NESTED FILTERING METHODS

FOR BAYESIAN INFERENCE IN STATE SPACE MODELS

by

Sara Pérez Vieites

in partial fulfillment of the requirements for the degree of

Doctor in Multimedia and Communications

Universidad Carlos III de Madrid

Advisor:

Joaquín Míguez

January, 2022

Acknowledgements

This thesis has been the result of a journey of four years, full of experiences that made me grow professionally and personally. I have learnt more than I thought I could and I have met many people along the way that have supported me and advised me. This growth wouldn't be possible without them.

First and foremost, I am truly grateful to my supervisor Joaquín Míguez, who guided me and taught me to do research. I admire him not only for his depth knowledge but also for his patience and kindness. Without his support this thesis wouldn't exist. I think I've been really lucky having him as my advisor and I will miss all the discussions we had.

Besides, I am thankful to all the great colleagues of the Signal Processing Group (GTS) at Carlos III, with whom I've worked and spent time during the last years. In particular, I want to thank Gonzalo Vázquez, Pablo M. Olmos, Tobias Koch, David Ramírez, Chao Qi, Jithin Ravi, Luca Martino, Manuel Vázquez and Javier López for the advice and the interesting conversations during lunch time. Also I am thankful to my laboratory workmates Pablo Bonilla, Alejandro Lancho, Fran Hernando, Alfredo Nazábal and Melanie Fernández for the chatting we had from time to time in our breaks. Special mention to Gonzalo Ríos and Grace Silvana Villacrés, that have shared with me space in the cubicle, being always the first ones willing to help. Finally, I am deeply grateful for the friendship with Deniz Akyıldız. Although he had the ability to get on my nerves frequently, I can still say that I trust him unconditionally.

In addition, I am thankful to all the people of the Department of Mathematics and Statistics at the University of Reading for hosting me during my stay. In particular, I appreciate the time and knowledge that my advisor in Reading, Jochen Broecker, shared with me. Also, I am very grateful to Elena Saggioro, Giulia Carigi, Ieva Dauzickaite, Manuel Santos and Sebastiano Roncoroni, that were so welcoming that made me feel like home.

Last but not least, I am deeply thankful to my family. Without my parents, Rosa and Javier, and my brother Diego, I wouldn't have even started this journey. With them I learned to challenge myself and to value my work. However, I wouldn't have made it to the end if it weren't for my partner, Alejandro, that was there encouraging me throughout the whole process.

Thank you.

PUBLISHED AND SUBMITTED CONTENT

The technical contributions presented in this thesis have been published in the following journals and conference papers:

> Sara Pérez-Vieites and Joaquín Míguez. "Kalman-based nested hybrid filters for recursive inference in state-space models". *2020 28th European Signal Processing Conference (EUSIPCO)*, 2468-2472.
> Available at `https://doi.org/10.23919/Eusipco47968.2020.9287359`.

The contribution of this paper is partially included in Chapter 4. The material from this source included in this thesis is not singled out with typographic means and references.

> Sara Pérez-Vieites and Joaquín Míguez. "A nested hybrid filter for parameter estimation and state tracking in homogeneous multi-scale models". *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 1-8.
> Available at `https://doi.org/10.23919/FUSION45008.2020.9190630`.

The contribution of this paper is partially included in Chapter 5. The material from this source included in this thesis is not singled out with typographic means and references.

> Sara Pérez-Vieites, Inés Pérez Mariño and Joaquín Míguez. "Probabilistic scheme for joint parameter estimation and state prediction in complex dynamical systems". *Physical Review E*, 98 (6), 063305.
> Available at `https://doi.org/10.1103/PhysRevE.98.063305`.

The whole paper is included in Chapter 3 and in Appendices A, B and C. The material from this source included in this thesis is not singled out with typographic means and references.

The corresponding preprint versions can be found on Arxiv:

> Sara Pérez-Vieites and Joaquín Míguez. "Nested Gaussian filters for recursive Bayesian inference and nonlinear tracking in state space models". *Signal Processing*, 189, 108295.
> Available at: `https://doi.org/10.1016/j.sigpro.2021.108295`.

It is wholly included in Chapter 4 and in Appendix D. The material from this source included in this thesis is not singled out with typographic means and references.

Abstract

A common feature to many problems in some of the most active fields of science is the need to calibrate (i.e., estimate the parameters) and then forecast the time evolution of high-dimensional dynamical systems using sequentially collected data. In this dissertation we introduce a generalised nested filtering methodology that is structured in (two or more) intertwined layers in order to estimate the static parameters and the dynamic state variables of nonlinear dynamical systems. This methodology is essentially probabilistic. It aims at recursively computing the sequence of posterior probability distributions of the unknown model parameters and its (time-varying) state variables conditional on the available observations. To be specific, in the first layer of the filter we approximate the posterior probability distribution of the static parameters and in the consecutive layers we employ filtering (or data assimilation) techniques to track and predict different conditional probability distributions of the state variables. We have investigated the use of different Monte Carlo-based methods and Gaussian filtering techniques in each of the layers, leading to a wealth of algorithms.

In a first approach, we have introduced a nested filtering methodology of two layers that aims at recursively estimating the static parameters and the dynamical state variables of a state space model. This probabilistic scheme uses Monte Carlo-based methods in the first layer of the filter, combined with the use of Gaussian filters in the second layer. Different from the nested particle filter (NPF) of [25], the use of Gaussian filtering techniques in the second layer allows for fast implementations, leading to algorithms that are better suited to high-dimensional systems. As each layer uses different types of methods, we refer to the proposed methodology as *nested hybrid filtering*. We specifically explore the combination of Monte Carlo and quasi–Monte Carlo approximations in the first layer, including sequential Monte Carlo (SMC) and sequential quasi-Monte Carlo (SQMC), with standard Gaussian filtering methods in the second layer, such as the ensemble Kalman filter (EnKF) and the extended Kalman filter (EKF). However, other algorithms can fit naturally within the framework. Additionally, we prove a general convergence result for a class of procedures that use SMC in the first layer and we show numerical results for a stochastic two-scale Lorenz 96 system, a model commonly used to assess data assimilation (filtering) procedures in Geophysics. We apply and compare different implementations of the methodology to the tracking of the state and the estimation of the fixed parameters. We show estimation and forecasting results, obtained with a desktop computer, for up to 5000 dynamic state variables.

As an extension of the nested hybrid filtering methodology, we have introduced a class of schemes that can incorporate deterministic sampling techniques (such as the cubature Kalman filter (CKF) or the unscented Kalman filter (UKF)) in the first layer of the algorithm, instead of the Monte Carlo-based methods employed in the original procedure. As all the methods used in this scheme are Gaussian, we refer to this class of algorithms as *nested Gaussian filters*. One more time, we reduce the computational cost with the proposed scheme, making the resulting algorithms potentially better-suited for high-dimensional state and parameter spaces. In the numerical results, we describe and implement a specific instance of the new method (a UKF-EKF algorithm) and evaluate its average performance in terms of estimation errors and running times for nonlinear stochastic models. Specifically, we present numerical results for a stochastic Lorenz 63 model using synthetic data, as well as for a stochastic volatility model with real-world data.

Finally, we have extended the proposed methodology in order to estimate the static parameters and the dynamical variables of a class of heterogeneous multi-scale state-space models [1]. This scheme combines three or more layers of filters, one inside the other. Each of the layers corresponds to the different time scales that are relevant to the dynamics of this kind of state-space models, allocating the variables with the greatest time scales (the slowest ones) in the outer-most layer and the variables with the smallest time scales (the fastest ones) to the inner-most layer. In particular, we describe a three-layer filter that approximates the posterior probability distribution of the parameters in a first layer of computation, in a second layer we approximate the posterior probability distribution of the slow state variables, and the posterior probability distribution of the fast state variables is approximated in a third layer. To be specific, we describe two possible algorithms that derive from this scheme, combining Monte Carlo methods and Gaussian filters at different layers. The first method uses SMC methods in both first and second layers, together with a bank of UKFs in the third layer (i.e., a SMC-SMC-UKF algorithm). The second method employs a SMC in the first layer, EnKFs at the second layer and introduces the use of a bank of EKFs in the third layer (i.e., a SMC-EnKF-EKF algorithm). We present numerical results for a two-scale stochastic Lorenz 96 model with synthetic data.

