness of a candidate solution given the available data. The smaller the cost, the better the candidate.

described by a transition density τ_t and the relationship between y_t and each x_t is defined via a likelihood function, denoted g_t. When τ_t describes a linear stochastic dynamical system with Gaussian perturbations and g_t describes a linear observation model with additive Gaussian noise, the exact solution of the filtering problem can be found via the procedure known as the Kalman filter (KF), first developed by Rudolf Kalman and Robert Bucy [6–8]. Since then, the KF has received significant attention and has been successfully used for many signal processing and control problems [9], including the control of the Apollo moon landing [10]. Several variants of the KF for Gaussian models with nonlinearities were also developed, such as extended Kalman filters [11, 5] for systems which can be linearized (this scheme uses Taylor expansions of the transition and observation models) or unscented Kalman filters (UKF) [12–14] that propagate well-placed points in the state-space to construct the moments of the conditional densities, which do not require any gradient information. Another variant of the Kalman filter, called the ensemble Kalman filter (EnKF), has been introduced to tackle high-dimensional dynamical problems in geophysics, see, e.g. [15–19].

The computation of the posterior distribution for general transitions and likelihoods, i.e., for nonlinear and non-Gaussian models, however, remained elusive for a long time. Finally, the bootstrap particle filter (BPF) was developed in [20], which was motivated by the earlier works of [21, 22] and it became possible to solve the filtering problem numerically for a general model. The BPF is a Monte Carlo algorithm [23], which aims at drawing samples in order to construct the conditional distributions of the state variables and estimate the quantities of interest via the use of a sampling-weighting-resampling scheme [22]. The algorithm, for relatively low-dimensional systems, turned out to be quite efficient and accurate. This has resulted in a plethora of applications of the BPF scheme to various problems; see, e.g., [24–26]. As a result of this interest, a comprehensive theory for the class of algorithms called sequential Monte Carlo (SMC) methods, including the BPF as a special case, has been developed, framing the algorithm as an interacting particle system and investigating its long-term behaviour. See, e.g., [2, 27–34] and references therein.

Despite the success of PFs in relatively low dimensional settings, their use has been regarded impractical in models where $(x_t)_{t\geq 0}$ and $(y_t)_{t\geq 1}$ are sequences of high-dimensional random variables. This is an instance of a general problem coined as the *the curse of dimensionality*. It is referred to as *the collapse of the particle filter* within the particle filtering context and it has been studied from various empirical and theoretical viewpoints in recent years. The authors of [35] study the weights of the BPF in a high-dimensional setting and they show that the

maximum weight tends to 1 (under simplifying assumptions) which, they argue, causes the filter to collapse effectively (also see [36, 37] for additional research on the collapse of the BPF). Recently, a "block PF" has been proposed [38] that is applicable to certain network-like high-dimensional SSMs, and can be proved to converge for a subset of marginal posterior distributions on the state space under assumptions on the correlations between state variables. Another scheme, the space-time particle filter [39] has also been proved to converge even when the state dimension increases without bound, under certain assumptions on the structure of the state dynamics.

The collapse of the PF has received significant attention in the weather dynamics literature, where models are high-dimensional, obviously approximate, and do not yield analytic solutions. In particular, in the data assimilation literature, high-dimensional systems are often dealt with via an operation called *nudging* [40–43]. Within the particle filtering context, nudging can be defined as a transformation of the particles, which are pushed towards the observations using some observation-dependent map [44–47]. If the dimensions of the observations and the hidden states are different, which is often the case, a gain matrix is computed in order to perform the nudging operation. In [44–47] nudging is performed after the sampling step of the particle filter. The importance weights are then computed accordingly, so that they remain proper. Hence, nudging in this version amounts to a sophisticated choice of the importance function that generates the particles. It has been shown (numerically) that the schemes proposed by [44–47] can track high-dimensional systems with a low number of particles. However, generating samples from the nudged proposal requires costly computations for each particle and the evaluation of weights becomes heavier as well. It is also unclear how to apply existing nudging schemes when non-Gaussianity and nontrivial nonlinearities are present in the observation model.

A related class of algorithms includes the so-called implicit particle filters (IPFs) [48–50]. Similar to nudging schemes, IPFs rely on the principle of pushing particles to high-probability regions in order to prevent the collapse of the filter in high-dimensional state spaces. In a typical IPF, the region where particles should be generated is determined by solving an algebraic equation. This equation is model dependent, yet it can be solved for a variety of different cases (general procedures for finding solutions are given by [48] and [49]). The fundamental principle underlying IPFs, moving the particles towards high-probability regions, is similar to nudging. Note, however, that unlike IPFs, nudging-based methods are not designed to *guarantee* that the resulting particles land on high-probability regions; it can be the case that nudged particles are moved to relatively low probability

regions (at least occasionally). Since an IPF requires the solution of a model-dependent algebraic equation for every particle, it can be computationally costly, similar to the nudging methods by [44–47]. Moreover, it is not straightforward to derive the map for the translation of particles in general models, hence the applicability of IPFs depends heavily on the specific model at hand.

In this thesis, we will be interested in high-dimensional stochastic filtering problem. In particular, we propose a nudging based PF scheme in order to alleviate some of the problems outlined above. Our contributions to this field is summarized in Section 1.2.

1.1.2 Stochastic optimization

Our secondary focus in this thesis is the problem of *stochastic optimization*. Very broadly, optimization methods can be classified into two main classes: deterministic and stochastic. In deterministic schemes, all of the observed data can be used in every iteration to achieve the descent direction, e.g., gradient descent uses full gradients at every iteration. Deterministic optimization methods have received significant attention in the last decades due to their efficiency and usefulness in several areas. Especially, *convex optimization*, where the cost function to be minimized is a convex function, has been one of the most active research areas in recent years [51–53], and it has found applications in compressed sensing [54], matrix completion [55], machine learning [52], signal processing and communications [56], to name a few.

With the increasing abundance of data, however, these deterministic schemes have been rendered inefficient. In particular, deterministic methods struggle when the cost function is of the form

$$f(\theta) = \sum_{k=1}^{n} f_k(\theta), \tag{1.1}$$

with n very large, where the f_k's are called individual component functions and f is the cost function. In this case, deterministic methods lead to computationally very demanding solutions. In response to this challenge, the class of *stochastic optimization* methods has emerged. Originated by a paper in 50s [57], stochastic optimization methods have become widely used in modern signal processing and machine learning [58–69], due to their efficiency in big data settings. Notably, stochastic optimization methods do not need to perform computations using all data points at each iteration. Instead, they obtain noisy estimates of the necessary quantities for optimization using subsampling. The most basic scheme, the *stochastic gradient descent* (SGD) algorithm [60], obtains a subsample from the

dataset and construct an estimator of the gradient. The resulting estimate is then used to perform a stochastic descent step. The majority of these stochastic gradient methods construct the subsamples using sampling with replacement to obtain *unbiased* estimates of the gradient. The latter can then be seen as a noisy gradient estimate with additive, zero-mean noise. In practice, however, there are schemes that subsample the data set without replacement (hence producing biased gradient estimators) and it has been argued that such methods can attain better numerical performance [70, 71]. There has been a massive research effort to improve the SGD, see, e.g., [65, 68, 69], including second-order extensions [72–75].

An alternative to the gradient-based methods outlined above is the family of *proximal methods* [76, 77]. In order to move along the parameter space, proximal methods regularize the cost function with a term depending on the current value of the estimate of the solution. The optimization procedure resulting from this idea is different from a gradient step and can be viewed as a generalization of the gradient step (see Chapter 2). Proximal methods have received interest over the past decades in the optimization literature; see, e.g., [76–80]. Such methods have particularly attracted interest in signal processing [77] and have been used, e.g., for signal recovery [81], sparse signal processing [82], and machine learning [83]. Similar to the gradient based methods, several extensions to the stochastic setting have been proposed, see, e.g., [84–89].

A typical problem of stochastic optimization methods is that they can be inefficient and unstable depending on the choice of parameters and the problem. It is a well-known fact among practitioners that the performance of the SGD is very sensitive to the step-size parameter and the SGD may diverge easily. To alleviate such problems, a number of schemes were introduced, e.g., AdaGrad [65], Adam [68], RMSprop [90], or AdaDelta [64]; see [69] for a review. Such methods usually precondition the gradient by multiplying it with a (usually diagonal) matrix, which is updated over the iterations of the algorithm. Because methods of this type adapt their parameters to the problem over time, they are also referred to as *adaptive stochastic optimization* methods [69]. However, despite such methods have convergence guarantees and practitioners have derived empirical rules to choose their parameters, the meaning of the parameters is not well-understood. One way to understand the role played by these quantities is to develop a probabilistic interpretation of the stochastic optimization problem and seeing these algorithms as *inference* methods. In recent years, this view has proved fruitful; see, e.g., [91–93]. This line of investigation is also related to the recently emerging field of *probabilistic numerical methods* [94, 95], which comprehends a class of numerical algorithms that propagate and quantify the uncertainty over the solutions

of numerical problems. Provided that the stochastic optimization problem can be recast as an inference problem, one can then interpret the existing adaptive stochastic optimization methods as mimicking optimal inference methods. Therefore, one can also identify potential problems of existing methods by comparing them to optimal inference rules.

A probabilistic interpretation of the stochastic optimization problem also opens up new paths for developing sampling-based optimization schemes. Although we have highlighted the successes of gradient-based stochastic optimization methods above, the gradient information may not be always available, however, due to different reasons. For example, in an engineering application, the system to be optimized might be a black-box, e.g., a piece of closed software code with free parameters, which can be *evaluated* but cannot be differentiated [96]. In these cases, one needs to use a *gradient-free optimization* scheme, meaning that the scheme must rely only on *function evaluations*, rather than any sort of actual gradient information. Classical gradient-free optimization methods have attracted significant interest over the past decades [97, 98]. These methods proceed either by a random search which is based on evaluating the cost function at random points and update the parameter whenever a descent in the function evaluation is achieved [97], or by constructing a numerical (finite-difference type) approximation of the gradient that can be used to take a descent step [96].

The techniques in [96–98] and others of the same class are not applicable, however, if one can only obtain noisy function evaluations or one can only evaluate certain subsets of component functions. In this case, since the function evaluations are not exact, random search methods cannot be used reliably. To address this problem, in recent years, a number of gradient-free stochastic optimization methods have been proposed, see, e.g., [99–102]. Similar to the classical case, these methods are based on the use of noisy function evaluations in order to construct a finite-difference type approximation of the gradient. However, when the cost function has multiple minima or has some regions where the gradients are nearly zero, these methods may suffer from poor numerical performance. In particular, the optimizer can get stuck in a local minimum easily, due to its reliance on gradient approximations. Moreover, when the gradient contains little information about any minimum (e.g., in flat regions), gradient-free stochastic optimizers (as well as perfect gradient schemes) can suffer from slow convergence.

An alternative to constructing a numerical approximation of the gradient is to build up a probability measure endowed with a probability density function (pdf) whose maxima coincide with the minima of the cost function. In this way, the optimization problem can be recast as an inference problem; e.g. [103]. Indeed, one

can then resort to a set of sampling techniques to obtain the probability measure and then estimate the maxima of its pdf. This approach has the advantage of enabling the reconstruction of multiple minima and finding a global minimum, since the probability measure is matched to the cost function. Methods of this nature have long been considered in the literature, including simulated annealing [104], Monte Carlo expectation maximization [105], Markov chain Monte Carlo (MCMC) based methods [106] or methods using sequential Monte Carlo (SMC) [107, 108, 103]. These methods have been restricted to the case where one can utilize the exact function evaluation to assess the quality of each sample. The stochastic setting, where it is only possible to compute noisy evaluations of $f(\theta)$, has also received some attention, see, e.g., [109] for a survey. However, the methods reviewed in [109] contain gradient-based Monte Carlo estimators and address a different class of stochastic optimizaton problems, where the cost function itself is defined as an *expectation*, rather than a *finite-sum* as in (1.1). In recent years, extensions of MCMC based sampling methods have been developed in order to sample from pdfs whose minima match those of a function in the form of eq. (1.1), see, e.g., [110, 111]. However, these schemes rely on the computation of noisy gradients, which we herein assume is not possible. There are also other tecniques using MCMC (see, e.g., [112] which employs noisy Metropolis steps) which do not require gradients. However, these techniques are primarily designed as sampling algorithms, rather than optimization methods. A perspective which is closer to our approach in this thesis was taken in [113], where an adaptive importance sampler was developed using subsampling to compute biased weights. However, the method in [113] lacks convergence guarantees.

We summarize our contributions to stochastic optimization in Section 1.2.

1.2 Organization and contributions

As explained in Section 1.1, we are interested in a broad class of problems in this thesis. The particular contributions we have attained can be summarized as follows.

In **Chapter 2**, titled **Fundamentals of inference and optimization**, we survey statistical inference and optimization methods which are of interest to us in this thesis. In particular, we survey Gaussian inference rules and Monte Carlo statistical inference methods for static and sequential settings. We provide basic convergence theory for the Monte Carlo schemes with self-contained proofs provided in Appendix A.1. We also provide a brief survey of the optimization algorithms which are of interest to us in this work.

In **Chapter 3**, titled **Nudging the particle filter**, we propose a particle filter, termed *the nudged particle filter* (NuPF), aimed at improving the performance of particle filters whenever (a) there is a significant mismatch between the assumed model dynamics and the actual system, or (b) the posterior probability tends to concentrate in relatively small regions of the state space. The proposed scheme pushes some particles towards specific regions where the likelihood is expected to be high, an operation known as *nudging* in the geophysics literature. We prove analytically that nudged particle filters can still attain asymptotic convergence with the same error rates as conventional particle methods while numerical experiments show that they can perform very efficiently in practice. We also provide an alternative interpretation of the nudging scheme as a conventional SMC algorithm applied to a modified dynamical model.

In **Chapter 4**, titled **Stochastic optimization as Bayesian inference**, we develop an interpretation of stochastic optimization problems as inference problems. In particular, we frame stochastic optimization problems as sequential Bayesian inference problems. We then identify this connection in a Gaussian setting, by showing that Kalman and extended Kalman filters can be shown to match a well-known stochastic optimization algorithm, the incremental proximal method [84]. Then, using the probabilistic interpretation of stochastic optimization, we develop a Monte Carlo method, namely a *parallel sequential Monte Carlo optimizer* (PSMCO), to solve general stochastic global optimization problems where the cost function is built up by many components. We prove analytically that this sampler converges to a global maximum. Finally, we demonstrate the utility of the proposed scheme on two challenging stochastic optimization problems.

In **Chapter 5**, titled **Dictionary filtering**, we investigate links between matrix factorisation algorithms and recursive linear filters. In particular, we describe a probabilistic model in which probabilistic inference naturally leads to a matrix factorisation procedure. Using this probabilistic model, we derive a *matrix-variate recursive linear filter* that can be run efficiently in high dimensional settings and leads to the factorisation of the data matrix into a dictionary matrix and a coefficient matrix. The resulting algorithm is inherently online and we refer to it as the *dictionary filter*. Numerical results are provided for image restoration and video modeling problems. We also show that the algorithm bears relationships to stochastic optimization-based schemes, hence this algorithm can be seen as an application of the concepts developed in Chapter 4.

Finally, in **Chapter 6**, titled **Conclusions**, we summarize our contributions and lay out the plans for future work.

The contributions of this thesis have been published in [114, 115, 91, 116, 117].

1.3 Notation and preliminaries

1.3.1 Notation

We denote the set of real numbers as $\mathbb{R}$, while $\mathbb{R}^d = \mathbb{R} \times \overset{d}{\cdots} \times \mathbb{R}$ is the space of d-dimensional real vectors. We denote the set of positive integers with $\mathbb{N}$ and the set of positive reals with $\mathbb{R}_+$. We represent the state space with $\mathsf{X} \subset \mathbb{R}^{d_x}$ and the observation space with $\mathsf{Y} \subset \mathbb{R}^{d_y}$.

In order to denote sequences, we use the shorthand notation $x_{i_1:i_2} = \{x_{i_1}, \ldots, x_{i_2}\}$. For sets of integers, we use $[n] = \{1, \ldots, n\}$. The Euclidean p-norm of a vector $x \in \mathbb{R}^d$ is defined by $\|x\|_p = (x_1^p + \cdots + x_d^p)^{1/p}$. The supremum norm of a real function $\varphi : \mathsf{X} \to \mathbb{R}$ is denoted $\|\varphi\|_\infty = \sup_{x \in \mathsf{X}} |\varphi(x)|$. The L_p norm of a random variable z with probability density function (pdf) $p(z)$ is denoted $\|z\|_p$ and defined as

$$\|z\|_p = \left(\int |z|^p p(z) \mathrm{d}z \right)^{1/p},$$

for $p \geq 1$. It is important to observe that L_p norm of a random variable is directly related to its expectation: $\mathbb{E}[|z|^p] = \|z\|_p^p$. A function is bounded if $\|\varphi\|_\infty < \infty$ and we indicate the space of real bounded functions $\mathsf{X} \to \mathbb{R}$ as $B(\mathsf{X})$.

The set of probability measures on X is denoted $\mathcal{P}(\mathsf{X})$, the Borel σ-algebra of subsets of X is denoted $\mathcal{B}(\mathsf{X})$ and the integral of a function $\varphi : \mathsf{X} \to \mathbb{R}$ with respect to (w.r.t.) a measure μ on the measurable space $(\mathsf{X}, \mathcal{B}(\mathsf{X}))$ is denoted

$$(\varphi, \mu) := \int \varphi(x) \mu(\mathrm{d}x).$$

The unit Dirac delta measure located at $x' \in \mathbb{R}^d$ is denoted $\delta_{x'}(\mathrm{d}x)$ and we note that $\int \varphi(x) \delta_{x'}(\mathrm{d}x) = \varphi(x')$ for any $\varphi \in B(\mathsf{X})$. The Monte Carlo approximation of a measure μ constructed using N samples is denoted as μ^N. More precisely, given N samples $\{x^{(i)}\}_{i=1}^N \sim \mu$, we write the empirical random measure μ^N associated to μ as,

$$\mu^N(\mathrm{d}x) = \frac{1}{N} \sum_{i=1}^N \delta_{x^{(i)}}(\mathrm{d}x).$$

Given a Markov kernel $\tau(\mathrm{d}x'|x)$ and a measure $\pi(\mathrm{d}x)$, we define the notation $\xi(\mathrm{d}x') = \tau\pi(\mathrm{d}x') \triangleq \int \tau(\mathrm{d}x'|x)\pi(\mathrm{d}x)$. We say that π is absolutely continuous w.r.t. q, and write $\pi \ll q$, if $q(A) = 0$ implies $\pi(A) = 0$ for any $A \in \mathcal{B}(\mathsf{X})$.

We often abuse the notation and denote the probability measures and their densities, i.e., probability density functions (pdfs), with respect to the Lebesgue measure with same letters. For instance, while $\mu(\mathrm{d}x)$ denotes a probability measure, we use the notation $\mu(x)$ to denote its density. Whether a given object, say

μ, denotes a probability measure or a pdf should be clarified by the context and the notation for the argument ($\mu(\mathrm{d}x)$ or $\mu(A)$, for a set A versus $\mu(x)$ for a point x).

Let $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}^* \times \cdots \times \mathbb{N}^*$, where $\mathbb{N}^* = \mathbb{N} \cup \{0\}$, be a multi-index. We define the partial derivative operator D^α as

$$\mathsf{D}^\alpha h = \frac{\partial^{\alpha_1} \cdots \partial^{\alpha_d} h}{\partial \theta_1^{\alpha_1} \cdots \partial \theta_d^{\alpha_d}}$$

for a sufficiently differentiable function $h : \mathbb{R}^d \to \mathbb{R}$. We use $|\alpha| = \sum_{i=1}^d \alpha_i$ to denote the order of the derivative. Finally, the notation $\lfloor x \rfloor$ indicates the floor function for a real number x, which returns the biggest integer $k \leq x$.

Matrix notations

Throughout the thesis, I_d denotes the $d \times d$ identity matrix and $\mathrm{vec}(\cdot)$ stands for the column-vectorisation operation. Specifically, for an $m \times r$ matrix A, $a = \mathrm{vec}(A)$ is an $mr \times 1$ vector constructed by stacking the columns of A. To revert this operation, we define the reshaping operator $A = \mathrm{vec}^{-1}_{m \times r}(a)$, where the subscript indicates the dimension of A. We usually denote vectors with lower-case letters and matrices with capital letters.

We recall some useful equalities below. Let A be of dimension $m \times r$ and B be of dimension $r \times n$; then [118],

$$\mathrm{vec}(AXB) = (B^\top \otimes A)\mathrm{vec}(X). \tag{1.2}$$

where $\otimes$ denotes the Kronecker product [118]. As a particular case, for an $r \times 1$ vector x we have

$$Ax = \mathrm{vec}(Ax) = (x^\top \otimes I_m)\mathrm{vec}(A). \tag{1.3}$$

We also draw from properties of the Kronecker product, namely the mixed product property [118],

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD), \tag{1.4}$$

and the inversion property [118],

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \tag{1.5}$$

Notation for distributions

We often rely on an argument-wise notation for probability density functions. Given two random vectors (r.v.'s) x and y, $p(x)$ denotes the pdf of x and $p(y)$ denotes the, possibly different, pdf of y. Similarly, we use $p(x, y)$ for the joint pdf of the two r.v.'s and $p(y|x)$ is the conditional density of y given x.

The Gaussian (normal) probability density evaluated at x with mean m and covariance matrix C is denoted $\mathcal{N}(x; m, C)$. The Student's-t distribution evaluated at x with mean m, scaling parameter σ^2, and degree of freedom ν is denoted $\mathcal{T}(x; m, \sigma^2, \nu)$. The Gamma distribution with parameters (a, θ) is denoted $\mathcal{G}(a, \theta)$ and the Beta distribution with parameters (α, β) is denoted $\mathcal{B}(\alpha, \beta)$.

1.3.2 Preliminary definitions

In this section, we list some preliminary definitions. The following definitions about Markov chain can be found in, e.g., [105].

Definition 1.1. (Markov kernel) A *Markov transition kernel* $\kappa(\cdot|\cdot)$ defined on $\mathsf{X} \times \mathcal{B}(\mathsf{X})$ is a function such that,

(i) For every $x \in \mathsf{X}$, $\kappa(\cdot|x)$ is a probability measure,

(ii) For every $A \in \mathcal{B}(\mathsf{X})$, $\kappa(A|\cdot)$ is measurable.

Next, Markov chains are defined using Markov kernels.

Definition 1.2. (Markov chain) Consider a probability space $(\mathsf{X}, \mathcal{B}(\mathsf{X}), \mathbb{P})$. A sequence of random variables $x_0, \ldots, x_t$ forms a *Markov chain*, if for some transition kernel κ, the following holds,

$$\mathbb{P}(x_{t+1} \in A|x_0, \ldots, x_t) = \mathbb{P}(x_{t+1} \in A|x_t) = \int_A \kappa(\mathrm{d}x_{t+1}|x_t).$$

Definition 1.3. (Convexity) A function $f : \mathsf{X} \to \mathbb{R}$ is said to be convex if, for any $x, y \in \mathsf{X}$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y),$$

for every $\lambda \in [0, 1]$.

2

Fundamentals of inference and optimization

In this section, we provide an introduction to inference and optimization, together with a brief survey of methods which are commonly used in the field and, particularly, in the rest of this thesis. In particular, we review some exact inference methods, Monte Carlo methods for statistical inference, and numerical optimization algorithms.

2.1 Introduction

2.1.1 Bayesian update

The problem of Bayesian inference starts with a *probabilistic model*. The model is usually in the form of a collection of probability distributions. In its most simple form, we are given a model that consists of a prior distribution $\pi_0(x)$, which reflects the belief of the model designer about the variable x, and a likelihood function $g(y|x)$, which characterizes the relationship between data and the quantity of interest. Given the model, a typical interest is the computation of the posterior

density, denoted $\pi(x|y)$, using the Bayes' rule,

$$\pi(x|y) = \pi_0(x)\frac{g(y|x)}{\int_{\mathsf{X}} g(y|x)\pi_0(x)\mathrm{d}x}.$$

Bayes' rule was first proposed by Thomas Bayes in his 1763 essay titled *An Essay towards solving a Problem in the Doctrine of Chances* [119]. However, the modern form of the rule, as written above, is due to Laplace [120], and introduced in 1820.

More formally, let us assume we are given a prior measure π_0 defined on a space $\mathsf{X} \subset \mathbb{R}^{d_x}$ with a *likelihood function* $g(y|x) : \mathsf{X} \to \mathbb{R}$, where y is fixed. The posterior probability measure in this case is given by

$$\pi(\mathrm{d}x) = \pi_0(\mathrm{d}x)\frac{g(y|x)}{\int_{\mathsf{X}} g(y|x)\pi_0(\mathrm{d}x)}. \tag{2.1}$$

In this thesis, we use the measure and density notation interchangeably and we will denote the densities and measures with the same letters as indicated in the notation section.

Often, the goal in Bayesian inference is to compute expectations of test functions $\varphi \in B(\mathsf{X})$ with respect to the posterior distribution π. Given $\varphi \in B(\mathsf{X})$, it is useful express the update in terms of expectations using (2.1), i.e.,

$$(\varphi, \pi) = \frac{(\varphi g, \pi_0)}{(g, \pi_0)}, \tag{2.2}$$

which is the Bayes update in integral form.

We also note that the update (2.2) can be defined for general positive real-valued functions $G : \mathsf{X} \to \mathbb{R}_+$, termed *potential functions*, which does not necessarily correspond to any likelihood function associated to an observation. This view is pertinent in several fields of physics, e.g., statistical mechanics or computational physics, where the dynamics of objects are defined via potential functions. In recent years, this view has been quite popular for the analysis of Monte Carlo methods as well [33, 121]. In this case, the update can be written as

$$\pi(\mathrm{d}x) = \pi_0(\mathrm{d}x)\frac{G(x)}{\int_{\mathsf{X}} G(x)\pi_0(\mathrm{d}x)} \tag{2.3}$$

for a potential $G : \mathsf{X} \to \mathbb{R}_+$. This view will be useful for us when we deal with optimization problems in a probabilistic setup. Also, it is customary to write the likelihood functions by dropping the observation dependence from the notation, as we will see in the next section.

2.1.2 Sequential Bayesian update for static models

In some models, which are amenable to sequential computation, one can use the Bayes' rule described above sequentially. The principle of sequential Bayesian inference is that *the posterior becomes the new prior* over the iterations.

Let us assume in this case that we have a sequence of observations $(y_t)_{t\geq 1}$ each defining a likelihood function, which we denote as[1] $g_t(x) := g_t(y_t|x)$. We also denote the prior measure as π_0. Given the posterior distribution $\pi_{t-1}(\mathrm{d}x)$ conditioned on $y_{1:t-1}$, the update is given as

$$\pi_t(\mathrm{d}x) = \pi_{t-1}(\mathrm{d}x)\frac{g_t(x)}{\int_{\mathsf{X}} g_t(x)\pi_{t-1}(\mathrm{d}x)}. \tag{2.4}$$

One can also write this update in terms of expectations, similar to (2.2),

$$(\varphi, \pi_t) = \frac{(\varphi g_t, \pi_{t-1})}{(g_t, \pi_{t-1})}. \tag{2.5}$$

The expressions (2.4) and (2.5) cannot be computed in closed form in general. In this static setup, this computation can only be done exactly for models that retain certain conjugacy properties. An example we will use repeatedly is the case where π_0 and g_t are both Gaussian. In this case, the posterior distribution is also Gaussian [5], and its moments can be sequentially computed in closed form, see Section 2.3.1.

2.1.3 State-space models and the filtering problem

In this section we introduce the class of state-space models (SSMs). An SSM consists of two stochastic processes: the *hidden state process* $(x_t)_{t\geq 0}$ and the *observation process* $(y_t)_{t\geq 1}$. The hidden state is a Markov process whereas the observations are conditionally independent given the hidden state process. Compactly, we can write an SSM as

$$x_0 \sim \pi_0(\mathrm{d}x_0), \tag{2.6}$$

$$x_t|x_{t-1} \sim \tau_t(\mathrm{d}x_t|x_{t-1}), \tag{2.7}$$

$$y_t|x_t \sim g_t(y_t|x_t), \quad t \in \mathbb{N}, \tag{2.8}$$

where $x_t \in \mathsf{X} \subset \mathbb{R}^{d_x}$ is the d_x-dimensional system state at time t, $y_t \in \mathsf{Y} \subset \mathbb{R}^{d_y}$ is the t-th d_y-dimensional observation, the measure π_0 describes the prior probability distribution of the initial state, τ_t is a Markov transition kernel on X, and $g_t(y_t|x_t)$

[1] We often drop the observation from the notation, called implicit conditioning, which clarifies the notation significantly.

is the (possibly non-normalized) pdf of the observation y_t conditional on the state x_t. We assume the observation sequence $(y_t)_{t\geq 1}$ is arbitrary but fixed. Hence, it is convenient to think of the conditional pdf g_t as a likelihood function and we write $g_t(x_t) := g_t(y_t|x_t)$ for conciseness. Also from now on, we also denote SSMs compactly as $(\pi_0, \tau_t, g_t)_{t\geq 1}$.

We are interested in the sequence of posterior probability distributions $(\pi_t)_{t\geq 1}$ of the states generated by the SSM. To be specific, at each time $t = 1, 2, ...$ we aim at computing (or, at least, approximating) the probability measure π_t which describes the probability distribution of the state x_t conditional on the observation of the sequence $y_{1:t}$. When it exists, we use $\pi(x_t|y_{1:t})$ to denote the pdf of x_t given $y_{1:t}$ with respect to the Lebesgue measure, i.e., $\pi_t(\mathrm{d}x_t) = \pi(x_t|y_{1:t})\mathrm{d}x_t$. The measure π_t is often termed the *optimal filter* at time t. The computation of the sequence of measures $(\pi_t)_{t\geq 1}$ is referred to as *the filtering problem*, as introduced in Chapter 1.

The optimal filter π_t is closely related to another probability measure ξ_t, which describes the probability distribution of the state x_t conditional on $y_{1:t-1}$ and it is, therefore, termed the *predictive* measure at time t. As for the case of the optimal filter, we use $\xi(x_t|y_{1:t-1})$ to denote the pdf (when it exists), with respect to the Lebesgue measure, of x_t given $y_{1:t-1}$. The predictive measure and the optimal filter can be constructed recursively. Assume that we are given the optimal filter π_{t-1}. In order to obtain the filter π_t, one needs to first compute the predictive measure ξ_t via the Chapman-Kolmogorov equation [2, 122, 123],

$$\xi_t(\mathrm{d}x_t) = \int \tau_t(\mathrm{d}x_t|x_{t-1})\pi_{t-1}(\mathrm{d}x_{t-1}). \tag{2.9}$$

Note that we can similarly write the prediction Eq. (2.9) for the expectation of a test function $\varphi \in B(\mathsf{X})$, as

$$(\varphi, \xi_t) = ((\varphi, \tau_t), \pi_{t-1}).$$

Once we obtain the predictive measure ξ_t, we can compute the filter at time t as

$$\pi_t(\mathrm{d}x_t) = \xi_t(\mathrm{d}x_t)\frac{g_t(x_t)}{\int_{\mathsf{X}} g_t(x_t)\xi_t(\mathrm{d}x_t)}. \tag{2.10}$$

Similarly to (2.5), we can write the update Eq. (2.10) as

$$(\varphi, \pi_t) = \frac{(\varphi g_t, \xi_t)}{(g_t, \xi_t)} \tag{2.11}$$

for any test function $\varphi \in B(\mathsf{X})$. In general, the recursions (2.9) and (2.10), or the expectations with respect to them, are not available in closed form, apart

from a restricted class of models, e.g., models with linear-Gaussian transitions and likelihoods (for which the Kalman filter [6] yields exact solutions). We will review numerical methods for both Gaussian and general cases in the following sections.

2.2 Static inference

In this section, we review numerical methods for the problem of inferring a posterior probability measure π and then compute integrals of the form

$$(\varphi, \pi) = \int_{\mathsf{X}} \varphi(x)\pi(\mathrm{d}x), \tag{2.12}$$

which are analytically intractable in general. Specifically, we review Monte Carlo (MC) methods which aim at constructing a random grid by sampling from the base distribution $\pi(\mathrm{d}x)$ and then approximating the integral using these samples [23]. Moreover, since it is often the case that we cannot sample from π directly, it is crucial to devise schemes that generate a suitable random grid by other means. We remark that the integral (2.12) is directly related to the problem of Bayesian inference since it is often of crucial interest to estimate integrals with respect to posterior distributions.

2.2.1 Exact inference for Gaussian distributions

As indicated in the previous section, for special cases it is possible to obtain the posterior distribution using the Bayes' rule. An important case is the one where *conjugate priors* are chosen for specific likelihoods, in such a way that the posterior distribution can be obtained in closed form and usually in the form of the prior [124, 125]. In this section, we review one specific (but important) case where exact inference is possible. More specifically, we focus on the case where the prior distribution and the likelihood function are chosen as Gaussian and free from nonlinearities in their moments.

In this section, we summarize the inference rules for a Gaussian likelihood and prior. In particular, we assume that we have a model

$$\pi_0(x) = \mathcal{N}(x; \mu_0, V_0), \tag{2.13}$$

$$g(y|x) = \mathcal{N}(y; Hx, R), \tag{2.14}$$

where $\mu_0 \in \mathbb{R}^{d_x}$ and $V_0 \in \mathbb{R}^{d_x \times d_x}$ are the mean and covariance parameters of the prior, respectively, $H \in \mathbb{R}^{d_y \times d_x}$ is the observation matrix, and $R \in \mathbb{R}^{d_x \times d_x}$ is the observation covariance matrix. In this case, the posterior pdf $\pi(x|y)$ can be computed in closed form and it is given by the following lemma [125, 126].

Lemma 2.1. *Assume that the prior and the likelihood are defined as in Eqs. (2.13)–(2.14). Then the posterior distribution is Gaussian, specifically*

$$\pi(x|y) = \mathcal{N}(x; \mu, V),$$

where

$$\begin{aligned} \mu &= \mu_0 + V_0 H^\top (R + H V_0 H^\top)^{-1}(y - H\mu_0), \qquad \text{and} \\ V &= V_0 - V_0 H^\top (R + H V_0 H^\top)^{-1} H V_0. \end{aligned}$$

Proof. See, e.g., [125, 126]. □

This Gaussian update can be implemented recursively when one has a data sequence $(y_t)_{t\geq 0}$. This is the key operation of the Kalman filter, which we will review in Section 2.3.1. Given π, one can compute integrals of the form (2.12) either in closed form or by using perfect sampling (which is introduced in the next section) if the test function φ prevents the exact computation of the integral (2.12).

2.2.2 Perfect Monte Carlo

The numerical estimation of the integral in (2.12) is the simplest when we can sample from π directly. Using independent samples from a certain probability measure to calculate integrals is the core principle of Monte Carlo methods [23]. When independent sampling is achievable, the integral (2.12) can be estimated efficiently. This can be done in various ways, e.g., via the inversion method (provided that one has an access to a uniform random number generator and the inverse of the cdf for the target distribution) or various other schemes which makes it possible to obtain i.i.d. samples from π [127, 128]. In this subsection, we just assume that we have access to an i.i.d. sampling scheme for π.

One can approximate (2.12) using samples from π and this type of approximations are called Monte Carlo (or particle) approximations of the integral. Formally, given a collection of samples $\{x^{(i)}\}_{i=1}^N$, where $x^{(i)} \sim \pi$ i.i.d, we first construct the empirical measure, i.e., the empirical approximation of π, as

$$\pi^N(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^N \delta_{x^{(i)}}(\mathrm{d}x). \tag{2.15}$$

Given the empirical measure π^N, the integral (2.12) can be approximated as

$$(\varphi, \pi) \approx (\varphi, \pi^N) = \frac{1}{N}\sum_{i=1}^N \varphi(x^{(i)}), \tag{2.16}$$

i.e., we replace a (possibly complicated) integral w.r.t. π by an integral w.r.t. π^N which is straightforward to compute. As a classical result, the estimate (2.16) converges to the true integral by the strong law of large numbers [129], i.e.,

$$\lim_{N\to\infty}(\varphi,\pi^N)=(\varphi,\pi)\quad \text{a.s.} \tag{2.17}$$

provided the variance of $\varphi(x)$ is finite. The Monte Carlo estimator (2.16) is unbiased and its variance is given by [105]

$$\operatorname{var}\left[(\varphi,\pi^N)\right]=\frac{1}{N^2}\sum_{i=1}^{N}\operatorname{var}_\pi[\varphi(x^{(i)})]=\frac{\operatorname{var}_\pi[\varphi(x)]}{N},$$

which vanishes as $N\to\infty$. In the sequel, we prove a well-known L_p convergence result for the case where we can sample from π directly.

Theorem 2.1. *Given a test function $\varphi\in B(\mathsf{X})$, we have the explicit error bound*

$$\left\|(\varphi,\pi)-(\varphi,\pi^N)\right\|_p\leq\frac{c_p\|\varphi\|_\infty}{\sqrt{N}},$$

where $c_p<\infty$ is a constant independent of N.

Proof. A simple proof for $p=2$ and general integer p are both given in Appendix A.1. □

This theorem is a typical result that displays the classical $\mathcal{O}(N^{-1/2})$ Monte Carlo rate of convergence. We will see that many other Monte Carlo methods also preserve this rate. It is also possible to give an explicit convergence rate for the random approximation error (instead of the expected error in Theorem 2.1).

Corollary 2.1. *Assume Theorem 2.1 holds. Then we have the following bound for the random error,*

$$|(\varphi,\pi^N)-(\varphi,\pi)|\leq\frac{U_\delta}{N^{\frac{1}{2}-\delta}}, \tag{2.18}$$

where U_δ is an almost surely (a.s.) finite random variable independent of N and $\delta<\frac{1}{2}$ is an arbitrarily small positive constant. As a consequence,

$$\lim_{N\to\infty}(\varphi,\pi^N)=(\varphi,\pi),\qquad \text{a.s.} \tag{2.19}$$

for any $\varphi\in B(\mathsf{X})$.

Proof. See the Appendix A.1. □

Since $\delta>0$ can be arbitrarily small, the random error $|(\varphi,\pi^N)-(\varphi,\pi)|$ also converges with a rate $\mathcal{O}(N^{-\frac{1}{2}})$. Corollary 2.1 is implied by Theorem 2.1, hence it will be possible to prove the same type of result for other Monte Carlo samplers which converge in L_p.

2.2.3 Importance sampling

In many cases, it is not possible to directly sample from the probability measure π. In that situation, one can use an instrumental distribution, also called a *proposal* distribution, which is easy to sample from in order to obtain samples and use a weighting scheme to obtain a proper approximation. We introduce here the importance sampling (IS) method, which utilizes this idea. The IS method was first introduced in [130]. Since then, it has been one of the most popular Monte Carlo integration methods. For the basic theory and some applications see, e.g., [105, 123].

We denote the proposal measure with $q(\mathrm{d}x)$ (whose density w.r.t. Lebesgue measure will be denoted by q) and we assume that it is easy to sample from it directly. Formally, we assume that the target is absolutely continuous w.r.t. the proposal, i.e., $\pi \ll q$. Given that we can sample from q, the IS estimate is constructed as follows. First recall that our aim is to estimate integrals of form (2.12). Provided that $\pi \ll q$, we readily have,

$$(\varphi, \pi) = \int_{\mathsf{X}} \varphi(x) \frac{\mathrm{d}\pi}{\mathrm{d}q}(x) q(\mathrm{d}x) := \int_{\mathsf{X}} \varphi(x) w(x) q(\mathrm{d}x), \tag{2.20}$$

where the Radon-Nikodym derivative $\frac{\mathrm{d}\pi}{\mathrm{d}q} : \mathsf{X} \to \mathbb{R}_+$ is also referred to as the weight function and denoted as $w(x)$. Assuming that both π and q have densities w.r.t. Lebesgue measure (denoted with same letters), we can write

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

Now the integral in (2.20) can be estimated using the empirical measures

$$q^N(\mathrm{d}x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^{(i)}}(\mathrm{d}x) \quad \text{and} \quad \pi^N(\mathrm{d}x) = \frac{1}{N} \sum_{i=1}^{N} \mathsf{w}^{(i)} \delta_{x^{(i)}}(\mathrm{d}x) \tag{2.21}$$

where $x^{(i)} \sim q(\mathrm{d}x)$ i.i.d. for $i = 1, \ldots, N$ and $\mathsf{w}^{(i)} = w(x^{(i)})$ are called importance weights. We denote the IS estimate in this case as

$$(\varphi, \pi^N) = \frac{1}{N} \sum_{i=1}^{N} \mathsf{w}^{(i)} \varphi(x^{(i)}). \tag{2.22}$$

Remark 2.1. The estimator (2.22) is an unbiased estimator of the integral in (2.20), as shown easily by

$$\mathbb{E}[(\varphi, \pi^N)] = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[w(x)\varphi(x)] = \mathbb{E}_\pi[\varphi(x)] = (\varphi, \pi).$$

□

Algorithm 2.1 Self-normalized importance sampling

1: Sample from proposal $x^{(i)} \sim q(\mathrm{d}x)$ for $i = 1, \ldots, N$.