Contents

List of Figures

List of Tables

List of Acronyms

APF	auxiliary particle filter
CKF	cubature Kalman filter
EKF	extended Kalman filter
EnKF	ensemble Kalman filter
i.i.d.	independent and identically distributed
KF	Kalman filter
MSE	mean square error
NGF	nested Gaussian filter
NHF	nested hybrid filter
NMSE	normalized mean square error
NPF	nested particle filter
NSMC	nested sequential Monte Carlo
PDE	partial differential equation
pdf	probability density function
PF	particle filter
PMCMC	particle Markov chain Monte Carlo
QMC	quasi Monte Carlo
r.v.	random variable
RML	recursive maximum likelihood
SDE	stochastic differential equation
SIR	sequential importance resampling
SMC	sequential Monte Carlo
SMC^2	sequential Monte Carlo square
SQMC	sequential quasi-Monte Carlo

ST-PF	space-time particle filter
UKF	unscented Kalman filter
UT	unscented transform
w.r.t.	with respect to

1 Introduction

1.1 Motivation

Many problems in science and engineering involve the calibration of dynamical models and their subsequent use to track and predict the evolution, over time, of a dynamical system. ‘Calibration’ may have different implications in different problems, but most often it refers to the estimation of a set of unknown, static parameters using real-world data. Of course, the processes of prediction and tracking, and parameter estimation are closely related. Typically, the same data are used for both tasks and, in problems where observations are collected sequentially and online, we would ideally like to have algorithms for *joint* parameter estimation, and tracking and prediction of dynamical variables.

In this work, we address precisely the problem of joint model inference (i.e., parameter estimation) and filtering (i.e., tracking and prediction) for the broad class of dynamical systems that can be represented by state space models. State-space models are often used because they provide a general framework that describes the probabilistic dependence between the latent state (dynamical) variables, the unknown parameters and the observations. A typical state space model consists of:

- A random sequence of state vectors, $\boldsymbol{x}_t$, that contain the variables of interest for the description of the real-world system at hand but cannot be observed (at least completely).
- A random sequence of noisy observation vectors, $\boldsymbol{y}_t$, where each observation $\boldsymbol{y}_t$ can be related to the state $\boldsymbol{x}_t$ through some conditional probability distribution.
- A vector $\boldsymbol{\theta}$ of static model parameters that determine the model behavior and, typically, have to be estimated from the available data.

Many examples can be found in meteorology [22], oceanography [58] and climate modelling [27], where current models for global weather forecasting involve the tracking of millions of time-varying state variables.

This problem is not constrained to geophysics, though. In biochemistry it is often necessary to estimate the evolution of populations of different types of reacting molecules, which usually involves the estimation of the parameters that govern the interaction between them as well [41]. A similar problem needs to be solved in ecology, forecasting the populations of prey and predator species as they interact [14, 17]. In neuroscience we can also find problems that need state tracking and parameter estimation, such as the ones involving the FitzHugh–Nagumo model [51] (characterizing the functioning of an excitable system like a cell or a neuron) and the Hodgkin–Huxley model [19] (describing how action potentials in neurons are initiated and propagated). Additionally, we can find other similar examples in other fields such as quantitative finance and engineering. One of the most typical problems in finance is related to stochastic volatility models [3, 54, 98] that evaluate derivative securities such as option pricing, while target tracking [101] is a classical problem in engineering that has a wide range of applications such as surveillance, air traffic control, aerospace, robotics, remote sensing and computer vision.

1.2 State of the art

Traditionally, model calibration (i.e., the estimation of the model static parameters $\boldsymbol{\theta}$) and the tracking and forecasting of the time-varying state variables, $\boldsymbol{x}_t$, have been addressed separately. The problem of tracking the state of the system using sequentially-collected observations, $\boldsymbol{y}_t$, is often termed *data assimilation* in geophysics, while it is referred to as *stochastic* or *Bayesian* filtering by researchers in computational statistics and applied probability. Classical filtering methods [5, 31, 35, 42, 88, 90], including both Kalman-based algorithms and Monte Carlo schemes (particle filters [31, 33, 35, 42]) tackle the problem of predicting and tracking the states $\boldsymbol{x}_t$ using the observations $\boldsymbol{y}_t$, while assuming that the parameters $\boldsymbol{\theta}$ are given. This is hardly ever the case in practice, though, and the fixed parameters $\boldsymbol{\theta}$ have to be estimated from the data $\boldsymbol{y}_t$ as well. The joint tracking of $\boldsymbol{x}_t$ and estimation of $\boldsymbol{\theta}$ involves several practical and theoretical difficulties.

Many procedures have been suggested over the years (see, e.g., [62, 2, 37, 20], as well as [53] for a survey), however they are subject to problems related to observability (i.e., ambiguities), lack of performance guarantees or prohibitive computational cost. Some of the most relevant techniques can be classified in one or more of the categories below.

- *State augmentation methods with artificial dynamics.* The state vector, which contains the dynamical variables $\boldsymbol{x}_t$ that describe the physical system, is extended with any static unknown $\boldsymbol{\theta}$ (commonly reinterpreted as slowly changing dynamical variables) in the model [8, 46, 56, 62, 65, 105]. Standard filtering (or data assimilation) techniques are then used in order to track and forecast the extended state vector. Therefore, this methodology can be applied with Kalman-like methods (either extended [65] or sigma-point-based [46] approximations) as well as particle filters (PFs) [56, 62, 64]. In the case of PFs, artificial dynamics are usually introduced for the fixed parameters, reinterpreting them as slow-changing dynamical variables in order to avoid the degeneracy of the Monte Carlo approximation. Both with Kalman and particle filtering techniques, state augmentation is easy to apply but the resulting methods are often inefficient and lack theoretical guarantees.

- *Particle learning techniques.* For some models, the posterior probability distribution of the static parameters $\boldsymbol{\theta}$, conditional on the system states $\boldsymbol{x}_0, \dots, \boldsymbol{x}_t$, can be computed in closed form and it depends only on a set of finite-dimensional statistics [20, 32, 79, 93]. In a Monte Carlo setting, e.g., for particle filters, this means that the static parameters can be efficiently represented by sampling. Then, the classical state-augmentation can be replaced by a two-stage procedure where one first samples the posterior of the parameters and then the states (conditional on the parameters). Unfortunately, this approach is restricted to very specific models (an attempt to extend this idea to a more general setting can be found in [30]). The term particle learning was coined in [20], although the fundamental ideas were introduced earlier [32, 93].

- *Classical importance resampling methods.* Several authors have studied the performance of classical sequential importance sampling for static parameters [81, 83, 84]. Unfortunately, such algorithms tend to degenerate quickly over time unless certain conditions are met by the prior and posterior distributions [83, 84] or computationally heavy interpolation schemes are adopted for the static parameters [81].

- *Recursive maximum likelihood (RML) methods.* As an alternative to the previous algorithms, the RML methods [8, 9, 29, 53, 94] enable the sequential processing of the observed data as they are collected in order to approximate the posterior probability distribution of the parameters and the states. These techniques are well-principled and can be applied to a broad class of models. However, they do not yield full posterior distributions of the unknowns. They only output point estimates instead. Therefore, it is not possible to quantify the uncertainty of the estimates or forecasts. Moreover, they are subject to various convergence (and complexity) issues, e.g., when the posterior probability distribution is multimodal, when it contains singularities or when the parameter likelihoods cannot be computed exactly.

Only in the last few years there have been advances leading to methods that aim at calculating the *full* posterior probability distribution of all the unknown variables and parameters of the model. They are well-principled probabilistic methods that solve the joint problem numerically and supported by rigorous performance analyses [7, 21, 25, 68, 75]. From the viewpoint of Bayesian analysis, these conditional, or *posterior*, distributions contain all the information relevant for the estimation task. From them, one can compute point estimates of the parameters and states but also quantify the estimation error. Some examples are the sequential Monte Carlo square (SMC^2) [21], the particle Markov chain Monte Carlo (PMCMC) [7] and the nested particle filter (NPF) [25] methods. However, both SMC^2 and PMCMC are batch (non recursive) techniques. In other words, every time a new observation arrives, the whole sequence of observations may have to to be re-processed from scratch in order to update the estimates, leading to a quadratic increase of the computational effort over time. As an alternative, NPFs [25] apply the same principles as SMC^2 in a recursive way. It is a scheme with two intertwined layers of Monte Carlo methods, one inside the other, using the "inner" layer to track the dynamic state variables $\boldsymbol{x}_t$ and the "outer" layer for parameter estimation. However, since the NPF uses Monte Carlo in both layers of filters, its computational cost becomes prohibitive in high-dimensional problems.