2: Compute weights,

$$\mathsf{w}^{(i)} = \frac{W^{(i)}}{\sum_{i=1}^{N} W^{(i)}} \quad \text{where } W^{(i)} = \frac{\Pi(x^{(i)})}{q(x^{(i)})}, \quad \text{for} \quad i = 1, \ldots, N.$$

3: Report,

$$\tilde{\pi}^N(\mathrm{d}x) = \sum_{i=1}^{N} \mathsf{w}^{(i)} \delta_{x^{(i)}}(\mathrm{d}x) \quad \text{and} \quad (\varphi, \tilde{\pi}^N) = \sum w^{(i)} \varphi(x^{(i)}).$$

The unbiasedness property of the IS estimator only holds for the case where one can evaluate $\pi(x)$ exactly. In the next section, we review the IS scheme for the (practically more relevant) case in which one can only evaluate $\pi(x)$ up to a normalization constant.

Self-normalized importance sampling

While the IS scheme we have introduced is convenient, it is often not possible to implement it exactly as it requires evaluations of the target density π. In practice, the target π is only known up to a normalization constant, denoted Z_π. The unnormalized target is denoted with Π and we can write

$$\pi(\mathrm{d}x) = \frac{\Pi(\mathrm{d}x)}{Z_\pi} \quad \text{and} \quad \pi(x) = \frac{\Pi(x)}{Z_\pi}$$

for the target measure and the target pdf, respectively, where

$$Z_\pi = \int_{\mathsf{X}} \Pi(\mathrm{d}x) = \Pi(\mathsf{X}).$$

Let us again assume $\Pi \ll q$. In this case, we can rewrite the integral (φ, π) as

$$(\varphi, \pi) = \frac{1}{Z_\pi} \int_{\mathsf{X}} \varphi(x) \Pi(\mathrm{d}x).$$

In a similar way to the derivation in (2.20), we write,

$$(\varphi, \pi) = \frac{\int_{\mathsf{X}} \varphi(x) \frac{\mathrm{d}\Pi}{\mathrm{d}q}(x) q(\mathrm{d}x)}{\int_{\mathsf{X}} \frac{\mathrm{d}\Pi}{\mathrm{d}q}(x) q(\mathrm{d}x)} := \frac{\int_{\mathsf{X}} \varphi(x) W(x) q(\mathrm{d}x)}{\int_{\mathsf{X}} W(x) q(\mathrm{d}x)}, \tag{2.23}$$

where now the Radon-Nikodym derivative $\frac{\mathrm{d}\Pi}{\mathrm{d}q} : \mathsf{X} \to \mathbb{R}_+$ is called the unnormalized weight function, which we also denote as $W(x)$. The normalized weight function

$w : \mathsf{X} \to \mathbb{R}_+$, i.e., the Radon-Nikodym derivative of π w.r.t. q, $w(x) := \frac{\mathrm{d}\pi}{\mathrm{d}q}(x)$, can be written in terms of the unnormalized weight function as

$$w(x) := \frac{\mathrm{d}\pi}{\mathrm{d}q}(x) = \frac{W(x)}{\int_{\mathsf{X}} W(x')q(\mathrm{d}x')}.$$

Given these constructions, the IS algorithm for this case can be constructed as follows. Assume that we sample $\{x^{(i)}\}_{i=1}^N$ i.i.d. from $q(\mathrm{d}x)$ and obtain the empirical measure (2.21). Then, (2.23) yields the *self-normalized importance sampling (SNIS) estimator*

$$(\varphi, \tilde{\pi}^N) = \frac{\sum_{i=1}^N W^{(i)}\varphi(x^{(i)})}{\sum_{i=1}^N W^{(i)}},$$

where $W^{(i)} := W(x^{(i)})$ for $i = 1, \ldots, N$. We can write this estimate more compactly as

$$(\varphi, \tilde{\pi}^N) = \sum_{i=1}^N \mathsf{w}^{(i)}\varphi(x^{(i)}), \quad \text{where} \quad \mathsf{w}^{(i)} = \frac{W^{(i)}}{\sum_{i=1}^N W^{(i)}}. \tag{2.24}$$

This estimator is biased as it is defined as a ratio of two unbiased estimators, which follows from Jensen's inequality[2]. Although the estimator is biased for finite N, the bias vanishes as $N \to \infty$ with a rate $\mathcal{O}(1/N)$. Actually, the estimator (2.24) is *consistent* [105]. In what follows, we present an L_p convergence result concerning the estimator (2.24), i.e., an IS version of Theorem 2.1.

Theorem 2.2. *Assume that $\|W\|_\infty < \infty$. Then, for any $\varphi \in B(\mathsf{X})$, we have*

$$\|(\varphi, \pi) - (\varphi, \tilde{\pi}^N)\|_p \leq \frac{c_p\|\varphi\|_\infty}{\sqrt{N}}$$

for any $p \geq 1$, where c_p is a constant independent of N.

Proof. See Appendix A.1. □

Similarly to the perfect Monte Carlo, we have the following result for SNIS estimators.

Corollary 2.2. *Under the assumptions of Theorem 2.2, for any $\varphi \in B(\mathsf{X})$ we have*

$$|(\varphi, \tilde{\pi}^N) - (\varphi, \pi)| \leq \frac{\tilde{U}_\delta}{N^{\frac{1}{2}-\delta}}, \tag{2.25}$$

[2]The bias also follows from the fact that $\mathsf{w}^{(i)} \neq w(x^{(i)})$.

where $\tilde{U}_\delta$ is an almost surely finite random variable and $\delta < \frac{1}{2}$ is an arbitrarily small and positive constant independent of N. As a consequence,

$$\lim_{N\to\infty} (\varphi, \tilde{\pi}^N) = (\varphi, \pi) \qquad \text{a.s.} \tag{2.26}$$

for any $\varphi \in B(\mathsf{X})$.

Proof. The proof of this corollary follows from Theorem 2.2 and the proof of Corollary 2.1. □

The SNIS scheme is outlined in Algorithm 2.1.

2.2.4 Markov chain Monte Carlo methods

An alternative to building an importance sampling-type sampler is to use Markov chain Monte Carlo (MCMC) methods. MCMC techniques are based on the principle of simulating a Markov chain $(x_t)_{t\geq 0}$ with a stationary distribution π, which coincides with the target distribution (see [105] for precise definitions of a Markov chain and its stationary distribution). This is done via designing a Markov kernel $\kappa(\cdot|\cdot)$ which leaves π invariant, i.e., $\pi = \kappa\pi$. More precisely, the kernel κ should satisfy,

$$\pi(\mathrm{d}x'|x) = \int \kappa(\mathrm{d}x'|x)\pi(\mathrm{d}x).$$

Provided that one can design such a kernel and obtain random variables $(x_t)_{t\geq 1}$, the marginal distribution of x_t converges to π as $t \to \infty$. For conditions and modes of convergence see [105].

2.3 Sequential inference

In many scenarios for probabilistic inference, such as sequential inference or filtering, it is required to estimate a sequence of distributions $(\pi_t)_{t\geq 0}$ and integrals of the form

$$(\varphi, \pi_t) = \int_{\mathsf{X}} \varphi(x)\pi_t(\mathrm{d}x), \quad \text{for} \quad t \in \mathbb{N}. \tag{2.27}$$

For example, in Bayesian statistics, each π_t is conditioned on more and more data as t grows. In this case, it is computationally prohibitive to use the methods we have introduced in the previous section, as the complexity of the problem grows with the number of iterations. It is therefore crucial to develop methods tailored for the sequential structure of the problem. In the following sections, we review methods for sequential inference.

2.3.1 Kalman update

Assume that we are given a Gaussian prior $\pi_0(x)$ and a sequence of Gaussian likelihoods $g(y_t|x)$ for $t \geq 1$. As we have summarized in Sec. 2.2.1, one-step Bayes update can be performed in closed form when the prior and a single likelihood are Gaussian. It is therefore natural to generalize this idea for sequential Bayesian updating.

More specifically, we consider the following Gaussian probability model,

$$\pi_0(x) = \mathcal{N}(x; \mu_0, V_0) \tag{2.28}$$
$$g(y_t|x) = \mathcal{N}(y_t; H_t x, R_t), \tag{2.29}$$

where (μ_0, V_0) are parameters of the prior, $(H_t)_{t\geq 1}$ defines the sequence of observation matrices, and $(R_t)_{t\geq 1}$ are observation noise covariances. Given the posterior pdf $\pi_t(x|y_{1:t-1})$, the posterior pdf at time t is given by

$$\pi_t(x|y_{1:t}) = \pi_{t-1}(x|y_{1:t-1}) \frac{g_t(y_t|x)}{\int_{\mathsf{X}} \pi_{t-1}(x|y_{1:t-1}) g_t(y_t|x) \mathrm{d}x},$$

where $\pi(x|y_0) := \pi_0(x)$. The parameters of this posterior distribution can be found via the following recursions, which are referred to as *Kalman updates*.

Lemma 2.2. *Given the model (2.28)–(2.29) and the posterior distribution $\pi(x|y_{1:t-1}) = \mathcal{N}(x; \mu_{t-1}, V_{t-1})$, we obtain*

$$\pi(x|y_{1:t}) = \mathcal{N}(x; \mu_t, V_t),$$

where,

$$\mu_t = \mu_{t-1} + V_{t-1} H_t^\top (R_t + H_t V_{t-1} H_t^\top)^{-1} (y_t - H_t \mu_{t-1}),$$
$$V_t = V_{t-1} - V_{t-1} H_t^\top (R_t + H_t V_{t-1} H_t^\top)^{-1} H_t V_{t-1},$$

for $t \geq 1$ and $\pi(x|y_0) := \pi_0(x)$.

Proof. See, e.g., [6, 5, 125, 126]. □

Again, the integrals (φ, π_t) for certain φ can be computed, since π_t can be obtained in closed form. However, again, for general φ, one may need to resort to sampling methods to estimate the integral if it is not possible to integrate φ against a Gaussian.

2.3.2 Kalman filters

Assume that we are given an SSM $(\pi_0, \tau_t, g_t)_{t\geq 1}$ of the form,

$$\pi_0(x_0) = \mathcal{N}(x_0; \mu_0, V_0), \tag{2.30}$$
$$\tau_t(x_t|x_{t-1}) = \mathcal{N}(x_t; A_t x_{t-1} + B_t u_t, Q_t), \tag{2.31}$$
$$g_t(y_t|x_t) = \mathcal{N}(y_t; H_t x_t, R_t), \tag{2.32}$$

where $(A_t)_{t\geq 1} \in \mathbb{R}^{d_x \times d_x}$ and $(Q_t)_{t\geq 1} \in \mathbb{R}^{d_x \times d_x}$ denote the *linear transition matrices* and the *process covariance matrices*, respectively. The sequences $(H_t)_{t\geq 1}$ and $(R_t)_{t\geq 1}$ characterize the observation model. In particular, $(H_t)_{t\geq 1}$ denotes the *linear observation model* and the sequence $(R_t)_{t\geq 1}$ stands for the *observation noise covariance*, respectively. Finally, the sequences $(B_t, u_t)_{t\geq 1}$ define the control inputs. We refer to models of the form (2.30)–(2.32) as linear-Gaussian state-space models (LGSSMs), indicating that the transition and observation models of the SSM are linear-Gaussian. For this type of models, the posterior distribution of the state $\pi_t(x_t|y_{1:t})$ is available in closed form. The procedure to compute the sufficient statistics of $\pi_t(x_t|y_{1:t})$ is called the Kalman filter (KF) [6, 5]. More specifically, the moments of the π_t are given by the following lemma.

Lemma 2.3. *Assume we are given an SSM of form (2.30)–(2.32). Given the optimal filter at time $t-1$,*

$$\pi_{t-1}(x_{t-1}|y_{1:t-1}) = \mathcal{N}(x_{t-1}; \mu_{t-1}, V_{t-1}),$$

the predictive distribution $\xi_t(x_t|y_{1:t-1})$ is given by

$$\xi_t(x_t|y_{1:t-1}) = \mathcal{N}(x_t; \tilde{\mu}_t, \tilde{V}_t),$$

where,

$$\tilde{\mu}_t = A_t \mu_{t-1} + B_t u_t, \tag{2.33}$$
$$\tilde{V}_t = A_t V_{t-1} A_t^\top + Q_t. \tag{2.34}$$

Finally, the optimal filter $\pi_t(x_t|y_{1:t})$ is given by

$$\pi_t(x_t|y_{1:t-1}) = \mathcal{N}(x_t; \mu_t, V_t),$$

where,

$$\mu_t = \tilde{\mu}_t + \tilde{V}_t H_t^\top (R_t + H_t \tilde{V}_t H_t^\top)^{-1}(y_t - H_t \tilde{\mu}_t), \tag{2.35}$$
$$V_t = \tilde{V}_t - \tilde{V}_t H_t^\top (R_t + H_t \tilde{V}_t H_t^\top)^{-1} H_t \tilde{V}_t, \tag{2.36}$$

from Lemma 2.2.

Proof. See, e.g., [6, 5, 125, 126]. □

The recursions (2.33)–(2.36) are together referred to as the *Kalman filtering recursions*.

2.3.3 Extended Kalman filters

Given an SSM where the transition and observation models are nonlinear, it is not possible to utilize the KF. However, it is reasonable to assume that when the transition and observation models are *locally linear*, and the process and the observation noise are Gaussian, one can linearize the SSM and obtain a Kalman-type filter for a nonlinear model, which is called as the extended Kalman filter (EKF) [5]. Note, however, that the distributions approximated by the EKF are just *approximate*. We will make this fact clear by denoting the posterior distributions as π_t^E, instead of π_t. The EKF can be shown to converge if the model satisfies certain conditions, see, e.g., [2] for a discussion.

Assume that we are given the SSM

$$\begin{aligned}\pi_0(x_0) &= \mathcal{N}(x_0; \mu_0, V_0),\\ \tau_t(x_t|x_{t-1}) &= \mathcal{N}(x_t; a_t(x_{t-1}), Q_t)\\ g_t(y_t|x_t) &= \mathcal{N}(y_t; h_t(x_t), R_t).\end{aligned}$$

where $a_t : \mathsf{X} \to \mathsf{X}$, $h_t : \mathsf{X} \to \mathsf{Y}$, $Q_t \in \mathbb{R}^{d_x \times d_x}$, and $R_t \in \mathbb{R}^{d_y \times d_y}$. Assume that the approximate posterior distribution at time $t-1$ is $\pi_{t-1}^E(x_{t-1}) = \mathcal{N}(x_{t-1}; \mu_{t-1}^E, V_{t-1}^E)$. If the model is approximately locally linear, one can linearize $a_t(x_t)$ around μ_{t-1}^E and obtain the dynamical model

$$\bar{a}_t(x_t) = a_t(\mu_{t-1}^E) + A_t(x_t - \mu_{t-1}^E) = a_t(\mu_{t-1}^E) + A_t x_t - A_t \mu_{t-1}^E, \tag{2.37}$$

where

$$A_t = \left.\frac{\partial a_t(x)}{\partial x}\right|_{x=\mu_{t-1}^E}.$$

We can see (2.37) as a linear model with control inputs. Hence, the prediction step with this linearized model simply becomes

$$\tilde{\mu}_t^E = a_t(\mu_{t-1}).$$

The uncertainty is propagated also as in the KF, since (2.37) is a linear model, hence we obtain

$$\tilde{V}_t^E = A_t V_{t-1}^E A_t^\top + Q_t.$$

Similarly, given $\tilde{\mu}_t^E$, in order to proceed with the observation model we can linearize h_t around $\tilde{\mu}_t^E$, i.e., we construct

$$\bar{h}_t(x_t) = h_t(\tilde{\mu}_t^E) + H_t(x_t - \tilde{\mu}_t^E),$$

where

$$H_t = \left.\frac{\partial h_t(x)}{\partial x}\right|_{x=\tilde{\mu}_t}.$$

Given the linearization, the EKF update step now becomes

$$\begin{aligned}
\mu_t^E &= \tilde{\mu}_{t-1}^E + \tilde{V}_t^E H_t^\top (R_t + H_t \tilde{V}_t^E H_t^\top)^{-1}(y_t - h_t(\tilde{\mu}_t^E)),\\
V_t^E &= \tilde{V}_t^E - \tilde{V}_t^E H_t^\top (R_t + H_t \tilde{V}_t^E H_t^\top)^{-1} H_t \tilde{V}_t^E.
\end{aligned}$$

Finally, one can compactly summarize the EKF as follows. Given $\pi_{t-1}^E(x_{t-1}) = \mathcal{N}(x_{t-1}; \mu_{t-1}^E, V_{t-1}^E)$, the new posterior pdf $\pi_t^E(x_t) = \mathcal{N}(x_t; \mu_t^E, V_t^E)$ is obtained via

$$\tilde{\mu}_t^E = a_t(\mu_{t-1}), \tag{2.38}$$
$$\tilde{V}_t^E = A_t V_{t-1}^E A_t^\top + Q_t, \tag{2.39}$$
$$\mu_t^E = \tilde{\mu}_{t-1}^E + \tilde{V}_t^E H_t^\top (R_t + H_t \tilde{V}_t^E H_t^\top)^{-1}(y_t - h_t(\tilde{\mu}_t^E)), \tag{2.40}$$
$$V_t^E = \tilde{V}_t^E - \tilde{V}_t^E H_t^\top (R_t + H_t \tilde{V}_t^E H_t^\top)^{-1} H_t \tilde{V}_t^E. \tag{2.41}$$

From now on, the equations (2.38)–(2.41) for $t \geq 1$ are referred to as the EKF recursions.

2.3.4 Other Kalman-type filters

The EKF is not the only way to deal with nonlinearities when one is interested in inferring nonlinear Gaussian state-space models. Actually, when there are severe nonlinearities present in an SSM, the EKF may not be suitable to use. Furthermore, as it requires derivatives of the transition and observation models, it may not be applicable to certain models.

An alternative approach to obtain a Kalman-like filtering algorithm for a nonlinear SSM is the so-called unscented Kalman filter (UKF) [12–14]. These methods define a precomputed grid, which is designed so that the moments before and after the nonlinear transformations stay consistent. There is a class of similar methods called sigma-point filters, which rely on the same principle [126]. The UKF can be successfully applied for cases where derivatives of the models are not available. The second Kalman-type filter, which is widely used by weather prediction practitioners, is the ensemble Kalman filter (EnKF). The EnKF resembles a Monte

Algorithm 2.2 Bootstrap Particle Filter

1: Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
2: **for** $t \geq 1$ **do**
3: Sampling: draw $\bar{x}_t^{(i)} \sim \tau_t(\mathrm{d}x_t|x_{t-1}^{(i)})$ independently for every $i = 1, \ldots, N$.
4: Weighting: compute $\mathsf{w}_t^{(i)} = g_t(\bar{x}_t^{(i)})/\bar{Z}_t^N$ for every $i = 1, \ldots, N$, where $\bar{Z}_t^N = \sum_{i=1}^N g_t(\bar{x}_t^{(i)})$.
5: Resampling: draw $x_t^{(i)}$, $i = 1, ..., N$ from the discrete distribution $\sum_i \mathsf{w}_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x)$, independently for $i = 1, ..., N$.
6: **end for**

Carlo method since it proceeds by generating samples in the state space using the state equation. The covariance matrices and some other key quantities for the Kalman filter are approximated empirically in the filter using the ensemble of samples. The EnKF has been proved successful, especially for high-dimensional problems, and has been consistently used in practice, e.g., for weather prediction [15–19].

2.3.5 Particle filters

Consider a general SSM $(\pi_0, \tau_t, g_t)_{t\geq 1}$,

$$\begin{aligned} x_0 &\sim \pi_0(x_0), \\ x_t|x_{t-1} &\sim \tau_t(x_t|x_{t-1}), \\ y_t|x_t &\sim g_t(y_t|x_t), \quad t \in \mathbb{N}_+, \end{aligned}$$

where π_0, τ_t, and g_t denote the prior, the transition model, and the likelihood function, respectively. Suppose that τ_t and g_t are such that none of the Kalman-type filters summarized above are applicable due to non-Gaussianity and nonlinearities. In this case, a generic inference method, based on the Monte Carlo principle, can be developed. This type of Monte Carlo estimators are called sequential Monte Carlo (SMC) methods [131–133]. In the rest of this section, we introduce a special class of SMC methods called bootstrap particle filters (BPFs); then we review general particle filters (PFs).

Bootstrap particle filters

Suppose that we are given a collection of particles $\{x_{t-1}^{(i)}\}_{i=1}^N$ at time $t-1$, forming an empirical approximation π_{t-1}^N of the optimal filter π_{t-1}. The BPF first obtains

the empirical approximation ξ_t^N of the predictive measure ξ_t, by propagating the existing samples using the transition kernel, i.e., by sampling

$$\bar{x}_t^{(i)} \sim \tau_t(\mathrm{d}x_t|x_{t-1}^{(i)}), \quad i = 1, \ldots, N,$$

and then constructs the *approximate predictive measure*

$$\xi_t^N(\mathrm{d}x_t) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x_t).$$

Next, each sample is assessed and given weights using the likelihood. Recalling the notation $g_t(x_t) := g_t(y_t|x_t)$, the weight of the i-th particle becomes

$$\mathsf{w}_t^{(i)} = \frac{g_t(\bar{x}_t^{(i)})}{\sum_{i=1}^{N} g_t(\bar{x}_t^{(i)})}, \quad i = 1, \ldots, N,$$

and it yields the *weighted measure*

$$\tilde{\pi}_t^N(\mathrm{d}x_t) = \sum_{i=1}^{N} \mathsf{w}_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x_t).$$

Although, $\tilde{\pi}_t^N$ is enough to estimate expectations w.r.t. π_t, in practice weights $\mathsf{w}_t^{(i)}$ become degenerate, rendering the algorithm practically useless [131–133]. In order to tackle this problem, a resampling step is usually performed after the weighting step, by sampling N i.i.d. particles

$$x_t^{(i)} \sim \tilde{\pi}_t^N(\mathrm{d}x_t), \qquad i = 1, \ldots, N,$$

and constructing the resampled measure

$$\pi_t^N(\mathrm{d}x_t) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_t^{(i)}}(\mathrm{d}x_t).$$

We note that the resampling step, as we have described it here, corresponds to *multinomial resampling scheme*. There are alternative resampling schemes with practically improved performance [134].

The BPF possesses several useful theoretical properties. In the following, we state an L_p convergence proof.

Assumption 2.1. *The likelihood function is positive and bounded, i.e.,*

$$g_t(x_t) > 0 \quad \text{and} \quad \|g_t\|_\infty = \sup_{x_t \in \mathsf{X}} |g_t(x_t)| < \infty,$$

for $t = 1, \ldots, T$.

Theorem 2.3. *Let $y_{1:T}$ be arbitrary but fixed observation sequence with $T < \infty$. If Assumption 2.1 is satisfied, then*

$$\|(\varphi, \pi_t) - (\varphi, \pi_t^N)\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}} \tag{2.42}$$

for $t = 1, \ldots, T$, any $\varphi \in B(\mathsf{X})$, any $p \geq 1$ and some constant $c_{t,p} < \infty$ independent of N.

Proof. See Appendix A.1. □

Note that Theorem 2.3 can be used to prove a uniform convergence result with additional assumptions [30]. We make this aspect clearer in Chapter 3, when we present a result similar to Theorem 2.3 for the nudged particle filter.

Moreover, we have the following bound for the random error made by the BPF.

Corollary 2.3. *Assume Theorem 2.3 holds. Then for every $t \geq 1$,*

$$|(\varphi, \pi_t^N) - (\varphi, \pi_t)| \leq \frac{U_{t,\delta}}{N^{\frac{1}{2}-\delta}}, \tag{2.43}$$

where $U_{t,\delta}$ is an almost surely finite random variable and $0 < \delta < \frac{1}{2}$ is a constant independent of N. As a consequence,

$$\lim_{N\to\infty} (\varphi, \pi_t^N) = (\varphi, \pi_t), \qquad \text{a.s.} \tag{2.44}$$

for any $\varphi \in B(\mathsf{X})$.

Proof. The proof of this corollary follows from Theorem 2.3 and the proof of Corollary 2.1. □

The full procedure of the BPF is outlined in Algorithm 2.2.

General particle filters

The BPF can be seen as a special case of a general sequential Monte Carlo method. In this section, we introduce the general PF formulation as an importance sampling method for increasing dimension. To be more precise, we introduce the PF with general proposals. The convergence of this scheme has been studied but we will not go into the details, instead refer to, e.g., [135] for L^4 convergence. We assume all distributions are absolutely continuous with respect to Lebesgue measure for simplicity.

Algorithm 2.3 General Particle Filter

1: Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
2: **for** $t \geq 1$ **do**
3: Sampling: draw $\bar{x}_t^{(i)} \sim q_t(\mathrm{d}x_t|x_{t-1}^{(i)})$ independently for every $i = 1, \ldots, N$.
4: Weighting: compute

$$\mathsf{w}_t^{(i)} \propto \frac{\tau(\bar{x}_t^{(i)}|x_{t-1}^{(i)})g_t(\bar{x}_t^{(i)})}{q_t(\bar{x}_t^{(i)}|x_{t-1}^{(i)})},$$

for every $i = 1, \ldots, N$.
5: Resampling: draw $x_t^{(i)}$, $i = 1, ..., N$ from the discrete distribution $\sum_i \mathsf{w}_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x)$, independently for $i = 1, ..., N$.
6: **end for**

Let us define a product space $\mathsf{X}^{\otimes t} := \bigotimes_{k=1}^t \mathsf{X}$ at time t. The unnormalized distribution of the SSM defined on $\mathsf{X}^{\otimes t}$ is given as,

$$\Pi_t(x_{1:t}) = \pi_0(x_0) \prod_{k=1}^t \tau_k(x_k|x_{k-1}) g_k(x_k).$$

In order to define an importance sampling procedure, let us define a proposal distribution on $\mathsf{X}^{\otimes t}$ as

$$\tilde{q}_t(x_{1:t}) = \prod_{k=1}^t q_k(x_k|x_{1:k-1}).$$

Now, we can write the *weights* of the importance sampler at time t for the entire trajectory as

$$\begin{aligned} W_t(x_{1:t}) &= \frac{\mathrm{d}\Pi_t}{\mathrm{d}q_t}(x_{1:t}), \\ &= \underbrace{\frac{\Pi_{t-1}(x_{1:t-1})}{\tilde{q}_{t-1}(x_{1:t-1})}}_{W_{1:t-1}(x_{1:t-1})} \frac{\Pi_t(x_{1:t})}{\Pi_{t-1}(x_{1:t-1})} \frac{1}{q_t(x_t|x_{1:t-1})}. \end{aligned} \tag{2.45}$$

Expression (2.45) finally yields

$$W_{1:t}(x_{1:t}) = W_{t-1}(x_{1:t-1}) w_t(x_{1:t}),$$

where $w_t(x_{1:t})$ is called the incremental weight function and can be written as

$$w_t(x_{1:t}) = \frac{\tau_t(x_t|x_{t-1})g_t(x_t)}{q_t(x_t|x_{1:t-1})}. \tag{2.46}$$

The PF is a method which takes advantage of the sequential structure of the weights. It can be seen that one can update the weights by computing only the incremental weight. When the proposal $q_t(x_t|x_{1:t-1})$ depends only on the pair (x_{t-1}, x_t), this leads to very efficient importance samplers on the path space, in the sense that they can be implemented without the need of increasing the computational cost per iteration.

Assume that we are given the particle system in the path space $\{x_{1:t-1}^{(i)}\}_{i=1}^N$. Also, let us assume that the proposal q_t only depends on the previous time, that is, it is a Markov kernel which admits a density of the form $q_t(x_t|x_{t-1})$. The general PF first samples from this kernel, i.e.,

$$\bar{x}_t^{(i)} \sim q_t(\mathrm{d}x_t|x_{t-1}^{(i)}) \quad \text{i.i.d. for } i = 1, \ldots, N.$$

Next, we compute weights as shown in (2.46), i.e.,

$$\mathsf{w}_t^{(i)} \propto \frac{\tau_t(\bar{x}_t^{(i)}|x_{t-1}^{(i)}) g_t(\bar{x}_t^{(i)})}{q_t(\bar{x}_t^{(i)}|x_{t-1}^{(i)})}, \quad \text{for } i = 1, \ldots, N.$$

Note that the incremental weight in (2.46) is unnormalized, we thus use the notation $\propto$ to indicate that the weights $\mathsf{w}_t^{(i)}$ are normalized accordingly. Finally, a resampling scheme is implemented in order to prevent weights from degenerating,

$$x_t^{(i)} \sim \sum_{i=1}^N \mathsf{w}_t^{(i)} \delta_{\bar{x}_t^{(i)}}(\mathrm{d}x_t), \quad \text{for } i = 1, \ldots, N,$$

and the approximation of π_t is constructed as

$$\pi_t^N(\mathrm{d}x_t) = \frac{1}{N} \sum_{i=1}^N \delta_{x_t^{(i)}}(\mathrm{d}x_t).$$

The full procedure of the general PF is outlined in Algorithm 2.3.

2.4 Numerical optimization

2.4.1 Introduction

Optimization methods play a central role in this thesis. An optimization problem can be defined as finding some θ^* such that

$$\theta^\star \in \operatorname*{argmin}_{\theta \in \Theta} f(\theta), \tag{2.47}$$

where $f(\cdot)$ is some cost function. In this section, we are interested in unconstrained optimization, therefore $\Theta = \mathbb{R}^d$. If f is a convex function, one can say that

there is a unique function value for the global minima, i.e., there exists a unique $f^\star := \min_\theta f(\theta)$. To have a unique minimum $\theta^\star$, one needs strong convexity of f [53].

Most optimization problems cannot be solved analytically, hence typically a numerical optimization method is used. Given an initial point, a numerical method for optimization (which we term an *optimizer*) collects information about the function and utilizes this information to achieve descent moves, i.e., moves which reduce the value of the cost function. A numerical optimizer can use function evaluations, gradients (or higher order information), or tools like *proximal operators* (which we will introduce later) to achieve a descent direction. Broadly, we can classify optimizers into three general classes:

- Zeroth-order optimizers which only have access to function evaluations $f(\cdot)$,
- First-order optimizers which have access to function evaluations $f(\cdot)$ and gradients $\nabla f(\cdot)$,
- Second-order optimizers which have access to function evaluations $f(\cdot)$, gradients $\nabla f(\cdot)$, and Hessians $\nabla^2 f(\cdot)$.

In many important scenarios arising in signal processing and machine learning, however, it is often the case that quantities like function evaluations or gradients can only be computed approximately. This kind of problems are referred to as *stochastic optimization* problems. In general, stochastic optimizers can also be classified into the same three classes, namely: (i) zeroth order, (ii) first order, and (iii) second order optimizers. In this case, however, the quantities used by the optimizer contain some form of randomness. This randomness does not have to be limited to the case of additive noise, as we will make clear in the sequel.

In this work we are particularly interested in the stochastic setting that arises when the cost function has the additive form

$$f(\theta) = \frac{1}{n} \sum_{k=1}^{n} f_k(\theta) \tag{2.48}$$

with large n. In this case, it is inefficient to compute function evaluations or gradients since, in an applied setup, each f_k (which we refer to as the kth component function) is parameterized by a data-point. This means that computing the gradient of f requires processing all dataset for a single iteration. In this case, there are different strategies to move in the parameter space while using only a few of the component functions at each iteration. One such strategy is *subsampling with replacement* of the individual functions f_k and constructing a surrogate cost

function, which yields an unbiased estimator of the function evaluation and its gradient. In particular, let us assume that we obtain

$$\tilde{f}(\theta) = \frac{1}{K}\sum_{k=1}^{K} f_{i_k}(\theta),$$

where $i_1, \ldots, i_K$ are sampled uniformly from the index set $[n] = \{1, \ldots, n\}$. In this context, K is called as the *mini-batch size*. This leads to an unbiased estimate of the function, i.e.,

$$\mathbb{E}_{i_1,\ldots,i_K}\left[\tilde{f}(\theta)\right] = f(\theta).$$

Subsampling with replacement is especially used to estimate the gradients of the functions, i.e., at time t, one obtains

$$g_t = \frac{1}{K}\sum_{k=1}^{K} \nabla f_{i_k}(\theta_{t-1}),$$

where

$$\mathbb{E}_{i_1,\ldots,i_K}\left[g_t\right] = \nabla f(\theta_{t-1}).$$

Although this estimate is unbiased, it is not possible to obtain the exact form of the randomness in this case. It is clear that, as K increases, the noise induced on the function evaluations and gradients approaches a Gaussian random variable, whose variance is related to the mini-batch size K.

Although stochastic gradient methods use subsampling with replacement, and the theory of SGD methods rely heavily on this assumption, there are many other stochastic optimization methods which use, e.g., subsampling without replacement, see [84, 85] for some convergence results regarding subgradient methods with different subsampling methods.

2.4.2 Gradient descent

One of the most basic and popular optimization algorithms is the well-known gradient descent (GD) method. Given a function f and an initial point θ_0, the GD recursion has the form

$$\theta_t = \theta_{t-1} - \gamma \nabla f(\theta_{t-1}), \tag{2.49}$$

where $\gamma > 0$ is a step-size parameter. Eq. (2.49) describes a dynamical system that moves θ_t in the direction of negative gradient, hence obtaining a sequence $(\theta_t)_{t\geq 0}$ that progressively moves toward a minimum of f. Under suitable conditions, the GD is known to converge asymptotically as $t \to \infty$ with a rate $\mathcal{O}(1/t)$ for convex f [53].

2.4.3 Stochastic gradient descent

As we have mentioned in Section 2.4.1, it is often the case that gradients can only be obtained with some noise. In this section, we introduce the stochastic gradient descent (SGD) method, the most basic extension of the GD that uses noisy gradients instead of exact gradients.

In particular, at iteration t, we denote the noisy gradient as g_t, and it can be written as

$$g_t = \nabla f(\theta_{t-1}) + \eta_t,$$

where η_t is a random variable. If $\mathbb{E}[\eta_t] = 0$, then the noisy gradient is unbiased. Having obtained the noisy gradient at time $t-1$, the SGD recursion is defined as

$$\theta_t = \theta_{t-1} - \gamma_t g_t,$$

where $(\gamma_t)_{t\geq 1}$ is a sequence of step-sizes, which has to satisfy certain conditions for convergence. For asymptotic convergence, i.e., for the regime $t \to \infty$, it is enough to assume [60, 61] that

$$\sum_t \gamma_t = \infty \quad \text{and} \quad \sum_t \gamma_t^2 < \infty.$$

In finite time, under suitable assumptions, the SGD converges with a rate $\mathcal{O}(1/\sqrt{t})$, slower than the GD rate $\mathcal{O}(1/t)$, due to the noise on the gradients. The rate can be improved under different conditions, such as strong convexity. See [62, 67, 136, 137] and references therein for the detailed investigation of the convergence of the SGD.

2.4.4 Proximal point iteration

Despite GD-based optimization has dominated many application areas, there are cases where it is not possible to apply GD based optimization, e.g., the case of nondifferentiable cost functions. An alternative to GD methods for such a scenario is to use a *proximal operator* in order to identify a descent direction. The methods utilizing proximal operators are collectively called *proximal methods* and they have become increasingly popular for nonsmooth optimization- See, e.g., [77, 76] for an introduction and applications in signal processing.

In this section, we introduce the proximal point iteration (PPI), the most basic proximal algorithm. First, we start by defining the proximal operator [76].

Definition 2.1. Given a function f, its proximal operator $\text{prox}_{f,V}(\theta_0)$ with a symmetric positive definite matrix $V \in \mathbb{R}^{d\times d}$ is defined as

$$\text{prox}_{f,V}(\theta_0) = \underset{\theta\in\mathbb{R}^{d_\theta}}{\text{argmin}}\, f(\theta) + \frac{1}{2}\|\theta - \theta_0\|_{2,V}^2, \tag{2.50}$$

where

$$\|\theta\|_{2,V} = \sqrt{\theta^\top V^{-1}\theta}.$$

When $V = I$, we simplify the notation and write $\text{prox}_f(\theta_0)$. We also note the proximal operator of γf, which follows from Definition 2.1,

$$\text{prox}_{\gamma f}(\theta_0) = \underset{\theta\in\mathbb{R}^d}{\text{argmin}}\, f(\theta) + \frac{1}{2\gamma}\|\theta - \theta_0\|_2^2.$$

We have noted that the proximal operator achieves a descent step in the parameter space. Therefore, it can be viewed as an alternative to the gradient step, when the latter is not possible to implement. Although we will not prove the fact that the proximal operator achieves a descent step, we discuss some interpretations of the proximal operators, which are useful in order to understand its relationship with the gradient descent.

Remark 2.2. Proximal operator implements an implicit gradient step. More precisely, we can interpret the proximal operator as an implicit gradient step for differentiable f. Note that, in order to compute $\text{prox}_{\gamma f}(\theta_0)$, we have to solve the equation

$$\nabla_\theta \left(f(\theta) + \frac{1}{2\gamma}\|\theta - \theta_0\|_2^2 \right) = 0,$$

which yields

$$\theta = \theta_0 - \gamma\nabla f(\theta).$$

This is an implicit equation that has to be solved for θ. □

Remark 2.3. Conversely, an explicit gradient step can be seen as an approximation to the proximal operator [138]. In order to see this, consider the following linear approximation of $f(\theta)$ around θ_0,

$$\bar{f}(\theta) = f(\theta_0) + \nabla f(\theta_0)^\top(\theta - \theta_0).$$

Dropping the terms unrelated to θ, the proximal operator of $\bar{f}$, i.e., $\text{prox}_{\gamma\bar{f}}(\theta_0)$ can be written as

$$\theta = \underset{\theta\in\mathbb{R}^d}{\text{argmax}}\, \nabla f(\theta_0)^\top\theta + \frac{1}{2\gamma}\|\theta - \theta_0\|_2^2.$$

Solving this problem for θ yields the usual explicit gradient step:

$$\theta = \theta_0 - \gamma\nabla f(\theta_0).$$

□

Next, we summarize a proximal point iteration to solve the minimization problem of f. In order to minimize a general f (which is not necessarily smooth) starting from an initial parameter θ_0, one can use the following recursion, which is called as the PPI [76],

$$\theta_t = \text{prox}_{\gamma f}(\theta_{t-1}). \tag{2.51}$$

The PPI has a convergence rate $\mathcal{O}(1/t)$ for convex f, when $\gamma > 0$, and constant or bounded away from zero for every t.

Remark 2.4. We note that, in the light of Remark 2.2, the iteration (2.51) has an intrinsic relationship with gradient descent for smooth f. In particular, it is well known that the gradient descent scheme can be seen as an explicit Euler integration of the continuous-time gradient flow ordinary differential equation (ODE) [76],

$$\frac{\mathrm{d}\theta(t)}{\mathrm{d}t} = -\nabla f(\theta(t)).$$

Using Remark 2.2, on the other hand, we can conclude that recursion (2.51) can be seen as an implicit Euler integration of the same ODE [76]. Therefore, the GD and the PPI can be seen as different numerical methods for solving the same ODE. □

2.4.5 The incremental proximal method

The incremental proximal method (IPM) is a *stochastic* proximal method, meaning that it minimizes cost functions of the form (2.48), while only using one (or a few) component functions at each iteration. The IPM method has been analyzed from different perspectives, see e.g., [84, 85, 138]. To be specific, at iteration t, the incremental proximal method solves

$$\theta_t = \text{prox}_{\gamma f_{i_t}}(\theta_{t-1})$$

where f_{i_t} is the i_tth component function and $i_t \sim [n]$ is drawn uniformly at random. However, as analyzed in [84, 85], the algorithm has also convergence guarantees when the component functions are chosen without replacement or in cyclic order, i.e., when one scans the dataset multiple times in same order. The IPM converges with rate $\mathcal{O}(1/\sqrt{t})$ and, similar to the SGD, this rate can be improved to $\mathcal{O}(1/t)$ under additional assumptions on the cost function; see, e.g., [139]. We remark that the IPM is of special interest to us in this thesis, since it turns out that it can be understood as a sequential inference method (see Chapter 4 for details).

3

Nudging the particle filter

In this chapter, we introduce the nudged particle filter (NuPF), a novel particle filtering method aimed at addressing some shortcomings of the bootstrap particle filter (BPF). In particular, the NuPF incorporates a sampling scheme based on the general idea of nudging the particles towards high-likelihood regions. This approach improves performance for mis-modeled dynamical systems as well as for some high-dimensional examples.

3.1 Introduction

In this chapter, we propose a modification of the PF, termed *the nudged particle filter* (NuPF) and assess its performance in high dimensional settings and with misspecified models.

We use the same idea for nudging that is presented in the literature, as reviewed in Chapter 1. However, our algorithm has subtle but crucial differences. Recall that nudging is defined as a step aiming at moving state particles towards observed data, via some observation dependent operator. Although, we use the same concept to move our particles, we note the following crucial differences.

- First, we define the nudging step not just as a relaxation step towards observations but as a step that strictly increases the likelihood of a subset of particles. This definition paves the way for using different nudging schemes, such

as using the gradients of likelihoods or employing random search schemes to move around the state-space. In particular, classical nudging (relaxation) operations arise as a special case of nudging using gradients when the likelihood is assumed to be Gaussian. Compared to IPFs, the nudging operation we propose is easier to implement as we only demand the likelihood to increase (rather than the posterior density). Indeed, nudging operators can be implemented in relatively straightforward forms, without the need to solve model-dependent equations.

- Second, unlike the other nudging based PFs, we do not correct the bias induced by the nudging operation during the weighting step. Instead, we compute the weights in the same way they would be computed in a conventional (non-nudged) PF. However, we carry out the nudging step in a way that preserves the convergence rate of the PF under mild standard assumptions. In particular, we push only a fraction of particles and leave others untouched, which is key to preserving the Monte Carlo convergence rate, in addition to a significant computational gain. Also, computing biased weights is usually faster than computing proper (unbiased) weights. Depending on the choice of nudging scheme, the proposed algorithm can have an almost negligible computational overhead compared to the conventional PF from which it is derived.

In order to illustrate the contributions outlined above, we present computer the results of several computer experiments with both synthetic and real data. In the first example, we assess the performance of the NuPF when applied to a linear-Gaussian SSM. The aim of these computer simulations is to compare the estimation accuracy and the computational cost of the proposed scheme with several other competing algorithms, namely a standard BPF, a PF with optimal proposal function and a NuPF with proper weights. The fact that the underlying SSM is linear-Gaussian enables the computation of the optimal importance function (intractable in a general setting) and proper weights for the NuPF. We implement the latter scheme because of its similarity to standard nudging filters in the literature. This example shows that the NuPF suffers just from a slight performance degradation compared to the PF with optimal importance function or the NuPF with proper weights, while the latter two algorithms are computationally more demanding.

The second and third examples are aimed at testing the robustness of the NuPF when there is a significant misspecification in the state equation of the SSM. This is helpful in real-world applications because practitioners often have more control

over measurement systems, which determine the likelihood, than they have over the state dynamics. We present computer simulation results for a stochastic Lorenz 63 model and a maneuvering target tracking problem.

In the fourth example, we present numerical results for a stochastic Lorenz 96 model, in order to show how a relatively high-dimensional system can be tracked without a major increase of the computational effort compared to the standard BPF. For this set of computer simulations we have also compared the NuPF with the Ensemble Kalman filter (EnKF), which is the de facto choice for tackling this type of systems.

Let us remark that, for the two stochastic Lorenz systems, the Markov kernel in the SSM can be sampled in a relatively straightforward way, yet transition probability densities cannot be computed (as they involve a sequence of noise variables mapped by a composition of nonlinear functions). Therefore, computing proper weights for proposal functions other than the Markov kernel itself is, in general, not possible for these examples.

Finally, we demonstrate the practical use of the NuPF on a problem where a real dataset is used to fit a stochastic volatility model using either particle Markov chain Monte Carlo (pMCMC) [140] or nested particle filters [141].

Organization

The chapter is structured as follows. We describe the SSMs of interest and the BPF in Section 3.2. Then in Section 3.3, we outline the general algorithm and the specific nudging schemes we propose to use within the PF. We prove a convergence result in Section 3.4 which shows that the new algorithm has the same asymptotic convergence rate as the BPF. We also provide an alternative interpretation of the nudging operation which shows that it defines an implicit model adapted to the data which explains its robustness in scenarios where there is a mismatch between the observed data and the assumed SSM. We discuss the computer simulation experiments in Section 3.5 and present results for real data in Section 3.6.

3.2 Background

In this section, we briefly recall the notions we have introduced in Chapter 2.

3.2.1 State space models

We consider SSMs of the form

$$x_0 \sim \pi_0(\mathrm{d}x_0) \tag{3.1}$$

$$x_t|x_{t-1} \sim \tau_t(\mathrm{d}x_t|x_{t-1}) \tag{3.2}$$

$$y_t|x_t \sim g_t(y_t|x_t), \quad t \in \mathbb{N}, \tag{3.3}$$

where $x_t \in \mathsf{X}$ is the system state at time t, $y_t \in \mathsf{Y}$ is the t-th observation, the measure π_0 describes the prior probability distribution of the initial state, τ_t is a Markov transition kernel on X, and $g_t(y_t|x_t)$ is the (possibly non-normalized) pdf of the observation y_t conditional on the state x_t. We assume the observation sequence $(y_t)_{t\in\mathbb{N}}$ is arbitrary but fixed. Hence, it is convenient to think of the conditional pdf g_t as a likelihood function and we write $g_t(x_t) := g_t(y_t|x_t)$ for conciseness.

We are interested in the sequence of posterior probability distributions of the states generated by the SSM. In particular, we aim at computing the sequence of probability distributions π_t, for $t = 1, \ldots, T$, which describes the probability distribution of the state x_t conditional on the observation of the sequence $y_{1:t}$. The measure π_t is often termed the *optimal filter* at time t. We denote the predictive measure as ξ_t.

3.2.2 Bootstrap particle filter

As we already discussed in Chapter 2, the BPF [20] is a recursive algorithm that produces successive Monte Carlo approximations of ξ_t and π_t for $t = 1, 2, \ldots$. The method can be outlined as shown in Algorithm 2.2. After an initialization stage, where a set of independent and identically distributed (i.i.d.) samples from the prior are drawn, the BPF consists of three recursive steps which can be schematically represented as

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N. \tag{3.4}$$

Given a Monte Carlo approximation $\pi_{t-1}^N = \frac{1}{N}\sum_{i=1}^N \delta_{x_{t-1}^{(i)}}$ computed at time $t-1$, the sampling step yields an approximation of the predictive measure ξ_t of the form

$$\xi_t^N = \frac{1}{N}\sum_{i=1}^N \delta_{\bar{x}_t^{(i)}}$$

by propagating the *particles* $\{x_{t-1}^{(i)}\}_{i=1}^N$ via the Markov kernel $\tau_t(\cdot|x_{t-1}^{(i)})$. The observation y_t is assimilated via the importance weights $w_t^{(i)} \propto g_t(x_t^{(i)})$, to obtain the

approximate filter

$$\tilde{\pi}_t^N = \sum_{i=1}^N w_t^{(i)} \delta_{\bar{x}_t^{(i)}},$$

and the resampling step produces a set of un-weighted particles that completes the recursive loop and yields the approximation

$$\pi_t^N = \frac{1}{N} \sum_{i=1}^N \delta_{x_t^{(i)}}.$$

The random measures ξ_t^N, $\tilde{\pi}_t^N$ and π_t^N are commonly used to estimate *a posteriori* expectations conditional on the available observations. For example, if φ is a function $\mathsf{X} \to \mathbb{R}$, then the expectation of the random variable $\varphi(x_t)$ conditional on $y_{1:t-1}$ is $\mathbb{E}[\varphi(x_t)|y_{1:t-1}] = (\varphi, \xi_t)$. The latter integral can be approximated using ξ_t^N, namely,

$$\begin{aligned}(\varphi, \xi_t) &= \int \varphi(x_t) \xi_t(\mathrm{d}x_t) \approx (\varphi, \xi_t^N) \\ &= \int \varphi(x_t) \xi_t^N(\mathrm{d}x_t) = \frac{1}{N} \sum_{i=1}^N \varphi(\bar{x}_t^{(i)}).\end{aligned}$$

Similarly, we can have estimators $(\varphi, \tilde{\pi}_t^N) \approx (\varphi, \pi_t)$ and $(\varphi, \pi_t^N) \approx (\varphi, \pi_t)$. Classical convergence results are usually proved for real bounded functions, e.g., if $\varphi \in B(\mathsf{X})$ then

$$\lim_{N \to \infty} |(\varphi, \pi_t) - (\varphi, \pi_t^N)| = 0 \quad \text{almost surely (a.s.)}$$

under mild assumptions; see [33, 2] and references therein.

We recall that, as summarized in Section 2.3.5, the BPF can be generalized by using arbitrary proposal pdf's $q_t(x_t|x_{t-1}^{(i)}, y_t)$, possibly observation-dependent, instead of the Markov kernel $\tau_t(\cdot|x_{t-1}^{(i)})$, in order to generate the particles $\{\bar{x}_t^{(i)}\}_{i=1}^N$ in the sampling step. This can lead to more efficient algorithms, but the weight computation has to account for the new proposal and we obtain [131],

$$w_t^{(i)} \propto \frac{g_t(\bar{x}_t^{(i)}) \tau_t(\bar{x}_t^{(i)}|x_t^{(i)})}{q_t(\bar{x}_t^{(i)}|x_{t-1}^{(i)}, y_t)}, \tag{3.5}$$

which can be more costly to evaluate. This issue is related to the nudged PF to be introduced in Section 3.3 below, which can be interpreted as a scheme to choose a certain observation-dependent proposal $q_t(x_t|x_{t-1}^{(i)}, y_t)$. However, the new method does not require that the weights be computed as in (3.5) in order to ensure convergence of the estimators.

3.3 Nudged particle filter

3.3.1 General algorithm

Compared to the standard BPF, the nudged particle filter (NuPF) incorporates one additional step right after the sampling of the particles $\{\bar{x}_t^{(i)}\}_{i=1}^N$ at time t. The schematic depiction of the BPF in (3.4) now becomes

$$\pi_{t-1}^N \underbrace{\rightarrow}_{\text{sampling}} \xi_t^N \underbrace{\rightarrow}_{\text{nudging}} \tilde{\xi}_t^N \underbrace{\rightarrow}_{\text{weighting}} \tilde{\pi}_t^N \underbrace{\rightarrow}_{\text{resampling}} \pi_t^N, \tag{3.6}$$

where the new *nudging step* intuitively consists in pushing a subset of the generated particles $\{\bar{x}_t^{(i)}\}_{i=1}^N$ towards regions of the state space X where the likelihood function $g_t(x)$ takes higher values.

When considered jointly, the sampling and nudging steps in (3.6) can be seen as sampling from a proposal distribution which is obtained by modifying the kernel $\tau_t(\cdot|x_{t-1})$ in a way that depends on the observation y_t. Indeed, this is the classical view of nudging in the literature [44–47]. However, unlike in this classical approach, here the weighting step does not account for the effect of nudging. In the proposed NuPF, the weights are kept the same as in the original filter, $w_t^{(i)} \propto g_t(x_t^{(i)})$. In doing so, we save computations but, at the same time, introduce bias in the Monte Carlo estimators. One of the contributions of this chapter is to show that this bias can be controlled using simple design rules for the nudging step, while practical performance can be improved at the same time.

In order to provide an explicit description of the NuPF, let us first state a definition for the nudging step.

Definition 3.1. A nudging operator $\alpha_t^{y_t} : \mathsf{X} \to \mathsf{X}$ associated with the likelihood function $g_t(x)$ is a map such that

$$\text{if} \quad x' = \alpha_t^{y_t}(x) \quad \text{then} \quad g_t(x') \geq g_t(x) \tag{3.7}$$

for every $x, x' \in \mathsf{X}$.

Intuitively, we define nudging herein as an operation that increases the likelihood. There are several ways in which this can be achieved and we discuss some examples in Sections 3.3.2 and 3.3.3. The NuPF with nudging operator $\alpha_t^{y_t} : \mathsf{X} \to \mathsf{X}$ is outlined in Algorithm 3.1.

It can be seen that the nudging operation is implemented in two stages.

- First, we choose a set of indices $\mathcal{I}_t \subset [N]$ that identifies the particles to be nudged. Let $M = |\mathcal{I}_t|$ denote the number of elements in $\mathcal{I}_t$. We prove in

Algorithm 3.1 Nudged Particle Filter (NuPF)

1: Generate the initial particle system $\{x_0^{(i)}\}_{i=1}^N$ by drawing N times independently from the prior π_0.
2: **for** $t \geq 1$ **do**
3: Sampling: draw $\bar{x}_t^{(i)} \sim \tau_t(\mathrm{d}x_t|x_{t-1}^{(i)})$ independently for every $i = 1, \ldots, N$.
4: **Nudging**: choose a set of indices $\mathcal{I}_t \subset [N]$, then compute $\tilde{x}_t^{(i)} = \alpha_t^{y_t}(\bar{x}_t^{(i)})$ for every $i \in \mathcal{I}_t$. Keep $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)}$ for every $i \in [N] \backslash \mathcal{I}_t$.
5: Weighting: compute $w_t^{(i)} = g_t(\tilde{x}_t^{(i)})/\tilde{Z}_t^N$ for every $i = 1, \ldots, N$, where $\tilde{Z}_t^N = \sum_{i=1}^N g(\tilde{x}_t^{(i)})$.
6: Resample: draw $x_t^{(i)}$ from $\sum_i w_t^{(i)} \delta_{\tilde{x}_t^{(i)}}(\mathrm{d}x)$ independently for $i = 1, ..., N$.
7: **end for**

Section 3.4 that keeping $M \leq \mathcal{O}\left(\sqrt{N}\right)$ allows the NuPF to converge with the same error rates $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$ as the BPF. In Section 3.3.2 we discuss two simple methods to build $\mathcal{I}_t$ in practice.

- Second, we choose an operator $\alpha_t^{y_t}$ that guarantees an increase of the likelihood of any particle. We discuss different implementations of $\alpha_t^{y_t}$ in Section 3.3.3.

We devote the rest of this section to a discussion of how these two steps can be implemented (in several ways).

3.3.2 Selection of particles to be nudged

The set of indices $\mathcal{I}_t$, that identifies the particles to be nudged in Algorithm 3.1, can be constructed in several different ways, either random or deterministic. In this chapter, we describe two simple random procedures with little computational overhead.

- *Batch nudging:* Let the number of nudged particles M be fixed. A simple way to construct $\mathcal{I}_t$ is to draw indices $i_1, i_2, \ldots, i_M$ uniformly from $[N]$ without replacement, and then let $\mathcal{I}_t = i_{1:M}$. We refer to this scheme as *batch nudging*, referring to selection of the indices at once. One advantage of this scheme is that the number of particles to be nudged, M, is deterministic and can be set a priori.

- *Independent nudging:* The size and the elements of $\mathcal{I}_t$ can also be selected randomly in a number of ways. Here, we have studied a procedure in which,

for each index $i = 1, ..., N$, we assign $i \in \mathcal{I}_t$ with probability $\frac{M}{N}$. In this way, the actual cardinality $|\mathcal{I}_t|$ is random, but its expected value is exactly M. This procedure is particularly suitable for parallel implementations, since each index can be assigned to $\mathcal{I}_t$ (or not) at the same time as all others.

3.3.3 How to nudge

The nudging step is aimed at increasing the likelihood of a subset of individual particles, namely those with indices contained in $\mathcal{I}_t$. Therefore, any map $\alpha_t^{y_t} : \mathsf{X} \to \mathsf{X}$ such that $(g_t \circ \alpha_t^{y_t})(x) \geq g_t(x)$ when $x \in \mathsf{X}$ is a valid nudging operator. Typical procedures used for optimization, such as gradient moves or random search schemes, can be easily adapted to implement (relatively) inexpensive nudging steps. Here we briefly describe a few of such techniques.

- *Gradient nudging:* If $g_t(x_t)$ is a differentiable function of x_t, one straightforward way to nudge particles is to take gradient steps. In Algorithm 3.2 we show a simple procedure with one gradient step alone, and where $\gamma > 0$ is a step-size parameter and $\nabla_x g_t(x)$ denotes the vector of partial derivatives of g_t with respect to the state variables, i.e.,

$$\nabla_{x_t} g_t = \begin{bmatrix} \frac{\partial g_t}{\partial x_{1,t}} \\ \frac{\partial g_t}{\partial x_{2,t}} \\ \vdots \\ \frac{\partial g_t}{\partial x_{d_x,t}} \end{bmatrix} \quad \text{for} \quad x_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{d_x,t} \end{bmatrix} \in \mathsf{X}.$$

  Algorithms can obviously be designed where nudging involves several gradient steps. In this work we limit our study to the single-step case, which is shown to be effective and keeps the computational overhead to a minimum. We also note that the performance of gradient nudging can be sensitive to the choice of the step-size parameter $\gamma > 0$, and this is, in turn, model dependent.

- *Random nudging:* Gradient-free techniques inherited from the field of global optimization can also be employed in order to push particles towards regions where they have higher likelihoods. A simple stochastic-search technique adapted to the nudging framework is shown in Algorithm 3.3. We hereafter refer to the latter scheme as random-search nudging.

- *Model specific nudging:* Particles can also be nudged using the specific model information. For instance, in some applications the state vector x_t can be

split into two subvectors, x_t^{obs} and x_t^{unobs} (observed and unobserved, respectively), such that $g_t(x_t) = g_t(x_t^{\text{obs}})$, i.e., the likelihood depends only on x_t^{obs} and not on x_t^{unobs}. If the relationship between x_t^{obs} and x_t^{unobs} is tractable, one can first nudge x_t^{obs} in order to increase the likelihood and then modify x_t^{unobs} in order to keep it coherent with x_t^{obs}. A typical example of this kind arises in object tracking problems, where positions and velocities have a special and simple physical relationship but usually only position variables are observed through a linear or nonlinear transformation. In this case, nudging would only effect position variables. However, using position variables, one can also nudge velocity variables with simple rules. We discuss this idea and show numerical results in Section 3.5.

Algorithm 3.2 Gradient nudging

1: **for** every $i \in \mathcal{I}_t$ **do**

$$\tilde{x}_t^{(i)} = \bar{x}_t^{(i)} + \gamma \nabla_{x_t} g_t(\bar{x}_t^{(i)})$$

2: **end for**

Algorithm 3.3 Random search nudging

1: **repeat**
2: Generate $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)} + \eta_t$ where $\eta_t \sim \mathcal{N}(0, C)$ for some covariance matrix C.
3: If $g_t(\tilde{x}_t^{(i)}) > g_t(\bar{x}_t^{(i)})$ then keep $\tilde{x}_t^{(i)}$, otherwise set $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)}$.
4: **until** the particle is nudged.

3.3.4 Nudging general particle filters

In this chapter we limit our presentation to BPFs in order to focus on the key concepts of nudging and ease presentation. It should be apparent, however, that nudging steps can be plugged into general PFs. More specifically, since the nudging step is algorithmically detached from the sampling and weighting steps, it can be easily used within any PF, even if it relies on different proposals and different weighting schemes. We leave for future work the investigation of the performance of nudging within widely used PFs, such as auxiliary particle filters (APFs). [142].

3.4 Analysis

The nudging step modifies the random generation of particles in a way that is not compensated by the importance weights. Therefore, we can expect nudging to introduce bias in the resulting estimators in general. However, in Section 3.4.1 we prove that, as long as some basic guidelines are followed, the estimators of integrals with respect to the filtering measure π_t and the predictive measure ξ_t converge in L_p as $N \to \infty$ with the usual Monte Carlo rate $\mathcal{O}(1/\sqrt{N})$. The analysis is based on a simple induction argument and ensures the consistency of the estimators. In Section 3.4.2 we briefly comment on the conditions needed to guarantee that convergence is attained uniformly over time. We do not provide a full proof, but this can be done extending the classical arguments in [30] or [33] and using the same treatment of the nudging step as in the induction proof of Section 3.4.1. Finally, in Section 3.4.3, we provide an interpretation of nudging as a conventional particle filtering for a modified model. In particular, we show that the NuPF can be seen as a *standard* PF for a modified SSM.

3.4.1 Convergence in L_p

The goal in this section is to provide theoretical guarantees of convergence for the NuPF under mild assumptions. First, we analyze a general NuPF (with arbitrary nudging operator $\alpha_t^{y_t}$ and an upper bound on the size M of the index set $\mathcal{I}_t$) and then we provide a result for a NuPF with gradient nudging.

Before proceeding with the analysis, let us note that the NuPF produces several approximate measures, depending on the set of particles (and weights) used to construct them. After the sampling step, we have the random probability measure

$$\xi_t^N = \frac{1}{N}\sum_{i=1}^{N} \delta_{\bar{x}_t^{(i)}}, \tag{3.8}$$

which converts into

$$\tilde{\xi}_t^N = \frac{1}{N}\sum_{i=1}^{N} \delta_{\tilde{x}_t^{(i)}} \tag{3.9}$$

after nudging. Once the weights $w_t^{(i)}$ are computed, we obtain the approximate filter

$$\tilde{\pi}_t^N = \sum_{i=1}^{N} w_t^{(i)} \delta_{\tilde{x}_t^{(i)}}, \tag{3.10}$$

which finally yields

$$\pi_t^N = \frac{1}{N}\sum_{i=1}^{N} \delta_{x_t^{(i)}} \tag{3.11}$$

after the resampling step.

Similar to the BPF, the simple Assumption 3.1 stated next is sufficient for consistency and to obtain explicit error rates [29, 143, 103] for the NuPF, as stated in Theorem 3.1 below.

Assumption 3.1. *The likelihood function is positive and bounded, i.e.,*

$$g_t(x_t) > 0 \quad \text{and} \quad \|g_t\|_\infty = \sup_{x_t \in \mathsf{X}} |g_t(x_t)| < \infty$$

for $t = 1, \ldots, T$.

Theorem 3.1. *Let* $y_{1:T}$ *be an arbitrary but fixed sequence of observations, with* $T < \infty$*, and choose any* $M \leq \sqrt{N}$ *and any map* $\alpha_t^{y_t} : \mathsf{X} \to \mathsf{X}$*. If Assumption 3.1 is satisfied and* $|\mathcal{I}_t| = M$*, then*

$$\|(\varphi, \pi_t^N) - (\varphi, \pi_t)\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}} \tag{3.12}$$

for every $t = 1, 2, ..., T$*, any* $\varphi \in B(\mathsf{X})$*, any* $p \geq 1$ *and some constant* $c_{t,p} < \infty$ *independent of* N.

See Appendix A.2 for a proof.

Theorem 3.1 is very general; it actually holds for any map $\alpha_t^{y_t} : \mathsf{X} \to \mathsf{X}$, i.e., not necessarily a nudging operator. We can also obtain error rates for specific choices of the nudging scheme. A simple, yet practically appealing, setup is the combination of batch and gradient nudging, as described in Sections 3.3.2 and 3.3.3, respectively.

Assumption 3.2. *The gradient of the likelihood is bounded. In particular, there are constants* $G_t < \infty$ *such that*

$$\|\nabla_x g_t(x)\|_2 \leq G_t < \infty$$

for every $x \in \mathsf{X}$ *and* $t = 1, 2, \ldots, T$.

Lemma 3.1. *Choose a step-size* $\gamma > 0$ *and the number of nudged particles* $M > 0$ *in such a way that* $\gamma M \leq \sqrt{N}$*. If Assumption 3.2 holds and* φ *is a Lipschitz test function, then the error introduced by the batch gradient nudging step with* $|\mathcal{I}_t| = M$ *can be bounded as,*

$$\left\|(\varphi, \xi_t^N) - (\varphi, \tilde{\xi}_t^N)\right\|_p \leq \frac{LG_t}{\sqrt{N}},$$

where L *is the Lipschitz constant of* φ*, for every* $t = 1, \ldots, T$.

See Appendix A.2 for a proof.

It is straightforward to apply Lemma 3.1 to prove convergence of the NuPF with a batch gradient-nudging step. Specifically, we have the following result.

Theorem 3.2. *Let $y_{1:T}$ be an arbitrary but fixed sequence of observations, with $T < \infty$, and choose a step size $\gamma > 0$ and an integer M such that $\gamma M \leq \sqrt{N}$. Let π_t^N denote the filter approximation obtained with a NuPF with batch gradient nudging. If Assumptions 3.1 and 3.2 are satisfied and $|\mathcal{I}_t| = M$, then*

$$\|(\varphi, \pi_t^N) - (\varphi, \pi_t)\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}} \tag{3.13}$$

for every $t = 1, 2, ..., T$, any bounded Lipschitz function φ, some constant $c_{t,p} < \infty$ independent of N for any integer $p \geq 1$.

The proof is straightforward (using the same argument as in the proof of Theorem 3.1 combined with Lemma 3.1) and we omit it here. We note that Lemma 3.1 provides a guideline for the choice of M and γ. In particular, one can select $M = N^\beta$, where $0 < \beta < 1$, together with $\gamma \leq N^{\frac{1}{2}-\beta}$ in order to ensure $\gamma M \leq \sqrt{N}$. Actually, it would be sufficient to set $\gamma \leq CN^{\frac{1}{2}-\beta}$ for some constant $C < \infty$ in order to keep the same error rate (albeit with a different constant in the numerator of the bound). Therefore, Lemma 3.1 provides a heuristic to balance the step size with the number of nudged particles. We can increase the number of nudged particles but in that case we need to shrink the step size accordingly, so as to keep $\gamma M \leq \sqrt{N}$. Similar results can be obtained using the gradient of the log-likelihood, $\log g_t$, if the g_t comes from the exponential family of densities.

3.4.2 Uniform convergence

Uniform convergence can be proved for the NuPF under the same standard assumptions as for the conventional BPF; see, e.g., [30, 33]. The latter can be summarised as follows [33]:

(i) The likelihood function is bounded and bounded away from zero, i.e., $g_t \in B(\mathsf{X})$ and there is some constant $a > 0$ such that $\inf_{t>0, x\in\mathsf{X}} g_t(x) \geq a$.

(ii) The kernel mixes sufficiently well, namely, for any given integer m there is a constant $0 < \varepsilon < 1$ such that

$$\inf_{t>0;(x,x')\in\mathsf{X}^2} \frac{\tau_{t+m|t}(A|x)}{\tau_{t+m|t}(A|x')} > \varepsilon$$

for any Borel set A, where $\tau_{t+m|t}$ is the composition of the kernels $\tau_{t+m} \circ \tau_{t+m-1} \circ \cdots \circ \tau_t$.

When (i) and (ii) above hold, the sequence of optimal filters $\{\pi_t\}_{t\geq 0}$ is stable and it can be proved that

$$\sup_{t>0} \|(\varphi, \pi_t) - (\varphi, \pi_t^N)\|_p \leq \frac{c_p}{\sqrt{N}}$$

for any bounded function $\varphi \in B(\mathsf{X})$, $c_p < \infty$ is constant with respect to N and t and π_t^N is the particle approximation produced by either the NuPF (as in Theorem 3.1 or, provided $\sup_{t>0} G_t < \infty$, as in Theorem 3.2) or the BPF algorithms. We skip a formal proof as, again, it is straightforward combination of the standard argument by [33] (see also, e.g., [144] and [141]) with the same handling of the nudging operator in the proofs of Theorem 3.1 or Lemma 3.1 .

3.4.3 An alternative interpretation of nudging

We have found in computer simulation experiments that the NuPF is consistently more robust to model errors than the conventional BPF. In order to obtain some analytical insight of this scenario, in this section we reinterpret the NuPF as a standard BPF for a modified dynamical model and discuss some features of the method from this point of view. In particular, we argue that the model implicitly defined by the NuPF can be seen as an automatic adaptation of the underlying SSM to the available data.

Recall that we define SSMs with the prior τ_0, the kernels τ_t and the likelihood functions $g_t(x) = g_t(y_t|x)$, for $t \geq 1$. In this section we write the latter as $g_t^{y_t}(x) = g_t(y_t|x)$. Hence, we can formally represent the SSM defined by (3.1), (3.2) and (3.3) as $\mathcal{M}_0 = (\tau_0, \tau_t, g_t^{y_t})_{t\geq 1}$, which refers to the model assumed by the user of the algorithm. Now, let us assume $y_{1:T}$ to be fixed and construct the alternative SSM $\mathcal{M}_1 = (\tau_0, \tilde{\tau}_t^{y_t}, g_t^{y_t})_{t\geq 1}$, where

$$\tilde{\tau}_t^{y_t}(\mathrm{d}x_t|x_{t-1}) := (1-\varepsilon_M)\tau_t(\mathrm{d}x_t|x_{t-1}) + \varepsilon_M \int \delta_{\alpha_t^{y_t}(\bar{x}_t)}(\mathrm{d}x_t)\tau_t(\mathrm{d}\bar{x}_t|x_{t-1}), \quad (3.14)$$

$\varepsilon_M = \frac{M}{N}$ and the nudging operator $\alpha_t^{y_t}$ is a one-to-one map that depends on the (fixed) observation y_t. We note that the kernel $\tilde{\tau}_t^{y_t}$ jointly represents the Markov transition induced by the original kernel τ_t followed by an independent nudging transformation (namely, each particle is independently nudged with probability ε_M). As a consequence, the standard BPF for model $\mathcal{M}_1$ coincides exactly with a NuPF for model $\mathcal{M}_0$ with independent nudging and operator $\alpha_t^{y_t}$. Indeed, according to the definition of $\tilde{\tau}_t^{y_t}$ in (3.14), generating a sample $\tilde{x}_t^{(i)}$ from $\tilde{\tau}_t^{y_t}(\mathrm{d}x_t|x_{t-1}^{(i)})$ is a three-step process where

- we first draw $\bar{x}_t^{(i)}$ from $\tau_t(\mathrm{d}x_t|x_{t-1}^{(i)})$,

- then generate a sample $u_t^{(i)}$ from the uniform distribution $\mathcal{U}(0,1)$, and
- if $u_t^{(i)} < \varepsilon_M$ then we set $\tilde{x}_t^{(i)} = \alpha_t^{y_t}(\bar{x}_t^{(i)})$, else we set $\tilde{x}_t^{(i)} = \bar{x}_t^{(i)}$.

After sampling, the importance weight for the BPF applied to model $\mathcal{M}_1$ is $w_t^{(i)} \propto g_t^{y_t}(\tilde{x}_t^{(i)})$. This is exactly the same procedure as in the NuPF applied to the original SSM $\mathcal{M}_0$ (see Algorithm 3.1).

Some remarks are in order regarding this interpretation of the nudging scheme.

- Model $\mathcal{M}_1$ is not Markov; transition kernel at time t depends on the observation y_t, not only on x_{t-1}.
- It has been discussed in Section 3.3 that nudging can be implemented in a number of ways, depending on the selection of the particles and how they are nudged. Different schemes, e.g., for the selection of particles, may translate into different implicit models.
- Intuitively, nudging enables us to embed observations into the transition kernels in a systematic way. We show in the experimental section that this operation typically results in estimates with higher marginal likelihoods. We believe that this feature makes the NuPF more robust to modeling errors compared to conventional BPFs.

3.5 Computer simulations

In this section, we present the results of several computer experiments. In the first one, we address the tracking of a linear-Gaussian system. This is a very simple model which enables a clearcut comparison of the NuPF and other competing schemes, including a conventional PF with optimal importance function (which is intractable for all other examples) and a PF with nudging and proper importance weights. Then, we study three nonlinear tracking problems:

- a stochastic Lorenz 63 model with misspecified parameters,
- a maneuvering target monitored by a network of sensors collecting nonlinear observations corrupted with heavy-tailed noise,
- and, finally, a high-dimensional stochastic Lorenz 96 model[1].

[1]For the experiments involving Lorenz 96 model, simulation from the model is implemented in C++ and integrated into Matlab. The rest of the simulations are fully implemented in Matlab.

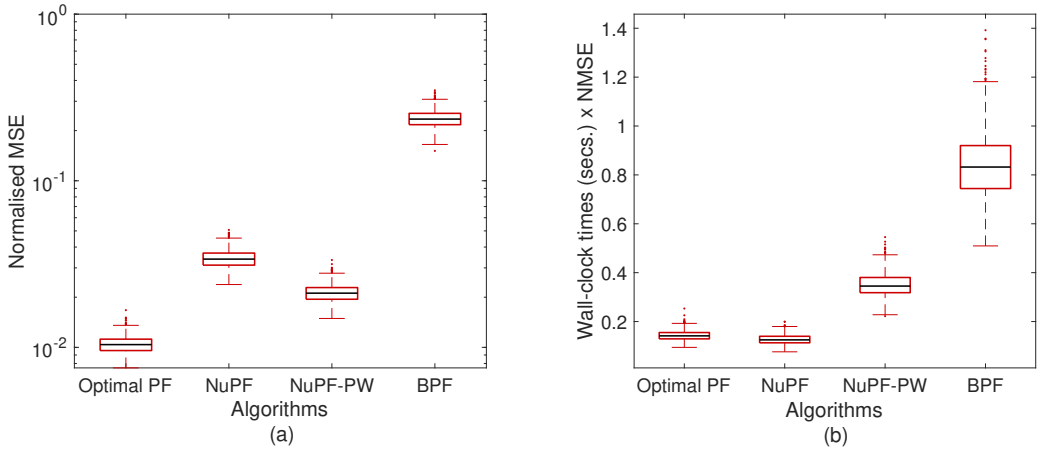

Figure 3.1: (a) Empirical NMSE of the Optimal PF, NuPF-PW, NuPF, and BPF methods implemented for the high-dimensional linear-Gaussian SSM given in (3.15)–(3.17). The box-plots are constructed from 1,000 independent Monte Carlo runs. It can be seen that the NMSE of the NuPF is comparable to the error of the Optimal PF and the NuPF-PW methods. (b) Runtimes×NMSEs of all methods. This experiment shows that, in addition to the fact that the NuPF attains a comparable estimation performance, which can be seen in (a), it has a computational cost similar to the plain BPF. The figure demonstrates that the NuPF has a comparable performance to the optimal PF for this model.

We have used gradient nudging in all experiments, with either $M \leq \sqrt{N}$ (deterministically, with batch nudging) or $\mathbb{E}[M] \leq \sqrt{N}$ (with independent nudging). This ensures the validity of the theoretical results presented in Section 3.4. For the object tracking experiment, we have used a large step-size, but this choice does not affect the convergence rate of the NuPF algorithm either.

3.5.1 A high-dimensional, inhomogeneous Linear-Gaussian state-space model

In this experiment we compare different PFs implemented to track a high-dimensional linear Gaussian SSM. In particular, the model under consideration is,

$$x_0 \sim \mathcal{N}(0, I_{d_x}), \tag{3.15}$$

$$x_t | x_{t-1} \sim \mathcal{N}(x_{t-1}, Q), \tag{3.16}$$

$$y_t | x_t \sim \mathcal{N}(C_t x_t, R), \tag{3.17}$$

where $(x_t)_{t\geq 0}$ are hidden states, $(y_t)_{t\geq 1}$ are observations, and Q and R are the process and the observation noise covariance matrices, respectively. The latter are diagonal matrices, namely $Q = qI_{d_x}$ and $R = I_{d_y}$, where $q = 0.1$, $d_x = 100$

and $d_y = 20$. The sequence $(C_t)_{t\geq 1}$ defines a time-varying observation model. The elements of this sequence are chosen as random binary matrices, i.e., $C_t \in \{0,1\}^{d_y \times d_x}$ where each entry is generated as an independent Bernoulli random variable with $p = 0.5$. Once generated, they are fixed and fed into all algorithms we describe below for each independent Monte Carlo run.

We compare the NuPF with three alternative PFs. The first method we implement is the PF with the optimal proposal pdf $p(x_t|x_{t-1}, y_t)$, abbreviated as Optimal PF. The pdf $p(x_t|x_{t-1}, y_t)$ leads to an analytically tractable Gaussian density for the model (3.15)–(3.17) [131] but not in the nonlinear tracking examples below. Note, however, that at each time step, the mean and covariance matrix of this proposal have to be explicitly evaluated in order to compute the importance weights.

The second filter is a nudged PF with proper importance weights (NuPF-PW). In this case, we treat the generation of the nudged particles as a proposal function to be accounted for during the weighting step. To be specific, the proposal distribution resulting from the NuPF has the form

$$\tilde{\tau}_t(\mathrm{d}x_t|x_{t-1}) = (1-\epsilon_N)\tau_t(\mathrm{d}x_t|x_{t-1}) + \epsilon_N \bar{\tau}_t(\mathrm{d}x_t|x_{t-1}), \tag{3.18}$$

where $\epsilon_N = \frac{1}{\sqrt{N}}$ and

$$\bar{\tau}_t(\mathrm{d}x_t|x_{t-1}) = \int \delta_{\alpha_t^{y_t}(\bar{x}_t)}(\mathrm{d}x_t)\tau_t(\mathrm{d}\bar{x}_t|x_{t-1}).$$

The latter conditional distribution admits an explicit representation as a Gaussian for model (3.15)-(3.17) when α_t operator is designed as a gradient step, but this approach is intractable for the examples in Section 3.5.2 and Section 3.5.4. Note that $\tilde{\tau}_t$ is a mixture of two time-varying Gaussians and this fact adds to the cost of the sampling and weighting steps. Specifically, computing weights for the NuPF-PW is significantly more costly, compared to the BPF or the NuPF, because the mixture (3.18) has to be evaluated together with the likelihood and the transition pdf.

The third tracking algorithm implemented for model (3.15)–(3.17) is the conventional BPF.

For all filters, we have set the number of particles as $N = 100$. In order to implement the NuPF and NuPF-PW schemes, we have selected the step size $\gamma = 2 \times 10^{-2}$. We have run 1,000 independent Monte Carlo runs for this experiment. To evaluate different methods, we have computed the empirical normalised mean

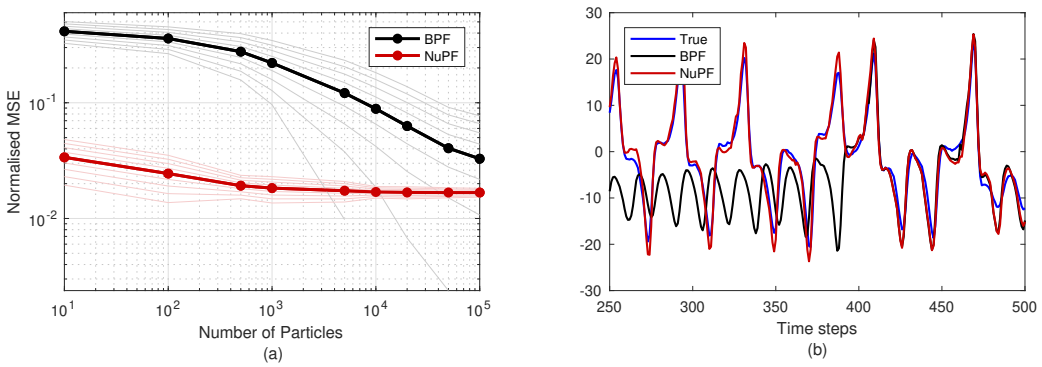

Figure 3.2: (a) NMSE results of the BPF and NuPF algorithms for a misspecified Lorenz 63 system. The results have been obtained from 1,000 independent Monte Carlo runs for each $N \in \{10, 100, 500, 1K, 5K, 10K, 20K, 50K, 100K\}$. The dashed lines indicate 1 standard deviation. The misspecified parameter is $\mathsf{b}_\epsilon = \mathsf{b} + \epsilon$, where $\mathsf{b} = 8/3$ and $\epsilon = 0.75$. (b) A sample path of the true state variable $x_{2,t}$ and its estimates in a run with $N = 500$ particles.

squared errors (NMSEs). Specifically, the NMSE for the j-th simulation is

$$\text{NMSE}(j) = \frac{\sum_{t=1}^{t_f} \|\bar{x}_t - \hat{x}_t(j)\|_2^2}{\sum_{t=1}^{t_f} \|x_t\|_2^2}, \tag{3.19}$$

where $\bar{x}_t$ is the exact posterior mean of the state x_t conditioned on the observations up to time t and $\hat{x}_t(j)$ is the estimate of the state vector in the j-th simulation run. In the figures, we usually plot the mean and the standard deviation of the sample of errors, $\text{NMSE}(1), \ldots, \text{NMSE}(1000)$.

The results are shown in Fig. 3.1. In particular, in Fig. 3.1(a), we observe that the NMSE performance of the NuPF compared to the optimal PF and NuPF-PW (which is similar to a classical PF with nudging) is comparable. However, Fig. 3.1(b) reveals that the NuPF is significantly less demanding compared to the optimal PF and the NuPF-PW method. Indeed, the runtimes of the NuPF are almost identical to the those of the plain BPF. As a result, the plot of the NMSEs multiplied by the running times displayed in Fig. 3.1(b) reveals that the proposed algorithm is as favorable as the optimal PF, which can be implemented for this model, but not for general models unlike the NuPF.