Recently, other algorithms with nested, or layered, structures (in the vein of SMC^2 or NPF) have been proposed in order to address inference in high-dimensional, complex models. The most recent examples are the space-time particle filter (ST-PF) [15] and the nested sequential Monte Carlo (NSMC) method [78]. Both methods are intended to outperform classical sequential Monte Carlo (SMC) in high dimensional systems. They rely on spatial structures within the state space (a Markov random field in [78] and an auto-regressive structure in [15]) and, therefore, they may be useful to tackle multiple spatial scales. However, these algorithms only address the state tracking assuming that the parameters $\boldsymbol{\theta}$ are known.

In the physics literature, approximation schemes have been proposed that exploit the conditional dependences between the static parameters and the dynamic state variables, in a way that resembles the SMC^2 or NPF schemes. The authors of [92] introduce a two-stage filter that alternates the estimation of static parameters (conditional on a fixed state estimate) and the tracking of the dynamic variables (conditional on a fixed estimate of the static parameters). Another alternating scheme, that combines Monte Carlo estimators with ensemble Kalman filters in order to handle the static parameters and dynamic states, can be found in [38].

In [12], an expectation-maximization (EM) algorithm is used to track a particle whose dynamics are governed by a hidden Markov model. The expectation step involves a (Monte Carlo based) forward-filtering, backward-smoothing step that is computationally heavy and prevents the online application of the method. The authors of [104] investigate a variational scheme (based on the Laplace integral approximation) for data assimilation (including state and parameter estimation) and illustrate it with applications to the Lorenz 63 and Lorenz 96 models in a low dimensional setting. The same task of data assimilation with parameter estimation is tackled in [50]. In this case, the estimation of the states and parameter is reduced to an optimization problem that can be solved via an adjoint method for the estimation of a Hessian matrix. The schemes in [12, 50, 104] require one to process the data in batches, rather than recursively, and hence they are not well suited for online implementations. A sequential method, based on variational Bayes techniques, that admits an online (recursive) implementation can be found in [100]. However, the latter contribution is limited to the estimation of the time-varying states and does not deal with unkown static parameters.

1.3 Contribution

In this thesis we propose a general probabilistic scheme to perform the joint task of parameter estimation and state tracking and forecasting. The methodology is Bayesian, i.e., it aims at the computation of the posterior probability distribution of the unknowns given the available data. It involves two layers of estimators, one for the static parameters and another one for the time-varying state variables. It can be interpreted that the state estimators and predictors are *nested*, or inserted, within a main algorithm that tackles the estimation of the parameters. The estimation of the static parameters and the dynamic variables is carried out in a purely sequential and recursive manner. This property makes the proposed methods well-suited for problems where long time series of data have to be handled.

It can be shown that a particular case of the proposed methodology is the NPF of [25], which relies on a sequential Monte Carlo sampler in the parameter space and bank of particle filters [35, 42] in the space of the dynamic variables. However, the key feature and advantage of the general scheme that we advocate here is the ability to combine different types of algorithms in the two layers of inference (parameters and dynamic variables). Any grid-based method (where the probability distribution of the static parameters is represented by a set of points in the parameter space) can be employed in the first layer, while the computationally-heavy particle filters in the second layer of the NPF can be replaced by simpler algorithms, easier to apply in practical problems.

In Chapter 3, we investigate the use of sequential Monte Carlo and quasi-Monte Carlo [40] techniques in the parameter estimation layer. We note that the quasi-Monte Carlo scheme is a *deterministic* technique, although it formally resembles the Monte Carlo approach (hence the name). For the second layer, we have assessed two Gaussian filters, namely the extended Kalman filter (EKF) and the ensemble Kalman filter (EnKF). These two types of Gaussian filters have been well-studied in the geophysics literature and there are a number of numerical techniques to ease their practical implementation for large-scale systems (e.g., covariance inflation [6, 60] or localization [49, 82, 99]).

Because of the flexibility to combine estimation techniques of different types within the same overall scheme, we refer to the resulting algorithms in general as nested hybrid filters (NHFs). Besides the numerical examples, we also provide a theoretical analysis of the asymptotic convergence of NHFs that use a sequential Monte Carlo scheme in the first layer (for the static parameters) and finite-variance estimators of the state variables in the second layer. Our analysis shows that the NHF can be biased if the filters in the second layer are so (as it is the case in general with approximate Gaussian filters). However, it also ensures that the approximate posterior distribution of the parameters generated by the NHF, consisting of N samples in the parameter space, converges to a well-defined limit distribution with rate $\mathcal{O}\left(N^{-\frac{1}{2}}\right)$ under mild assumptions.

To illustrate the performance of the methodology, we present the results of computer simulations with a stochastic two-scale Lorenz 96 model [11] with underlying chaotic dynamics. In meteorology, the Lorenz 96 model is commonly used as a benchmark system for data assimilation [59, 85] and parameter estimation techniques [43, 87] because it displays the basic physical features of atmospheric dynamics [11] (e.g., convection and sensitivity to perturbations) and its dimension (number of state variables) can be selected arbitrarily, so it is possible to make the system as high-dimensional as one wishes. We have implemented, and compared numerically, four NHFs that combine Monte Carlo, quasi-Monte Carlo, EKF and EnKF schemes in different ways. All the combinations that we have tried yield significant reductions of running times in comparison with the NPF for this model, without a significant loss of accuracy. We report simulation results for systems with up to 5,000 dynamical variables to track and forecast.

In Chapter 4, we extend the NHF methodology to enable the use of non-Monte Carlo schemes in both layers of the nested filtering procedure, resulting in a new set of algorithms that we refer to as nested Gaussian filters (NGFs). The new scheme, therefore, is a methodological extension of the algorithms in [23, 25, 91] that comprises a broad class of nested filters for which it is possible to use and combine Gaussian or particle filters in any of the two layers. The new algorithms remain purely recursive and yield numerical approximations of the posterior probability of the unknown state variables and parameters using the sequentially collected observations.

To be specific, in this chapter we explain in detail the use of a deterministic-sampling Gaussian approximation (such as the unscented Kalman filter (UKF) [90] or the cubature Kalman filter (CKF) [10]) in the outer layer of the nested filtering scheme. Either particle or Gaussian (Kalman-based) filters can be easily plugged into the inner layer (we implement extended Kalman filters in our experiments for simplicity). The key difficulty to be tackled when using non-Monte Carlo methods in the outer layer is to keep the algorithm recursive. This was achieved for the Monte Carlo methods in [25] and [91] using a “jittering” procedure that cannot be extended to Gaussian filters in a practical way. Instead, we place a condition on the update of the filter in the outer layer that depends on a distance defined on the parameter space. When the distance between consecutive parameter updates falls below a prescribed threshold the algorithm operates in a purely recursive manner. This approach can work adequately when the posterior probability distributions of the state variables are continuous with respect to (w.r.t.) the unknown parameters, and we prove that this is the case under regularity assumptions on the state-space model.

In order to assess the performance of the proposed nested methods we have implemented a recursive scheme that employs a UKF in the outer layer (for parameter estimation) and a bank of EKFs in the inner layer (for state tracking). We have carried out a simulation study to compare the performance of this algorithm with two state-augmented Gaussian filters (a UKF and an EnKF [36]) as well as another nested algorithm that combines a particle filter in the outer layer with EKFs in the inner layer [91]. The methods are applied in two examples: the first one consists in tracking a stochastic Lorenz 63 model with three unknown parameters in the state equation, while in the second one we perform inference on a stochastic volatility model using real-world time-series data (namely, euro-to-USD exchange rates between 2014 and 2016).

In Chapter 5, we introduce a further generalization of the NHF methodology aimed at performing recursive Bayesian inference for a class of heterogeneous multi-scale state-space models [1] that can be numerically approximated with a micro-macro solver [96, 102]. We analyze the case of a Lorenz 96 system that displays three time scales (static parameters, slow dynamic state variables at the macro-scale and fast dynamic state variables at the micro-scale), but the methodology works in the same way for more general examples (namely, systems with n scales either in time or space).

The new scheme can be described as a three-layer nested smoother that approximates, in a recursive manner, the posterior probability distributions of the parameters and the two sets of state variables given the sequence of available observations. Specifically, we approximate the posterior probability distribution of the parameters in a first layer of computation, the posterior probability distribution of the slow state variables in a second layer, and the posterior probability distribution of the fast state variables in a third layer. The computations in the second layer are conditional on the candidate parameter values generated on the first layer, while the calculations on the third layer are conditional on the candidates drawn at the first and second layers. The inference techniques used in each layer can vary, leading to different computational costs and degrees of accuracy. As examples that illustrate the methodology, we propose two methods. The first one uses SMC algorithms in the first and second layers, intertwined with a UKF [90] in the third layer. Similarly, the second method uses a SMC algorithm in the first layer, but incorporates the use of EnKFs [36] and an EKFs in the second and third layers of the scheme, respectively.

1.4 Organization of the thesis

The rest of the thesis is organized as follows. After a brief comment on notation, we describe in Chapter 2 the standard filtering techniques (mainly Kalman-based filters in Section 2.2 and Monte Carlo filters in Section 2.3) that are used in the layers of the proposed methodology. In addition, we include in Section 2.4 the description of the NPF, since it can be considered as a specific configuration of the methodology explained in this thesis.

In Chapter 3, NHFs are introduced. To be specific, they are derived in Section 3.2, followed by the discussion of an asymptotic convergence theorem which is stated in Section 3.3. The Lorenz 96 model, which is described in Section 3.4, is used in the simulations presented in the numerical results in Section 3.5.

In Chapter 4, we introduce the NGFs. We describe the class of state-space models with unknown parameters to be studied in Section 4.1. Then, we describe the family of nested Gaussian filters with sigma-point approximations in the outer layer in Section 4.2. In Sections 4.3 and 4.4, numerical results are presented.

Multi-scale nested filters are introduced in Chapter 5. In Section 5.1, we describe the class of heterogeneous multi-scale state-space models. Then, in Section 5.2 we define the optimal smoother for multi-scale systems with static parameters and two sets of dynamic state variables. In Section 5.3, we describe two specific methods derived from the general methodology and, finally in Section 5.4 we illustrate the numerical results for the stochastic two-scale Lorenz 96 model.

Finally, Chapter 6 is devoted to the conclusions, including a brief summary of the obtained results and an outlook on possible future work.

1.5 Notation

We denote vectors and matrices by bold-face letters, either lower-case (for vectors) or upper-case (for matrices). Scalar magnitudes are denoted using regular-face letters. For example, $d \in \mathbb{N}$ and $x \in \mathbb{R}$ are scalars, $\boldsymbol{x} \in \mathbb{R}^d$ is a vector and $\boldsymbol{X} \in \mathbb{R}^{d\times d}$ is a matrix.

Most of the magnitudes of interest in this paper are random vectors (r.v.'s). If $\boldsymbol{x}$ is a d-dimensional r.v. taking values in $\mathbb{R}^d$, we use the generic notation $p(\boldsymbol{x})$ for its probability density function (pdf). This is an argument-wise notation. If we have two r.v.'s, $\boldsymbol{x}$ and $\boldsymbol{y}$, we write $p(\boldsymbol{x})$ and $p(\boldsymbol{y})$ for their respective pdfs, which are possibly different. In a similar way, $p(\boldsymbol{x}, \boldsymbol{y})$ denotes the joint pdf of the two r.v.'s and $p(\boldsymbol{x}|\boldsymbol{y})$ denotes the conditional pdf of $\boldsymbol{x}$ given $\boldsymbol{y}$. We find this simple notation convenient for the presentation of the model and methods and introduce a more specific terminology only for the analysis of convergence. We assume, for simplicity, that all random magnitudes can be described by pdfs with respect to the Lebesgue measure. Notation $\boldsymbol{x} \sim p(\boldsymbol{x})$ is read as "the r.v. $\boldsymbol{x}$ is distributed according to the pdf $p(\boldsymbol{x})$".

We also resort to a more specific notation for Gaussian pdfs. If $\boldsymbol{x}$ is a d-dimensional Gaussian random vector with mean $\bar{\boldsymbol{x}}$ and covariance matrix $\boldsymbol{C} > 0$ then we can explicitly write the pdf $p(\boldsymbol{x})$ as

$$\mathcal{N}(\boldsymbol{x}|\bar{\boldsymbol{x}}, \boldsymbol{C}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{C}|^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2} (\boldsymbol{x} - \bar{\boldsymbol{x}})^\top \boldsymbol{C}^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}}) \right\},$$

where the superscript $^\top$ indicates transposition and $|\boldsymbol{C}|$ denotes the determinant of matrix $\boldsymbol{C}$.

2 Filtering Techniques

This chapter is devoted to the standard filtering techniques that set the background of this thesis. First, we outline a description of the state-space models we are interested in, followed by an introduction to the family of Gaussian filters including the extended Kalman filter (EKF) [5], the ensemble Kalman filter (EnKF) [36] and the unscented Kalman filter (UKF) [90]. Also, we introduce the Monte Carlo filters that, with the Gaussian filters, are going to be the basis of the new algorithms. Finally, we also describe the nested particle filter (NPF) which can be considered one specific configuration of the methodology introduced in Chapters 3 and 4. Table 2.1 summarises the notation in this chapter.

2.1 State-space models

We are interested in the class of Markov state-space dynamical systems with additive noise that can be described by the pair of equations

$$\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) + \boldsymbol{v}_t, \tag{2.1}$$

$$\boldsymbol{y}_t = \boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{\theta}) + \boldsymbol{r}_t, \tag{2.2}$$

where

- $t \in \mathbb{N}$ denotes discrete time,
- the vector $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ is the d_x-dimensional system state,
- the vector $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$ is the observation at time t,
- $\boldsymbol{v}_t$ and $\boldsymbol{r}_t$, which are zero-mean random variables (r.v.s) with covariances $\boldsymbol{V}_t$ and $\boldsymbol{R}_t$ respectively, are the state and observations noises,

- the functions $\boldsymbol{f}\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \longrightarrow \mathbb{R}^{d_x}$ and $\boldsymbol{g}\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \longrightarrow \mathbb{R}^{d_y}$, $d_x \geq d_y$, are possibly nonlinear and
- $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ is a (random but fixed) vector of unknown parameters that parameterize functions $\boldsymbol{f}$ and $\boldsymbol{g}$.

Table 2.1: Summary of notation for Chapter 2.

$\boldsymbol{C}_0$	Initial state covariance matrix
$\boldsymbol{C}_{t\|t-1}$	Predictive or *a priori* state covariance matrix
$\boldsymbol{C}_{t\|t}$	Updated or *a posteriori* state covariance matrix
$\boldsymbol{C}_t^{\boldsymbol{y}}$	Observation covariance matrix
$\boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}}$	Cross-covariance matrix
d_θ, d_x, d_y	Parameter, state and observation dimension, respectively
$\boldsymbol{f}(\cdot)$	State transition function
$\boldsymbol{F}$	State transition matrix
$\boldsymbol{g}(\cdot)$	Observation function
$\boldsymbol{G}$	Observation matrix
$\boldsymbol{J}_{\boldsymbol{f},\boldsymbol{x}'}$	Jacobian matrix of the state transition function $\boldsymbol{f}$ evaluated in $\boldsymbol{x}'$
$\boldsymbol{J}_{\boldsymbol{g},\boldsymbol{x}'}$	Jacobian matrix of the observation function $\boldsymbol{g}$ evaluated in $\boldsymbol{x}'$
$\boldsymbol{K}_t$	Kalman gain at time t
M	Number of sigma-points (in UKF), ensemble members (in EnKF) or particles (in particle filter (PF) and NPF)
N	Number of particles in the first layer of the NPF
$\boldsymbol{r}_t$	Noise observation vector
t	Discrete time
u_t^i	Normalised weights of the first layer (in NPF)
$\boldsymbol{v}_t$ and $\boldsymbol{v}_t^i$	Noise state vector
w^i	Weights of the sigma-points (in UKF)
w_t^i	Normalised weights in PF
$w_t^{i,j}$	Normalised weights of the second layer (in NPF)
$\boldsymbol{x}_t$	State vector at time t
$\hat{\boldsymbol{x}}_0$	Initial mean state vector
$\hat{\boldsymbol{x}}_{t\|t-1}$	Predictive (*a priori*) mean state vector
$\hat{\boldsymbol{x}}_{t\|t}$	Updated (*a posteriori*) mean state vector
$\boldsymbol{x}_{t\|t-1}^i$ and $\boldsymbol{x}_{t\|t-1}^{i,j}$	Predictive state samples or particles (in EnKF, PF and NPF)
$\boldsymbol{x}_{t\|t}^i$ and $\boldsymbol{x}_{t\|t}^{i,j}$	Updated state samples or particles (in EnKF, PF and NPF)
$\boldsymbol{X}_{t\|t-1}$ and $\boldsymbol{X}_{t\|t}$	Predictive and updated ensemble (in EnKF), respectively
$\boldsymbol{y}_t$	Observation vector at time t
$\hat{\boldsymbol{y}}_t$	Mean observation vector (in UKF and EnKF)
γ	Parameter for the configuration of sigma-points (in UKF)
$\boldsymbol{\theta}$	Parameter vector
$\boldsymbol{\theta}_t^i$	i-th parameter sample or particle (in NPF)
κ_N	Kernel in the jittering step (in NPF)