3.5.2 Stochastic Lorenz 63 model with misspecified parameters

In this experiment, we demonstrate the performance of the NuPF a misspecified stochastic Lorenz 63 model. The dynamics of the system is described by a

stochastic differential equation (SDE) in three dimensions,

$$\begin{aligned} \mathrm{d}x_1 &= -\mathsf{s}(x_1 - x_2) + \mathrm{d}w_1, \\ \mathrm{d}x_2 &= \mathsf{r}x_1 - x_2 - x_1 x_3 + \mathrm{d}w_2, \\ \mathrm{d}x_3 &= x_1 x_2 - \mathsf{b}x_3 + \mathrm{d}w_3, \end{aligned}$$

where $\{w_i(s)\}_{s\in(0,\infty)}$ for $i = 1, 2, 3$ are 1-dimensional independent Wiener processes and $\mathsf{s}, \mathsf{r}, \mathsf{b} \in \mathbb{R}$ are fixed model parameters. We discretise the model using the Euler-Maruyama scheme with integration step $\mathsf{T} > 0$ and obtain the system of difference equations

$$\begin{aligned} x_{1,t} &= x_{1,t-1} - \mathsf{T}\mathsf{s}(x_{1,t-1} - x_{2,t-1}) + \sqrt{\mathsf{T}}u_{1,t} \\ x_{2,t} &= x_{2,t-1} + \mathsf{T}(\mathsf{r}x_{1,t-1} - x_{2,t-1} - x_{1,t-1}x_{3,t-1}) + \sqrt{\mathsf{T}}u_{2,t} \\ x_{3,t} &= x_{3,t-1} + \mathsf{T}(x_{1,t-1}x_{2,t-1} - \mathsf{b}x_{3,t-1}) + \sqrt{\mathsf{T}}u_{3,t} \end{aligned} \tag{3.20}$$

where $\{u_{i,t}\}_{t\in\mathbb{N}}$, $i = 1, 2, 3$ are i.i.d. Gaussian random variables with zero mean and unit variance. We assume that we can only observe the variable $x_{1,t}$ every $t_s > 1$ discrete time steps and contaminated by additive noise. To be specific, we collect the sequence of observations

$$y_n = k_o x_{1,nt_s} + v_n, \quad n = 1, 2, ...,$$

where $\{v_n\}_{n\in\mathbb{N}}$ is a sequence of i.i.d. Gaussian random variables with zero mean and unit variance and the scale parameter k_o is assumed known.

In order to simulate both the state signal and the synthetic observations from this model, we choose the so-called standard parameter values

$$(\mathsf{s}, \mathsf{r}, \mathsf{b}) = \left(10, 28, \frac{8}{3}\right),$$

which make the system dynamics chaotic. The initial condition is set as

$$x_0 = [-5.91652, -5.52332, 24.5723]^\top,$$

which corresponds to a deterministic trajectory of the system (i.e., with no state noise) with the same parameter set [145]. We assume that the system is observed every $t_s = 40$ discrete time steps and for each simulation we simulate the system for $t = 0, 1, \ldots, t_f$, with $t_f = 20,000$. Since $t_s = 40$, we have a sequence of $\frac{t_f}{t_s} = 500$ observations overall.

Let us note here that the Markov kernel which takes the state from time $n-1$ to time n (i.e., from the time of one observation to the time of the next observation) is straightforward to simulate using the Euler-Maruyama scheme (3.20), however

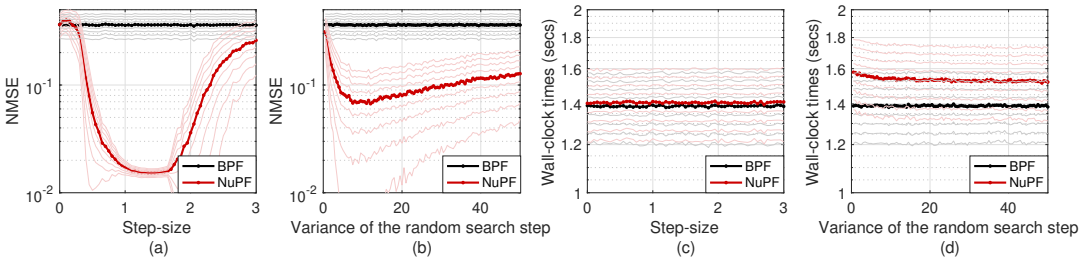

Figure 3.3: A comparison of gradient nudging and random search nudging for a variety of parameter settings. From (a), it can be seen that gradient nudging is robust within a large interval for γ. From (b), one can see that the same is true for random search nudging with the covariance of the form $C = \sigma^2 I$ for a wide range of σ^2. From (c)–(d), it can be seen that while gradient nudging causes negligible computational overhead, random search nudging is more demanding in terms of computation time and this behavior is expected to be more apparent in higher dimensional spaces. Comparing (a)–(b), it can also be seen that gradient nudging attains lower error rates in general.

the associated transition probability density cannot be evaluated because it involves the mapping of both the state and a sequence of t_s noise samples through a composition of nonlinear functions. This precludes the use of importance sampling schemes that require the evaluation of this density when computing the weights.

We run the BPF and NuPF algorithms for the model described above, except that the parameter b is replaced by $\mathsf{b}_\epsilon = \mathsf{b} + \epsilon$, with $\epsilon = 0.75$ (hence $\mathsf{b}_\epsilon \approx 3.417$ versus $\mathsf{b} \approx 2.667$ for the actual system). As the system underlying dynamics is chaotic, this mismatch affects the predictability of the system significantly.

We have implemented the NuPF with independent gradient nudging. Each particle is nudged with probability $\frac{1}{\sqrt{N}}$, where N is the number of particles (hence $\mathbb{E}[M] = \sqrt{N}$), and the size of the gradient steps is set to $\gamma = 0.75$ (see Algorithm 3.2). As a figure of merit, we evaluate the NMSE for the 3-dimensional state vector, averaged over 1,000 independent Monte Carlo simulations.

For this example (as well as in the rest of this section), it is not possible to compute the exact posterior mean of the state variables. Therefore, the NMSE values are computed with respect to the ground truth, i.e.,

$$\mathrm{NMSE}(j) = \frac{\sum_{t=1}^{t_f} \|x_t - \hat{x}_t(j)\|_2^2}{\sum_{t=1}^{t_f} \|x_t\|_2^2}, \tag{3.21}$$

where $(x_t)_{t\geq 1}$ is the ground truth signal.

Fig. 3.2 (a) displays the NMSE, attained for varying number of particles N, for the standard BPF and the NuPF. It is seen that the NuPF outperforms the BPF for the whole range of values of N in the experiment, both in terms of

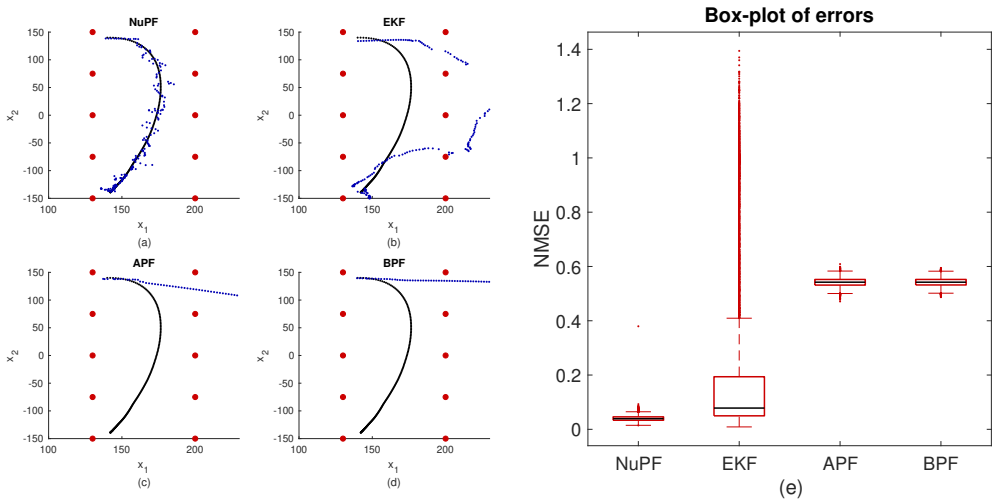

Figure 3.4: Plots (a)–(d): A typical simulation run for the BPF, APF, EKF and NuPF algorithms using $N = 500$ particles. The black dots denote the real trajectory of the object, the red dots are sensors and the blue dots are position estimates as provided by the filters. Plot (e): Box-plot of the errors $\mathrm{NMSE}(1), \ldots, \mathrm{NMSE}(10,000)$ obtained for the set of independent simulation runs. The NuPF achieves a low NMSE with a low variance whereas the EKF exhibits a large variance.

the mean and the standard deviation of the errors, although the NMSE values become closer for larger N. The plot on the right displays the values of $x_{2,t}$ and its estimates for a typical simulation. In general, the experiment shows that the NuPF can track the actual system using the misspecified model and a small number of particles, whereas the BPF requires a higher computational effort to attain a similar performance.

As a final experiment with this model, we have tested the robustness of the algorithms with respect to the choice of parameters in the nudging step. In particular, we have tested the NuPF with independent gradient nudging for a wide range of step-sizes γ. Also, we have tested the NuPF with random search nudging using a wide range of covariances of the form $C = \sigma^2 I$ by varying σ^2.

The results can be seen in Fig. 3.3. This figure shows that the algorithm is robust to the choice of parameters for a range of step-sizes and variances of the random search step. As expected, random search nudging takes longer running time compared to gradient steps. This difference in run-times is expected to be larger in higher-dimensional models since random search is expected to be harder in such scenarios.

3.5.3 Object tracking with a misspecified model

In this experiment, we consider a tracking scenario where a target is observed through sensors collecting radio signal strength (RSS) measurements contaminated with additive heavy-tailed noise. The target dynamics are described by the model,

$$x_t = Ax_{t-1} + BL(x_{t-1} - x_o) + u_t$$

where $x_t \in \mathbb{R}^4$ denotes the target state, consisting of its position $r_t \in \mathbb{R}^2$ and its velocity, $v_t \in \mathbb{R}^2$, hence $x_t = \begin{bmatrix} r_t \\ v_t \end{bmatrix} \in \mathbb{R}^4$. The parameter x^o is a deterministic, pre-chosen state to be attained by the target. Each element in the sequence $\{u_t\}_{t\in\mathbb{N}}$ is a zero-mean Gaussian random vector with covariance matrix Q. The parameters A, B, Q are selected as

$$A = \begin{bmatrix} I_2 & \kappa I_2 \\ 0 & 0.99 I_2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & I_2 \end{bmatrix}^\top,$$

and

$$Q = \begin{bmatrix} \frac{\kappa^3}{3} I_2 & \frac{\kappa^2}{2} I_2 \\ \frac{\kappa^2}{2} I_2 & \kappa I_2 \end{bmatrix},$$

where I_2 is the 2×2 identity matrix and $\kappa = 0.04$. The policy matrix $L \in \mathbb{R}^{2\times 4}$ determines the trajectory of the target from an initial position $x_0 = [140, 140, 50, 0]^\top$ to a final state $x_T = [140, -140, 0, 0]^\top$ and it is computed by solving a Riccati equation (see [146] for details), which yields

$$L = \begin{bmatrix} -0.0134 & 0 & -0.0381 & 0 \\ 0 & -0.0134 & 0 & -0.0381 \end{bmatrix}.$$

This policy results in a highly maneuvering trajectory. In order to design the NuPF, however, we assume the simpler dynamical model

$$x_t = Ax_{t-1} + u_t,$$

hence there is a considerable model mismatch.

The observations are nonlinear and coming from 10 sensors placed in the region where the target moves. The measurement collected at the i-th sensor, time t, is modelled as

$$y_{t,i} = 10 \log_{10} \left(\frac{P_0}{\|r_t - s_i\|^2} + \eta \right) + w_{t,i}$$

where $r_t \in \mathbb{R}^2$ is the position vector, s_i is the position of the ith sensor and $w_{t,i} \sim \mathcal{T}(0, 1, \nu)$ is an independent t-distributed random variable for each $i = 1, \ldots, 10$. Intuitively, the closer the parameter ν to 1, the more explosive the observations become. In particular, we set $\nu = 1.01$ to make the observations explosive and heavy-tailed. As for the sensor parameters, we set the transmitted RSS as $P_0 = 1$ and the sensitivity parameter as $\eta = 10^{-9}$. The latter yields a soft lower bound of -90 decibel (dB) for the RSS measurements.

We have implemented the NuPF with batch gradient nudging, with a large-step size $\gamma = 5.5$ and $M = \lfloor \sqrt{N} \rfloor$. Since the observations depend on the position vector r_t only, an additional model-specific nudging step is needed for the velocity vector v_t. In particular, after nudging the $r_t^{(i)} = [x_{1,t}^{(i)}, x_{2,t}^{(i)}]^\top$, we update the velocity variables as

$$v_t^{(i)} = \frac{1}{\kappa}(r_t^{(i)} - r_{t-1}^{(i)}), \quad \text{where} \quad v_t^{(i)} = [x_{3,t}^{(i)}, x_{4,t}^{(i)}]^\top,$$

where $\kappa = 0.04$ as defined for the model. The motivation for this additional transformation comes from the physical relationship between position and velocity. We note, however, that the NuPF also works without nudging the velocities.

We have run $10,000$ Monte Carlo runs with $N = 500$ particles in the auxiliary particle filter (APF) [142, 147, 148], the BPF [20] and the NuPF. We have also implemented the extended Kalman filter (EKF), which uses the gradient of the observation model.

Fig. 3.4 shows a typical simulation run with each one of the four algorithms (on the left side, plots (a)–(d)) and a box-plot of the NMSEs obtained for the 10,000 simulations (on the right, plot (e)). Plots (a)–(d) show that, while the EKF also uses the gradient of the observation model, it fails to handle the heavy-tailed noise, as it relies on Gaussian approximations. The BPF and the APF collapse due to the model mismatch in the state equation. Plot (d) shows that the NMSE of the NuPF is just slightly smaller in the mean than the NMSE of the EKF, but much more stable.

3.5.4 High-dimensional stochastic Lorenz 96 model

In this computer experiment, we compare the NuPF with the ensemble Kalman filter (EnKF) for the tracking of a stochastic Lorenz 96 system. The latter is described by the set of stochastic differential equations (SDEs)

$$\mathrm{d}x_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F + \mathrm{d}w_i, \quad i = 1, \ldots, d,$$

where $\{w_i(s)\}_{s\in(0,\infty)}$, $i = 1, \ldots, d$, are independent Wiener processes, d is the system dimension and the forcing parameter is set to $F = 8$, which ensures a

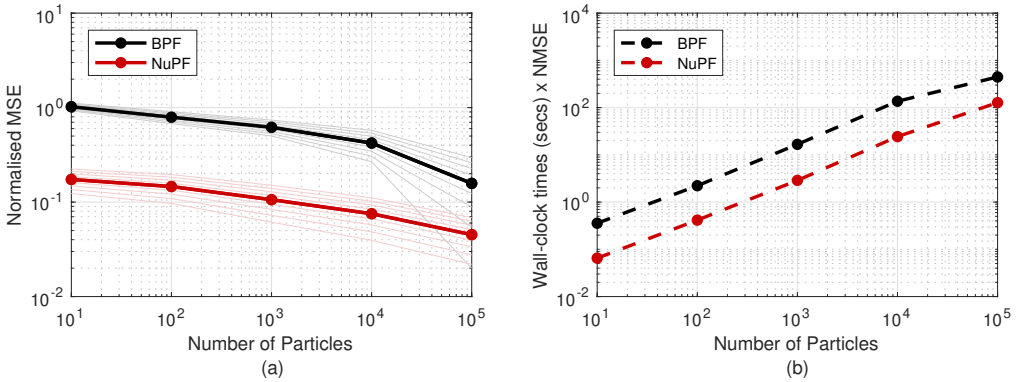

Figure 3.5: Comparison of the NuPF and the BPF for the stochastic Lorenz 96 system with model dimension $d = 40$. The results have been obtained from a set of $1,024$ independent Monte Carlo runs. Plot (a): average NMSE and standard deviation (dashed lines) as the number of particles N is increased. Plot (b): Running-times×NMSE of the BPF and the NuPF for the same set of simulations. Since the increase in computational cost of the NuPF, compared to the BPF, is negligible, it is clearly seen from (b) that the NuPF performs better when errors and run-times are considered jointly.

chaotic regime. The model is assumed to have a circular structure, so that $x_{-1} = x_{d-1}$, $x_0 = x_d$, and $x_{d+1} = x_1$. In order to simulate data from this model, we apply the Euler-Maruyama discretization scheme and obtain the difference equations,

$$x_{i,t} = x_{i,t-1} + \mathsf{T}[(x_{i+1,t-1} - x_{i-2,t-1})x_{i-1,t-1} - x_{i,t-1} + F] + \sqrt{\mathsf{T}}u_{i,t}$$

where $u_{i,t}$ are zero-mean, unit-variance Gaussian random variables.

We assume that the system is only partially observed. In particular, half of the state variables are observed, in Gaussian noise, every $t_s = 10$ time steps, namely,

$$y_{j,n} = x_{2j-1,nt_s} + u_{j,n}, \quad n = 1, 2, \ldots, \quad j = 1, 2, \ldots, \lfloor d/2 \rfloor,$$

where $u_{j,n}$ is a normal random variable with zero mean and unit variance. The same as in the stochastic Lorenz 63 example of Section 3.5.2, the transition pdf that takes the state from time $(n-1)t_s$ to time nt_s is simple to simulate but hard to evaluate, since it involves mapping a sequence of noise variables through a composition of nonlinearities.

In all the simulations for this system we run the NuPF with batch gradient nudging (with $M = \lfloor \sqrt{N} \rfloor$ nudged particles and step-size $\gamma = 0.075$). In the first computer experiment, we fixed the dimension $d = 40$ and run the BPF and the NuPF with increasing number of particles. The results can be seen in Fig. 3.5, which shows how the NuPF performs better than the BPF in terms of NMSE (plot (a)) while the running times of both algorithms are nearly identical (plot (b)).

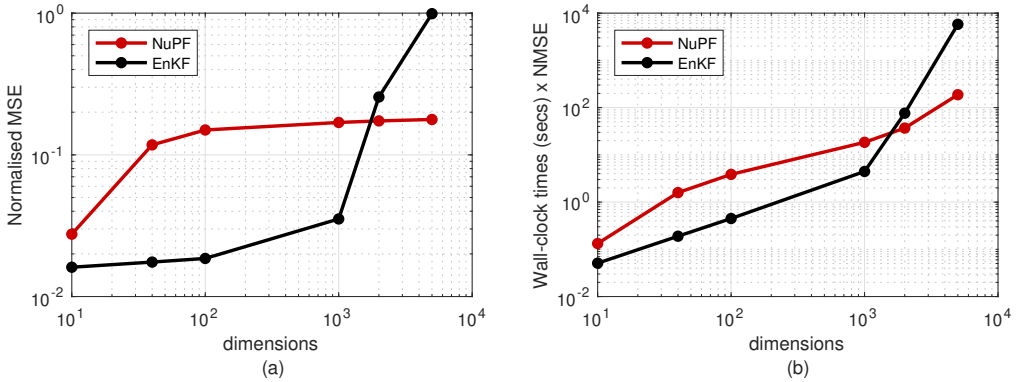

Figure 3.6: Comparison of the NuPF with the EnKF for the stochastic Lorenz 96 model with increasing dimension d and fixed number of particles $N = 500$ (this is the same as the number of ensemble members in the EnKF). We have run 1,000 independent Monte Carlo trials for this experiment. Plot (a): NMSE versus dimension d. The EnKF attains a smaller error for lower dimensions but then it explodes for $d > 10^3$, while the NuPF remains stable. Plot (b): Running-times×NMSE plot for the same set of simulations. It can be seen that the overall performance of the NuPF is better beyond 1K dimensions compared to the EnKF.

In a second computer experiment, we compared the NuPF with the EnKF. Fig. 3.6(a) shows how the NMSE of the two algorithms grows as the model dimension d increases and the number of particles N is kept fixed. In particular, the EnKF attains a better performance for smaller dimensions (up to $d = 10^3$), however its NMSE blows up for $d > 10^3$ while the performance of the NuPF remains stable. The running time of the EnKF was also higher than the running time of the NuPF in the range of higher dimensions ($d \geq 10^3$).

3.5.5 Assessment of bias

In this section, we numerically quantify the bias of our algorithm on a low-dimensional linear-Gaussian state-space model. To assess the bias, we compute the marginal likelihood estimates given by the BPF and the NuPF. The reason for this choice is the well-known unbiasedness property of the BPF [33], i.e., the marginal likelihood estimates given by the BPF are unbiased[2]. Since the NuPF leads to biased marginal likelihood estimates (with respect to the original model), but the BPF does not, our aim here is to quantify this bias by looking at the difference. To this end, we choose a simple linear-Gaussian state space model for which

[2]Note that the estimates of the expectations given by the BPF and the NuPF are both biased and the bias vanishes with the same rate for both algorithms as a result of Theorem 3.1.

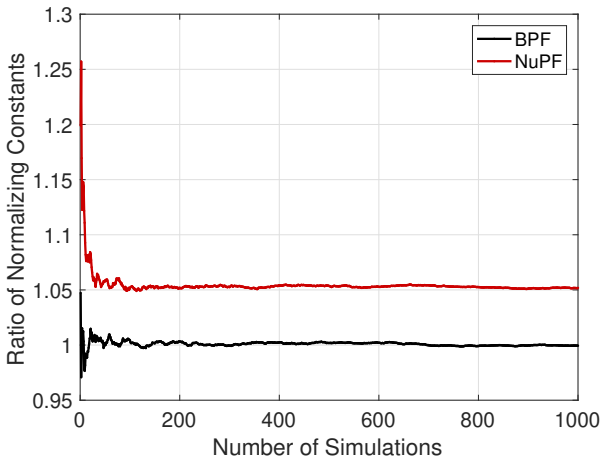

Figure 3.7: The evolution of $\bar{Z}^N_{\mathrm{BPF}}/Z^\star$ (black) and $\bar{Z}^N_{\mathrm{NuPF}}/Z^\star$ (red) for the number of simulations $K = 1, \ldots, 1000$ with $N = 100,000$ for both filters. It can be seen that the ratio for the BPF converges to 1, implying unbiasedness. The ratio for the NuPF converges to a number slightly larger than 1, showing the induced bias on marginal likelihood estimates.

the marginal likelihood can be exactly computed as a byproduct of the Kalman filter. We then compare this exact marginal likelihood to the estimates given by the BPF and the NuPF.

Particularly, we define the state-space model,

$$x_0 \sim \mathcal{N}(x_0; \mu_0, P_0) \tag{3.22}$$

$$x_t | x_{t-1} \sim \mathcal{N}(x_t; x_{t-1}, Q), \tag{3.23}$$

$$y_t | x_t \sim \mathcal{N}(y_t; C_t x_t, R), \tag{3.24}$$

where $(C_t)_{t\geq 0} \in [0,1]^{1\times 2}$ is a sequence of randomly chosen observation matrices, μ_0 is a zero vector, and $x_t \in \mathbb{R}^2$ and $y_t \in \mathbb{R}$. The dynamical model is correlated, i.e.,

$$Q = \begin{bmatrix} 2.7 & -0.48 \\ -0.48 & 2.05 \end{bmatrix},$$

and $R = 1$. We have chosen the prior covariance as $P_0 = I_{d_x}$. We have simulated the system for $T = 200$ time steps. Given a fixed observation sequence $y_{1:T}$, the marginal likelihood for the system given in Eqs. (3.23)-(3.24) is

$$Z^\star = \mathsf{p}(y_{1:T}),$$

which can be exactly computed via the Kalman filter.

We denote the estimate of $Z^\star$ given by the BPF and the NuPF as Z^N_{BPF} and Z^N_{NuPF}, respectively. It is well-known that the following unbiasedness property holds for the BPF [33],

$$\mathbb{E}[Z^N_{\mathrm{BPF}}] = Z^\star, \tag{3.25}$$

where $\mathbb{E}[\cdot]$ denotes the expectation with respect to the particles. Numerically, this suggests that as one runs identical Monte Carlo simulations to obtain $\{Z^{N,k}_{\mathrm{BPF}}\}_{k=1}^K$ and compute the average

$$\bar{Z}^N_{\mathrm{BPF}} = \frac{1}{K}\sum_{k=1}^{K} Z^{N,k}_{\mathrm{BPF}}, \tag{3.26}$$

then it follows from the unbiasedness property (3.25) that the ratio of the average in (3.26) and the true value $Z^\star$ should satisfy,

$$\frac{\bar{Z}^N_{\mathrm{BPF}}}{Z^\star} \to 1 \quad \text{as} \quad K \to \infty.$$

Since the marginal likelihood estimates provided by the NuPF are not unbiased with respect to the original SSM and tend to take higher values, if we define

$$\bar{Z}^N_{\mathrm{NuPF}} = \frac{1}{K}\sum_{k=1}^{K} Z^{N,k}_{\mathrm{NuPF}},$$

then as $K \to \infty$, we should see,

$$\frac{\bar{Z}^N_{\mathrm{NuPF}}}{Z^\star} \to 1 + \epsilon \quad \text{as} \quad K \to \infty,$$

for some $\epsilon > 0$.

We have conducted the experiment suggested by this reasoning by choosing $N = 100,000$ for the both filters and $\gamma = 0.1$ for the NuPF with gradient nudging. From Fig. 3.7, one can see that we exactly observe the behavior suggested by the theory. This shows that, with respect to the original SSM, the use of nudging induces some upward bias and provides higher marginal likelihood estimates. The induced bias is relatively small for this experiment.

3.6 Experimental results on model inference

In this section, we illustrate the application of the NuPF to estimate the parameters of a financial time-series model. In particular, we adopt a stochastic-volatility SSM and we aim at estimating its unknown parameters (and track its state variables) using the EURUSD log-return data from 2014-12-31 to 2016-12-31 (obtained

from `www.quandl.com`). For this task, we apply two recently proposed Monte Carlo schemes: the nested particle filter (NPF) [145] (a purely recursive, particle-filter style Monte Carlo method) and the particle Metropolis-Hastings (pMH) algorithm [140] (a batch Markov chain Monte Carlo procedure). In their original forms, both algorithms use the marginal likelihood estimators given by the BPF to construct a Monte Carlo approximation of the posterior distribution of the unknown model parameters. Here, we compare the performance of these algorithms when the marginal likelihoods are computed using either the BPF or the proposed NuPF.

We assume the stochastic volatility SSM [149],

$$x_0 \sim \mathcal{N}\left(\mu, \frac{\sigma_v^2}{1-\phi^2}\right), \tag{3.27}$$

$$x_t | x_{t-1} \sim \mathcal{N}(\mu + \phi(x_{t-1} - \mu), \sigma_v^2), \tag{3.28}$$

$$y_t | x_t \sim \mathcal{N}(0, \exp(x_t)), \tag{3.29}$$

where $\mu \in \mathbb{R}$, $\sigma_v \in \mathbb{R}_+$, and $\phi \in [-1, 1]$ are fixed but unknown parameters. The states $(x_t)_{1 \leq t \leq T}$ are log-volatilities and the observations $(y_t)_{1 \leq t \leq T}$ are log-returns. We follow the same procedure as [150] to pre-process the observations. Given the historical price sequence $s_0, \ldots, s_T$, the log-return at time t is calculated as

$$y_t = 100 \log(s_t / s_{t-1})$$

for $1 \leq t \leq T$. Then, given $y_{1:T}$, we tackle the joint Bayesian estimation of $x_{1:T}$ and the unknown parameters $\theta = (\mu, \sigma_v, \phi)$. In the next two subsections we compare the conventional BPF and the NuPF as building blocks of the NPF and the pMH algorithms.

3.6.1 Nudging the nested particle filter

The NPF in [145] consists of two layers of particle filters which are used to jointly approximate the posterior distributions of the parameters and the states. The filter in the first layer builds a particle approximation of the marginal posterior distribution of the parameters. Then, for each particle in the parameter space, say $\theta^{(i)}$, there is an *inner* filter that approximates the posterior distribution of the states conditional on the parameter vector $\theta^{(i)}$. The inner filters are classical particle filters, which are essentially used to compute the importance weights (marginal likelihoods) of the particles in the parameter space. In the implementation of [145], the inner filters are conventional BPFs. We have compared this conventional implementation with an alternative one where the BPFs are replaced by the NuPFs. For a detailed description of the NPF, see [145].

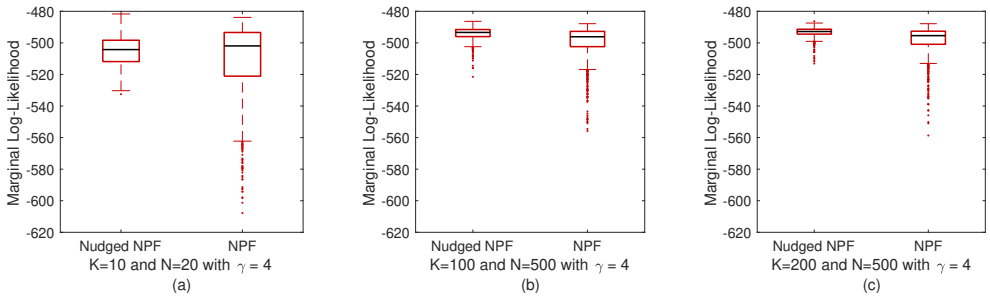

Figure 3.8: Model evidence estimates produced by the nudged NPF and the conventional NPF with varying computational effort. From (a) to (c), it can be seen that, as we increase the number of particles in the parameter space and the state space, the variances of the estimates are smaller. The nudged NPF results in much more stable estimates, with lower variance and few extreme values.

In order to assess the performances of the nudged and classical versions of the NPF, we compute the model evidence estimate estimate given by the nested filter by integrating out both the parameters and the states. In particular, if the set of particles in the parameter space at time t is $\{\theta_t^{(i)}\}_{i=1}^K$ and for each particle $\theta_t^{(i)}$ we have a set of particles in the state space $\{x_t^{(i,j)}\}_{j=1}^N$, we compute

$$\widehat{\mathsf{p}(y_{1:T})} = \prod_{t=1}^{T} \left[\frac{1}{KN} \sum_{i=1}^{K} \sum_{j=1}^{N} g_t(x_t^{(i,j)}) \right].$$

The model evidence quantifies the fitness of the stochastic volatility model for the given dataset, hence we expect to see a higher value when a method attains a better performance (the intuition is that if we have better estimates of the parameters and the states, then the model will fit better). For this experiment, we compute the model evidence for the nudged NPF *before* the nudging step, so as to make the comparison with the conventional algorithm fair.

We have conducted 1,000 independent Monte Carlo runs for each algorithm and computed the model evidence estimates. We have used the same parameters and the same setup for the two versions of the NPF (nudged and conventional). In particular, each unknown parameter is jittered independently. The parameter μ is jittered with a zero-mean Gaussian kernel variance $\sigma_\mu^2 = 10^{-3}$, the parameter σ_v is jittered with a truncated Gaussian kernel on $(0, \infty)$ with variance $\sigma_{\sigma_v}^2 = 10^{-4}$, and the parameter ϕ is jittered with a zero-mean truncated Gaussian kernel on $[-1, 1]$, with variance $\sigma_\phi^2 = 10^{-4}$. We have chosen a large step-size for the nudging step, $\gamma = 4$, and we have used batch nudging with $M = \lfloor \sqrt{N} \rfloor$.

The results in Fig. 3.8 demonstrate empirically that the use of the nudging

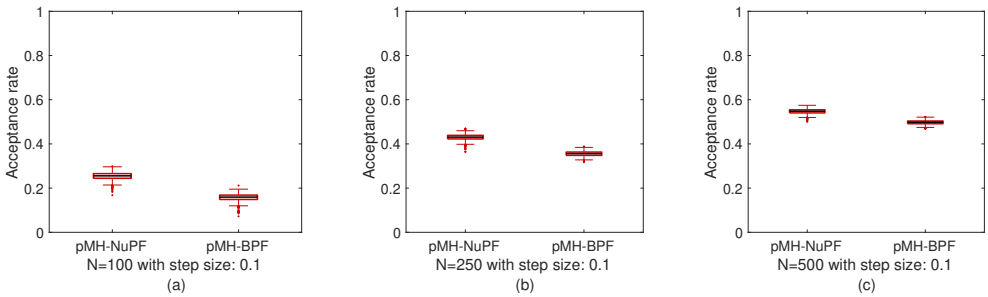

Figure 3.9: Empirical acceptance rates computed for the pMH running BPF and the pMH running NuPF. From (a), it can be seen that there is a significant increase in the acceptance rates when the number of particles are low, e.g., $N = 100$. From (b) and (c), it can be seen that the pMH-NuPF is still better for increasing number of particles but the pMH-BPF is catching up with the performance of the pMH-NuPF.

step within the NPF reduces the variance of the model evidence estimators, hence it improves the stability and reliability of the NPF.

3.6.2 Nudging the particle Metropolis-Hastings

The pMH algorithm is a Markov chain Monte Carlo (MCMC) method for inferring parameters of general SSMs [140]. The pMH uses PFs as auxiliary devices to estimate parameter likelihoods in a similar way as the NPF uses them to compute importance weights. In the case of the pMH, these estimates should be unbiased and they are needed to determine the acceptance probability for each element of the Markov chain. For the details of the algorithm, see [140] (or [150] for a tutorial-style introduction). Let us note that the use of NuPF does not lead to an unbiased estimate of the likelihood with respect to the assumed SSM. However, as discussed in Section 3.4.3, the use of nudging implies an implicit SSM, and the likelihood estimates are unbiased with respect to this model. Thus, one can view the use of nudging here as an implementation of pMH with an implicit SSM derived from the original SSM as well.

We have carried out a computer experiment to compare the performance of the pMH scheme using either BPFs or NuPFs to compute acceptance probabilities. The two algorithms are labeled pMH-BPF and pMH-NuPF, respectively, hereafter. The parameter priors in the experiment are

$$p(\mu) = \mathcal{N}(0, 1) \qquad p(\sigma_v) = \mathcal{G}(2, 0.1) \qquad p(\phi) = \mathcal{B}(120, 2)$$

where $\mathcal{G}(a, \theta)$ denotes the Gamma pdf with shape parameter a and scale parameter θ, and $\mathcal{B}(\alpha, \beta)$ denotes the Beta pdf with shape parameters (α, β). Unlike [150],

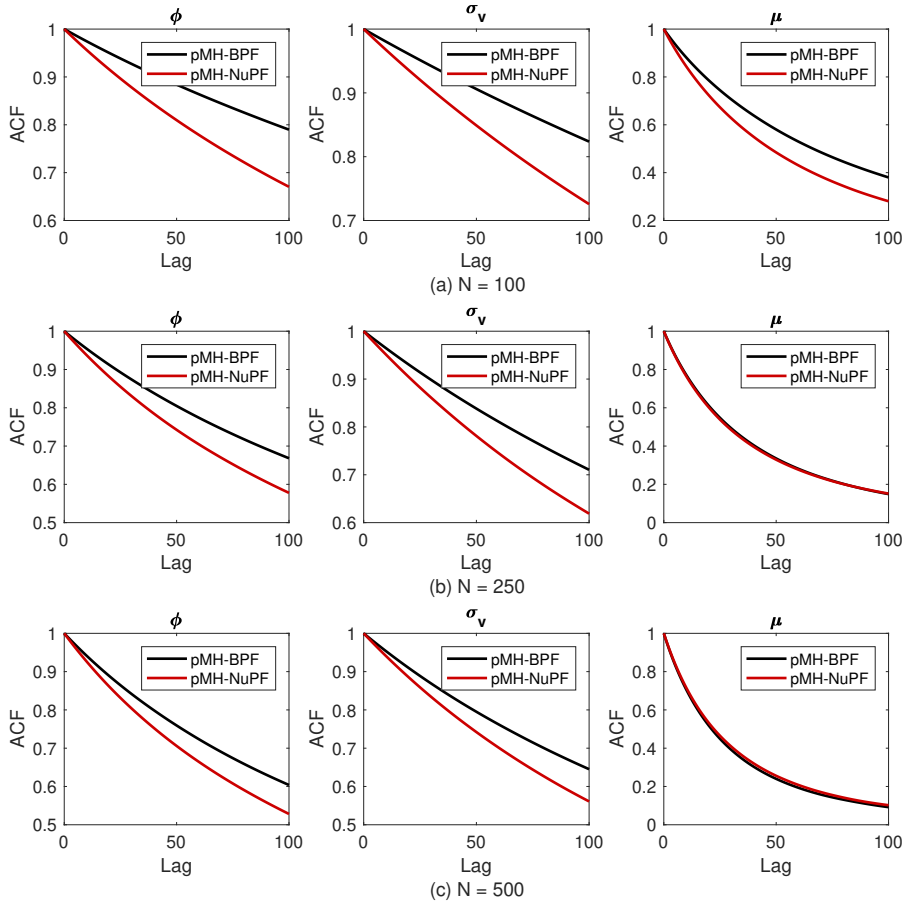

Figure 3.10: Empirical autocorrelation functions (ACFs) computed for the pMH -BPF and the pMH-NuPF. From (a)–(c), it can be seen that using the NuPF instead of BPF within the pMH causes faster autocorrelation decay. These results are obtained by averaging ACFs over $1,000$ Monte Carlo runs.

who use a truncated Gaussian prior centered on 0.95 with a small variance for ϕ, we use the Beta pdf, which is defined on $[0, 1]$, with mean $\alpha/(\alpha + \beta) = 0.9836$, which puts a significant probability mass on the interval $[0.9, 1]$.

We have compared the pMH-BPF algorithm and the pMH-NuPF scheme (using a batch nudging procedure with $\gamma = 0.1$ and $M = \lfloor\sqrt{N}\rfloor$) by running 1,000 independent Monte Carlo trials. We have computed the marginal likelihood estimates in the NuPF *after* the nudging step.

The resulting empirical acceptance rates can be seen from Fig. 3.9. It is observed that the use of nudging leads to higher acceptance rates, making the pMH more efficient to use.

Another key figure of merit for an MCMC algorithm is the degree of correlation

in the chain. Fig. 3.10 displays the average autocorrelation functions (ACFs) of the chains obtained in the 1,000 independent simulations. We see that the autocorrelation of the chains produced by the pMH-NuPF method decays more quickly than the autocorrelation of the chains output by the conventional pMH-BPF. Less correlation can be expected to translate into better estimates as well for a fixed length of the chain.

4

Stochastic optimization as Bayesian inference

In this chapter, we reformulate the stochastic optimization problem as an inference problem by describing probabilistic models where the maxima of the probability density of interest coincide with the minima of the original cost function. We then focus on the Gaussian case and show that the inference rules coincide with an incremental optimization procedure. Finally, we propose a more general Monte Carlo simulation method which aims at estimating the minima of the cost function by sampling from a posterior probability measure and constructing an estimate of the maxima of its associated probability density.

4.1 Introduction

In recent years, a surge of interest in stochastic optimization methods has been propelled by the recent developments in signal processing and machine learning. In particular, the following optimization problem,

$$\min_{\theta \in \mathbb{R}^d} f(\theta) := \sum_{i=1}^{n} f_i(\theta) \tag{4.1}$$

where n is very large, has attracted significant attention in the machine learning literature.

In this chapter, our goal is to develop new probabilistic methods for stochastic optimization. The algorithms we introduce are *incremental*, meaning that they subsample the data points without replacement. Since the stochastic optimization methods which draw our interest are sequential algorithms which use subsets of data at each iteration, it is natural to reframe them as sequential Bayesian inference methods. Thus, in Section 4.2 we first formulate a general Bayes recursion, defined via a sequence of potential functions $(G_t)_{t\geq 1}$ which do not necessarily correspond to any explicit likelihood. In particular, the potentials are constructed using mini-batches of the cost function, which are sampled without replacement. We show that the probability distributions resulting from this recursion can be used to find the global minima of the original cost function in (4.1).