The system of Eqs. (2.1) and (2.2) can be described in terms of a set of relevant probability density functions (pdfs), specifically

$$\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \tag{2.3}$$

$$\boldsymbol{\theta} \sim p(\boldsymbol{\theta}), \tag{2.4}$$

$$\boldsymbol{x}_t \sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \tag{2.5}$$

$$\boldsymbol{y}_t \sim p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}), \tag{2.6}$$

where $p(\boldsymbol{\theta})$ and $p(\boldsymbol{x}_0)$ are the a *priori* pdfs of the parameters and the state, respectively, $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ is the conditional density of the state $\boldsymbol{x}_t$ given $\boldsymbol{x}_{t-1}$ and the parameter vector $\boldsymbol{\theta}$, and $p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta})$ is the conditional pdf of the observation $\boldsymbol{y}_t$ given $\boldsymbol{x}_t$ and $\boldsymbol{\theta}$. We assume that $\boldsymbol{y}_t$ is conditionally independent of all other observations (given $\boldsymbol{x}_t$ and $\boldsymbol{\theta}$) and the prior pdfs of the state, $p(\boldsymbol{x}_0)$, and the parameters, $p(\boldsymbol{\theta})$, are known and the corresponding probability distributions are independent, i.e., the joint prior pdf of $\boldsymbol{x}_0$ and $\boldsymbol{\theta}$ factorizes as $p(\boldsymbol{x}_0, \boldsymbol{\theta}) = p(\boldsymbol{x}_0)p(\boldsymbol{\theta})$.

2.2 Kalman filters

2.2.1 Standard Kalman filter (KF)

The Kalman filter (KF) is a recursive algorithm that estimates the state variables of a dynamical system given a sequence of noisy observations. To be specific, the KF provides the optimal Bayesian estimator of the state, i.e., the exact pdfs $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})$ and $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$, as long as the dynamical system is linear and the noises in Eqs. (2.1) and (2.2) are Gaussian. Both pdfs are Gaussian, so they are completely characterized by the mean and the covariance matrix, i.e. $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) = \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t|t-1}, \boldsymbol{C}_{t|t-1})$ and $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t|t}, \boldsymbol{C}_{t|t})$. Therefore, the KF computes recursively the posterior mean and covariance matrix of the state.

The general state-space model of Eqs. (2.1) and (2.2) reduces, in the linear-Gaussian case, to

$$\boldsymbol{x}_t = \boldsymbol{F}\boldsymbol{x}_{t-1} + \boldsymbol{v}_t, \qquad \text{where} \quad \boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{v}_t|\boldsymbol{0}, \boldsymbol{V}_t), \tag{2.7}$$

$$\boldsymbol{y}_t = \boldsymbol{G}\boldsymbol{x}_t + \boldsymbol{r}_t, \qquad \text{where} \quad \boldsymbol{r}_t \sim \mathcal{N}(\boldsymbol{r}_t|\boldsymbol{0}, \boldsymbol{R}_t), \tag{2.8}$$

where $\boldsymbol{F}$ is a $d_x \times d_x$ transition matrix and $\boldsymbol{G}$ is a $d_y \times d_x$ observation matrix. Note that we ignore the parameter vector $\boldsymbol{\theta}$ for simplicity.

Figure 2.1 summarizes the functioning of this algorithm, that can be divided in two main phases: prediction and update. In the prediction phase, the *a priori* probability distribution of the state $\boldsymbol{x}_t$, which is Gaussian and hence represented by its mean $\hat{\boldsymbol{x}}_{t|t-1}$ and covariance matrix $\boldsymbol{C}_{t|t-1}$, is computed. The predictive estimates are obtained by propagating the previous estimation through the transition matrix $\boldsymbol{F}$. Once there is a new noisy observation $\boldsymbol{y}_t$ available, the update phase takes place. The previous mean and covariance matrix are updated using the Kalman gain and the innovation $\boldsymbol{y}_t - \boldsymbol{G}\hat{\boldsymbol{x}}_{t|t-1}$. Hence, we obtain the updated or filtered distribution, represented by the mean $\hat{\boldsymbol{x}}_{t|t}$ and the covariance matrix $\boldsymbol{C}_{t|t}$.

Although the original KF is designed to deal with linear dynamical systems, several extensions and generalizations have been developed to work with more complex nonlinear systems. Hereafter, we summarize some of the most common variants: the EKF, the EnKF and the UKF.

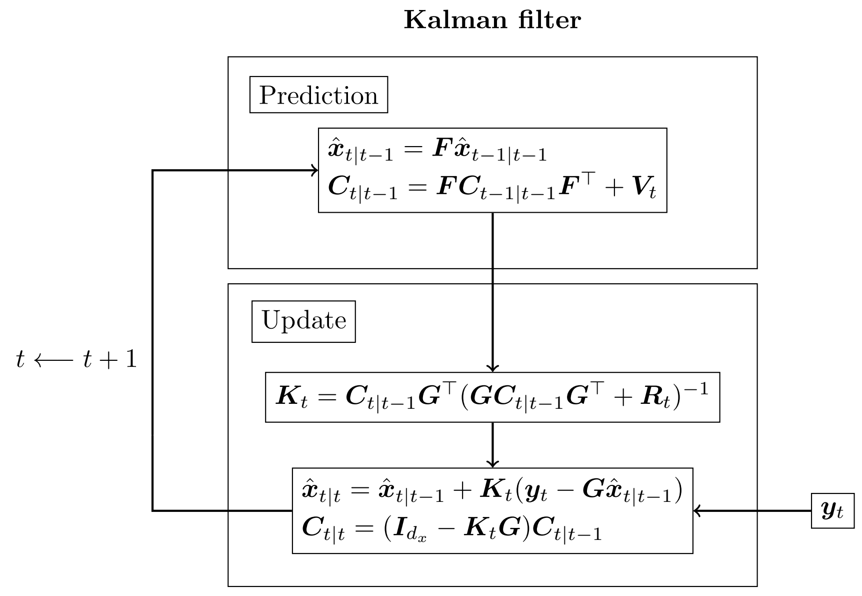

Kalman filter
Prediction
$\hat{\boldsymbol{x}}_{t|t-1} = \boldsymbol{F}\hat{\boldsymbol{x}}_{t-1|t-1}$
$\boldsymbol{C}_{t|t-1} = \boldsymbol{F}\boldsymbol{C}_{t-1|t-1}\boldsymbol{F}^\top + \boldsymbol{V}_t$
Update
$t \longleftarrow t+1$
$\boldsymbol{K}_t = \boldsymbol{C}_{t|t-1}\boldsymbol{G}^\top(\boldsymbol{G}\boldsymbol{C}_{t|t-1}\boldsymbol{G}^\top + \boldsymbol{R}_t)^{-1}$
$\hat{\boldsymbol{x}}_{t|t} = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{y}_t - \boldsymbol{G}\hat{\boldsymbol{x}}_{t|t-1})$
$\boldsymbol{C}_{t|t} = (\boldsymbol{I}_{d_x} - \boldsymbol{K}_t\boldsymbol{G})\boldsymbol{C}_{t|t-1}$
$\boldsymbol{y}_t$

Figure 2.1: Schematic description of the Kalman filter.

2.2.2 Extended Kalman filter (EKF)

The EKF was proposed as an extension to the original Kalman filter in order to work with nonlinear systems [5, 103]. This algorithm is not an optimal filter since a process of linearization is applied around the current state. However, it provides reasonable performance depending on the specific problem. It has been shown useful in applications such as radar [55], target tracking [70], state estimation of battery charging [47] and econometrics [72].