Next, in Section 4.3, we demonstrate the correspondence between optimization methods and probabilistic inference for the Gaussian case. In particular, we show that the proximal mappings can be seen as Bayesian updates. We show that this relationship can be made entirely explicit for the linear regression case. In this case, the update for the Bayesian linear regression coincides with the proximal mapping for the corresponding linear-quadratic cost function. Using this correspondence, we show that the update rules for the incremental proximal method in order to perform large scale regression take a very similar form to the updates of the Kalman filter. In other words, we show that for a specific choice of the f_i's in (4.1), it is possible to obtain a well-known incremental optimization method, called the incremental proximal method [84] (IPM), as a sequential Bayes update with Gaussian likelihoods. In this case, the Bayes recursion can be implemented exactly. Not surprisingly, these exact equations coincide with the *Kalman updates* and it can be shown that they correspond to a variable-metric extension of the IPM. Then we generalize this idea to a broad class of nonlinear least squares problems and develop an IPM-type optimizer in the form of an extended Kalman filter (EKF), which we argue can provide a systematic way for the derivation of practical procedures.

Finally, we move on to the general case with general potentials in Section 4.4. Since the Bayes update cannot be computed in closed form for general potentials, we propose a particle method to simulate these updates. To be specific, we devise a parallel sequential Monte Carlo optimizer (PSMCO) to minimize cost functions with finite-sum structure. The PSMCO is a zeroth-order stochastic optimization algorithm, in the sense that it only uses evaluations of small batches of individual components $f_i(\theta)$ in (1.1). The proposed scheme proceeds by constructing parallel samplers each of which aims at minimizing (1.1). Each sampler performs *subsampling without replacement* to obtain its mini-batches of individual components and

passes over the dataset only once. Using these mini-batches, the PSMCO constructs potential functions, propagates samples via a jittering scheme [151], and selects samples by applying a weighting-resampling procedure. The communication between parallel samplers is only necessary when an estimate of the minimum is required. In this case, the best performing sampler is selected and the minimum is estimated. We analytically prove that the estimate provided by each sampler converges almost surely to a global minimum of the cost function when the number of Monte Carlo samples at each sampler tends to infinity. We then provide numerical results for two optimization problems where classical stochastic optimization methods struggle to perform. We remark the difference between the proposed scheme and the SMC-based schemes in [107, 103] where the authors partitioned the parameter and modeled it as a dynamical system which is suitable for many global optimization problems [152]. In contrast, we aim at estimating the full parameter at each iteration.

4.2 Stochastic optimization as Bayesian inference

In this section, we describe how to construct a sequence of probability distributions that can be linked to the solution of problem (1.1). Let $\pi_0 \in \mathcal{P}(\Theta)$ be the initial element of the sequence. We construct the rest of the sequence recursively as

$$\pi_t(\mathrm{d}\theta) = \pi_{t-1}(\mathrm{d}\theta)\frac{G_t(\theta)}{\int_\Theta G_t(\theta)\pi_{t-1}(\mathrm{d}\theta)}, \quad \text{for } t \geq 1, \tag{4.2}$$

where $G_t : \Theta \to \mathbb{R}_+$ are termed *potential functions* [33]. The key idea is to associate these potentials $(G_t)_{t\geq 1}$ with mini-batches of individual components of the cost function (subsets of the f_i's) in order to construct a sequence of measures $\pi_0, \pi_1, \ldots, \pi_T$ such that (for a prescribed value of T) the global maxima of the density of π_T match the global minima of $f(\theta)$. We remark that the measures $\pi_1, \ldots, \pi_T$ are all absolutely continuous with respect to π_0 if the potential functions G_t, $t = 1, \ldots, T$ are bounded and positive.

To construct the potentials, we use mini-batches consisting of K individual functions f_i for each iteration t. To be specific, we randomly select subsets of indices $\mathcal{I}_t, t = 1, \ldots, T$, by drawing uniformly from $\{1, \ldots, n\}$ without replacement. Each subset has $|\mathcal{I}_t| = K$ elements, in such a way that we obtain T subsets satisfying $\bigcup_{i=1}^{T} \mathcal{I}_t = [L]$ and $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$ when $i \neq j$. Finally, we define the potential functions $(G_t)_{t\geq 1}$ as

$$G_t(\theta) = \exp\left(-\sum_{i\in\mathcal{I}_t} f_i(\theta)\right), \qquad t = 1, \ldots, T. \tag{4.3}$$

Below, we provide a result that establishes a precise connection between the optimization problem in (4.1) and the sequence of probability measures defined in (4.2), provided that Assumption 4.1 below is satisfied.

Assumption 4.1. *The sequence of functions $(G_t)_{t\geq 1}$ are positive and bounded, i.e., $G_t(\theta) > 0 \quad \forall \theta \in \Theta$ and $G_t \in B(\Theta)$.*

Next, we prove the result showing the relationship between the minima of $f(\theta)$ and the maxima of $\frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}$.

Proposition 4.1. *Assume that the potentials are selected as in (4.3) for $1 \leq t \leq T$, with $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$ and $\bigcup_i \mathcal{I}_i = [n]$. Let π_T be the T-th probability measure constructed by means of recursion (4.2). If Assumption 4.1 holds and $\pi_0 \in \mathcal{P}(\Theta)$, then*

$$\operatorname*{argmax}_{\theta\in\Theta} \frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta) = \operatorname*{argmin}_{\theta\in\Theta} \sum_{i=1}^{n} f_i(\theta),$$

where $\frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta) : \Theta \to \mathbb{R}_+$ denotes the Radon-Nikodym derivative of π_T with respect to the prior measure π_0.

Proof. See Appendix A.3. □

For conciseness, we abuse the notation and use $\pi(\theta)$, $\theta \in \Theta$, to indicate the pdf associated to a probability measure $\pi(\mathrm{d}\theta)$. The two objects are distinguished clearly by the context (e.g., for an integral (φ, π), π is necessarily a measure) but also by their arguments. The probability measure $\pi(\cdot)$ takes arguments $\mathrm{d}\theta$ or $A \in \mathcal{B}(\Theta)$, while the pdf $\pi(\theta)$ is a function $\Theta \to [0, \infty)$.

Remark 4.1. Notice that, when π_0 is a uniform probability measure on Θ, we simply have

$$\pi_T(\theta) \propto \exp\left(-\sum_{i=1}^{n} f_i(\theta)\right), \qquad \text{for } \theta \in \Theta,$$

where $\pi_T(\theta)$ denotes the pdf (w.r.t. Lebesgue measure) of the measure $\pi_T(\mathrm{d}\theta)$. □

Remark 4.2. Moreover, if we choose

$$\pi_0(\theta) \propto \exp\left(-f_1(\theta)\right) \tag{4.4}$$

and then select subsets of indices such that $\bigcup_{t=1}^{T} \mathcal{I}_t = \{2, \ldots, n\}$ then it readily follows that

$$\pi_T(\theta) \propto \exp\left(-\sum_{i=1}^{n} f_i(\theta)\right), \qquad \text{for } \theta \in \Theta.$$

When a Monte Carlo is scheme used to realize recursion (4.2), the use of a prior of the form (4.4) requires the ability to sample from it. □

Therefore, if we can construct the sequence described by (4.2), then we can replace the minimization problem of $f(\theta)$ in (1.1) by the maximization of the pdf $\pi_T(\theta)$. This relationship was exploited in a Gaussian setting in [91], i.e., the special case of a Gaussian prior π_0 and log-quadratic potentials $(G_t)_{t\geq 1}$ (corresponding to Gaussian likelihoods in a Bayesian model), which makes it possible to implement recursion (4.2) analytically. The solution of this special case can be shown to match a well-known stochastic optimization algorithm, called the incremental proximal method [84], with a variable-metric as we derive in the next section.

4.3 Incremental proximal method as inference

The recursion (4.2) is general, however, it is not possible implement it exactly for any π_0 and $(G_t)_{t\geq 1}$. In this section, we summarize a special case, where (4.2) can be implemented exactly. We also show that this special case matches to a well-known optimization scheme, called the incremental proximal method (IPM) [84]. We show that, in particular, the probabilistic interpretation of the IPM leads to a development of a variable-metric IPM scheme. For a review of variable-metric methods see, e.g., [153, 154]. We focus on the case where $\Theta = \mathbb{R}^d$ within this section.

4.3.1 Proximal operators as Bayes updates

Consider a generic, convex (for simplicity) function f. We recall the proximal operator of f is given as in Definition (2.1)

$$\text{prox}_{\gamma f,V}(\cdot) = \underset{\theta\in\mathbb{R}^d}{\text{argmin}}\, f(\theta) + \frac{1}{2\gamma}\|\theta - \cdot\|_{2,V}^2,$$

where $\|\theta\|_{2,V} = \sqrt{\theta^\top V^{-1}\theta}$ is the Mahalanobis norm. In this section, we first show that, for certain f, the proximal operator can be interpreted as a Bayes update. In particular, consider

$$f(\theta) = \frac{1}{2}(y - \mathbf{x}^\top\theta)^2,$$

where $y \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^d$. We first derive the proximal operator of f.

Proposition 4.2. *Let $f(\theta) = \frac{1}{2}(y - \mathbf{x}^\top\theta)^2$. Then the proximal operator of f,*

$$\tilde{\theta} = \text{prox}_{\gamma f,V_0}(\theta_0) = \underset{\theta\in\mathbb{R}^d}{\text{argmin}}\, f(\theta) + \frac{1}{2\gamma}\|\theta - \theta_0\|_{2,V_0}^2,$$

is realized by

$$\tilde{\theta} = \theta_0 + \frac{V_0 \mathbf{x}(y - \mathbf{x}^\top \theta_0)}{\gamma^{-1} + \mathbf{x}^\top V \mathbf{x}}. \tag{4.5}$$

Proof. See Appendix A.3. □

Interestingly, it is possible to see (4.5) as a Bayes update for a Gaussian probability model, as made explicit by the following proposition.

Proposition 4.3. *Consider a prior and a likelihood function, respectively,*

$$\pi_0(\theta) = \mathcal{N}(\theta; \theta_0, V_0),$$
$$G(\theta) = \mathcal{N}(y; \mathbf{x}^\top \theta, \gamma^{-1}).$$

Then, the pdf

$$\pi(\theta) = \frac{G(\theta)\pi_0(\theta)}{\int_{\mathbb{R}^d} G(\theta)\pi_0(\theta)\mathrm{d}\theta},$$

is Gaussian, namely,

$$\pi(\theta) = \mathcal{N}(\theta; \tilde{\theta}, V),$$

where

$$\tilde{\theta} = \theta_0 + \frac{V_0 \mathbf{x}(y - \mathbf{x}^\top \theta_0)}{\gamma^{-1} + \mathbf{x}^\top V \mathbf{x}}, \tag{4.6}$$

$$V = V_0 - \frac{V_0 \mathbf{x}\mathbf{x}^\top V_0}{\gamma^{-1} + \mathbf{x}^\top V_0 \mathbf{x}}. \tag{4.7}$$

Proof. This proposition is a special case of Lemma 2.1. □

Remark 4.3. Note that eqs. (4.5) and (4.6) are identical. This means that the realization of the proximal operator of f, defined with a symmetric positive definite matrix V_0 and parameter γ, is identical to the realization of the Bayes update. As a byproduct of the Bayes update, one obtains V via (4.7), which then can be used to quantify the uncertainty of the output of the proximal operator. We aim at utilizing this idea to develop uncertainty-aware stochastic optimization schemes. □

In the next section, we apply this interpretation to the family of incremental proximal methods in order to get an online probabilistic optimizer. Perhaps not surprisingly, this algorithm is related to the Kalman filter for the linear case. Then we discuss its extension to nonlinear optimization problems.

4.3.2 The IPM as a Kalman filter

As introduced in Section 2.4.5, at iteration t the incremental proximal method solves the problem

$$\theta_t = \text{prox}_{\gamma f_t}(\theta_{t-1}),$$

where f_t is selected randomly from $[T] = \{1, \dots, T\}$ without replacement[1]. In this section, we first define a general, variable-metric IPM with a sequence of (as yet unspecified) symmetric positive definite matrices $(V_t)_{t\geq 0}$. We define the variable-metric IPM as

$$\theta_t = \text{prox}_{\gamma f_t, V_{t-1}}(\theta_{t-1}) = \underset{\theta\in\mathbb{R}^d}{\text{argmin}}\, f_t(\theta) + \frac{1}{2\gamma}\|\theta - \theta_{t-1}\|^2_{2,V_{t-1}}, \tag{4.8}$$

where $\|\theta\|_{2,V} = \sqrt{\theta^\top V^{-1}\theta}$ and the sequence $(V_t)_{t\geq 1}$ is specified by the user. Now, assume that we are given a dataset $(y_t, \mathbf{x}_t)_{0\leq t\leq T}$ and aim at minimizing a cost function of the form (4.1) with $f_t(\theta) = \frac{1}{2}(y_t - \mathbf{x}_t^\top\theta)^2$. The next proposition states the explicit form of (4.8) for this case.

Proposition 4.4. *Let $f_t(\theta) = \frac{1}{2}(y_t - \mathbf{x}_t^\top\theta)^2$ for $t = 1, \dots, T$. Then, given a sequence of symmetric, positive definite matrices $(V_t)_{0\leq t\leq T-1}$, the recursion (4.8) can be explicitly written as*

$$\theta_t = \theta_{t-1} + \frac{V_{t-1}\mathbf{x}_t(y_t - \mathbf{x}_t^\top\theta_{t-1})}{\gamma^{-1} + \mathbf{x}_t^\top V_{t-1}\mathbf{x}_t}. \tag{4.9}$$

Proof. This result follows straightforwardly from the proof of Proposition 4.2. □

Next, we aim at building up a probabilistic version of the problem. As we summarized in Section 4.2, in order to minimize f we need to define a prior and a sequence of potential functions for each t. In this section, we choose a prior of the form

$$\pi_0(\theta) = \mathcal{N}(\theta; \theta_0, V_0). \tag{4.10}$$

and define the sequence of potentials $(G_t)_{t\geq 1}$ such that

$$G_t(\theta) = \mathcal{N}(y_t; \mathbf{x}_t^\top\theta, \gamma^{-1}). \tag{4.11}$$

[1]Throughout the chapter, we abuse the notation for randomly selected individual functions. In particular, we sample $i_t \sim \{1, \dots, T\}$ without replacement and set $f_t := f_{i_t}$ in our notation. Since the dataset can be reshuffled and then processed sequentially, we always assume that we process the dataset from 1 to T (unless otherwise specified) without any loss of generality. Note also that $T = n$ for this case, since we sample each individual function with $K = 1$.

Note, again, that this implies that we take a single component at each iteration, i.e., we select the mini-batch size $K = 1$. For the model (4.10) and (4.11), the recursion (4.2) can be implemented exactly and is given by the following proposition.

Proposition 4.5. *Assume that the prior is given by*

$$\pi_0(\theta) = \mathcal{N}(\theta; \theta_0, V_0)$$

and the potential functions G_t are specified as

$$G_t(\theta) = \mathcal{N}(y_t; \mathbf{x}_t^\top \theta, \gamma^{-1})$$

for $t = 1, \ldots, T$. Then the recursion (4.2) can be implemented exactly and yields

$$\pi_t(\theta) = \mathcal{N}(\theta; \theta_t, V_t), \tag{4.12}$$

where

$$\theta_t = \theta_{t-1} + \frac{V_{t-1}\mathbf{x}_t(y_t - \mathbf{x}_t^\top \theta_{t-1})}{\gamma^{-1} + \mathbf{x}_t^\top V_{t-1}\mathbf{x}_t}, \tag{4.13}$$

$$V_t = V_{t-1} - \frac{V_{t-1}\mathbf{x}_t\mathbf{x}_t^\top V_{t-1}}{\gamma^{-1} + \mathbf{x}_t^\top V_{t-1}\mathbf{x}_t}, \tag{4.14}$$

for $t \geq 1$.

Proof. This is a special case of Lemma 2.2. □

Remark 4.4. Note that recursions (4.9) and (4.13) are identical, similar to the static case we presented in the last section. However, in this case there is a fundamental difference between the two methods. For the variable-metric IPM, (4.9) can be only defined when the sequence of matrices $(V_t)_{t\geq 1}$ are predefined by the user. Otherwise, the variable-metric IPM does not yield an explicit method. The most common implementation of the IPM simply assumes $V_t = I_d$ for $t = 0, \ldots, T-1$. In contrast, the Kalman filter given in Proposition 4.5 produces mean estimates in the same form of (4.9) and, in addition, it produces the sequence $(V_t)_{t\geq 1}$ in the form of a sequence of posterior covariance matrices. In other words, the Kalman filter is identical to the variable-metric IPM, except that it automatically determines the metrics. In the next section, we develop a nonlinear extension of the variable-metric IPM, which cannot be implemented from an IPM perspective since the choice of general f_t will not yield a computable proximal operator. But given the probabilistic view, we can still develop an approximate probabilistic optimizer from the filtering perspective via the use of the extended Kalman filter (EKF). □

Remark 4.5. Recall that in Proposition 4.1, we have shown that the maxima of the Radon-Nikodym derivative $\frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta)$ coincide with the minima of the cost function. In particular, if the maximum is unique, i.e., we have unique $\theta^\star$ such that

$$\theta^\star = \operatorname*{argmax}_\theta \frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta), \tag{4.15}$$

then $\theta^\star$ is also the global minimum of the cost function. In this section, however, we take the maximum (i.e. the mean) of the posterior distribution $\pi_T(\theta)$, namely θ_T, as an estimate of the minimum instead of the maximum of the Radon-Nikodym derivative $\frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta)$ for the sake of computational simplicity. Note that the solution of the problem (4.15) can also be exactly solved for this case. More precisely, the exact solution can be straightforwardly derived using (4.10) and (4.12) and is given by

$$\theta^\star = (V_T^{-1} - V_0^{-1})^{-1}(V_T^{-1}\theta_T - V_0^{-1}\theta_0). \tag{4.16}$$

Then one can show that when the entries of V_T are close to zero and V_0 is chosen so that the initial uncertainty is large (e.g. $V_0 = v_0 I_d$ with $v_0 > 0$ large), the effect of the prior vanishes and $\theta^\star \approx \theta_T$. In order to see this, assume (for simplicity) that we have obtained $V_T = \epsilon I_d$ with $\epsilon > 0$ small and $V_0 = v_0 I_d$. Then,

$$\theta^\star = \frac{\epsilon v_0}{v_0 - \epsilon}\left(\frac{1}{\epsilon}\theta_T - \frac{1}{v_0}\theta_0\right),$$

which yields

$$\theta^\star = \frac{v_0}{v_0 - \epsilon}\theta_T - \frac{\epsilon}{v_0 - \epsilon}\theta_0$$

which justifies taking $\theta^\star \approx \theta_T$ for large v_0 and small ϵ. □

4.3.3 EKF as an approximate IPM

In this section, we consider a nonlinear regression problem. Given observations $\mathbf{y}$, we would like to obtain $y_t \approx h(\mathbf{x}_t, \theta)$ where $h(\cdot, \theta)$ is a nonlinear function of θ. Since the $\mathbf{x}_t$'s are fixed, we set $h_t(\theta) := h(\mathbf{x}_t, \theta)$ for notational conciseness. Note that $h_t : \mathbb{R}^d \to \mathbb{R}$. Then, we would like to solve a problem of the form (4.1). In particular, we aim at solving

$$\min_{\theta \in \mathbb{R}^d} f(\theta) = \min_{\theta \in \mathbb{R}^d} \sum_{t=1}^{T} f_t(\theta), \tag{4.17}$$

where $f_t(\theta) = \frac{1}{2}(y_t - h_t(\theta))^2$. The incremental proximal step for this problem is given by

$$\theta_t = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}(y_t - h_t(\theta))^2 + \frac{1}{2\gamma}\|\theta - \theta_{t-1}\|^2_{2,V_{t-1}} \tag{4.18}$$

for each iteration t. Because of the nonlinearity $h_t(\cdot)$, this proximal step is intractable in general. Therefore, the typical choice for problems like in Eq. (4.17) is the SGD. In what follows, we propose the use of EKF recursions as one-step approximations of the realization of the proximal operator.

To this end, let us consider the prior probability density

$$\pi_0(\theta) = \mathcal{N}(\theta; \theta_0, V_0), \tag{4.19}$$

and the sequence of potential functions

$$G_t(\theta) = \mathcal{N}(y_t; h_t(\theta), \gamma^{-1}). \tag{4.20}$$

Since the model defined by (4.19)–(4.20) contains a nonlinearity h_t, using the EKF is a natural way to solve the regression problem. Let us denote $d_t = \nabla_\theta h_t(\theta_{t-1})$. For the model (4.19)–(4.20), the EKF recursions given in eqs. (2.40)–(2.41), for $t \geq 1$, can be written as[2]

$$\theta_t = \theta_{t-1} + \frac{V_{t-1} d_t (y_t - h_t(\theta_{t-1}))}{\gamma^{-1} + d_t^\top V_{t-1} d_t} \tag{4.21}$$

and

$$V_t = V_{t-1} - \frac{V_{t-1} d_t d_t^\top V_{t-1}}{\gamma^{-1} + d_t^\top V_{t-1} d_t}.$$

Remark 4.6. Throughout our discussion, we have kept the prior $\pi_0(\theta)$ static, meaning that θ is assumed to be random but not changing over time. While this assumption is convenient when the cost function is not changing, it does not hold in most realistic settings. For this reason, the authors of [155] consider what they call *nonstationary* losses, where the cost function is also changing with time. Tackling such a scenario is trivial from our perspective, as one only needs to modify the algorithm slightly in order to get a dynamic algorithm. In particular, in addition to the update step, one needs to employ a prediction step, according to the assumed dynamics of the parameter. One can model the degree of nonstationarity by modifying the model over θ_t and filtering algorithms extend to such settings very naturally. We leave the detailed investigation of this issue for future work. □

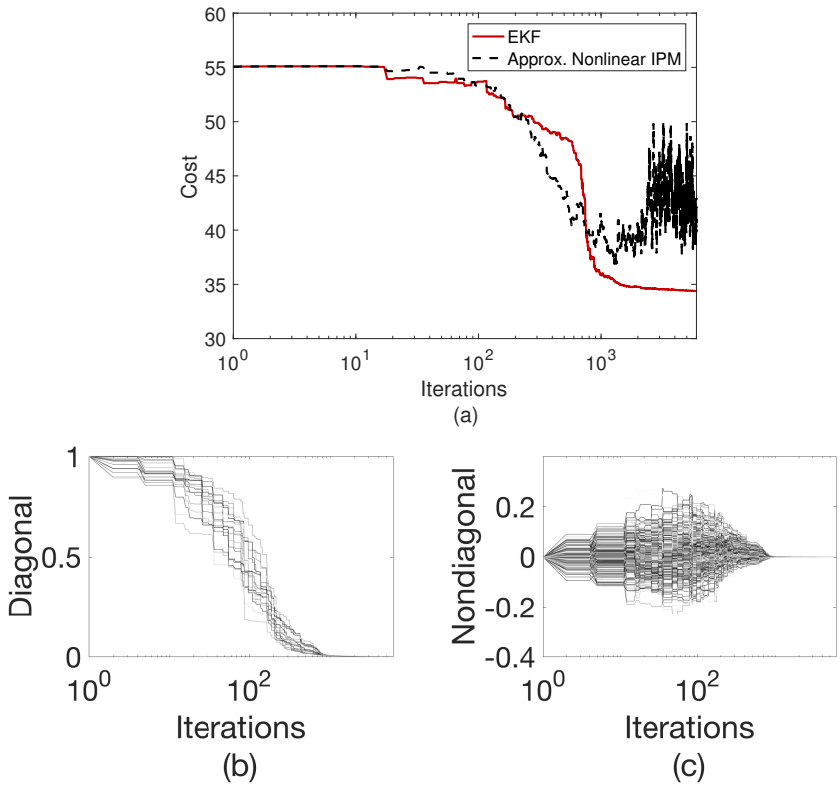

Figure 4.1: Results on fitting a sigmoid function using EKF and approximate nonlinear IPM. From (a), it can be seen that the approximate nonlinear IPM proceed towards minimum but suffers from instability while the EKF proceeds in a stable way. From (b)-(c), it can be seen that the entries of the diagonal and nondiagonal parts of the covariance matrix V_t converge to zero, which is the reason why the EKF does not suffer from instability.

4.3.4 Some numerical results

In this section, we investigate two algorithms on a simple problem of fitting a sigmoid function. The first algorithm, which we refer to as *approximate nonlinear IPM*, consists of applying a standard iterative solver for each subiteration, since the nonlinear problem of Eq. (4.18) is not solvable in general. The second algorithm is the EKF, as explained in Section 4.3.3. The model used in the experiment is of the form of Eq. (4.19)–(4.20), with

$$h_t(\theta) = \frac{1}{1 + \exp(-\alpha - \beta^\top \mathbf{x}_t)},$$

[2]Note that since this model does not have a dynamical component, the EKF consists only of the update steps (2.40)–(2.41), i.e., without the prediction steps (2.38)–(2.39).

where $\mathbf{x}_t \in \mathbb{R}^{d-1}$ denotes the inputs, and the parameter vector is $\theta = (\alpha, \beta)$ (i.e., $\theta \in \mathbb{R}^d$ with $d = 21$). Recall that, by choosing such a model, we aim at solving a problem of form given in Eq. (4.17). We set the value of the parameter $\gamma^{-1} = 0.2$ while generating the data and we use the same value in the algorithms. Also, the initial value θ_0 of the approximate IPM is set randomly while the proximal matrix is the $d \times d$ identity, denoted $V = I_d$. Similarly, the prior for the EKF is initialized with (θ_0, V_0) where $V_0 = I_d$.

Figure 4.1(a) shows that the approximate IPM suffers from numerical instability as the parameter estimate θ_t becomes close to the actual minimum. One reason for this instability is that, in proximal-type algorithms, there is no natural mechanism to reduce the size of the step taken by the algorithm. A natural remedy would be to update the proximal matrix in a way that it dampens the updates as the number of iterations increases, in a similar way to decreasing the step-size of the SGD. The EKF exactly employs this strategy in a natural way. This fact can be seen from Fig. 4.1(b)-(c) where the diagonal and nondiagonal entries of the covariance matrix V_t are plotted, respectively. It is evident that the entries of this matrix converge to zero, meaning that the update (4.21) eventually converges to some point in the parameter space.

4.4 SMC for stochastic optimization

As we have demonstrated in the previous section, the recursion (4.2) can be implemented exactly for some special cases. However, for general f, which consists of general component functions f_t, it is not possible to implement (4.2) exactly. In this section, we develop a general sequential Monte Carlo method in order to optimize cost functions of the form (4.1).

In this section we first describe a sampler to simulate from the distributions defined by recursion (4.2). We then describe an algorithm which runs these samplers in parallel. The parallelization here is not primarily motivated by the computational gain (although it can be substantial). We have found that non-interacting parallel samplers are able to keep track of multiple minima better than a single "big" sampler. For this reason, we will not focus on demonstrating computational gains in the experimental section. Rather, we will discuss what parallelization brings in terms of providing better estimates.

We consider M workers (corresponding to M samplers). Specifically, each worker sees a different configuration of the dataset, i.e., the m-th worker constructs a distinct sequence of index sets $(\mathcal{I}_t^{(m)})_{t \geq 1}$, which determine the mini-batches sampled from the full set of individual components. Having obtained different mini-

Algorithm 4.1 Sampler on a local node m

1: Sample $\theta_0^{(i,m)} \sim \pi_0$ for $i = 1, \dots, N$.
2: **for** $t \geq 1$ **do**
3: Jitter by generating samples

$$\hat{\theta}_t^{(i,m)} \sim \kappa(\mathrm{d}\theta | \theta_{t-1}^{(i,m)}) \qquad \text{for} \quad i = 1, \dots, N.$$

4: Compute weights,

$$w_t^{(i,m)} = \frac{G_t^{(m)}(\hat{\theta}_t^{(i,m)})}{\sum_{i=1}^N G_t^{(m)}(\hat{\theta}_t^{(i,m)})} \quad \text{for} \quad i = 1, \dots, N.$$

5: Resample by drawing N i.i.d. samples,

$$\theta_t^{(i,m)} \sim \hat{\pi}_t^{(m),N}(\mathrm{d}\theta) := \sum_{i=1}^N w_t^{(i,m)} \delta_{\hat{\theta}_t^{(i,m)}}(\mathrm{d}\theta), \qquad i = 1, \dots, N.$$

6: **end for**

batches which are randomly constructed, each worker then constructs different potentials $(G_t^{(m)})_{t \geq 1}$, as described in the previous section.

Workers, therefore, aim at estimating their own sequence of probability measures $\pi_t^{(m)}$ for $m \in \{1, \dots, M\}$. We denote the particle approximation of the posterior $\pi_t^{(m)}$ at time t as

$$\pi_t^{(m),N}(\mathrm{d}\theta) = \frac{1}{N} \sum_{i=1}^N \delta_{\theta^{(i,m)}}(\mathrm{d}\theta).$$

Overall, the algorithm retains M probability distributions. Note that these distributions are different for each $t < T$ as they depend on different potentials.

One iteration of the algorithm on a local worker m can be described as follows. Assume we collect a probability measure $\pi_{t-1}^{(m),N}$ from worker m, with the particle system $\{\theta_{t-1}^{(m,i)}\}_{i=1}^N$. First, we use a jittering kernel $\kappa(\mathrm{d}\theta | \theta_{t-1})$, which is a Markov kernel on Θ, to modify the particles [151] (see Subsection 4.4.1 for the precise definition of $\kappa(\cdot|\cdot)$). The idea is to *jitter* a subset of the particles in order to modify and propagate them into better regions of Θ with higher probability density and lower cost. The particles are jittered by sampling,

$$\hat{\theta}_t^{(i,m)} \sim \kappa(\cdot | \theta_{t-1}^{(i,m)}) \quad \text{for } i = 1, \dots, N.$$

Note that the jittering kernel may be designed so that it only modifies a subset of particles (again, see Section 4.4.1 for details). Next, we compute weights for the

new set of particles $\{\hat{\theta}_t^{(i,m)}\}_{i=1}^N$ according to the t-th potential, namely

$$w_t^{(i,m)} = \frac{G_t^{(m)}(\hat{\theta}_t^{(i,m)})}{\sum_{i=1}^N G_t^{(m)}(\hat{\theta}_t^{(i,m)})} \quad \text{for} \quad i = 1, \ldots, N.$$

After obtaining weights, each worker performs a resampling step where for $i = 1, \ldots, N$, we set $\theta_t^{(i,m)} = \hat{\theta}_t^{(i,k)}$ for $k \in \{1, \ldots, N\}$ with probability $w_t^{(i,m)}$. The procedure just described corresponds to a simple multinomial resampling scheme, but other standard methods can be applied as well [134]. We denote the resulting probability measure constructed at the t-th iteration of the m-th worker as

$$\pi_t^{(m),N}(\mathrm{d}\theta) = \frac{1}{N}\sum_{i=1}^N \delta_{\theta_t^{(i,m)}}(\mathrm{d}\theta).$$

The full procedure for the m-th worker is outlined in Algorithm 4.1. In Section 4.4.1, we elaborate on the selection of the jittering kernels to jitter particles and in Section 4.4.2, we detail the scheme for estimating a global minimum of $f(\theta)$ from the set of random measures $\{\pi_t^{(m),N}\}_{m=1}^M$.

4.4.1 Jittering kernel

The jittering kernel constitutes one of the key design choices of the proposed algorithm. Following [151], we put the following assumption on the kernel κ.

Assumption 4.2. *The Markov kernel κ satisfies*

$$\sup_{\theta' \in \Theta} \int_\Theta |\varphi(\theta) - \varphi(\theta')| \kappa(d\theta|\theta') \leq \frac{c_\kappa \|\varphi\|_\infty}{\sqrt{N}}$$

for any $\varphi \in B(\Theta)$ and some constant $c_\kappa < \infty$ independent of N.

In this paper, we use kernels of form

$$\kappa(\mathrm{d}\theta|\theta') = (1 - \epsilon_N)\delta_{\theta'}(\mathrm{d}\theta) + \epsilon_N \tau(\mathrm{d}\theta|\theta'), \tag{4.22}$$

where $\epsilon_N \leq \frac{1}{\sqrt{N}}$, which satisfy Assumption 4.2 [151]. The kernel τ can be rather simple, such as a multivariate Gaussian or multivariate-t distribution centered around $\theta' \in \Theta$. Other choices of τ are possible as well.

4.4.2 Estimating the global minima of $f(\theta)$

In order to estimate the global minima of $f(\theta)$, we first assess the performance of the samplers run by each worker. A typical performance measure is the *marginal likelihood estimate* resulting from $\pi_t^{(m),N}$. After choosing the worker which has

attained the highest marginal likelihood (say the m_0-th worker), we estimate a minimum of $f(\theta)$ by selecting the particle $\theta_t^{(i,m)}$ that yields the highest density $\pi_t^{(m_0)}(\theta_t^{(i,m_0)})$.

To be precise, let us start by denoting the *incremental* marginal likelihood associated to $\pi_t^{(m)}$ and its estimate $\pi_t^{(m),N}$ as $Z_{1:t}^{(m)}$ and $Z_{1:t}^{(m),N}$, respectively. They can be explicitly obtained by first computing

$$Z_t^{(m)} := \int G_t^{(m)}(\theta)\hat{\pi}_t^{(m)}(\mathrm{d}\theta) \approx \frac{1}{N}\sum_{i=1}^{N} G_t^{(m)}(\hat{\theta}_t^{(i,m)}) =: Z_t^{(m),N}$$

and then updating the running products

$$Z_{1:t}^{(m)} = Z_t^{(m)} Z_{1:t-1}^{(m)} = \prod_{k=1}^{t} Z_k^{(m)}$$

and

$$Z_{1:t}^{(m),N} = Z_t^{(m),N} Z_{1:t-1}^{(m),N} = \prod_{k=1}^{t} Z_k^{(m),N}.$$

The quantity $Z_{1:t}^{(m)}$ is a local performance index that keeps track of the "quality" of the m-th particle system $\{\theta_t^{(i,m)}\}_{i=1}^N$ [156]. This means that we can use $\{Z_{1:t}^{(m),N}\}_{m=1}^M$ to determine the best performing worker. Given the index of the best performing sampler, which is given by

$$m_t^\star = \underset{m\in\{1,\dots,M\}}{\operatorname{argmax}} Z_{1:t}^{(m),N},$$

we obtain a maximum-a-posteriori (MAP) estimator,

$$\theta_t^{\star,N} = \underset{i\in\{1,\dots,N\}}{\operatorname{argmax}} \mathsf{p}_t^{(m_t^\star),N}(\theta^{(i,m_t^\star)}), \tag{4.23}$$

where $\mathsf{p}_t^{(m_t^\star),N}(\theta)$ is the kernel density estimator [157, 158] described in Remark 4.7 below. Note that we do *not* construct the entire density estimator and maximize it. Since this operation is performed locally on the particles from the best performing sampler, it has $\mathcal{O}(N^2)$ cost, where N is the number of particles on a single worker, which is much smaller than the total number MN. The full procedure is outlined in Algorithm 4.2.

Remark 4.7. Let $\mathsf{k} : \Theta \to (0,\infty)$ be a bounded pdf with zero mean and finite second order moment, $\int_\Theta \|\theta\|_2^2 \mathsf{k}(\theta)\mathrm{d}\theta < \infty$. We can use the particle system $\{\theta_t^{(i,m)}\}_{i=1}^N$

and the pdf $\mathsf{k}(\cdot)$ to construct the kernel density estimator (KDE) of $\pi_t^{(m)}(\theta)$ as

$$\begin{aligned}\mathsf{p}_t^{(m),N}(\theta) &= \frac{1}{N}\sum_{i=1}^{N}\mathsf{k}(\theta-\theta_t^{(i,m)}) \\ &= (k^\theta, \pi_t^{(m),N}),\end{aligned} \tag{4.24}$$

where $\mathsf{k}^\theta(\theta') = \mathsf{k}(\theta-\theta')$. Note that $\mathsf{p}_t^{(m),N}(\theta)$ is not a standard KDE because the particles $\{\theta_t^{(i,m)}\}_{i=1}^N$ are not i.i.d. samples from $\pi_t^{(m)}(\theta)$. Eq. (4.24), however, suggests that the estimator, $\mathsf{p}_t^{(m),N}(\theta)$ converges when the approximate measure $\pi_t^{(m),N}$ does. See [159] for an analysis of particle KDE's. □

4.4.3 Analysis

In this section, we provide some basic theoretical guarantees for Algorithm 4.2. In particular, we prove results regarding a sampler on a single worker m, which hold for any $m \in \{1,\dots,M\}$. To ease the notation, we skip the superscript (m) in the rest of this section and simply note that results presented below hold for any $m \in \{1,\dots,M\}$. All proofs are deferred to Appendix A.3.

When constrained to a single worker m, the approximation π_t^N is provably convergent. In particular, we have the following result that holds for every worker $m = 1,\dots,M$.

Theorem 4.1. *If the sequence $(G_t)_{t\geq 1}$ satisfies Assumption 4.1 then, for any $\varphi \in B(\Theta)$, we have*

$$\left\|(\varphi,\pi_t) - \left(\varphi,\pi_t^N\right)\right\|_p \leq \frac{c_{t,p}\|\varphi\|_\infty}{\sqrt{N}}$$

for every $t = 1,\dots,T$ and for any $p \geq 1$, where $c_{t,p} > 0$ is a finite constant independent of N.

Proof. See Appendix A.3.□

Theorem 4.1 states that the samplers on local workers converge to their correct probability measures (for each m) with rate $\mathcal{O}(1/\sqrt{N})$, which is standard for Monte Carlo methods. This result is important since it enables us to analyze the properties of the KDEs constructed using the samples at each sampler. In order to be able to do so, we need to impose regularity conditions on the sequence of densities $\pi_t(\theta)$ and the kernels we use to approximate them.

Assumption 4.3. *For every $\theta \in \Theta$, the derivatives $\mathsf{D}^\alpha\pi_t(\theta)$ exist and they are Lipschitz continuous, i.e., there is a finite constant $L_{\alpha,t} > 0$ such that*

$$|\mathsf{D}^\alpha\pi_t(\theta) - \mathsf{D}^\alpha\pi_t(\theta')| \leq L_{\alpha,t}\|\theta-\theta'\|$$

for all $\theta, \theta' \in \Theta$, $t = 1, \ldots, T$ *and for all* $\alpha = (\alpha_1, \ldots, \alpha_d)$ *such that* $\alpha_i \in \{0, 1\}$.

Note that for $\alpha = (0, \ldots, 0)$, it is not hard to relate Assumption 4.3 directly to the cost function, as we do in the following proposition.

Proposition 4.6. *Assume that we define the incremental cost functions*

$$F_t(\theta) = \sum_{i \in \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_t} f_i(\theta)$$

and there exists some ℓ_t *such that*

$$|F_t(\theta) - F_t(\theta')| \leq \ell_t \|\theta - \theta'\|,$$

i.e., F_t *is Lipschitz. Assume there exists* $F_t^\star = \min_{\theta \in \Theta} F_t(\theta)$ *such that* $|F_t^\star| < \infty$ *and recall that* $\pi_t(\theta) \propto \exp(-F_t(\theta))$. *Then we have the following inequality,*

$$|\pi_t(\theta) - \pi_t(\theta')| \leq \frac{\ell_t \exp(-F_t^\star)}{Z_{\pi_t}} \|\theta - \theta'\|$$

where $Z_{\pi_t} = \int_\Theta \exp(-F_t(\theta)) \mathrm{d}\theta$.

Proof. See Appendix A.3. □

Next, we state some assumptions on the kernel k. We first note that the kernels in practice are defined with a bandwidth parameter $h \in \mathbb{R}_+$. In particular, given a kernel k, we can define scaled kernels k_h as

$$\mathsf{k}_h(\theta) = h^{-d} \mathsf{k}(h^{-1}\theta), \qquad h > 0.$$

Hence, given k we define a family of kernels $\{\mathsf{k}_h, h \in \mathbb{R}_+\}$.

Assumption 4.4. *The kernel* $\mathsf{k} : \Theta \to (0, \infty)$ *is a zero-mean bounded pdf, i.e.,* $\mathsf{k}(\theta) \geq 0 \ \forall \theta \in \Theta$ *and* $\int \mathsf{k}(\theta) \mathrm{d}\theta = 1$. *The second moment of this density is bounded, i.e.,* $\int_\Theta \|\theta\|^2 \mathsf{k}(\theta) \mathrm{d}\theta < \infty$. *Finally,* $\mathsf{D}^\alpha \mathsf{k} \in \mathsf{C}_b(\Theta)$, *i.e.,* $\|\mathsf{D}^\alpha \mathsf{k}\|_\infty < \infty$ *for any* $\alpha \in \{0, 1\}^d$.

Remark 4.8. We note that Assumption 4.4 implies that $\mathsf{D}^\alpha \mathsf{k}_h \in \mathsf{C}_b(\Theta)$ and we have $\|\mathsf{D}^\alpha \mathsf{k}_h\|_\infty = \frac{1}{h^{d+|\alpha|}} \|\mathsf{D}^\alpha \mathsf{k}\|_\infty$ for any $h > 0$ and $\alpha \in \{0, 1\}^d$. □

We denote *the kernel density estimator* defined using a scaled kernel k_h and the empirical measure π_t^N as $\mathsf{p}_t^{h,N}(\theta)$. In particular, given a normalized kernel (a pdf) $\mathsf{k} : \Theta \to (0, \infty)$, satisfying the assumptions in Assumption 4.4, we can construct the KDE

$$\mathsf{p}_t^{h,N}(\theta) = (\mathsf{k}_h^\theta, \pi_t^N).$$

where $\mathsf{k}_h^\theta(\theta') = \mathsf{k}_h(\theta - \theta')$ (see Remark 4.7). Now, we are ready to state our main results about the KDEs, adapted from [159].

Theorem 4.2. *Choose*

$$h = \left\lfloor N^{\frac{1}{2(d+1)}} \right\rfloor^{-1} \tag{4.25}$$

and denote $\mathsf{p}_t^N(\theta) = \mathsf{p}_t^{h,N}(\theta)$ *(since* $h = h(N)$*). If Assumptions 4.1, 4.3 and 4.4 hold then*

$$\sup_{\theta \in \Theta} |\mathsf{p}_t^N(\theta) - \pi_t(\theta)| \leq \frac{U^\varepsilon}{\left\lfloor N^{\frac{1}{2(d+1)}} \right\rfloor^{1-\varepsilon}} \tag{4.26}$$

where $U^\varepsilon \geq 0$ *is an almost surely finite random variable and* $0 < \varepsilon < 1$ *is a constant, both of which are independent of* N *and* θ*. As a consequence*

$$\lim_{N \to \infty} \sup_{\theta \in \Theta} |\mathsf{p}_t^N(\theta) - \pi_t(\theta)| = 0 \qquad \text{a.s.} \tag{4.27}$$

Proof. See the proof of Theorem 4.2 and Corollary 4.1 in [159]. Recall that $\Theta \subset \mathbb{R}^d$ is compact. □

This theorem is a uniform convergence result, i.e., it holds uniformly in a compact parameter space Θ. We note that Theorem 4.2 specifies a certain h, that is the bandwidth, in order for the results to hold. Based on this result, we can relate empirical maxima to the true maxima.

Theorem 4.3. *Let* $\theta_t^{\star,N} \in \operatorname{argmax}_{i \in \{1,\dots,N\}} \mathsf{p}_t^N(\theta_t^{(i)})$ *be an estimate of a global maximum of* π_t*. Then, under the assumptions of Theorem 4.2,*

$$\lim_{N \to \infty} \mathsf{p}_t^N(\theta_t^{\star,N}) = \pi_t(\theta_t^\star) \qquad \text{a.s.},$$

where $\theta_t^\star \in \operatorname{argmax}_{\theta \in \Theta} \pi_t(\theta)$*.*

Proof. See the proof of Theorem 5.2 in [159]. □

4.4.4 Experimental Results

In this section, we show numerical results for two optimization problems, which are hard to solve with conventional methods. In the first example, we focus on minimizing a function with multiple global minima. The aim of this experiment is to show that the algorithm populates multiple global minima successfully. In the second example, we minimize a challenging cost function for which standard stochastic gradient optimizers struggle.

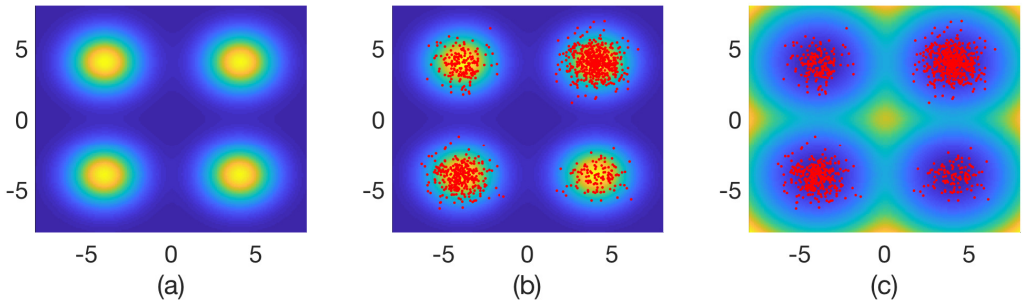

Figure 4.2: An illustration of the performance of the proposed algorithm for a cost function with four global minima. (a) The plot of $\pi_T(\theta) \propto \exp(-f(\theta))$. The blue regions indicate low values. It can be seen that there are four global maxima. (b) Samples drawn by the PSMCO at a single time instant. (c) The plot of the samples together with the actual cost function $f(\theta)$.

Minimization of a function with multiple global minima

In this experiment, we tackle the problem

$$\min_{\theta \in \mathbb{R}^2} f(\theta),$$

where

$$f(\theta) = \sum_{i=1}^{n} f_i(\theta) \qquad \text{and} \qquad f_i(\theta) = -\frac{1}{\lambda} \log \left(\sum_{k=1}^{4} \mathcal{N}(\theta; m_{i,k}, R) \right),$$

with $\lambda = 10$, $R = rI$ with $r = 0.2$. We choose the means $m_{i,k}$ randomly, namely $m_{i,k} \sim \mathcal{N}(m_{i,k}; m_k, \sigma^2)$ where,

$$m_1 = [4, 4]^\top, \quad m_2 = [-4, -4]^\top, \quad m_3 = [-4, 4]^\top, \quad m_4 = [4, -4]^\top,$$

and $\sigma^2 = 0.5$. This selection results in a cost function with four global minima. This cost function is a good model for cost functions arising in many machine learning problems where there are multiple global minima, see, e.g., [160]. In this experiment, we have chosen $n = 1,000$. Although a small number for stochastic optimization problems, we note that each $f_i(\theta)$ models a mini-batch in this scenario and we choose $K = 1$ in our algorithm.

In order to run the algorithm, we choose a uniform prior measure $\pi_0(\theta) = \mathcal{U}([-a, a] \times [-a, a])$ with $a = 50$. It follows from Proposition 4.1 that the pdf that matches the cost function $f(\theta)$ can be written as

$$\pi_T(\theta) \propto \exp(-f(\theta)),$$

and it has four global maxima. This pdf is displayed in Fig. 4.2(a). We run $M = 100$ samplers, each with each $N = 50$ particles, yielding a total number of particles $MN = 5,000$. We choose a Gaussian jittering scheme; specifically, the jittering kernel is defined as

$$\kappa(\mathrm{d}\theta|\theta') = (1 - \epsilon_N)\delta_{\theta'}(\mathrm{d}\theta) + \epsilon_N \mathcal{N}(\theta; \theta', \sigma_j^2)\mathrm{d}\theta, \tag{4.28}$$

where $\epsilon_N \leq 1/\sqrt{N}$ and $\sigma_j^2 = 0.5$.

Some illustrative results can be seen from Fig. 4.2. To be specific, we have run independent samplers and plot all samples for this experiment (instead of estimating a minimum with the best performing sampler). From Fig. 4.2(b), one can see that the algorithm populates all maxima with samples. Finally, Fig. 4.2(c) shows the location of the samples relative to the actual cost function $f(\theta)$. This plots illustrate how the algorithm populates multiple, distinct global maxima with independent samplers. This implies that different independent samplers can report different global maxima in practice. Note that this is in agreement with the analysis provided in Section 4.4.3.

Minimization of the sigmoid function

In this experiment, we address the problem

$$\min_{\theta \in \mathbb{R}^2} f(\theta) := \sum_{i=1}^{n} (y_i - g_i(\theta))^2, \quad \text{where} \quad g_i(\theta) = \frac{1}{1 + \exp(-\theta_1 - \theta_2 x_i)}, \tag{4.29}$$

with $x_i \in \mathbb{R}$, $f_i(\theta) = (y_i - g_i(\theta))^2$ and $\theta = [\theta_1, \theta_2]^\top$. The function g_i is known as the sigmoid function. Cost functions of the form in eq. (4.29) are widely used in nonlinear regression with neural networks in machine learning [125].

In this experiment, we have $n = 100,000$. We choose $M = 25$ and $MN = 1,000$, leading to $N = 40$ particles for every sampler. The mini-batch size is $K = 100$. The jittering kernel κ is defined in the same way as in (4.28), where the Gaussian pdf has a variance chosen as the ratio of the dataset size n to the mini-batch size K, i.e., $\sigma_j^2 = n/K$, which yields a rather large variance[3] $\sigma_j^2 = 1000$. To compute the maximum as described in Eq. (4.23), we use a Gaussian kernel with bandwidth $h = \lfloor N^{\frac{1}{6}} \rfloor^{-1}$, since $d = 2$, which yields $h = 1$.

The results can be seen from Fig. 4.3. We compare our method with a parallel stochastic gradient descent (PSGD) scheme [161] using M optimizers. We note

[3]Note that this is for efficient exploration of the global minima, which is hard to find for this example. A large jittering variance may not be adequate in practice when there are multiple minima close to each other, see, e.g., Section 4.4.4.

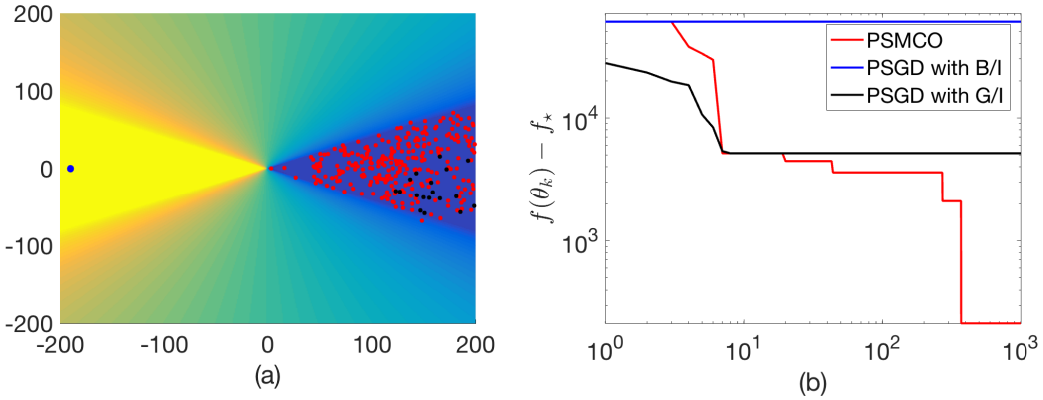

Figure 4.3: (a) The cost function and a snapshot of samples from the 50th iteration of PSMCO, PSGD with bad initialization (blue point) and PSGD with good initialization (black points). (b) The minimization performance of each algorithm. It can be seen that PSMCO first converges to the wide region with low values (blue triangle region) and then jumps to the minimum. This is because the marginal likelihood estimate of the sampler close to minimum dominates after a while. So there is effectively full communication only to determine the minimizer although there is no exchange of information.

that given a particular realization of $(x_i)_{i=1}^n$ (which is an iid sequence) with $x_k \sim \mathcal{U}([-2.5, 2.5]))$, the cost function landscape can be hard to optimize. One can see from Fig. 4.3(a) that the cost function (for a particular realization of $(x_i)_{i=1}^n$) has broad flat regions which make it difficult to minimize even for gradient based methods unless their initialization is sufficiently good. Accordingly, we have run two instances of PSGD with "bad" and "good" initializations.

The bad initial point for PSGD can be seen from Fig 4.3(a), at $[-190, 0]^\top$ (the blue dot). We initialize M parallel SGD optimizers around $[-190, 0]^\top$, each with a small zero-mean Gaussian perturbation with variance 10^{-8}. This is a poor initialization because gradients are nearly zero in this region (yellow triangle in Fig. 4.3(a)). We refer to the PSGD algorithm starting from this point as PSGD with B/I, which refers to bad initialization. We also initialize the PSMCO from this region, with Gaussian perturbations around $[-190, 0]^\top$, with the same small variance of $\sigma^2_{\text{init}} = 10^{-8}$.

The "good" initialization for the PSGD is selected from a better region, namely around the point $[0, -100]^\top$, where gradient values actually contain useful information about the minimum. We refer to the PSGD algorithm starting from this point as PSGD with G/I.

The results and some comments can be seen from Fig. 4.3(b). It can be seen

that the PSGD with good initialization (G/I) moves towards a better region, however, it gets stuck because gradients become zero. On the other hand, PSGD with B/I is unable to move at all, since it is initialized in a region where all gradients are zero (which is true even for the mini-batch observations). PSMCO, on the other hand, is able to search the space effectively to find the global minimum, which is clearly reflected in Fig. 4.3(b).

Algorithm 4.2 PSMCO

1: Sample $\theta_0^{(i,m)} \sim \pi_0$ for $i = 1, \ldots, N$.
2: **for** $t \geq 1$ **do**
3: **for** $m = 1, \ldots, M$ **do**
4: Jitter by generating samples

$$\hat{\theta}_t^{(i,m)} \sim \kappa(\mathrm{d}\theta|\theta_{t-1}^{(i,m)}) \qquad \text{for} \quad i = 1, \ldots, N.$$

5: Update the marginal likelihood,

$$Z_{1:t}^{(m),N} = Z_{1:t-1}^{(m),N} \times Z_t^{(m),N} \quad \text{where} \quad Z_t^{(m),N} = \frac{1}{N} \sum_{i=1}^{N} G_t^{(m)}(\hat{\theta}_t^{(i,m)}).$$

6: Compute weights,

$$w_t^{(i,m)} = \frac{G_t^{(m)}(\hat{\theta}_t^{(i,m)})}{\sum_{i=1}^{N} G_t^{(m)}(\hat{\theta}_t^{(i,m)})} \quad \text{for} \quad i = 1, \ldots, N.$$

7: Resample N i.i.d. samples,

$$\theta_t^{(i,m)} \sim \hat{\pi}_t^{(m),N}(\mathrm{d}\theta) = \sum_{i=1}^{N} w_t^{(i,m)} \delta_{\hat{\theta}_t^{(i,m)}}(\mathrm{d}\theta) \quad \text{for} \quad i = 1, \ldots, N.$$

8: **end for**
9: **if** an estimate of the solution of problem (1.1) is needed at time t **then**
10: Choose

$$m_t^\star = \underset{m \in \{1,\ldots,M\}}{\operatorname{argmax}} \; Z_{1:t}^{(m),N}$$

11: Estimate

$$\theta_t^{\star,N} = \underset{i \in \{1,\ldots,N\}}{\operatorname{argmax}} \; \mathsf{p}_t^{(m_t^\star),N}(\theta_t^{(i,m_t^\star)}).$$

12: **end if**
13: **end for**

5
Dictionary filtering

In this chapter, by using the duality between optimization and inference we have developed in Chapter 4, we investigate a link between matrix factorisation algorithms and recursive linear filters. We describe a probabilistic model in which sequential inference naturally leads to a matrix factorisation procedure. We refer to the resulting algorithm as the dictionary filter.

5.1 Introduction

Matrix factorization (MF) algorithms are a cornerstone of modern signal processing, machine learning, and, more generally, computational linear algebra. Formally, we are interested in solving the problem of factorizing a data matrix $Y \in \mathbb{R}^{m \times n}$ as

$$Y \approx CX \tag{5.1}$$

where $C \in \mathbb{R}^{m \times r}$ is the *dictionary matrix*, the columns of $X \in \mathbb{R}^{r \times n}$ are *coefficients*, and r is the *approximation rank*. In this chapter, we assume all matrices are real-valued. We are interested in computing both C and X recursively, or *online*, using a single (column) data vector at each time to update the factors.

Matrix factorization methods became popular with the work on nonnegative matrix factorization (NMF) of [162]. The authors considered the factorization of a nonnegative data matrix $Y \in \mathbb{R}_+^{m \times n}$ into nonnegative factors $C \in \mathbb{R}_+^{m \times r}$ and $X \in$

$\mathbb{R}_+^{r\times n}$ using a multiplicative gradient descent method (see [163] for a convergence proof). The algorithm has received significant attention due to its ability to learn important and interpretable features in an unsupervised way. Following [162], similar algorithms were also proposed for real-valued matrices and factors, as in our formulation (5.1), especially when the primary interest is the prediction of entries but not necessarily obtaining interpretable features. Optimization-based approaches became popular in that avenue, i.e., taking a cost function of the form $d(Y, CX)$ and minimizing it with respect to C and X using optimization algorithms such as projected gradient descent for NMF [164] or alternating least-squares for real MF [165]. There has been a seemingly inexhaustible research activity in this area and a full literature review is out of scope for this thesis.

Similar to the optimization-based ideas, probabilistic approaches to MF have received considerable attention. Compared to the optimization-based methods which try to obtain point estimates of the factors, probabilistic methods aim at capturing the posterior distribution over the factors, hence quantifying the uncertainty. The authors of [166] introduced a Gaussian model for real-valued MF to estimate movie ratings where factors were assumed to have independent and identically distributed (iid) entries. Similar ideas have been proposed for nonnegative factorizations [167]. Also a probabilistic interpretation of batch NMF was introduced in [168] deriving multiplicative update rules as a variational inference scheme.

All these methods were batch techniques, meaning that they require the whole dataset to update each factor at each iteration. With the rise of big datasets, these ideas became infeasible to apply. On the optimization side, the research focus increasingly shifted to stochastic optimization algorithms which enabled implementations of MF for large datasets. In one of the early works, NMF has been extended to the incremental setting [169]. A canonical stochastic gradient descent (SGD) based approach for general MF can be found in [170] which can be applied entry-wise or column-wise to solve the problem in (5.1). Similarly, one can apply the same idea to any type of differentiable cost, such as regularized versions of the cost function proposed in [170], and obtain a MF method. The idea is extended in several ways, see e.g. [171, 172]. One of the fundamental limitations of these algorithms are their step-size tuning problems, a problem which has received much attention recently, see e.g. [65, 68, 173]. Every different dataset requires a different step-size and decay rate of the step-size, usually set after conducting empirical tests.

Compared to stochastic optimization based works, online versions of probabilistic MF have received less attention. The work in [174] followed the probabilistic

interpretation of NMF given in [168] to propose a sequential Monte Carlo based NMF algorithm. However, this algorithm was only applied to low dimensional problems and its applicability to realistic settings remains unclear. In [175], the same batch NMF model of [168] was implemented with stochastic variational inference techniques in the online setting.

On the other hand, sequential inference for matrix-variate linear dynamic models independently received some interest for different applications, see, e.g. [176, 177]. A different perspective, closer to our approach in this chapter, was introduced in [178], where matrix-variate update rules for Hessian matrices were derived as analytic inference rules in probabilistic models. As a result, the authors of [178] obtained quasi-Newton algorithms from a probabilistic perspective. However, [178] focuses on symmetric matrices whereas in this thesis, our focus is more general, non-square matrices.

In this chapter, we highlight a connection between online matrix factorization and sequential probabilistic inference. In doing so, we propose a matrix-variate dynamic linear probabilistic model in which sequential approximate inference leads to an *online* matrix factorization algorithm. More specifically, we derive a *matrix-variate recursive filtering* method that can be readily interpreted as an online MF technique. This probabilistic characterization brings several advantages. First, since the proposed method is based on an explicit probabilistic model, it enables the user to naturally incorporate further prior knowledge on factors (by extending the model we put forward) or dealing with non-stationary data in a principled way by putting dynamics on the dictionary matrix (see Section 5.3.3). Therefore, the proposed framework makes it easier to develop application-specific models and inference procedures. Secondly, from a practical perspective, compared to other *online* methods, the inference method we propose removes the need of step-size tuning and involves only easy-to-tune parameters. Specifically, the proposed algorithm does not require any step-size parameter. Its role is played by an $r \times r$ covariance matrix that is updated automatically at each iteration. We note that this is different (and computationally much cheaper) than a second-order Hessian-based approach [51], where the Hessian matrices there would need to be of the same dimension as the vectorized dictionary matrix, which is impractical in this case. Finally, as opposed to the simulation-based probabilistic MFs such as [174, 179], which only obtain samples from the posterior of the dictionary, the proposed method obtains an analytical form of the posterior distribution in terms of a Gaussian, which enables the user to quantify the uncertainty over the dictionary or diagnose convergence.

The rest of the chapter is organised as follows. We introduce the probabilistic

model for the factorization problem in Section 5.2. Two online algorithms are derived in Section 5.3. In Section 5.4, we compare our algorithm with some popular optimization based methods. Some illustrative experimental results on image restoration and video modeling are presented in Section 5.5.

5.2 Probabilistic model

5.2.1 Model

Recall that $Y \in \mathbb{R}^{m\times n}$ denotes the data matrix, $C \in \mathbb{R}^{m\times r}$ is the dictionary matrix, with approximation rank r, and $X \in \mathbb{R}^{r\times n}$ is the coefficient matrix. The i-th column of the data matrix is denoted $Y(:,i)$ and we use $\{1,\dots,n\}$ to denote sets of consecutive indices.

Let us consider a random mechanism for the collection of data vectors. In particular, assume that, at time k, we sample an index i_k from the uniform probability distribution over the index set $\{1,\dots,n\}$, and use it to select the data vector $y_k = Y(:,i_k)$ (specifically note that y_k denotes the observation at time k, *not* the k-th column of Y). Similarly, the i_k-th column of X is denoted $x_k = X(:,i_k)$. We use $c = \mathrm{vec}(C)$ to denote the vector form of the dictionary, while $c_k = \mathrm{vec}(C_k)$ denotes its estimate at time k. We assume a probabilistic model linking these factors, namely,

$$p(c) = \mathcal{N}(c; c_0, V_0 \otimes I_m), \tag{5.2}$$

$$p(y_k|c, x_k) = \mathcal{N}(y_k; Cx_k, \lambda \otimes I_m), \tag{5.3}$$

where $p(c)$ is a prior pdf on the dictionary vector c, V_0 is an $r\times r$ a priori covariance matrix (identical for each column of C), c_0 is an $mr \times 1$ mean vector, $\lambda > 0$ is a scale parameter and $p(y_k|c,x_k)$ is the conditional pdf (assumed Gaussian) of the data vector y_k given the dictionary C and the coefficient vector x_k. The covariance matrix V_0 encodes the prior knowledge of correlations between columns of C and λ models how informative the observations are (similar to a regularisation parameter in optimization based approaches). In this model, x_k is a static unknown parameter, while c and y_k are random vectors.

Intuitively, model (5.2)–(5.3) implies that $y_k \approx Cx_k$ (and, hence, $Y \approx CX$), where λ controls the magnitude of the approximation error. Notice that, using the identity (1.3) for Cx_k, we can rewrite (5.3) as

$$p(y_k|c, x_k) = \mathcal{N}(y_k; (x_k^\top \otimes I_m)c, \lambda \otimes I_m) \tag{5.4}$$

and we express the model in terms of the vector form of the dictionary. Treating c as a latent vector with observation matrix $x_k^\top \otimes I_m$ enables us to use standard

linear filtering recursions in vector form. We will show, however, that working with the matrix form of the dictionary is also possible and leads to a significant reduction in computational complexity.

5.3 Algorithm

We assume the coefficient vectors x_k are deterministic parameters for which we aim at computing point-estimates, whereas the dictionary C is a latent random matrix and we aim at computing its posterior probability distribution (given the data in Y). The two problems are addressed in this section.

5.3.1 Parameter estimation

Let us assume that C_{k-1} is an estimate of the dictionary matrix computed at time $k-1$ (by a procedure to be specified later). We propose to compute a maximum likelihood estimator of the coefficient vector x_k, given the dictionary C_{k-1} and the data y_k, namely,

$$x_k^* = \underset{x_k}{\operatorname{argmax}}\, p(y_k|c_{k-1}, x_k), \tag{5.5}$$

where $c_{k-1} = \mathrm{vec}(C_{k-1})$. Since the density in (5.5) is Gaussian, with mean $C_{k-1}x_k$, the estimator can be easily computed and yields

$$x_k^* = (C_{k-1}^\top C_{k-1})^{-1} C_{k-1}^\top y_k. \tag{5.6}$$

We use this update rule for the coefficients in the experiments of Section 5.5.

5.3.2 Inference of the dictionary matrix

Let us assume that $x_k = x_k^*$ is fixed via the rule in Eq. (5.6) and drop it from the notation for simplicity. Model (5.2)–(5.4) can then be rewritten as

$$p(c) = \mathcal{N}(c; c_0, P_0), \tag{5.7}$$

$$p(y_k|c) = \mathcal{N}(y_k; H_k c, R), \tag{5.8}$$

where $x_k = x_k^*$ is implicit, $P_0 = V_0 \otimes I_m$ and $R = \lambda \otimes I_m$. The observation matrix for this model takes the form $H_k = x_k^{*,\top} \otimes I_m$ and hence it is assumed known. Given (5.7)–(5.8), the posterior distribution of c given the data sequence $y_{1:k}$ is Gaussian and can be computed exactly [5]. To be specific, the posterior pdf is Gaussian, $p(c|y_{1:k}) = \mathcal{N}(c; c_k, P_k)$, with mean c_k and covariance matrix P_k, and

can be computed exactly using a Kalman filter, which can be described by the recursive equations [5]

$$c_k = c_{k-1} + P_{k-1}H_k^\top (H_k P_{k-1} H_k^\top + R_k)^{-1}(y_k - H_k c_{k-1}), \tag{5.9}$$
$$P_k = P_{k-1} - P_{k-1}H_k^\top (H_k P_{k-1} H_k^\top + R_k)^{-1} H_k P_{k-1}. \tag{5.10}$$

The algorithm is initialised with c_0 and P_0 in (5.7).

Implementing the update rules (5.9)–(5.10) is computationally inefficient, however, when $c \in \mathbb{R}^{mr}$ is large dimensional. They require the storage of a potentially very large matrices (H_k is $m \times rm$ and P_k is $rm \times rm$) which can easily make the algorithm impractical. To circumvent this limitation, we propose an equivalent, yet computationally more efficient, pair of equations for the update of the dictionary matrix $C_k = \mathrm{vec}^{-1}_{m\times r}(c_k)$ and the covariance matrix V_k (with dimensions $r \times r$ and initialised with the matrix V_0 in (5.2)). The following two propositions state the form of the update rules.

Proposition 5.1. *The posterior covariance matrix P_k in (5.10) can be written as $P_k = V_k \otimes I_m$, for $k \geq 0$, where*

$$V_k = \left(V_{k-1} - \frac{V_{k-1} x_k x_k^\top V_{k-1}}{x_k^\top V_{k-1} x_k + \lambda} \right), \qquad \text{for } k \geq 1. \tag{5.11}$$

and V_0 is given by the prior pdf in (5.2).

Proof. See Appendix A.4. □

Proposition 5.2. *The posterior mean c_k in (5.9) can be rewritten, in matrix form, as*

$$C_k = C_{k-1} + \frac{(y_k - C_{k-1}x_k)x_k^\top V_{k-1}^\top}{x_k^\top V_{k-1} x_k + \lambda}, \tag{5.12}$$

where $C_k = \mathrm{vec}^{-1}_{m\times r}(c_k)$ is the posterior expectation of the dictionary matrix C and the sequence $\{V_k; k \geq 0\}$ is computed as in Proposition 5.1.

Proof. See Appendix A.4. □

The complete procedure is outlined in Algorithm 5.1. We hereafter refer to this method as the dictionary filter (DF). Note that this procedure has iteration complexity $\mathcal{O}(mr^2 + r^3)$. In high-dimensional scenarios where $m \gg r$, we have $\mathcal{O}(mr^2)$.

Algorithm 5.1 Dictionary Filter

1: Select C_0 randomly, choose an initial covariance matrix $V_0 > 0$, and set $k = 1$.
2: **repeat**
3: Pick $y_k = Y(:, i_k)$ where i_k is drawn from the uniform distribution over the index set $\{1, \ldots, n\}$.
4: Update the coefficient vector, the dictionary matrix and the covariance matrix as

$$x_k = (C_{k-1}^\top C_{k-1})^{-1} C_{k-1}^\top y_k$$
$$C_k = C_{k-1} + \frac{(y_k - C_{k-1}x_k)x_k^\top V_{k-1}}{\lambda + x_k^\top V_{k-1} x_k}$$
$$V_k = V_{k-1} - \frac{V_{k-1}x_k x_k^\top V_{k-1}}{x_k^\top V_{k-1} x_k + \lambda},$$

respectively.
5: $k \leftarrow k + 1$
6: **until** convergence

5.3.3 Dynamic dictionary filter

When the dataset is a (possibly nonstationary) time series, such as in video modeling problems, the prior (5.2) on the matrix C can be misleading since it assumes that a single dictionary for all data points can be sufficient. In these cases, one can allow C to be time-varying. Hence, we obtain a state-space model

$$p(\tilde{c}_0) = \mathcal{N}(\tilde{c}_0; c_0, V_0 \otimes I_m), \tag{5.13}$$
$$p(\tilde{c}_k|\tilde{c}_{k-1}) = \mathcal{N}(\tilde{c}_k; \tilde{c}_{k-1}, Q \otimes I_m), \tag{5.14}$$
$$p(y_k|\tilde{c}_k, x_k) = \mathcal{N}(y_k; \tilde{C}_k x_k, \lambda \otimes I_m), \tag{5.15}$$

where Q is a $r \times r$ covariance matrix. A simple modification of Algorithm 5.1 is sufficient to conduct inference in this model. Given the mean-covariance estimate (C_{k-1}, V_{k-1}) at time $k-1$, one needs to compute the predictive covariance matrix,

$$\tilde{V}_k = V_{k-1} + Q,$$

and use $\tilde{V}_k$ instead of V_{k-1} in all substeps within the step 4 of the Algorithm 5.1. We refer to this modified version as the *dynamic dictionary filter*. Intuitively, the dynamic version allows the algorithm to adapt the dictionary when the contents evolve quickly over time, such as in a sequence of video frames. We demonstrate how this property of the dynamic dictionary filter is useful in highly nonstationary problems.

5.4 Links with stochastic optimization

Our algorithm relates to the MF algorithms that use stochastic optimization based approaches. In the literature, MF problems are often formulated as [170]

$$\min_{C,X} \|Y - CX\|_F^2 := \sum_{k=1}^{n} \|y_k - Cx_k\|_2^2.$$

Let us assume that x_k is set as in Algorithm 5.1 given each y_k. Then, the SGD implementation for estimating C becomes the update rule

$$C_k = C_{k-1} + \gamma_k (y_k - C_{k-1}x_k)x_k^\top, \tag{5.16}$$

where the positive step-size γ_k must be tuned to achieve the best convergence rate. In particular, it must satisfy, $\sum_{k=1}^{\infty} \gamma_k = \infty$ and $\sum_{k=1}^{\infty} \gamma_k^2 < \infty$ in order to guarantee convergence [60]. In practice, it is usually chosen as $\gamma_k = \alpha/k^\beta$, where $\alpha > 0$ and $0.5 < \beta < 1$. The SGDMF method has iteration complexity $\mathcal{O}(mr + r^3)$ which reduces to $\mathcal{O}(mr)$ when $m \gg r$.

It can be seen that the update rule (5.16) is related to the filtering update rule (5.12). In our algorithm, in contrast to SGD, we do not have a step-size parameter γ_k, instead we have the gain matrix $V_{k-1}/(\lambda + x_k^\top V_{k-1} x_k))$ which is updated at each iteration with the posterior column-covariance V_k. Note that, our approach is not identical to the second-order SGD or the full filtering recursions, which are equivalent under some assumptions [180]. Those approaches would be infeasible for our problem because the full covariance or Hessian matrices are too big to store and update compared to V_k, which captures only the posterior column covariance.

It is also possible to relate our algorithm to another class of stochastic optimization methods, called incremental proximal methods [84]. A proximal approach to the same cost function would give the update rule (see [181] for an explicit derivation)

$$C_k = C_{k-1} + \frac{(y_k - C_{k-1}x_k)x_k^\top}{\lambda + x_k^\top x_k}$$

which is a special case of our algorithm (specifically if one sets $V_{k-1} = I_r$ at each iteration). This would be obviously unjustified from the probabilistic perspective. We refer to Chapter 4 for the relationship between filters and proximal algorithms, see also [91].

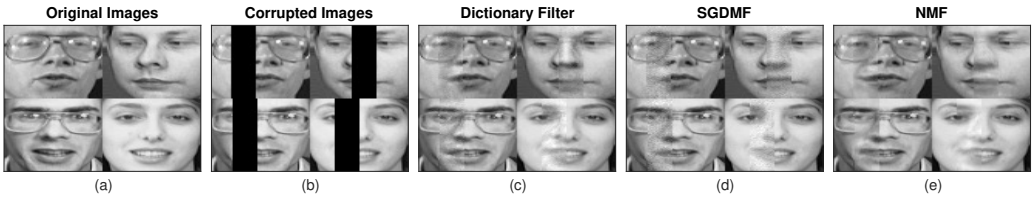

Figure 5.1: Comparison of the proposed algorithm (DF) with stochastic gradient descent matrix factorization (SGDMF), and nonnegative matrix factorization (NMF). The NMF method was iterated 1,000 times. (a) Four original images out of the 400-member dataset. (b) Corrupted images. (c) Output of the DF algorithm. (d) Output of the SGDMF algorithm. (e) Output of the NMF algorithm. The RMSE values attained by the algorithms are 10.26 (DF), 10.13 (NMF) and 10.79 (SGDMF), starting from an initial RMSE value 70.54.

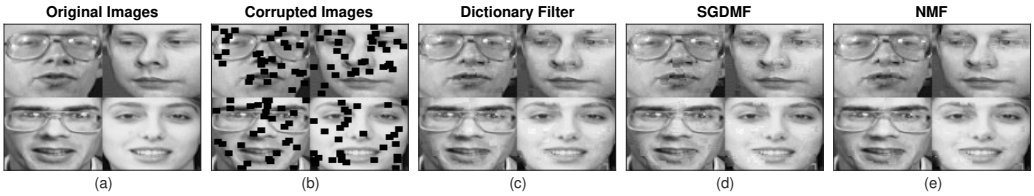

Figure 5.2: Results with non-uniform corruptions. The initial RMSE is given by 58.6536. The algorithms attain RMSEs as (i) Dictionary Filter: 6.8684, (ii) SGDMF: 7.9709, (iii) NMF RMSE: 6.9336. It can be seen that in this scenario the DF outperforms other methods in terms of the RMSE performance.

5.5 Experiments

5.5.1 Image restoration

We tackle an image restoration problem on the Olivetti dataset [168]. This dataset consists of 400 face images of size 64×64. We vectorize each face into a column vector with dimension $m = 4096$. Since there are 400 faces, $n = 400$. We chose an approximation rank $r = 40$ and $\lambda = 2$. The factors C_0 and X_0 are initialized randomly, without imposing any structure. We choose $V_0 = I_m$ for this particular dataset (other choices do not seem to lead to better performance). It is up to the user to encode any prior knowledge about the dictionary using the covariance matrix V_0.

We deal with missing data in the images by putting masks into the model and extending the update rules for the missing data case. First, the inference step can be extended easily. We define a mask M for the whole dataset Y and denote the mask associated with y_k as m_k, more precisely $m_k = M(:, i_k)$ as $y_k = Y(:, i_k)$. Note that M is known, i.e., we know the positions of missing entries. In Algorithm 5.1,

for the update of C_k (inference step), we simply replace the term $(y_k - C_{k-1}x_k)$ by $m_k \odot (y_k - C_{k-1}x_k)$ where $\odot$ denotes the Hadamard (elementwise) product. This corresponds to assuming having observations of the form $m_k \odot y_k$ with mean $m_k \odot (Cx_k)$. Then, in accordance, we also have to compute the maximum likelihood parameter estimator x_k^* with missing data. This corresponds to solving the least squares problem

$$\min_{x_k} \|m_k \odot (y_k - C_{k-1}x_k)\|_2^2 \tag{5.17}$$

For this purpose we construct a special mask, $M_k = \underbrace{[m_k, \ldots, m_k]}_{r \text{ times}}$. The rationale behind this mask can be made explicit by observing that

$$m_k \odot (y_k - C_{k-1}x_k) = m_k \odot y_k - (M_k \odot C_{k-1})x_k.$$

Hence solving (5.17) is equivalent to solving the following least squares problem

$$\min_{x_k} \|m_k \odot y_k - (M_k \odot C_{k-1})x_k\|_2^2$$

which, in turn, reduces to the solution of a problem of the form $b \approx Ax$ with $b = m_k \odot y_k$ and $A = M_k \odot C_{k-1}$. Hence the solution can be found by applying the update rule (pseudoinverse operation),

$$x_k^* = ((M_k \odot C_{k-1})^\top (M_k \odot C_{k-1}))^{-1} (M_k \odot C_{k-1})^\top (m_k \odot y_k),$$

in Algorithm 5.1 for x_k.

We compare the performance of the DF, SGDMF [170] and NMF [162] algorithms. NMF can be considered as a standard benchmark for image restoration, yet we recall that it is a batch (non-recursive or offline) method. SGD is an online procedure, the same as DF. The implementation of SGDMF is similar to ours – the C_k SGD update followed by the same update rule (5.6).

The results of applying the three techniques can be seen in Fig. 5.1 and Fig. 5.2, along with quantitative results (root-mean squared error (RMSE) values) in the captions. In the former, in order to construct images with missing data, we randomly removed a patch consisting of %25 of all columns (for all 400 faces)[1]. In the latter, we randomly remove patches of the images in order to construct images with missing data. The SGDMF and DF algorithms were passed 10 times over the dataset recursively (i.e., with $k = 1, \ldots, 10n$). The sample images show that the proposed algorithm works well perceptually, and achieves an RMSE very

[1]The relative performance of the three methods remains similar when we change the percentage of missing data.

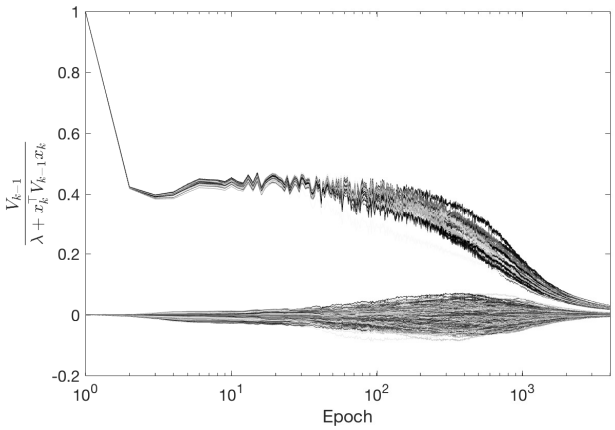

Figure 5.3: Values of diagonal and upper triangular entries of $V_{k-1}/(\lambda + x_k^\top V_{k-1} x_k)$. As our update rule can be decomposed as the "step-size" and the gradient, we can see these values as step-sizes. From this perspective, it can be seen that our algorithm automatically tunes the step-sizes.

close to the NMF algorithm (and better than SGDMF) with a significantly lesser computational cost.

The behavior of the entries of the matrix $V_{k-1}/(\lambda + x_k^\top V_{k-1} x_k)$ can be seen from Fig. 5.3. As we initialized $V_0 = I_r$, diagonal values can be seen in the upper cluster in the plot which are decreasing from one to zero. Non-diagonal entries are initialized as zero and took nonzero values before again converging to zero. The figure hints that, as the entries go to zero, the algorithm converges empirically. This provides a natural and automated step-size tuning procedure in the form of posterior covariance matrix, which frees the user from the notorious task of step-size tuning. Note that second-order optimization-based approaches, or the full filtering approach, would require to store an $mr \times mr$ matrix where $mr = 163,840$ in this case, which practically renders these approaches infeasible since a matrix of dimension $mr \times mr$ is not possible to fit into the memory. In contrast, our algorithm only requires to store $r \times r$ matrix where $r = 40$, which is a lightweight computation, hence it reduces the computational burden significantly.