To be specific, this linearization is applied by using the Jacobian of the nonlinear state and observation functions ($\boldsymbol{f}$ and $\boldsymbol{g}$) in the Kalman filter Eqs. (2.1) and (2.2) at each time step. These Jacobians replace the state and observation transition matrices and are constructed as

$$\boldsymbol{J}_{\boldsymbol{f},\boldsymbol{x}'} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_{d_x}} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_{d_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{d_x}}{\partial x_1} & \frac{\partial f_{d_x}}{\partial x_2} & \cdots & \frac{\partial f_{d_x}}{\partial x_{d_x}} \end{bmatrix} \quad \text{for} \quad \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_{d_x}(\boldsymbol{x}) \end{bmatrix}, \tag{2.9}$$

$$\boldsymbol{J}_{\boldsymbol{g},\boldsymbol{x}'} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_{d_x}} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_{d_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{d_y}}{\partial x_1} & \frac{\partial g_{d_y}}{\partial x_2} & \cdots & \frac{\partial g_{d_y}}{\partial x_{d_x}} \end{bmatrix} \quad \text{for} \quad \boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} g_1(\boldsymbol{x}) \\ g_2(\boldsymbol{x}) \\ \vdots \\ g_{d_y}(\boldsymbol{x}) \end{bmatrix}, \tag{2.10}$$

where $\boldsymbol{J}_{\boldsymbol{f},\boldsymbol{x}'}$ is the $d_x \times d_x$ Jacobian matrix of function $\boldsymbol{f}$ and $\boldsymbol{J}_{\boldsymbol{g},\boldsymbol{x}'}$ is the $d_y \times d_x$ Jacobian matrix of function $\boldsymbol{g}$, both evaluated at the current state $\boldsymbol{x}'$.

Algorithm 1 summarizes the EKF for the filtering of the state $\boldsymbol{x}_t$ (i.e., given known parameters $\boldsymbol{\theta}$). First, in step 1 we initialize the state from the prior pdf $p(\boldsymbol{x}_0)$. We assume that this pdf is Gaussian with mean $\hat{\boldsymbol{x}}_0$ and covariance matrix $\boldsymbol{C}_0$. For $t \geq 1$, the prediction and update phases are computed recursively.

The prediction is described in step 2a, where the *a priori* mean $\hat{\boldsymbol{x}}_{t|t-1}$ is computed by propagating the previous state estimation through the transition function $\boldsymbol{f}$ and the predictive covariance matrix $\boldsymbol{C}_{t|t-1}$ is calculated using the Jacobian $\boldsymbol{J}_{\boldsymbol{f},\hat{x}_{t-1|t-1}}$ as described in Eq. (2.9). The Kalman gain is approximated in step 2(b)i using the predictive covariance matrix $\boldsymbol{C}_{t|t-1}$, the Jacobian $\boldsymbol{J}_{\boldsymbol{g},\hat{x}_{t|t-1}}$ of Eq. (2.10) and the observation covariance matrix $\boldsymbol{R}_t$. Finally, when a new observation $\boldsymbol{y}_t$ arrives, the updated mean $\hat{\boldsymbol{x}}_{t|t}$ and the updated covariance matrix $\boldsymbol{C}_{t|t}$ are computed in step 2(b)ii. As a result, we approximate the posterior pdf $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t},\boldsymbol{\theta})$ as Gaussian, namely, $\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{x}}_{t|t},\boldsymbol{C}_{t|t})$.

Algorithm 1 *Extended Kalman filter*

Inputs:

- *A priori pdf* $p(\boldsymbol{x}_0)$.
- *Known parameters* $\boldsymbol{\theta}$.

Procedure:

1. *Initialization* ($\boldsymbol{x}_0$)

   Assume $p(\boldsymbol{x}_0)$ *is Gaussian with mean* $\hat{\boldsymbol{x}}_0$ *and covariance matrix* $\boldsymbol{C}_0$*, i.e.,* $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{x}}_0, \boldsymbol{C}_0)$.

2. *Recursive step*

   (a) *Prediction step:*

   Compute the predictive mean $\hat{\boldsymbol{x}}_{t|t-1}$ *and the predictive covariance matrix* $\boldsymbol{C}_{t|t-1}$*:*

$$\hat{\boldsymbol{x}}_{t|t-1} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1|t-1},\boldsymbol{\theta}), \tag{2.11}$$

$$\boldsymbol{C}_{t|t-1} = \boldsymbol{J}_{\boldsymbol{f},\hat{\boldsymbol{x}}_{t-1|t-1}}\boldsymbol{C}_{t-1|t-1}\boldsymbol{J}^{\top}_{\boldsymbol{f},\hat{\boldsymbol{x}}_{t-1|t-1}} + \boldsymbol{V}_t. \tag{2.12}$$

   (b) *Update step:*

   i. *Compute the Kalman Gain as*

$$\boldsymbol{K}_t = \boldsymbol{C}_{t|t-1}\boldsymbol{J}^{\top}_{\boldsymbol{g},\hat{\boldsymbol{x}}_{t|t-1}}\left(\boldsymbol{J}_{\boldsymbol{g},\hat{\boldsymbol{x}}_{t|t-1}}\boldsymbol{C}_{t|t-1}\boldsymbol{J}^{\top}_{\boldsymbol{g},\hat{\boldsymbol{x}}_{t|t-1}} + \boldsymbol{R}_t\right)^{-1}. \tag{2.13}$$

   ii. *Compute the updated mean* $\hat{\boldsymbol{x}}_{t|t}$ *and the updated covariance matrix* $\boldsymbol{C}_{t|t}$*:*

$$\hat{\boldsymbol{x}}_{t|t} = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{y}_t - \boldsymbol{g}(\hat{\boldsymbol{x}}_{t|t-1},\boldsymbol{\theta})), \tag{2.14}$$

$$\boldsymbol{C}_{t|t} = (\boldsymbol{I}_{d_x} - \boldsymbol{K}_t\boldsymbol{J}_{\boldsymbol{g},\hat{\boldsymbol{x}}_{t|t-1}})\boldsymbol{C}_{t|t-1}, \tag{2.15}$$

   where $\boldsymbol{I}_{d_x}$ *is* d_x*-dimensional identity matrix.*

Outputs: $\hat{\boldsymbol{x}}_{t|t}$ *and* $\boldsymbol{C}_{t|t}$.

2.2.3 Unscented Kalman filter (UKF)

The UKF [52, 71] is a suboptimal filter that has been proposed as an improvement of the EKF. This method uses a deterministic sampling technique known as the unscented transform (UT) [90] in order to approximate the predicted and updated posterior pdfs of the state. The UT is a mathematical device that can be used to approximate the integrals of a nonlinear function $h(\boldsymbol{x})$ with respect to (w.r.t.) a Gaussian distribution, that takes the form

$$\int h(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x},$$

where $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{x}}, \boldsymbol{C})$, $\boldsymbol{x} \in \mathbb{R}^{d_x}$, and the first and second order moments of the Gaussian vector $\boldsymbol{x}$ can be characterised by a collection of (deterministic) sigma-points in the state space, with associated weights, denoted $\{\boldsymbol{x}^i, w^i\}_{0\leq i\leq M-1}$, for $M = 2d_x + 1$. This yields to the approximation

$$\int h(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} \approx \sum_{i=0}^{M-1} w^i h(\boldsymbol{x}^i) \tag{2.16}$$

for any integrable test function $h(\cdot)$.

There are different ways in which the weighted sigma-points can be selected (see [71] for a survey). One of the most common ways to select the set $\{\boldsymbol{x}^i, w^i\}_{0\leq i\leq M-1}$ from a pdf $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{x}}, \boldsymbol{C})$ is [71]

$$\boldsymbol{x}^0 = \hat{\boldsymbol{x}}, \qquad w^0 = \frac{\gamma}{\gamma + d_x}, \tag{2.17}$$

$$\boldsymbol{x}^j = \hat{\boldsymbol{x}} + (\sqrt{(d_x+\gamma)\boldsymbol{C}})_j, \qquad w^j = \frac{1}{2(d_x+\gamma)}, \tag{2.18}$$

$$\boldsymbol{x}^{j+d_x} = \hat{\boldsymbol{x}} + (\sqrt{(d_x+\gamma)\boldsymbol{C}})_j, \qquad w^{j+d_x} = \frac{1}{2(d_x+\gamma)}, \tag{2.19}$$

where $\gamma > -d_x$ is a tunning parameter, $j = 1, \ldots, d_x$, and $(\sqrt{(d_x+\gamma)\boldsymbol{C}})_j$ is the j-th column of the matrix square-root of $(d_x+\gamma)\boldsymbol{C}$. The UKF is the scheme that results from the application of the UT to approximate the predictive and/or the updated mean and covariance in a nonlinear state-space model. Hence, this method is derivative-free (i.e., no need to compute Jacobians) and can be used with systems described by non-differentiable functions. Moreover, the computational complexity of this method is $\mathcal{O}(d_x^3)$, that is the same of the EKF [5, 103].