5.5.2 Video modeling

In this experiment, we test our algorithm on a video modeling task. The aim is to track a sequence of low dimensional subspaces of the video frames and reconstruct them successfully. The video is taken from Youtube 8M dataset [182] and it has $T = 450$ frames with size 360×640 after preprocessing. We vectorized the

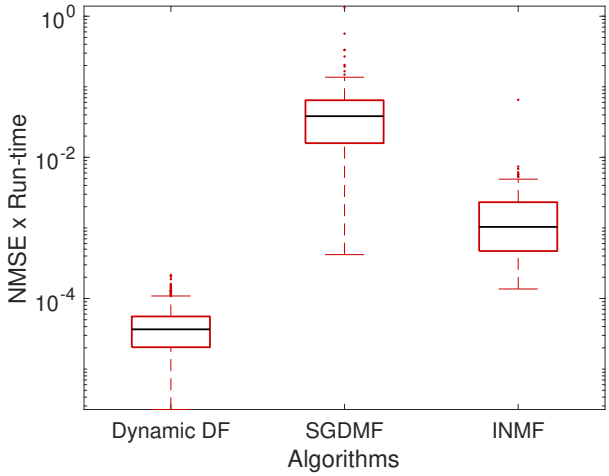

Figure 5.4: The boxplot of NMSEs×run-time for 450 frames of the video for each frame. It can be seen that dynamic DF attains lower error levels compared to SGDMF and INMF. The dynamic DF takes 26.7191 seconds to run while SGDMF and INMF take 23.8624 and 20.7956 seconds respectively. While our algorithm is slightly slower, it can be seen that it is the most favorable when combined with the NMSEs.

video and obtained a dataset consisting of a 230400×450 matrix. The task is to sequentially process the video frames and obtain a sequence of dictionaries and coefficients that result in low reconstruction errors for all frames of the video. In this experiment, we have used the dynamic dictionary filter (dynamic DF) algorithm as explained in Sec. 5.3.3 with $Q = qI_r$ where $q = 0.05$ and $\lambda = 2$. For comparison, we have implemented SGDMF [170] and the incremental nonnegative matrix factorization (INMF) [169]. We have chosen the rank $r = 10$ for all algorithms. For SGDMF, we have chosen a constant step-size 0.1 and for INMF, we have chosen $\alpha = 0.9$. The results can be seen from Fig. 5.4 in terms of running times and normalized mean squared errors (NMSE). From the results, it can be concluded that dynamic DF arises as a favorable dictionary learning scheme.

6

Conclusions and future work

6.1 Conclusions

The contribution of this thesis is twofold. First, we have proposed novel methods for stochastic filtering of high-dimensional and/or misspecified state-space models. Second, we have developed probabilistic methods for stochastic optimization in order to tackle inherent difficulties in mainstream stochastic optimization methods.

In particular, in Chapter 3, we have proposed a simple modification of the particle filter which, according to our computer experiments, can improve the performance of the algorithm (e.g., when tracking high-dimensional systems) or enhance its robustness to model mismatches in the state equation of a SSM. The modification of the standard particle filtering scheme consists of an additional step, which we term nudging, in which a subset of particles are pushed towards regions of the state space with a higher likelihood. In this way, the state space can be explored more efficiently while keeping the computational effort at nearly the same level as in a standard particle filter. We refer to the new algorithm as the "nudged particle filter" (NuPF). While, for clarity and simplicity, we have kept the discussion and the numerical comparisons restricted to the modification (nudging) of the conventional BPF, the new step can be naturally incorporated to most known particle filtering methods.

We have presented a basic analysis of the NuPF which indicates that the al-

gorithm converges (in L_p) with the same error rate as the standard particle filter. In addition, we have also presented an alternative interpretation of the NuPF as defining a data-dependent implicit model, which might shed some light onto why the NuPF tends to outperform the BPF in computer simulations when there is some mismatch in the state equation of the SSM.

The analytical results have been corroborated by a number of computer experiments, both with synthetic and real data. In the latter case, we have tackled the fitting of a stochastic volatility SSM using Bayesian methods for model inference and a time-series dataset consisting of euro to US dollar exchange rates over a period of two years. We have shown how different figures of merit (model evidence, acceptance probabilities or autocorrelation functions) improve when using the NuPF, instead of a standard BPF, in order to implement a nested particle filter [141] and a particle Metropolis-Hastings [140] algorithm.

Although we have shown through numerical examples how the NuPF improves the performance of conventional particle filters (e.g., with misspecified or high-dimensional models), we do not claim that it can alone solve the degeneracy problem [131, 35]. While pushing particles into high likelihood regions makes it more probable that nudged particles have even weights, degeneracy should still be expected for certain models.

Since the nudging step is fairly general, it can be used with a wide range of differentiable or nondifferentiable likelihoods. Besides, the new operation does not require any modification of the well-defined steps of the particle filter so it can be plugged into a variety of common particle filtering methods. Therefore, it can be adopted by a practitioner with hardly any additional effort. In particular, gradient nudging steps (for differentiable log-likelihoods) can be implemented using automatic differentiation tools, currently available in many software packages, hence relieving the user from explicitly calculating the gradient of the likelihood.

Similar to the resampling step, which is routinely employed for numerical stability, we believe the nudging step can be systematically used for improving the performance and robustness of particle filters.

Next, in Chapter 4, we have developed a probabilistic perspective for proximal and incremental proximal methods. We have shown that a probabilistic setting can provide a systematic way to derive algorithms when this is not possible from the classical perspective. In particular, within an online setup, we have argued that the use of filtering algorithms corresponds to employing an IPM-type scheme for optimization. However, filtering algorithms have natural dampening mechanisms for parameter updates, as they refine their uncertainty over the solution iteratively. This line of work can be pushed forward in a number of different di-

rections. First, different Kalman filters can be used in a similar way to get more advanced optimization schemes for more complicated problems. Among the candidates, the unscented Kalman filter (UKF) [12–14] and the Ensemble Kalman filter (EnKF) [15–19] can be useful to tackle high dimensional, possibly time-varying, optimization problems. Second, this approach can be extended beyond quadratic functions by exploiting the relationship between exponential families and Bregman divergences [183]. When the likelihood belongs to the exponential family, the cost function can be expressed in general as a Bregman divergence. In that case, since the Gaussianity assumption is violated, one needs to resort to more complicated numerical algorithms, such as particle filters [25] or other advanced filtering methods.

As our second main contribution in Chapter 4, we have proposed a parallel sequential Monte Carlo optimizer to minimize challenging cost functions, e.g., with multiple global minima or with wide flat regions. The algorithm uses jittering kernels to propagate samples [151] and particle kernel density estimators to find the minima [159], within a stochastic optimization setup. We have shown that, on a single (local) node, the algorithm is provably convergent. On the global level, we argue that the parallel setting where each sampler uses a different configuration of the same dataset can be useful to improve the practical convergence of the algorithms. The numerical performance of our algorithm in difficult scenarios shows that this is a promising direction.

Finally in Chapter 5, by exploiting the relationship between optimization and inference, we have recast the matrix factorization problem $Y \approx CX$ as a linear filtering problem and proposed efficient matrix-variate update rules for the posterior mean and covariance of the dictionary matrix C. As a result, we have obtained an online algorithm for matrix factorization that is flexible and computationally efficient. In particular, we have noted that the proposed method has $\mathcal{O}(mr^2)$ complexity which is independent of the number of data points. We have empirically demonstrated that the overhead of the DF compared to the SGDMF, which has $\mathcal{O}(mr)$ complexity, is small. We have also shown that the algorithm is competitive with the state-of-the-art methods in an image restoration example as well as on a video processing example which demonstrates that the proposed algorithm can successfully learn nonstationary and dynamic signals. Unlike the stochastic gradient based approaches, our algorithm does not need any parameter tuning.

6.2 Future work

The stochastic filtering and optimization methods that we have devised within this thesis can be extended in various ways.

In particular, the general nudging scheme we have proposed in Chapter 3 can be used improve the performance of other SMC methods, such as auxiliary particle filters [147]. Moreover, it can also be used within Kalman-type filters, such as in an EnKF [17], or even in a regular Kalman filter (when the model is misspecified) as we have demonstrated in Chapter 3. The method can also be used to improve parameter estimation methods which rely on SMC-type algorithms, as we have shown in Chapter 3 for the pMH and the NPF schemes. However, the method can be plugged in to any other generic parameter estimation method which uses SMC as well. From a practical perspective, the NuPF can be improved and be made parameter-free. Particularly, when the gradient step is used as a nudging step, one can incorporate line-search schemes in order to obtain a parameter-free nudging scheme. On the theoretical side, the analysis of the NuPF in misspecified case remains an open problem that we plan to tackle. In particular, the empirical evidence points to the fact that the NuPF leads to higher marginal likelihoods compared to the BPF, however this is a conjecture that remains to be proven. Quantifying this improvement compared to the classical case can give important insight to develop novel methods for tackling misspecified models in the real world.

The methods we have proposed in Chapter 4 can be extended in various ways. The probabilistic interpretation we have developed for the stochastic optimization methods is general. In other words, one can use the whole Bayesian machinery to implement the general Bayes recursion (4.2). Moreover, one can also extend the interpretation to tackle regularized cost functions. To be specific, the algorithms we have provided tackle the minimization of the cost functions of form $f := \sum_i f_i$ where f_i are component functions. One can also extend this probabilistic approach to minimize cost functions of the form $f + g$ where g is a regularizer. In particular, the probabilistic interpretation of the IPM algorithm can be extended to tackle this case. This approach will lead to the probabilistic extension of the incremental proximal gradient methods. In this direction, we have done some initial work, see, e.g., [184].

In order to test the PSMCO method, we have focused on challenging but low dimensional cost functions. We leave the potential applications of the PSMCO scheme to high-dimensional optimization problems as a future work. Also the design of an interacting extension of our method similar to particle islands [185] can be potentially useful in more challenging settings. Note that as we have provided

a general recursion which leads to a probabilistic interpretation of the stochastic optimization problem, one can use different approximate Bayesian techniques other than SMC-type methods to solve the stochastic optimization problem. In particular, variational Bayesian inference [125], optimal transport [186, 187], or Stein variational gradient descent (SVGD) [188] based methods can be utilized in order to obtain a probabilistic stochastic optimizer.

Finally, the dictionary filter we have presented in Chapter 5 can also be developed further in different ways. In particular, extending the model we have proposed for the MF problems can lead to online and efficient algorithms to factorize structured data, such as text, video, audio, or a spatio-temporal time series. We believe that extensions and explorations of these dynamic settings could lead to useful algorithms to deal with high-dimensional time-series data.

A
Proofs

A.1 Proofs of Chapter 2

A.1.1 Lemmata for Chapter 2

In order to prove theorems in this section, we need a preliminary lemma, which can be found, e.g., in [31].

Lemma A.1. *Let $\alpha, \beta, \bar{\alpha}, \bar{\beta} \in \mathcal{P}(\mathsf{X})$ be probability measures and $f, h \in B(\mathsf{X})$ be two real bounded functions on X such that $(h, \bar{\alpha}) > 0$ and $(h, \bar{\beta}) > 0$. If the identities,*

$$(f, \alpha) = \frac{(fh, \bar{\alpha})}{(h, \bar{\alpha})} \quad \text{and} \quad (f, \beta) = \frac{(fh, \bar{\beta})}{(h, \bar{\beta})}$$

hold, then we have,

$$|(f, \alpha) - (f, \beta)| \leq \frac{1}{(h, \bar{\alpha})} \left|(fh, \bar{\alpha}) - (fh, \bar{\beta})\right| + \frac{\|f\|_\infty}{(h, \bar{\alpha})} \left|(h, \bar{\alpha}) - (h, \bar{\beta})\right|. \tag{A.1}$$

Proof. We write,

$$\begin{aligned}
|(f, \alpha) - (f, \beta)| &= \left| \frac{(fh, \bar{\alpha})}{(h, \bar{\alpha})} - \frac{(fh, \bar{\beta})}{(h, \bar{\beta})} \right| \\
&\leq \frac{1}{(h, \bar{\alpha})} \left|(fh, \bar{\alpha}) - (fh, \bar{\beta})\right| + |(fh, \bar{\beta})| \left| \frac{1}{(f, \bar{\alpha})} - \frac{1}{(f, \bar{\beta})} \right| \\
&\leq \frac{1}{(h, \bar{\alpha})} \left|(fh, \bar{\alpha}) - (fh, \bar{\beta})\right| + \|f\|_\infty \cancel{|(h, \bar{\beta})|} \left| \frac{(h, \bar{\alpha}) - (h, \bar{\beta})}{(h, \bar{\alpha}) \cancel{(h, \bar{\beta})}} \right|,
\end{aligned}$$

which leads to (A.1). □

A.1.2 Proofs of Chapter 2

Proof of Theorem 2.1

We first provide the proof for $p = 2$ for simplicity. We rewrite the L_2 norm using its definition as,

$$\left\|(\varphi,\pi)-(\varphi,\pi^N)\right\|_2 = \left\|(\varphi,\pi)-\frac{1}{N}\sum_{k=1}^{N}\varphi\left(x^{(k)}\right)\right\|_2 = \mathbb{E}\left[\left|(\varphi,\pi)-\frac{1}{N}\sum_{k=1}^{N}\varphi\left(x^{(k)}\right)\right|^2\right]^{1/2}.$$

Writing explicitly, we have,

$$\mathbb{E}\left[\left|(\varphi,\pi)-\frac{1}{N}\sum_{k=1}^{N}\varphi\left(x^{(k)}\right)\right|^2\right] = \frac{1}{N^2}\mathbb{E}\left[\left|\sum_{i=1}^{N}\left(\varphi(x^{(i)})-(\varphi,\pi)\right)\right|^2\right].$$

We define $S^{(i)} = \varphi(x^{(i)})-(\varphi,\pi)$ and note that $\mathbb{E}[S^{(i)}] = 0$ and $S^{(i)}$ are independent random variables. We therefore have,

$$\begin{aligned}\mathbb{E}\left[\left|(\varphi,\pi)-\frac{1}{N}\sum_{k=1}^{N}\varphi\left(x^{(k)}\right)\right|^2\right] &= \frac{1}{N^2}\mathbb{E}\left[\left|\sum_{i=1}^{N}S^{(i)}\right|^2\right],\\ &= \frac{1}{N^2}\sum_{i=1}^{N}\mathbb{E}\left[\left|S^{(i)}\right|^2\right],\\ &\leq \frac{N4\|\varphi\|_\infty^2}{N^2},\end{aligned}$$

since $\left|S^{(i)}\right| = \left|\varphi(x^{(i)})-(\varphi,\pi)\right| \leq 2\|\varphi\|_\infty$. Therefore, we have,

$$\left\|(\varphi,\pi)-(\varphi,\pi^N)\right\|_2 \leq \frac{2\|\varphi\|_\infty}{\sqrt{N}},$$

where $c_2 = 2$ for the simplest case.

We now proceed to general integer $p \geq 1$. Assume we have $x^{(i)}$, i.i.d. from π as before. We first write,

$$\|(\varphi,\pi^N)-(\varphi,\pi)\|_p = \mathbb{E}\left[\left|\frac{1}{N}\sum_{i=1}^{N}\left(\varphi(x^{(i)})-(\varphi,\pi)\right)\right|^p\right]^{1/p}.$$

We define $S^{(i)} = \varphi(x^{(i)})-(\varphi,\pi)$ and note that $S^{(i)}$, $i = 1,\ldots,N$ are zero-mean and independent random variables. Using the Marcinkiewicz-Zygmund inequality

[129], we arrive at,

$$\mathbb{E}\left[\left|\frac{1}{N}\sum_{i=1}^{N} S^{(i)}\right|^p\right] \leq \frac{B_{0,p}}{N^p}\mathbb{E}\left[\left(\sum_{i=1}^{N}\left|S^{(i)}\right|^2\right)^{\frac{p}{2}}\right] \leq \frac{B_{0,p}}{N^p}\left(N4\|\varphi\|_\infty^2\right)^{\frac{p}{2}},$$

where B_p is a constant independent of N and the last inequality follows from $\left|S^{(i)}\right| = \left|\varphi(x^{(i)}) - (\varphi,\pi)\right| \leq 2\|\varphi\|_\infty$. Therefore, we have proved that

$$\|(\varphi,\pi^N) - (\varphi,\pi)\|_p \leq \frac{c_p\|\varphi\|_\infty}{\sqrt{N}},$$

where $c_p = 2B_p^{1/p}$ is a constant independent of N. □

Proof of Corollary 2.1

Take $p \geq 4$. We recall that, by Theorem 2.1, for any $p \geq 1$, we have,

$$\|(\varphi,\pi^N) - (\varphi,\pi)\|_p \leq \frac{c_p\|\varphi\|_\infty}{\sqrt{N}}.$$

We start by defining a nonnegative random variable $U_{p,\epsilon}$,

$$U_{p,\epsilon} = \lim_{k\to\infty}\sum_{N=1}^{k} N^{\frac{p}{2}-1-\epsilon}\left|(\varphi,\pi) - (\varphi,\pi^N\right|^p.$$

Using Fatou's lemma and Theorem 2.1 together, we obtain,

$$\begin{aligned}\mathbb{E}[U_{p,\epsilon}] &\leq \sum_{N=1}^{\infty} N^{\frac{p}{2}-1-\epsilon}\mathbb{E}\left[\left|(\varphi,\pi) - (\varphi,\pi^N\right|^p\right] \\ &\leq \sum_{N=1}^{\infty} N^{\frac{p}{2}-1-\epsilon}\frac{c^p\|\varphi\|_\infty^p}{N^{\frac{p}{2}}} \\ &= c^p\|\varphi\|_\infty^p\sum_{N=1}^{\infty} N^{-1-\epsilon} < \infty.\end{aligned}$$

Since a random variable with finite expectation implies that the r.v. is almost surely finite, we conclude that $U_{p,\epsilon}$ is a.s. finite. Next, we write,

$$N^{\frac{p}{2}-1-\epsilon}\left|(\varphi,\pi) - (\varphi,\pi^N)\right|^p \leq U_{p,\epsilon},$$

which yields,

$$\left|(\varphi,\pi) - (\varphi,\pi^N\right| \leq \frac{U_\delta}{N^{\frac{1}{2}-\delta}},$$

where $\delta = \frac{1+\epsilon}{p}$ and $U_\delta = (U_{p,\epsilon})^{\frac{1}{p}}$. Since $p \geq 4$ and $0 < \epsilon < 1$, we conclude $0 < \delta < \frac{1}{2}$. The almost sure convergence follows when we take $N \to \infty$. □

Proof of Theorem 2.2

We recall the weighted measure,

$$\tilde{\pi}^N(\mathrm{d}x) = \sum_{i=1}^{N} w^{(i)} \delta_{x^{(i)}}(\mathrm{d}x) \quad \text{where} \quad w^{(i)} = \frac{W^{(i)}}{\sum_{i=1}^{N} W^{(i)}}.$$

and the following relations,

$$(\varphi, \pi) = \frac{(\varphi W, q)}{(W, q)}, \quad \text{and} \quad (\varphi, \tilde{\pi}^N) = \frac{(\varphi W, q^N)}{(W, q^N)}, \tag{A.2}$$

for the true integral (φ, π) and its SNIS estimate $(\varphi, \tilde{\pi}^N)$. Using Lemma A.1 together with (A.2), we obtain,

$$\left|(\varphi, \pi) - (\varphi, \tilde{\pi}^N)\right| \leq \frac{1}{(W, q)} \left(\|\varphi\|_\infty \left|(W, q) - (W, q^N)\right| + \left|(\varphi W, q) - (\varphi W, q^N)\right| \right). \tag{A.3}$$

where $(W, q) > 0$ by assumption. Using Minskowski's inequality, we can obtain from (A.3) that,

$$\left\|(\varphi, \pi) - (\varphi, \tilde{\pi}^N)\right\|_p \leq \frac{1}{(W, q)} \left(\|\varphi\|_\infty \left\|(W, q) - (W, q^N)\right\|_p + \left\|(\varphi W, q) - (\varphi W, q^N)\right\|_p \right). \tag{A.4}$$

Now Theorem 2.1 and (A.4) together yield,

$$\left\|(\varphi, \pi) - (\varphi, \tilde{\pi}^N)\right\|_p \leq \frac{1}{(W, q)} \left(\|\varphi\|_\infty \frac{B_p \|W\|_\infty}{\sqrt{N}} + \frac{B_p \|W\varphi\|_\infty}{\sqrt{N}} \right),$$

Noting that we have $\|\varphi W\|_\infty \leq \|\varphi\|_\infty \|W\|_\infty$, we obtain,

$$\left\|(\varphi, \pi) - (\varphi, \tilde{\pi}^N)\right\|_p \leq \frac{c_p \|\varphi\|_\infty}{\sqrt{N}},$$

where,

$$c_p = \frac{2\|W\|_\infty B_p}{(W, q)} = 2\|w\|_\infty B_p,$$

is a constant independent of N. □

Proof of Theorem 2.3

We follow the same kind of induction argument as in, e.g., [30] and [145].

For the base case, i.e. $t = 0$, we draw $x_0^{(i)}$, $i = 1, ..., N$, i.i.d. from π_0 and obtain,

$$\|(\varphi, \pi_0^N) - (\varphi, \pi_0)\|_p = \mathbb{E}\left[\left|\frac{1}{N}\sum_{i=1}^{N}\left(\varphi(x_0^{(i)}) - (\varphi, \pi_0)\right)\right|^p\right]^{1/p}.$$

We define $S_0^{(i)} = \varphi(x_0^{(i)}) - (\varphi, \pi_0)$ and note that $S_0^{(i)}$, $i = 1, \ldots, N$ are zero-mean and independent random variables. Using the Marcinkiewicz-Zygmund inequality [129], we arrive at,

$$\begin{aligned}\mathbb{E}\left[\left|\frac{1}{N}\sum_{i=1}^{N} S_0^{(i)}\right|^p\right] &\leq \frac{B_{0,p}}{N^p}\mathbb{E}\left[\left(\sum_{i=1}^{N}\left|S_0^{(i)}\right|^2\right)^{\frac{p}{2}}\right] \\ &\leq \frac{B_{0,p}}{N^p}\left(N4\|\varphi\|_\infty^2\right)^{\frac{p}{2}},\end{aligned}$$

where $B_{0,p}$ is a constant independent of N and the last inequality follows from $\left|S_0^{(i)}\right| = \left|\varphi(x_0^{(i)}) - (\varphi, \pi_0)\right| \leq 2\|\varphi\|_\infty$. Therefore, we have proved that Eq. (2.42) holds for the base case,

$$\|(\varphi, \pi_0^N) - (\varphi, \pi_0)\|_p \leq \frac{c_{0,p}\|\varphi\|_\infty}{\sqrt{N}},$$

where $c_{0,p} = 2B_{0,p}^{1/p}$ is a constant independent of N.

The induction hypothesis is that, at time $t-1$,

$$\left\|(\varphi, \pi_{t-1}^N) - (\varphi, \pi_{t-1})\right\|_p \leq \frac{c_{t-1,p}\|\varphi\|_\infty}{\sqrt{N}}$$

for some constant $c_{t-1,p} < \infty$ independent of N.

We start analyzing the predictive measure ξ_t^N,

$$\xi_t^N(\mathrm{d}x) = \frac{1}{N}\sum_{i=1}^{N}\delta_{\bar{x}_t^{(i)}}(\mathrm{d}x),$$

where $\bar{x}_t^{(i)}$, $i = 1, \ldots, N$ are the particles sampled from the transition kernels $\tau_t^{x_{t-1}^{(i)}}(\mathrm{d}x_t) \triangleq \tau_t(\mathrm{d}x_t|x_{t-1}^{(i)})$. Since we have $\xi_t = \tau_t\pi_{t-1}$ (see Sec. 1.4), a simple triangle inequality yields,

$$\begin{aligned}\left\|(\varphi, \xi_t^N) - (\varphi, \xi_t)\right\|_p &= \left\|(\varphi, \xi_t^N) - (\varphi, \tau_t\pi_{t-1})\right\|_p \\ &\leq \left\|(\varphi, \xi_t^N) - (\varphi, \tau_t\pi_{t-1}^N)\right\|_p \\ &+ \left\|(\varphi, \tau_t\pi_{t-1}^N) - (\varphi, \tau_t\pi_{t-1})\right\|_p,\end{aligned} \tag{A.5}$$

where,

$$(\varphi, \tau_t \pi_{t-1}^N) = \frac{1}{N} \sum_{i=1}^{N} (\varphi, \tau_t^{x_{t-1}^{(i)}}). \tag{A.6}$$

For the sampling step, we aim at bounding the two terms on the rhs of (A.5).

For the first term, we introduce the σ-algebra generated by the random variables $x_{0:t}^{(i)}$ and $\bar{x}_{1:t}^{(i)}$, $i = 1, \ldots, N$, denoted $\mathcal{F}_t = \sigma(x_{0:t}^{(i)}, \bar{x}_{1:t}^{(i)}, i = 1, \ldots, N)$. Since π_{t-1}^N is measurable w.r.t. $\mathcal{F}_{t-1}$, we can write

$$\mathbb{E}[(\varphi, \xi_t^N) | \mathcal{F}_{t-1}] = \frac{1}{N} \sum_{i=1}^{N} (\varphi, \tau_t^{x_{t-1}^{(i)}}) = (\varphi, \tau_t \pi_{t-1}^N).$$

Next, we define the random variables $S_t^{(i)} = \varphi(\bar{x}_t^{(i)}) - (\varphi, \tau_t \pi_{t-1}^N)$ and note that, conditional on $\mathcal{F}_{t-1}$, $S_t^{(i)}$, $i = 1, \ldots, N$ are zero-mean and independent. Then, the approximation error of ξ_t^N can be written as,

$$\mathbb{E}[\left|(\varphi, \xi_t^N\ ) - (\varphi, \tau_t \pi_{t-1}^N)\right|^p | \mathcal{F}_{t-1}] = \mathbb{E}\left[\left| \frac{1}{N} \sum_{i=1}^{N} S_t^{(i)} \right|^p \middle| \mathcal{F}_{t-1} \right].$$

Resorting again to the Marcinkiewicz-Zygmund inequality, we can write,

$$\mathbb{E}\left[\left| \frac{1}{N} \sum_{i=1}^{N} S_t^{(i)} \right|^p \middle| \mathcal{F}_{t-1} \right] \leq \frac{B_{t,p}}{N^p} \mathbb{E}\left[\left(\sum_{i=1}^{N} \left| S_t^{(i)} \right|^2 \right)^{\frac{p}{2}} \middle| \mathcal{F}_{t-1} \right],$$

where $B_{t,p} < \infty$ is a constant independent of N. Moreover, since $\left|S_t^{(i)}\right| = \left|\varphi(\bar{x}_t^{(i)}) - (\varphi, \tau_t \pi_{t-1}^N)\right| \leq 2\|\varphi\|_\infty$, we have,

$$\mathbb{E}\left[\left| \frac{1}{N} \sum_{i=1}^{N} S_t^{(i)} \right|^p \middle| \mathcal{F}_{t-1} \right] \leq \frac{B_{t,p}}{N^p} \left(N 4 \|\varphi\|_\infty^2 \right)^{\frac{p}{2}} = \frac{B_{t,p}}{N^{p/2}} 2^p \|\varphi\|_\infty^p.$$

If we take unconditional expectations on both sides of the equation above, then we arrive at

$$\|(\varphi, \xi_t^N) - (\varphi, \tau_t \pi_{t-1}^N)\|_p \leq \frac{c_{1,p} \|\varphi\|_\infty}{\sqrt{N}}, \tag{A.7}$$

where $c_{1,p} = 2B_{t,p}^{1/p} < \infty$ is a constant independent of N.

To handle the second term in the rhs of (A.5), we define $(\bar{\varphi}, \pi_{t-1}) = (\varphi, \tau_t \pi_{t-1})$ where $\bar{\varphi} \in B(\mathsf{X})$ and given by,

$$\bar{\varphi}(x) = (\varphi, \tau_t^x).$$

We also write $(\bar{\varphi}, \pi_{t-1}^N) = (\varphi, \tau_t \pi_{t-1}^N)$. Since $\|\bar{\varphi}\|_\infty \leq \|\varphi\|_\infty$, the induction hypothesis leads,

$$\begin{aligned}\|(\varphi, \tau_t \pi_{t-1}^N) - (\varphi, \tau_t \pi_{t-1})\|_p &= \|(\bar{\varphi}, \pi_{t-1}^N) - (\bar{\varphi}, \pi_{t-1})\|_p \\ &\leq \frac{c_{t-1,p}\|\varphi\|_\infty}{\sqrt{N}},\end{aligned} \tag{A.8}$$

where $c_{t-1,p}$ is a constant independent of N. Combining (A.5) and (A.8) yields,

$$\left\|(\varphi, \xi_t^N) - (\varphi, \xi_t)\right\|_p \leq \frac{c_{1,t,p}\|\varphi\|_\infty}{\sqrt{N}} \tag{A.9}$$

where $c_{1,t,p} = c_{t-1,p} + c_{1,p} < \infty$ is a constant independent of N.

Next, we aim at bounding $\|(\varphi, \pi_t) - (\varphi, \tilde{\pi}_t^N)\|_p$ using (A.9). We note that, after the computation of weights, we define the weighted random measure,

$$\tilde{\pi}_t^N = \sum_{i=1}^N w_t^{(i)} \delta_{\bar{x}_t^{(i)}} \quad \text{where} \quad w_t^{(i)} = \frac{g_t(\bar{x}_t^{(i)})}{\sum_{i=1}^N g_t(\bar{x}_t^{(i)})}.$$

The integrals computed with respect to the weighted measure $\tilde{\pi}_t^N$ takes the form,

$$(\varphi, \tilde{\pi}_t^N) = \frac{(\varphi g_t, \xi^N)}{(g_t, \xi_t^N)}. \tag{A.10}$$

On the other hand, using Bayes theorem, integrals with respect to the optimal filter can also be written in a similar form as,

$$(\varphi, \pi_t) = \frac{(\varphi g_t, \xi_t)}{(g_t, \xi_t)}. \tag{A.11}$$

Using Lemma A.1 together with (A.10) and (A.11), we can readily obtain,

$$\begin{aligned}\left|(\varphi, \tilde{\pi}_t^N) - (\varphi, \pi_t)\right| \leq &\frac{1}{(g_t, \xi_t)} \left(\|\varphi\|_\infty \left|(g_t, \xi_t) - (g_t, \xi_t^N)\right|\right. \\ &\left. + \left|(\varphi g_t, \xi_t) - (\varphi g_t, \xi_t^N)\right|\right),\end{aligned} \tag{A.12}$$

where $(g_t, \xi_t) > 0$ by assumption. Using Minkowski's inequality, we can deduce from (A.12) that

$$\begin{aligned}\left\|(\varphi, \tilde{\pi}_t^N) - (\varphi, \pi_t)\right\|_p \leq &\frac{1}{(g_t, \xi_t)} \left(\|\varphi\|_\infty \left\|(g_t, \xi_t) - (g_t, \xi_t^N)\right\|_p\right. \\ &\left. + \left\|(\varphi g_t, \xi_t) - (\varphi g_t, \xi_t^N)\right\|_p\right).\end{aligned} \tag{A.13}$$

Noting that we have $\|\varphi g_t\|_\infty \leq \|\varphi\|_\infty \|g_t\|_\infty$, (A.9) and (A.13) together yield,

$$\left\|(\varphi, \pi_t) - (\varphi, \tilde{\pi}_t^N)\right\|_p \leq \frac{c_{2,t,p}\|\varphi\|_\infty}{\sqrt{N}}, \tag{A.14}$$

where

$$c_{2,t,p} = \frac{2\|g_t\|_\infty c_{1,t,p}}{(g_t, \xi_t)} < \infty$$

is a finite constant independent of N (the denominator is positive and the numerator is finite as a consequence of Assumption 2.1).

Finally, the analysis of the multinomial resampling step is also standard. We denote the resampled measure as π_t^N. Since the random variables which are used to construct π_t^N are sampled i.i.d from $\tilde{\pi}_t^N$, the argument for the base case can also be applied here to yield,

$$\left\|(\varphi, \tilde{\pi}_t^N) - (\varphi, \pi_t^N)\right\|_p \leq \frac{c_{3,t,p}\|\varphi\|_\infty}{\sqrt{N}}, \tag{A.15}$$

where $c_{3,t,p} < \infty$ is a constant independent of N. Combining bounds (A.14) and (A.15) to obtain the inequality (2.42), with $c_{t,p} = c_{2,t,p} + c_{3,t,p} < \infty$, concludes the proof. □

A.2 Proofs of Chapter 3

Proof of Theorem 3.1

We follow the same kind of induction argument as in the proof of Theorem 2.3. For $t = 0$, Eq. (3.12) is satisfied trivially as we draw $x_0^{(i)}$, $i = 1, ..., N$, i.i.d. from π_0 as we have proved in Theorem 2.1. Then the induction hypothesis is that, at time $t-1$,

$$\left\|(\varphi, \pi_{t-1}^N) - (\varphi, \pi_{t-1})\right\|_p \leq \frac{c_{t-1}\|\varphi\|_\infty}{\sqrt{N}}$$

for some constant $c_{t-1} < \infty$ independent of N.

The analysis of the approximate predictive measure ξ_t^N is standard and it can be easily shown that (see the proof of Theorem 2.3)

$$\left\|(\varphi, \xi_t^N) - (\varphi, \xi_t)\right\|_p \leq \frac{c_{1,t}\|\varphi\|_\infty}{\sqrt{N}} \tag{A.16}$$

where $c_{1,t} < \infty$ is a constant independent of N. After the nudging step we obtain the random measure $\tilde{\xi}_t^N$. The sets of samples $\{\bar{x}_t^{(i)}\}_{i=1}^N$, used to construct ξ_t^N, and $\{\tilde{x}_t^{(i)}\}_{i=1}^N$, used to construct $\tilde{\xi}_t^N$ as shown in (3.9), differ exactly in M particles, namely $\tilde{x}_t^{(j_1)}, \ldots, \tilde{x}_t^{(j_M)}$, where $\{j_1, \ldots, j_M\} = \mathcal{I}_t$. Therefore, we readily obtain the

relationship

$$\begin{aligned}\left\|(\varphi,\xi_t^N)-(\varphi,\tilde{\xi}_t^N)\right\|_p &= \left\|\frac{1}{N}\sum_{i\in\mathcal{I}_t}\left(\varphi(\bar{x}_t^{(i)})-\varphi(\tilde{x}_t^{(i)})\right)\right\|_p \\ &\leq \frac{2\|\varphi\|_\infty M}{N} \\ &\leq \frac{2\|\varphi\|_\infty}{\sqrt{N}} \end{aligned} \tag{A.17}$$

where the first inequality holds trivially (since $|\varphi(x)-\varphi(x')|\leq 2\|\varphi\|_\infty$ for every $(x,x')\in\mathsf{X}^2$) and the second inequality follows from the assumption $M\leq\sqrt{N}$. Combining (A.16) and (A.17) we arrive at

$$\left\|(\varphi,\xi_t)-(\varphi,\tilde{\xi}_t^N)\right\|_p \leq \frac{\tilde{c}_{1,t}\|\varphi\|_\infty}{\sqrt{N}}, \tag{A.18}$$

where the constant $\tilde{c}_{1,t}=2+c_{1,t}<\infty$ is independent of N.

The rest of the proof is standard. Following the same steps as in the proof of Theorem 2.3, we can prove that

$$\left\|(\varphi,\pi_t)-(\varphi,\tilde{\pi}_t^N)\right\|_p \leq \frac{c_{2,t}\|\varphi\|_\infty}{\sqrt{N}}, \tag{A.19}$$

where

$$c_{2,t}=\frac{2\|g_t\|_\infty\tilde{c}_{1,t}}{(g_t,\xi_t)}<\infty$$

is finite a constant independent of N (the denominator is positive and the numerator is finite as a consequence of Assumption 3.1). The analysis of the multinomial resampling step is also standard (see, again, the proof of Theorem 2.3) and it yields

$$\left\|(\varphi,\tilde{\pi}_t^N)-(\varphi,\pi_t^N)\right\|_p \leq \frac{c_{3,t}\|\varphi\|_\infty}{\sqrt{N}}, \tag{A.20}$$

where $c_{3,t}<\infty$ is a constant independent of N. We can combine bounds (A.19) and (A.20) to obtain the inequality (3.12), with $c_t=c_{2,t}+c_{3,t}<\infty$, and conclude the proof. □

Proof of Lemma 3.1

Since $\tilde{x}_t^{(i)}=\bar{x}_t^{(i)}+\gamma\nabla_{x_t}g_t(\bar{x}_t^{(i)})$, we readily obtain the relationships

$$\begin{aligned}\left|\varphi(\tilde{x}_t^{(i)})-\varphi(\bar{x}_t^{(i)})\right| &\leq L\left\|\tilde{x}_t^{(i)}-\bar{x}_t^{(i)}\right\|_2 \\ &= L\gamma\left\|\nabla_{x_t}g(x_t^{(i)})\right\|_2 \\ &\leq \gamma L G_t \end{aligned} \tag{A.21}$$

where the first inequality follows from the Lipschitz assumption, the identity is due to the implementation of the gradient-nudging step and the second inequality follows from Assumption 3.2. Then we bound the error $\|(\varphi, \xi_t^N) - (\varphi, \tilde{\xi}_t^N)\|_p$ as

$$\begin{aligned}\left\|(\varphi, \xi_t^N) - (\varphi, \tilde{\xi}_t^N)\right\|_p &= \left\|\frac{1}{N}\sum_{i\in\mathcal{I}_t}\left(\varphi(\bar{x}_t^{(i)}) - \varphi(\tilde{x}_t^{(i)})\right)\right\|_p \\ &\leq \frac{1}{N}\sum_{i\in\mathcal{I}}\left\|\varphi(\bar{x}_t^{(i)}) - \varphi(\tilde{x}_t^{(i)})\right\|_p \\ &\leq \frac{M}{N}\gamma L G_t \qquad \text{(A.22)}\end{aligned}$$

where the identity is a consequence of the construction of $\mathcal{I}_t$ and we apply Minkowski's inequality, (A.21) and the assumption $|\mathcal{I}_t| = M$ to obtain (A.22). However, we have assumed that $\gamma M \leq \sqrt{N}$, hence

$$\left\|(\varphi, \xi_t^N) - (\varphi, \tilde{\xi}_t^N)\right\|_p \leq \frac{L G_t}{\sqrt{N}}.$$

□

A.3 Proofs of Chapter 4

Proof of Proposition 4.1

We prove this result by induction. For $t = 1$, let

$$\pi_1(\mathrm{d}\theta) = \pi_0(\mathrm{d}\theta)\frac{G_1(\theta)}{\int_\Theta G_1(\theta)\pi_0(\mathrm{d}\theta)}.$$

Since $G_1 \in B(\Theta)$ it follows that

$$\sup_{\theta\in\Theta}\left|\frac{G_1(\theta)}{(G_1, \pi_0)}\right| = \frac{\sup_{\theta\in\Theta} G_1(\theta)}{(G_1, \pi_0)} < \infty$$

because of Assumption 4.1. Hence $\pi_1 \ll \pi_0$ is a proper measure. Assume next, as an induction hypothesis, that $\pi_{T-1} \ll \pi_0$. Then

$$\pi_T(\mathrm{d}\theta) = \pi_{T-1}(\mathrm{d}\theta)\frac{G_T(\theta)}{(G_T, \pi_{T-1})}$$

and Assumption 4.1 implies (again) that

$$\frac{\sup_{\theta\in\Theta} G_T(\theta)}{(G_T, \pi_{T-1})} < \infty,$$

hence π_T is proper and $\pi_T \ll \pi_{T-1} \ll \pi_0$. Moreover, the Radon-Nikodym derivative of the final measure π_T with respect to the prior π_0 is

$$\frac{\mathrm{d}\pi_T}{\mathrm{d}\pi_0}(\theta) \propto \prod_{t=1}^{T} G_t(\theta) = \exp\left(-\sum_{i=1}^{n} f_i(\theta)\right).$$

From here, it easily follows that maximizing this Radon-Nikodym derivative is equivalent to solving problem (1.1). □

Proof of Proposition 4.2

We aim at minimizing

$$s(\theta) = \frac{1}{2}(y - \mathbf{x}^\top\theta)^2 + \frac{1}{2\gamma}\|\theta - \theta_0\|^2_{2,V_0}.$$

Computing the gradient $\nabla s(\theta)$ and setting it to zero yields

$$\nabla_\theta s(\theta) = -\mathbf{x}(y - \mathbf{x}^\top\theta) + \gamma^{-1}V_0^{-1}(\theta - \theta_0) = 0.$$

By rearranging terms, we obtain

$$\mathbf{x}y - \mathbf{x}\mathbf{x}^\top\theta = \gamma^{-1}V_0^{-1}(\theta - \theta_0),$$

which implies

$$(\gamma^{-1}V_0^{-1} + \mathbf{x}\mathbf{x}^\top)\theta = \gamma^{-1}V_0^{-1}\theta_0 + \mathbf{x}y.$$

Finally, solving for θ leaves us with

$$\theta = (\gamma^{-1}V_0^{-1} + \mathbf{x}\mathbf{x}^\top)^{-1}(\gamma^{-1}V_0^{-1}\theta_0 + \mathbf{x}y).$$

Applying the Sherman-Morrison lemma[1] (see, e. g., [189]), we can construct the sequence of identities

$$\begin{aligned}
\theta &= \left(\gamma V_0 - \frac{\gamma V_0 \mathbf{x}\mathbf{x}^\top \gamma V_0}{1+\gamma \mathbf{x}^\top V_0 \mathbf{x}}\right)(\gamma^{-1} V_0^{-1}\theta_0 + \mathbf{x}y) \\
&= \left(\gamma V_0 - \frac{\gamma V_0 \mathbf{x}\mathbf{x}^\top V}{\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x}}\right)(\gamma^{-1} V_0^{-1}\theta_0 + \mathbf{x}y) \\
&= \theta_0 + \gamma V_0 \mathbf{x}y - \frac{V_0 \mathbf{x}\mathbf{x}^\top \theta_0}{\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x}} - \frac{\gamma V_0 \mathbf{x}\mathbf{x}^\top V_0 \mathbf{x}y}{\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x}} \\
&= \theta_0 + \frac{V_0 \mathbf{x}y(\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x})}{\gamma^{-1}(\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x})} - \frac{\gamma^{-1} V_0 \mathbf{x}\mathbf{x}^\top \theta_0}{\gamma^{-1}(\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x})} - \frac{V_0 \mathbf{x}\mathbf{x}^\top V_0 \mathbf{x}y}{\gamma^{-1}(\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x})} \\
&= \theta_0 + \cancel{\frac{V_0 \mathbf{x}\mathbf{x}^\top V_0 \mathbf{x}y}{\gamma^{-1}(\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x})}} + \frac{\gamma^{-1} V_0 \mathbf{x}y}{\gamma^{-1}(\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x})} - \frac{\gamma^{-1} V_0 \mathbf{x}\mathbf{x}^\top \theta_0}{\gamma^{-1}(\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x})} - \cancel{\frac{V_0 \mathbf{x}\mathbf{x}^\top V_0 \mathbf{x}y}{\gamma^{-1}(\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x})}} \\
&= \theta_0 + \frac{V_0 \mathbf{x}y}{\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x}} - \frac{V_0 \mathbf{x}\mathbf{x}^\top \theta_0}{\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x}} \\
&= \theta_0 + \frac{V_0 \mathbf{x}(y - \mathbf{x}^\top \theta_0)}{\gamma^{-1}+\mathbf{x}^\top V_0 \mathbf{x}},
\end{aligned}$$

which is the desired result. □

Proof of Theorem 4.1

We proceed by an induction argument. At time $t = 0$, the bound

$$\|(\varphi, \pi_0^N) - (\varphi, \pi_0)\|_p \leq \frac{c_{0,p}\|\varphi\|_\infty}{\sqrt{N}}$$

is a straightforward consequence of the Marcinkiewicz–Zygmund inequality [129] because the particles $\{\theta_0^{(i)}\}_{i=1}^N$ are i.i.d samples from π_0.