Algorithm 2 summarizes the UKF for state tracking given known parameters $\boldsymbol{\theta}$. First, in step 1 we initialize the state from the prior pdf $p(\boldsymbol{x}_0)$ generating a set of sigma-points and weights $\{\boldsymbol{x}_0^i, w^i\}_{0\leq i\leq M-1}$ as in Eqs. (2.17)-(2.19). We assume this pdf is Gaussian with mean $\hat{\boldsymbol{x}}_0$ and covariance matrix $\boldsymbol{C}_0$. For $t \geq 1$, the prediction and update phases are computed recursively. The prediction is described in steps 2(a)i and 2(a)ii, where the sigma-points $\{\boldsymbol{x}_0^i\}_{0\leq i\leq M-1}$ are propagated through the transition function $\boldsymbol{f}$ and the *a priori* mean $\hat{\boldsymbol{x}}_{t|t-1}$ and covariance matrix $\boldsymbol{C}_{t|t-1}$ are computed. The previous sigma-points are also propagated through the observation function $\boldsymbol{g}$ in order to calculate the empirical mean $\hat{\boldsymbol{y}}_t$ and covariance matrix $\boldsymbol{C}_t^{\boldsymbol{y}}$ in steps 2(b)i and 2(b)ii. Then, the cross-covariance matrix $\boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}}$ is computed in step 2(b)iii in order to approximate the Kalman gain $\boldsymbol{K}_t$ in step 2(b)iv. Finally, when a new observation $\boldsymbol{y}_t$ arrives, the updated mean $\hat{\boldsymbol{x}}_{t|t}$ and the updated covariance matrix $\boldsymbol{C}_{t|t}$ are computed in step 2(b)v. The approximate posterior distribution $\mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t|t}, \boldsymbol{C}_{t|t})$ is used to provide a new set of sigma-points $\{\boldsymbol{x}_{t|t}^i\}_{0\leq i\leq M-1}$ in step 2(b)vi.

Algorithm 2 *Unscented Kalman filter (UKF)*

```
Inputs:
- A priori pdf $p(\boldsymbol{x}_0)$ and known parameters $\boldsymbol{\theta}$.
Procedure:
1. Initialization. Draw a set of $M = 2d_x + 1$ points and weights $\{\boldsymbol{x}_0^i, w^i\}_{0\leq i\leq M-1}$ as in Eqs. (2.17)–(2.19).
2. Recursive step
    (a) Prediction step:
        i. For $i = 0, \dots, M-1$, propagate the sigma-points through the state equation $\boldsymbol{x}_{t|t-1}^i = \boldsymbol{f}(\boldsymbol{x}_{t-1|t-1}^i, \boldsymbol{\theta})$.
        ii. Compute the predictive mean $\hat{\boldsymbol{x}}_{t|t-1}$ and the predictive covariance matrix $\boldsymbol{C}_{t|t-1}$:
            $\hat{\boldsymbol{x}}_{t|t-1} = \sum_{i=0}^{M-1} w^i \boldsymbol{x}_{t|t-1}^i,$   (2.20)
            $\boldsymbol{C}_{t|t-1} = \sum_{i=0}^{M-1} w^i (\boldsymbol{x}_{t|t-1}^i - \hat{\boldsymbol{x}}_{t|t-1})(\boldsymbol{x}_{t|t-1}^i - \hat{\boldsymbol{x}}_{t|t-1})^\top + \boldsymbol{V}_t.$   (2.21)
    (b) Update step:
        i. For $i = 0, \dots, M-1$, propagate the sigma-points through the observation function
            $\boldsymbol{y}_t^i = \boldsymbol{g}(\boldsymbol{x}_{t|t-1}^i, \boldsymbol{\theta}).$   (2.22)
        ii. Compute the empirical mean $\hat{\boldsymbol{y}}_t$ and covariance matrix $\boldsymbol{C}_t^{\boldsymbol{y}}$ of the transformed points:
            $\hat{\boldsymbol{y}}_t = \sum_{i=0}^{M-1} w^i \boldsymbol{y}_t^i,$   (2.23)
            $\boldsymbol{C}_t^{\boldsymbol{y}} = \sum_{i=0}^{M-1} w^i (\boldsymbol{y}_t^i - \hat{\boldsymbol{y}}_t)(\boldsymbol{y}_t^i - \hat{\boldsymbol{y}}_t)^\top + \boldsymbol{R}_t.$   (2.24)
        iii. Additionally, compute the cross covariance matrix $\boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}}$ as
            $\boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}} = \sum_{i=0}^{M-1} w^i (\boldsymbol{x}_{t|t-1}^i - \hat{\boldsymbol{x}}_{t|t-1})(\boldsymbol{y}_t^i - \hat{\boldsymbol{y}}_t)^\top.$   (2.25)
        iv. Compute the Kalman Gain as $\boldsymbol{K}_t = \boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}} \left(\boldsymbol{C}_t^{\boldsymbol{y}}\right)^{-1}$.
        v. Compute the updated mean $\hat{\boldsymbol{x}}_{t|t}$ and the updated covariance matrix $\boldsymbol{C}_{t|t}$:
            $\hat{\boldsymbol{x}}_{t|t} = \boldsymbol{x}_{t|t-1} + \boldsymbol{K}_t(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t),$   (2.26)
            $\boldsymbol{C}_{t|t} = \boldsymbol{C}_{t|t-1} - \boldsymbol{K}_t \boldsymbol{C}_t^{\boldsymbol{y}} \boldsymbol{K}_t^\top.$   (2.27)
        vi. Calculate new sigma-points $\{\boldsymbol{x}_{t|t}^i\}_{0\leq i\leq M-1}$ from $\mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t|t}, \boldsymbol{C}_{t|t})$, as in Eqs. (2.17)–(2.19).
Outputs: $\hat{\boldsymbol{x}}_{t|t}$, $\boldsymbol{C}_{t|t}$ and the set of sigma-points $\{\boldsymbol{x}_{t|t}^i\}_{0\leq i\leq M-1}$.
```

2.2.4 Ensemble Kalman filter (EnKF)

The EnKF [36] is another version of the Kalman filter that was originated to deal with nonlinear high-dimensional systems. While in problems with a large number of variables other nonlinear Kalman filters struggle with evolving the covariance matrix of the state, in the EnKF this matrix is not propagated explicitly. Instead, this information is encoded in an ensemble of state samples, i.e., a collection of state vectors that represents the distribution of the state $\boldsymbol{x}$. The ensemble is denoted as

$$\boldsymbol{X}_t = \left[\boldsymbol{x}_t^1 \ldots, \boldsymbol{x}_t^M\right] = \left[\boldsymbol{x}_t^i\right]. \tag{2.28}$$

where $\boldsymbol{X}_t$ is an $d_x \times M$ matrix whose columns are the ensemble members for $i = 1, \ldots, M$.

Similar to the UKF, the ensemble is used to approximate integrals of a function $h(\boldsymbol{x})$ w.r.t. the posterior probability distribution of the state vector $\boldsymbol{x}_t$. For example, if $\boldsymbol{X}_{t|t-1} = \left[\boldsymbol{x}_{t|t-1}^i\right]$ denotes the ensemble for the one-step-ahead predictive pdf $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})$ then we can approximate

$$\int h(\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})d\boldsymbol{x} \approx \frac{1}{M}\sum_{i=1}^{M} h(\boldsymbol{x}_{t|t-1}^i) \tag{2.29}$$

for any integrable test function $h(\cdot)$.

Algorithm 3 summarizes the EnKF for the state tracking of the dynamical system assuming the parameter vector $\boldsymbol{\theta}$ is known. First, we initialize the state from the prior pdf $p(\boldsymbol{x}_0)$ in step 1, generating the samples $\{\boldsymbol{x}_0^i\}_{1\leq i\leq M}$ that form the ensemble $\boldsymbol{X}_0$. For $t \geq 1$, the prediction and update phases are computed in a recursive manner. The prediction is described in steps 2(a)i and 2(a)ii, where the samples $\{\boldsymbol{x}_0^i\}_{1\leq i\leq M}$ are propagated through the state Eq. (2.1) and the predictive mean $\hat{\boldsymbol{x}}_{t|t-1}$ and predictive covariance matrix $\boldsymbol{C}_{t|t-1}$ are computed[1]. The previous ensemble $\boldsymbol{X}_{t|t-1}$ is also propagated through the observation function $\boldsymbol{g}$ in order to calculate the empirical mean $\hat{\boldsymbol{y}}_t$ and covariance matrix $\boldsymbol{C}_t^{\boldsymbol{y}}$ in steps 2(b)i and 2(b)ii. Then, the cross-covariance matrix $\boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}}$ is approximated in step 2(b)iii in order to compute the Kalman gain $\boldsymbol{K}_t$ in step 2(b)iv. Finally, with a new observation $\boldsymbol{y}_t$, the ensemble can be updated as well as the mean $\hat{\boldsymbol{x}}_{t|t}$ and the covariance matrix $\boldsymbol{C}_{t|t}$ in steps 2(b)v and 2(b)vi. These yield an approximation of the posterior density $p(\boldsymbol{x}_{t|t}|\boldsymbol{y}_t, \boldsymbol{\theta})$ as a Gaussian pdf $\mathcal{N}(\boldsymbol{x}_{t|t}|\hat{\boldsymbol{x}}_{t|t}, \boldsymbol{C}_{t|t})$.