Assume now that, after iteration $t-1$, we have a particle set $\{\theta_{t-1}^{(i)}\}_{i=1}^N$ and the empirical measure $\pi_{t-1}^N(\mathrm{d}\theta_{t-1}) = \frac{1}{N}\sum_{i=1}^N \delta_{\theta_{t-1}^{(i)}}(\mathrm{d}\theta_{t-1})$, which satisfies

$$\left\|(\varphi, \pi_{t-1}) - (\varphi, \pi_{t-1}^N)\right\|_p \leq \frac{c_{t-1,p}\|\varphi\|_\infty}{\sqrt{N}}. \tag{A.23}$$

We first analyze the error in the jittering step. To this end, we construct the *jittered* random measure

$$\hat{\pi}_t^N(\mathrm{d}\theta) = \frac{1}{N}\sum_{i=1}^N \delta_{\hat{\theta}_t^{(i)}}(\mathrm{d}\theta)$$

[1] If A is a square invertible matrix and u, v are column vectors such that $A + uv^\top$ is also invertible,

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}.$$

and iterate the triangle inequality to obtain

$$\begin{aligned}\|(\varphi, \pi_{t-1}) - (\varphi, \hat{\pi}_t^N)\|_p \leq & \|(\varphi, \pi_{t-1}) - (\varphi, \pi_{t-1}^N)\|_p + \|(\varphi, \pi_{t-1}^N) - (\varphi, \kappa\pi_{t-1}^N)\|_p \\ & + \|(\varphi, \kappa\pi_{t-1}^N) - (\varphi, \hat{\pi}_t^N)\|_p, \end{aligned} \tag{A.24}$$

where

$$\kappa\pi_{t-1}^N = \int \kappa(\mathrm{d}\theta|\theta_{t-1})\pi_{t-1}^N(\mathrm{d}\theta_{t-1}) = \frac{1}{N}\sum_{i=1}^N \kappa(\mathrm{d}\theta|\theta_{t-1}^{(i)}).$$

The first term on the right hand side (rhs) of (A.24) is bounded by the induction hypothesis (A.23). For the second term, we note that,

$$\begin{aligned}\left|(\varphi, \pi_{t-1}^N) - (\varphi, \kappa\pi_{t-1}^N)\right| &= \left|\frac{1}{N}\sum_{i=1}^N \varphi(\theta_{t-1}^{(i)}) - \frac{1}{N}\sum_{i=1}^N \int \varphi(\theta)\kappa(\mathrm{d}\theta|\theta_{t-1}^{(i)})\right| \\ &= \left|\frac{1}{N}\sum_{i=1}^N \int \left(\varphi(\theta_{t-1}^{(i)}) - \varphi(\theta)\right)\kappa(\mathrm{d}\theta|\theta_{t-1}^{(i)})\right| \\ &\leq \frac{1}{N}\sum_{i=1}^N \int \left|\varphi(\theta_{t-1}^{(i)}) - \varphi(\theta)\right|\kappa(\mathrm{d}\theta|\theta_{t-1}^{(i)}) \\ &\leq \frac{c_\kappa\|\varphi\|_\infty}{\sqrt{N}}, \end{aligned} \tag{A.25}$$

where the last inequality follows from Assumption 4.2. The upper bound in (A.25) is deterministic, so the inequality readily implies that

$$\|(\varphi, \pi_{t-1}^N) - (\varphi, \kappa\pi_{t-1}^N\|_p \leq \frac{c_\kappa\|\varphi\|_\infty}{\sqrt{N}}. \tag{A.26}$$

For the last term in (A.24), we let $\mathcal{F}_{t-1}$ be the σ-algebra generated by the random sequence $\{\theta_{0:t-1}^{(i)}, \hat{\theta}_{1:t-1}^{(i)}\}_{i=1}^N$. Let us first note that

$$\mathbb{E}\left[(\varphi, \hat{\pi}_t)|\mathcal{F}_{t-1}\right] = \frac{1}{N}\sum_{i=1}^N \mathbb{E}\left[\varphi(\hat{\theta}_t^{(i)})\right] = \frac{1}{N}\sum_{i=1}^N \int \varphi(\theta)\kappa(\mathrm{d}\theta|\theta_{t-1}^{(i)}) = (\varphi, \kappa\pi_{t-1}^N).$$

Therefore, the difference $(\varphi, \hat{\pi}_t^N) - (\varphi, \kappa\pi_{t-1}^N)$ takes the form

$$(\varphi, \hat{\pi}_t^N) - (\varphi, \kappa\pi_{t-1}^N) = \frac{1}{N}\sum_{i=1}^N S^{(i)},$$

where $S^{(i)} = \varphi(\hat{\theta}_t^{(i)}) - \mathbb{E}[\varphi(\hat{\theta}_t^{(i)})|\mathcal{F}_{t-1}]$, $i = 1, \ldots, N$, are zero-mean and independent (conditionally on $\mathcal{F}_{t-1}$) random variables, with $|S^{(i)}| \leq 2\|\varphi\|_\infty$. Then we readily

obtain the bound

$$\mathbb{E}\left[\left|(\varphi,\hat{\pi}_t^N)-(\varphi,\kappa\pi_{t-1}^N)\right|^p\middle|\mathcal{F}_{t-1}\right]=\frac{1}{N^p}\mathbb{E}\left[\left|\sum_{i=1}^N S^{(i)}\right|^p\middle|\mathcal{F}_{t-1}\right]$$
$$\leq\frac{B_{t,p}N^{\frac{p}{2}}\|\varphi\|_\infty^p}{N^p}. \tag{A.27}$$

where the relation (A.27) follows from the Marcinkiewicz–Zygmund inequality [129] and $B_{t,p}<\infty$ is some constant independent of N. Taking unconditional expectation on both sides of (A.27) and then computing $(\cdot)^{\frac{1}{p}}$ yields

$$\|(\varphi,\hat{\pi}_t^N)-(\varphi,\kappa\pi_{t-1}^N)\|_p\leq\frac{\hat{c}_{t,p}\|\varphi\|_\infty}{\sqrt{N}}. \tag{A.28}$$

where $\hat{c}_{t,p}=B_{t,p}^{\frac{1}{p}}$ is a finite constant independent of N. Therefore, taking together (A.23), (A.26) and (A.28) we have established that

$$\|(\varphi,\pi_{t-1})-(\varphi,\hat{\pi}_t^N)\|_p\leq\frac{c_{1,t,p}\|\varphi\|_\infty}{\sqrt{N}}, \tag{A.29}$$

where $c_{1,t,p}=c_{t-1,p}+c_\kappa+\hat{c}_{t,p}<\infty$ is a finite constant independent of N.

Next, we have to bound the error after the weighting step. We recall

$$\pi_t(\mathrm{d}\theta)=\pi_{t-1}(\mathrm{d}\theta)\frac{G_t(\theta)}{(G_t,\pi_{t-1})}\qquad\text{and define}\qquad\tilde{\pi}_t^N(\mathrm{d}\theta)=\hat{\pi}_t^N(\mathrm{d}\theta)\frac{G_t(\theta)}{(G_t,\hat{\pi}_t^N)}$$

where $\tilde{\pi}_t^N$ denotes the *weighted* measure. We first note that

$$|(\varphi,\pi_t)-(\varphi,\tilde{\pi}_t^N)|=\left|\frac{(\varphi G_t,\pi_{t-1})}{(G_t,\pi_{t-1})}-\frac{(\varphi G_t,\hat{\pi}_t^N)}{(G_t,\hat{\pi}_t^N)}+\frac{(\varphi G_t,\hat{\pi}_t^N)}{(G_t,\pi_{t-1})}-\frac{(\varphi G_t,\hat{\pi}_t^N)}{(G_t,\pi_{t-1})}\right|$$
$$\leq\frac{\left|(\varphi G_t,\pi_{t-1})-(\varphi G_t,\hat{\pi}_t^N)\right|+\|\varphi\|_\infty|(G_t,\hat{\pi}_t^N)-(G_t,\pi_{t-1})|}{(G_t,\pi_{t-1})}. \tag{A.30}$$

Then, using Minkowski's inequality together with (A.29), the inequality (A.30) readily yields

$$\|(\varphi,\pi_t)-(\varphi,\tilde{\pi}_t^N)\|_p\leq\frac{c_{1,t,p}\|\varphi G_t\|_\infty+c_{1,t,p}\|\varphi\|_\infty\|G_t\|_\infty}{(G_t,\pi_{t-1})\sqrt{N}},$$
$$\leq\frac{2c_{1,t,p}\|\varphi\|_\infty\|G_t\|_\infty}{(G_t,\pi_{t-1})\sqrt{N}}$$

where the second inequality follows from $\|\varphi G_t\|_\infty\leq\|\varphi\|_\infty\|G_t\|_\infty$. More concisely, we have

$$\|(\varphi,\pi_t)-(\varphi,\tilde{\pi}_t^N)\|_p\leq\frac{c_{2,t,p}\|\varphi\|_\infty}{\sqrt{N}} \tag{A.31}$$

where the constant

$$c_{2,t,p} = \frac{2c_{1,t,p}\|G_t\|_\infty}{(G_t, \pi_{t-1})} < \infty$$

is independent of N. Note that the assumptions on $(G_t)_{t\geq 1}$ imply that $(G_t, \pi_{t-1}) > 0$.

Finally, we bound the resampling step. Note that the resampling step consists of drawing N i.i.d samples from $\tilde{\pi}_t^N$, i.e. $\theta_t^{(i)} \sim \tilde{\pi}_t^N$ i.i.d for $i = 1, \ldots, N$, and then constructing

$$\pi_t^N(\mathrm{d}\theta) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\theta_t^{(i)}}(\mathrm{d}\theta).$$

Since samples are i.i.d, as in the base case, we have,

$$\|(\varphi, \tilde{\pi}_t^N) - (\varphi, \pi_t^N)\|_p \leq \frac{\tilde{c}_p\|\varphi\|_\infty}{\sqrt{N}}, \tag{A.32}$$

for some constant $\tilde{c}_p < \infty$ independent of N. Now combining (A.31) and (A.32), we have the desired result,

$$\|(\varphi, \pi_t) - (\varphi, \pi_t^N)\|_p \leq \frac{c_t\|\varphi\|_\infty}{\sqrt{N}}$$

where $c_t = c_{2,t,p} + \tilde{c}_p$ is a finite constant independent of N. □

Proof of Proposition 4.6

Recall the assumption

$$|F_t(\theta) - F_t(\theta')| \leq \ell_t\|\theta - \theta'\|.$$

We write $F_t^\star = \min_{\theta\in\Theta} F_t(\theta)$, which is assumed to be finite, but not necessarily nonnegative. We first prove that $\exp(-F_t(\theta))$ is also Lipschitz continuous. Note that we trivially have $\exp(-F_t(\theta)) \leq \exp(-F_t^\star)$ for all θ since $F_t(\theta) \geq F_t^\star$ for all θ. Now consider any $(\theta, \theta') \in \Theta \times \Theta$. We first consider the case where $F_t(\theta) \leq F_t(\theta')$. We obtain

$$\begin{aligned} 0 < e^{-F_t(\theta)} - e^{-F_t(\theta')} &= e^{-F_t(\theta)}\left(1 - e^{F_t(\theta)-F_t(\theta')}\right), \\ &\leq e^{-F_t(\theta)}\left(1 - (1 + F_t(\theta) - F_t(\theta'))\right), \end{aligned} \tag{A.33}$$

where we have used the inequality $e^a \geq 1 + a$. Therefore, we readily obtain from (A.33)

$$\begin{aligned} 0 < e^{-F_t(\theta)} - e^{-F_t(\theta')} &\leq e^{-F_t(\theta)}\left(F_t(\theta') - F_t(\theta)\right), \\ &\leq e^{-F_t^\star}\left(F_t(\theta') - F_t(\theta)\right) = e^{-F_t^\star}|F_t(\theta') - F_t(\theta)|, \end{aligned} \tag{A.34}$$

since $F_t(\theta) \leq F_t(\theta')$. Next, assume otherwise, i.e., $F_t(\theta) \geq F_t(\theta')$. In this case, we can also show using the same line of reasoning that

$$e^{-F_t(\theta')} - e^{-F_t(\theta)} \leq e^{-F_t^\star} \left(F_t(\theta) - F_t(\theta')\right) = e^{-F_t^\star} |F_t(\theta') - F_t(\theta)|, \tag{A.35}$$

since $F_t(\theta) \geq F_t(\theta')$. Therefore, we can conclude (combining (A.34) and (A.35)) that

$$|e^{-F_t(\theta)} - e^{-F_t(\theta')}| \leq e^{-F_t^\star} |F_t(\theta') - F_t(\theta)| \leq e^{-F_t^\star} \ell_t \|\theta - \theta'\|,$$

where the last inequality holds because F_t is Lipschitz. Finally recall that

$$\pi_t(\theta) = \frac{\exp(-F_t(\theta))}{Z_{\pi_t}},$$

where $Z_{\pi_t} = \int_\Theta \exp(-F_t(\theta)) \mathrm{d}\theta$. We straightforwardly obtain

$$|\pi_t(\theta) - \pi_t(\theta')| \leq \frac{1}{Z_{\pi_t}} e^{-F_t^\star} \ell_t \|\theta - \theta'\|.$$

□

A.4 Proofs of Chapter 5

Proof of Proposition 5.1

We prove this result by induction. The model is constructed in such a way that $P_0 = V_0 \otimes I_m$ and we assume that $P_{k-1} = V_{k-1} \otimes I_m$, with the sequence $\{V_l; 1 \leq l \leq k-1\}$ computed as in (5.11). To obtain an expression for V_k at time k, we start substituting $H_k = x_k^\top \otimes I_m$, $R = \lambda \otimes I_m$ and $P_{k-1} = V_{k-1} \otimes I_m$ into (5.10), which yields

$$\begin{aligned} P_k = (V_{k-1} \otimes I_m) - (V_{k-1} \otimes I_m)(x_k \otimes I_m)((x_k^\top \otimes I_m)(V_{k-1} \otimes I_m)(x_k \otimes I_m) + \lambda \otimes I_m)^{-1} \\ \times (x_k^\top \otimes I_m)(V_{k-1} \otimes I_m). \end{aligned}$$

Applying the mixed product property (1.4) repeatedly in the equation above we arrive at

$$P_k = (V_{k-1} \otimes I_m) - (V_{k-1}x_k \otimes I_m)((x_k^\top V_{k-1} x_k + \lambda)^{-1} \otimes I_m)(x_k^\top V_{k-1} \otimes I_m),$$

where we also resorted to property (1.5). Applying the mixed product property (1.4) again leads to

$$P_k = \left(V_{k-1} - \frac{V_{k-1}x_k x_k^\top V_{k-1}}{x_k^\top V_{k-1} x_k + \lambda}\right) \otimes I_m, \tag{A.36}$$

where the expression between brackets matches the right-hand side of (5.11), hence $P_k = V_k \otimes I_m$ and the proof is complete. □

Proposition 5.2

Substituting $P_{k-1} = V_{k-1} \otimes I_m$ (given by Proposition 5.1) $H_k = x_k^\top \otimes I_m$ and $R_k = \lambda \otimes I_m$ into (5.9) we obtain

$$c_k = c_{k-1} + (V_{k-1} \otimes I_m)(x_k \otimes I_m)\Big((x_k^\top \otimes I_m)(V_{k-1} \otimes I_m)(x_k \otimes I_m) + \lambda \otimes I_m\Big)^{-1} \times (y_k - (x_k^\top \otimes I_m)c_{k-1}).$$

Repeatedly using the mixed product property (1.4) in the equation above we arrive at

$$c_k = c_{k-1} + (V_{k-1}x_k \otimes I_m)\Big((x_k^\top V_{k-1}x_k + \lambda) \otimes I_m\Big)^{-1}(y_k - (x_k^\top \otimes I_m)c_{k-1})$$

and applying (1.5), together with the mixed product property again, yields

$$c_k = c_{k-1} + \left[\frac{V_{k-1}x_k}{x_k^\top V_{k-1}x_k + \lambda} \otimes I_m\right] \times (y_k - (x_k^\top \otimes I_m)c_{k-1}). \tag{A.37}$$

If we now use (1.3) on the last term of the right-hand side of (A.37) we obtain

$$c_k = c_{k-1} + \left[\frac{V_{k-1}x_k}{x_k^\top V_{k-1}x_k + \lambda} \otimes I_m\right](y_k - C_{k-1}x_k),$$

Since $(y_k - C_{k-1}x_k)$ and $\frac{V_{k-1}x_k}{x_k^\top V_{k-1}x_k + \lambda}$ are vectors, we can rewrite this expression as,

$$c_k = c_{k-1} + \left[\operatorname{vec}\left(\frac{V_{k-1}x_k}{x_k^\top V_{k-1}x_k + \lambda}\right) \otimes I_m\right]\operatorname{vec}(y_k - C_{k-1}x_k). \tag{A.38}$$

Finally, applying (1.3) to the second term of the sum in the right-hand side of Eq. (A.38) yields

$$c_k = c_{k-1} + \operatorname{vec}\left(\frac{(y_k - C_{k-1}x_k)x_k^\top V_{k-1}^\top}{x_k^\top V_{k-1}x_k + \lambda}\right), \tag{A.39}$$

Now using inverse vectorisation $\operatorname{vec}_{m\times r}^{-1}(\cdot)$, we recover the update rule (5.12) and conclude the proof. □

References

[1] Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[2] Alan Bain and Dan Crisan. *Fundamentals of stochastic filtering*. Springer, 2009.

[3] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer, New York, NY, USA, second edition, 2006.

[4] Andrew H Jazwinski. *Stochastic processes and filtering theory*. Academic Press Inc., New York, 1970.

[5] Brian DO Anderson and John B Moore. Optimal filtering. *Englewood Cliffs, NJ: Pren*, 1979.

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

[7] Rudolf Emil Kalman. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5(2):102–119, 1960.

[8] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. *Journal of basic engineering*, 83(1):95–108, 1961.

[9] Harold Wayne Sorenson. *Kalman filtering: theory and application*. IEEE, 1985.

[10] Mohinder S Grewal and Angus P Andrews. Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems*, 30(3):69–78, 2010.

[11] Arthur Gelb. *Applied optimal estimation*. MIT press, 1974.

[12] Simon J Julier and Jeffrey K Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194. International Society for Optics and Photonics, 1997.

[13] Eric A Wan and Rudolph Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. IEEE, 2000.

[14] Eric A Wan and Rudolph Van Der Merwe. The unscented Kalman filter. *Kalman filtering and neural networks*, pages 221–280, 2001.

[15] Geir Evensen. Using the extended Kalman filter with a multilayer quasi-geostrophic ocean model. *Journal of Geophysical Research: Oceans*, 97(C11):17905–17924, 1992.

[16] Gerrit Burgers, Peter Jan van Leeuwen, and Geir Evensen. Analysis scheme in the ensemble Kalman filter. *Monthly weather review*, 126(6):1719–1724, 1998.

[17] Geir Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[18] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

[19] Michael Roth, Carsten Fritsche, Gustaf Hendeby, and Fredrik Gustafison. The ensemble Kalman filter and its relations to other nonlinear filters. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 1236–1240. IEEE, 2015.

[20] 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.

[21] Peter Müller. Monte Carlo integration in general dynamic models. *Contemporary Mathematics*, 115:145–163, 1991.

[22] Adrian FM Smith and Alan E Gelfand. Bayesian statistics without tears: A sampling–resampling perspective. *The American Statistician*, 46(2):84–88, 1992.

[23] Nicholas Metropolis and Stanislaw Ulam. The Monte Carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[24] 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.

[25] Petar M Djuric, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. Particle filtering. *IEEE signal processing magazine*, 20(5):19–38, 2003.

[26] 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.

[27] Dan Crisan, Pierre Del Moral, and Terry Lyons. *Discrete filtering using branching and interacting particle systems*. Université de Toulouse. Laboratoire de Statistique et Probabilités [LSP], 1998.

[28] Pierre Del Moral and Alice Guionnet. Central limit theorem for nonlinear filtering and interacting particle systems. *Annals of Applied Probability*, 9(2):275–297, 1999.

[29] P. Del Moral and L. Miclo. Branching and interacting particle systems. Approximations of Feynman-Kac formulae with applications to non-linear filtering. In *Azéma J., Ledoux M., Émery M., Yor M. (eds) Séminaire de Probabilités XXXIV. Lecture Notes in Mathematics*, volume 1729, pages 1–145. Springer, Berlin, Heidelberg, 2000.

[30] Pierre Del Moral and Alice Guionnet. On the stability of interacting processes with applications to filtering and genetic algorithms. *Annales de l'Institut Henri Poincare (B) Probability and Statistics*, 37(2):155–194, 2001.

[31] Dan Crisan. Particle filters–a theoretical perspective. In *Doucet A., de Freitas N., Gordon N. (eds) Sequential Monte Carlo Methods in Practice. Statistics for Engineering and Information Science*, pages 17–41. Springer, New York, NY, 2001.

[32] Dan Crisan and Arnaud Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on signal processing*, 50(3):736–746, 2002.

[33] Pierre Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, 2004.

[34] Nicolas Chopin. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.

[35] Thomas Bengtsson, Peter Bickel, and Bo Li. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In *Probability and statistics: Essays in honor of David A. Freedman*, pages 316–334. Institute of Mathematical Statistics, 2008.

[36] Chris Snyder, Thomas Bengtsson, Peter Bickel, and Jeff Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640, 2008.

[37] Peter Bickel, Bo Li, and Thomas Bengtsson. Sharp failure rates for the bootstrap particle filter in high dimensions. In *Pushing the limits of contemporary statistics: Contributions in honor of Jayanta K. Ghosh*, pages 318–329. Institute of Mathematical Statistics, 2008.

[38] Patrick Rebeschini and Ramon Van Handel. Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866, 2015.

[39] Alexandros Beskos, Dan Crisan, Ajay Jasra, Kengo Kamatani, and Yan Zhou. A stable particle filter for a class of high-dimensional state-space models. *Advances in Applied Probability*, 49(1):24–48, 2017.

[40] James E Hoke and Richard A Anthes. The initialization of numerical models by a dynamic-initialization technique. *Monthly Weather Review*, 104(12):1551–1556, 1976.

[41] Paola Malanotte-Rizzoli and William R Holland. Data constraints applied to models of the ocean general circulation. Part I: The steady case. *Journal of Physical Oceanography*, 16(10):1665–1682, 1986.

[42] Paola Malanotte-Rizzoli and William R Holland. Data constraints applied to models of the ocean general circulation. Part II: the transient, eddy-resolving case. *Journal of Physical Oceanography*, 18(8):1093–1107, 1988.

[43] X Zou, IM Navon, and FX LeDimet. An optimal nudging data assimilation scheme using parameter estimation. *Quarterly Journal of the Royal Meteorological Society*, 118(508):1163–1186, 1992.

[44] Peter Jan van Leeuwen. Particle filtering in geophysical systems. *Monthly Weather Review*, 137(12):4089–4114, 2009.

[45] Peter Jan van Leeuwen. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, 136(653):1991–1999, 2010.

[46] Melanie Ades and Peter Jan van Leeuwen. An exploration of the equivalent weights particle filter. *Quarterly Journal of the Royal Meteorological Society*, 139(672):820–840, 2013.

[47] Melanie Ades and Peter J van Leeuwen. The equivalent-weights particle filter in a high-dimensional system. *Quarterly Journal of the Royal Meteorological Society*, 141(687):484–503, 2015.

[48] Alexandre J Chorin and Xuemin Tu. Implicit sampling for particle filters. *Proceedings of the National Academy of Sciences*, 106(41):17249–17254, 2009.

[49] Alexandre Chorin, Matthias Morzfeld, and Xuemin Tu. Implicit particle filters for data assimilation. *Communications in Applied Mathematics and Computational Science*, 5(2):221–240, 2010.

[50] Ethan Atkins, Matthias Morzfeld, and Alexandre J Chorin. Implicit particle methods and their connection with variational data assimilation. *Monthly Weather Review*, 141(6):1786–1803, 2013.

[51] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific, 1999.

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

[53] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[54] Yonina C Eldar and Gitta Kutyniok. *Compressed sensing: theory and applications*. Cambridge University Press, 2012.

[55] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[56] Ahron Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, volume 2. SIAM, 2001.

[57] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[58] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

[59] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[60] Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

[61] Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

[62] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[63] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[64] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[65] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[66] Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 265–272. Omnipress, 2011.

[67] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011.

[68] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[69] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[70] Mert Gürbüzbalaban, Asu Ozdaglar, and Pablo Parrilo. Why random reshuffling beats stochastic gradient descent. *arXiv preprint arXiv:1510.08560*, 2015.

[71] Ohad Shamir. Without-replacement sampling for stochastic gradient methods. In *Advances in Neural Information Processing Systems*, pages 46–54, 2016.

[72] Nicol N Schraudolph, Jin Yu, and Simon Günter. A stochastic quasi-newton method for online convex optimization. In *Artificial Intelligence and Statistics*, pages 436–443, 2007.

[73] Antoine Bordes, Léon Bottou, and Patrick Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

[74] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

[75] Xiao Wang, Shiqian Ma, Donald Goldfarb, and Wei Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

[76] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[77] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[78] Bernard Lemaire. The proximal algorithm. *International series of numerical mathematics*, 87:73–87, 1989.

[79] Alfredo N Iusem. Augmented lagrangian methods and proximal point methods for convex optimization. *Investigación Operativa*, 8(11-49):7, 1999.

[80] Lieven Vandenberghe. Optimization methods for large-scale systems. *Lecture Notes*, 2009.

[81] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[82] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

[83] Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and Francis Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12(Jul):2297–2334, 2011.

[84] Dimitri P Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010:1–38, 2011.

[85] Dimitri P Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical programming*, 129(2):163–195, 2011.

[86] Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2014.

[87] Pascal Bianchi. Ergodic convergence of a stochastic proximal point algorithm. *SIAM Journal on Optimization*, 26(4):2235–2260, 2016.

[88] Yves F Atchadé, Gersende Fort, and Eric Moulines. On perturbed proximal gradient algorithms. *J. Mach. Learn. Res*, 18(1):310–342, 2017.

[89] Hilal Asi and John C Duchi. Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *arXiv preprint arXiv:1810.05633*, 2018.

[90] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[91] Omer Deniz Akyildiz, Victor Elvira, and Joaquin Miguez. The Incremental Proximal Method: A Probabilistic Perspective. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, April 2018.

[92] James Vuckovic. Kalman gradient descent: Adaptive variance reduction in stochastic optimization. *arXiv preprint arXiv:1810.12273*, 2018.

[93] Laurence Aitchison. A unified theory of adaptive stochastic gradient descent as Bayesian filtering. *arXiv preprint arXiv:1807.07540*, 2018.

[94] Philipp Hennig, Michael A Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. In *Proc. R. Soc. A*, volume 471, page 20150142. The Royal Society, 2015.

[95] Jon Cockayne, Chris Oates, Tim Sullivan, and Mark Girolami. Bayesian probabilistic numerical methods. *arXiv preprint arXiv:1702.03673*, 2017.

[96] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. Technical report, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2011.

[97] James C Spall. *Introduction to stochastic search and optimization: Estimation, simulation, and control*, volume 65. John Wiley & Sons, 2005.

[98] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*, volume 8 of *MPS-SIAM Series on Optimization*. SIAM, 2009.

[99] Andre Wibisono, Martin J Wainwright, Michael I Jordan, and John C Duchi. Finite sample convergence rates of zero-order stochastic optimization methods. In *Advances in Neural Information Processing Systems*, pages 1439–1447, 2012.

[100] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[101] Ruobing Chen and Stefan Wild. Randomized derivative-free optimization of noisy convex functions. *arXiv preprint arXiv:1507.03332*, 2015.

[102] Francis Bach and Vianney Perchet. Highly-smooth zero-th order online optimization. In *Conference on Learning Theory*, pages 257–283, 2016.

[103] Joaquín Míguez, Dan Crisan, and Petar M Djurić. On the convergence of two sequential Monte Carlo methods for maximum a posteriori sequence estimation and stochastic global optimization. *Statistics and Computing*, 23(1):91–107, 2013.

[104] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[105] Christian P Robert and George Casella. *Monte Carlo statistical methods*. John Wiley & Sons, 2004.

[106] Luca Martino, Víctor Elvira, David Luengo, Jukka Corander, and Francisco Louzada. Orthogonal parallel MCMC methods for sampling and optimization. *Digital Signal Processing*, 58:64–84, 2016.

[107] Joaquin Miguez, Cristina S Maiz, Petar M Djuric, and Dan Crisan. Sequential Monte Carlo optimization using artificial state-space models. In *IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop*, pages 268–273. IEEE, 2009.

[108] Joaquin Miguez. Analysis of a sequential monte carlo method for optimization in dynamical systems. *Signal Processing*, 90(5):1609–1622, 2010.

[109] Tito Homem-de Mello and Güzin Bayraksan. Monte Carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science*, 19(1):56–85, 2014.

[110] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[111] Changyou Chen, David Carlson, Zhe Gan, Chunyuan Li, and Lawrence Carin. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In *Artificial Intelligence and Statistics*, pages 1051–1060, 2016.

[112] Pierre Alquier, Nial Friel, Richard Everitt, and Aidan Boland. Noisy Monte Carlo: Convergence of Markov chains with approximate transition kernels. *Statistics and Computing*, 26(1-2):29–47, 2016.

[113] Omer Deniz Akyildiz, Ines P Mariño, and Joaquín Míguez. Adaptive noisy importance sampling for stochastic optimization. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2017 IEEE 7th International Workshop on*, pages 1–5. IEEE, 2017.

[114] Ömer Deniz Akyıldız, Víctor Elvira, Jesus Fernandez-Bes, and Joaquín Miguez. On the relationship between online optimizers and stochastic filters. *NIPS 2016 Workshop on Optimizing the Optimizers*, 2016.

[115] Ömer Deniz Akyıldız and Joaquín Míguez. Nudging the particle filter. *arXiv:1708.07801*, 2017.

[116] Ömer Deniz Akyildiz, Dan Crisan, and Joaquín Míguez. Parallel sequential monte carlo for stochastic optimization. *arXiv:1811.09469*, 2018.

[117] Ömer Deniz Akyildiz and Joaquín Míguez. Dictionary filtering: a probabilistic approach to online matrix factorisation. *Signal, Image and Video Processing*, Dec 2018.

[118] David A Harville. *Matrix algebra from a statistician's perspective*, volume 1. Springer, 1997.

[119] Thomas Bayes, Richard Price, and John Canton. An essay towards solving a problem in the doctrine of chances. 1763.

[120] Pierre Simon Laplace. *Théorie analytique des probabilités*. Courcier, 1820.

[121] Pierre Del Moral. *Mean field simulation for Monte Carlo integration*. CRC Press, 2013.

[122] Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[123] Randal Douc, Éric Moulines, and David Stoffer. *Nonlinear Time Series: Theory, Methods and Applications with R Examples*. Chapman & Hall, 2013.

[124] Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[125] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

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

[127] Luc Devroye. *Non-uniform random variate generation*. Springer-Verlag, New York, 1986.

[128] Luca Martino, David Luengo, and Joaquín Míguez. *Independent random sampling methods*. Springer, 2018.

[129] Albert N Shiryaev. *Probability*. Springer, 1996.

[130] John M Hammersley and David C Handscomb. *Monte Carlo methods*. Methuen & Co., London, 1965.

[131] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[132] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo methods in Practice*. Springer, New York, NY, 2001.

[133] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential Monte Carlo methods. In *Doucet A., de Freitas N., Gordon N. (eds) Sequential Monte Carlo Methods in Practice. Statistics for Engineering and Information Science*, pages 3–14. Springer, New York, NY, 2001.

[134] Randal Douc and Olivier Cappé. Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69. IEEE, 2005.

[135] Isambi S Mbalawata and Simo Sarkka. On the L^4 convergence of particle filters with general importance distributions. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 8048–8052. IEEE, 2014.

[136] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[137] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[138] Ernest K Ryu and Stephen Boyd. Stochastic proximal iteration: A non-asymptotic improvement upon stochastic gradient descent. *Author website, early draft*, 2014.

[139] Andrei Patrascu and Ion Necoara. Nonasymptotic convergence of stochastic proximal point methods for constrained convex optimization. *The Journal of Machine Learning Research*, 18(1):7204–7245, 2017.

[140] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[141] D. Crisan and J. Miguez. Uniform convergence over time of a nested particle filtering scheme for recursive parameter estimation in state–space Markov models. *Advances in Applied Probability*, 49(4):1170–1200, 2017.

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

[143] D. Crisan and A. Doucet. A survey of convergence results on particle filtering. *IEEE Transactions on Signal Processing*, 50(3):736–746, March 2002.

[144] B. N. Oreshkin and M. J. Coates. Analysis of error propagation in particle filters with approximation. *The Annals of Applied Probability*, 21(6):2343–2378, 2011.

[145] Dan Crisan and Joaquin Miguez. Nested particle filters for online parameter estimation in discrete-time state-space Markov models. *Bernoulli*, 24(4A):3039–3086, 2018.

[146] Dimitri P Bertsekas. *Dynamic Programming and Optimal Control, Vol. I*. Athena Scientific, Belmont, MA, 2001.

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

[148] Randal Douc, Eric Moulines, and Jimmy Olsson. Optimality of the auxiliary particle filter. *Probability and Mathematical Statistics*, 29(1):1–28, 2009.

[149] Ruey S Tsay. *Analysis of Financial Time Series*. John Wiley & Sons, 2005.

[150] Johan Dahlin and Thomas B Schön. Getting started with particle Metropolis-Hastings for inference in nonlinear dynamical models. *arXiv*, 1511.01707, 2015.

[151] Dan Crisan and Joaquin Miguez. Nested particle filters for online parameter estimation in discrete-time state-space markov models. *Bernoulli*, 24(4A):3039–3086, 2018.

[152] M Montaz Ali, Charoenchai Khompatraporn, and Zelda B Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of global optimization*, 31(4):635–672, 2005.

[153] Emilie Chouzenoux, Jean-Christophe Pesquet, and Audrey Repetti. Variable metric forward–backward algorithm for minimizing the sum of a differentiable function and a convex function. *Journal of Optimization Theory and Applications*, 162(1):107–132, 2014.

[154] Emilie Chouzenoux, Jean-Christophe Pesquet, and Audrey Repetti. A block coordinate variable metric forward–backward algorithm. *Journal of Global Optimization*, 66(3):457–485, 2016.

[155] Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. *ICML (3)*, 28:343–351, 2013.

[156] Víctor Elvira, Joaquín Míguez, and Petar M Djurić. Adapting the number of particles in sequential monte carlo methods through an online scheme for convergence assessment. *IEEE Transactions on Signal Processing*, 65(7):1781–1794, 2017.

[157] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 1998.

[158] Matt P Wand and M Chris Jones. *Kernel smoothing*. Chapman and Hall/CRC, 1994.

[159] Dan Crisan and Joaquín Míguez. Particle-kernel estimation of the filter density in state-space models. *Bernoulli*, 20(4):1879–1929, 2014.

[160] Song Mei, Yu Bai, and Andrea Montanari. The landscape of empirical risk for nonconvex losses. *The Annals of Statistics*, 46(6A):2747–2774, 2018.

[161] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

[162] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999.

[163] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2001.

[164] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[165] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[166] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Neural Information Processing Systems (NIPS) Conference*, volume 1, pages 2–1, 2007.

[167] Mikkel N Schmidt, Ole Winther, and Lars Kai Hansen. Bayesian non-negative matrix factorization. In *International Conference on Independent Component Analysis and Signal Separation*, pages 540–547. Springer, 2009.

[168] Ali Taylan Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Computational Intelligence and Neuroscience*, pages 4:1–4:17, January 2009.

[169] Serhat S Bucak and Bilge Gunsel. Incremental subspace learning via non-negative matrix factorization. *Pattern recognition*, 42(5):788–797, 2009.

[170] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011.

[171] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[172] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Online non-negative matrix factorization with robust stochastic approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1087–1099, 2012.

[173] Maren Mahsereci and Philipp Hennig. Probabilistic line searches for stochastic optimization. In *Advances In Neural Information Processing Systems*, pages 181–189, 2015.

[174] S. Yildirim, A. T. Cemgil, and S. S. Singh. An online expectation-maximisation algorithm for nonnegative matrix factorisation models. In *16th IFAC Symposium on System Identification (SYSID 2012)*, 2012.

[175] John Paisley, D Blei, and Michael I Jordan. Bayesian nonnegative matrix factorization with stochastic variational inference. In *volume Handbook of Mixed Membership Models and Their Applications, chapter 11.* Chapman and Hall/CRC, 2015.

[176] Carlos M Carvalho and Mike West. Dynamic matrix-variate graphical models. *Bayesian analysis*, 2(1):69–97, 2007.

[177] K Triantafyllopoulos. Reference priors for matrix-variate dynamic linear models. *Communications in Statistics—Theory and Methods*, 37(6):947–958, 2008.

[178] Philipp Hennig and Martin Kiefel. Quasi-newton methods: A new direction. *The Journal of Machine Learning Research*, 14(1):843–865, 2013.

[179] Sungjin Ahn, Anoop Korattikara, Nathan Liu, Suju Rajan, and Max Welling. Large-scale distributed Bayesian matrix factorization using stochastic gradient MCMC. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 9–18. ACM, 2015.

[180] Yann Ollivier. Online natural gradient as a Kalman filter. *arXiv:1703.00209*, 2017.

[181] Ömer Deniz Akyıldız. Online matrix factorization via Broyden updates. *arXiv preprint, arXiv:1506.04389*, 2015.

[182] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv:1609.08675*, 2016.

[183] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6(Oct):1705–1749, 2005.

[184] Ömer Deniz Akyildiz, Emilie Chouzenoux, Víctor Elvira, and Joaquín Míguez. A probabilistic incremental proximal gradient method. *arXiv preprint arXiv:1812.01655*, 2018.

[185] Christelle Vergé, Cyrille Dubarry, Pierre Del Moral, and Eric Moulines. On parallel implementation of sequential Monte Carlo methods: the island particle model. *Statistics and Computing*, 25(2):243–260, 2015.

[186] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[187] Tarek A El Moselhy and Youssef M Marzouk. Bayesian inference with optimal maps. *Journal of Computational Physics*, 231(23):7815–7850, 2012.

[188] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386, 2016.

[189] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook. *Technical University of Denmark*, 7:15, 2008.