The use of this ensemble of samples in the EnKF makes this method better-suited for high-dimensional nonlinear systems (i.e., $d_x >> 1$) than the UKF and the EKF. However, when the dimension of the observation d_y is also very large, the computation of the inverse matrix $\left(\boldsymbol{C}_t^{\boldsymbol{y}}\right)^{-1}$ in Eq. (2.39) becomes a bottleneck. In order to alleviate this operation there is an alternative formula to compute this inverse that turns out advantageous when the dimension of the observation d_y is large compared to the number of ensemble members (i.e., $d_y >> M$) but the inversion of the observation noise covariance matrix $\boldsymbol{R}_t$ is computationally cheap to obtain (or it has to be done just once, if $\boldsymbol{R}_t = \boldsymbol{R}$). Using the Sherman–Morrison–Woodbury formula [66] we can rewrite the Kalman gain in Eq. (2.39) as

$$\boldsymbol{K}_t = \boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}}\big(\boldsymbol{R}_t^{-1} - \boldsymbol{R}_t^{-1}\frac{1}{M}\tilde{\boldsymbol{Y}}_{t|t-1}\big(\boldsymbol{I}_M + \tilde{\boldsymbol{Y}}_{t|t-1}^\top\boldsymbol{R}_t^{-1}\frac{1}{M}\tilde{\boldsymbol{Y}}_{t|t-1}\big)^{-1}\tilde{\boldsymbol{Y}}_{t|t-1}^\top\boldsymbol{R}_t^{-1}\big), \tag{2.31}$$

where $\tilde{\boldsymbol{Y}}_{t|t-1} = \boldsymbol{Y}_{t|t-1} - \hat{\boldsymbol{y}}_t\mathbb{1}_M^\top$ and $\boldsymbol{I}_M$ is a M-dimensional identity matrix.

[1]Note that $\hat{\boldsymbol{x}}_{t|t-1}$ and $\boldsymbol{C}_{t|t-1}$ are the ensemble mean and covariance matrix respectively, i.e.,

$$\hat{\boldsymbol{x}}_{t|t-1} = \frac{1}{M}\sum_{i=1}^{M}\boldsymbol{x}_{t|t-1}^i \quad \text{and} \quad \boldsymbol{C}_{t|t-1} = \frac{1}{M-1}\sum_{i=1}^{M}(\boldsymbol{x}_{t|t-1}^i - \hat{\boldsymbol{x}}_{t|t-1})(\boldsymbol{x}_{t|t-1}^i - \hat{\boldsymbol{x}}_{t|t-1})^\top. \tag{2.30}$$

Algorithm 3 *Ensemble Kalman filter (EnKF)*

Inputs:

- *A priori pdf* $p(\boldsymbol{x}_0)$.
- *Known parameters* $\boldsymbol{\theta}$.
- *The number of ensemble members* M.

Procedure:

1. *Initialization*

   Draw a set of M *ensemble members from the prior* $p(\boldsymbol{x}_0)$ *to form the ensemble* $\boldsymbol{X}_0 = \left[\boldsymbol{x}_0^i\right]$ *as in Eq.* (2.28)*, for* $1 \leq i \leq M$.

2. *Recursive step*

   (a) *Prediction step:*

   i. *For* $i = 1, \ldots, N$*, propagate the ensemble members through the transition function*

   $$\boldsymbol{x}_{t|t-1}^i = \boldsymbol{f}(\boldsymbol{x}_{t-1|t-1}^i, \boldsymbol{\theta}) + \boldsymbol{v}_t^i \quad \text{for} \quad \boldsymbol{v}_t^i \sim \mathcal{N}(\boldsymbol{v}_t|\boldsymbol{0}, \boldsymbol{V}_t), \tag{2.32}$$

   in order to obtain the predictive ensemble $\boldsymbol{X}_{t|t-1} = \left[\boldsymbol{x}_{t|t-1}^i\right]$*, for* $1 \leq i \leq M$.

   ii. *Compute the predictive mean* $\hat{\boldsymbol{x}}_{t|t-1}$ *and the predictive covariance matrix* $\boldsymbol{C}_{t|t-1}$*:*

   $$\hat{\boldsymbol{x}}_{t|t-1} = \frac{1}{M}\boldsymbol{X}_{t|t-1}\mathbb{1}_M, \tag{2.33}$$

   $$\boldsymbol{C}_{t|t-1} = \frac{1}{M-1}\tilde{\boldsymbol{X}}_{t|t-1}\tilde{\boldsymbol{X}}_{t|t-1}^\top, \tag{2.34}$$

   where $\mathbb{1}_M = [1, \ldots, 1]^\top$ *is an* M*-dimensional column vector and* $\tilde{\boldsymbol{X}}_{t|t-1} = \boldsymbol{X}_{t|t-1} - \hat{\boldsymbol{x}}_{t|t-1}\mathbb{1}_M^\top$.

   (b) *Update step:*

   i. *For* $i = 1, \ldots, M$*, propagate the ensemble members through the observation function*

   $$\boldsymbol{y}_{t|t-1}^i = \boldsymbol{g}(\boldsymbol{x}_{t|t-1}^i, \boldsymbol{\theta}) \tag{2.35}$$

   in order to obtain the predictive ensemble $\mathbf{Y}_{t|t-1} = \left[\boldsymbol{y}_{t|t-1}^i\right]$*, for* $1 \leq i \leq M$.

   ii. *Compute the mean* $\hat{\boldsymbol{y}}_t$ *and covariance matrix* $\boldsymbol{C}_t^{\boldsymbol{y}}$*:*

   $$\hat{\boldsymbol{y}}_t = \frac{1}{M}\mathbf{Y}_{t|t-1}\mathbb{1}_M, \tag{2.36}$$

   $$\boldsymbol{C}_t^{\boldsymbol{y}} = \frac{1}{M-1}\tilde{\boldsymbol{Y}}_{t|t-1}\tilde{\boldsymbol{Y}}_{t|t-1}^\top + \boldsymbol{R}_t, \tag{2.37}$$

   where $\tilde{\boldsymbol{Y}}_{t|t-1} = \mathbf{Y}_{t|t-1} - \hat{\boldsymbol{y}}_t\mathbb{1}_M^\top$.

   iii. *Additionally, compute the cross covariance matrix* $\boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}}$ *as*

   $$\boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}} = \frac{1}{M-1}\tilde{\boldsymbol{X}}_{t|t-1}\tilde{\boldsymbol{Y}}_{t|t-1}^\top. \tag{2.38}$$

iv. Compute the Kalman Gain as

$$\boldsymbol{K}_t = \boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}} \left(\boldsymbol{C}_t^{\boldsymbol{y}}\right)^{-1}. \tag{2.39}$$

v. With a new available observation $\boldsymbol{y}_t$, compute the updated ensemble $\boldsymbol{X}_{t|t}$

$$\boldsymbol{X}_{t|t} = \boldsymbol{X}_{t|t-1} + \boldsymbol{K}_t(\check{\boldsymbol{Y}}_t - \mathbf{Y}_{t|t-1}), \tag{2.40}$$

where $\check{\boldsymbol{Y}}_t = \left[\check{\boldsymbol{y}}_t^i\right]$, for $1 \le i \le M$, and

$$\check{\boldsymbol{y}}_t^i = \boldsymbol{y}_t + \boldsymbol{r}_t^i \quad \textit{for} \quad \boldsymbol{r}_t^i \sim \mathcal{N}(\boldsymbol{r}_t|\mathbf{0}, \boldsymbol{R}_t). \tag{2.41}$$

vi. Compute the updated mean $\hat{\boldsymbol{x}}_{t|t}$ and the updated covariance matrix $\boldsymbol{C}_{t|t}$:

$$\hat{\boldsymbol{x}}_{t|t} = \frac{1}{M}\boldsymbol{X}_{t|t}\mathbb{1}_M, \tag{2.42}$$

$$\boldsymbol{C}_{t|t} = \frac{1}{M-1}\tilde{\boldsymbol{X}}_{t|t}\tilde{\boldsymbol{X}}_{t|t}^\top, \tag{2.43}$$

where $\tilde{\boldsymbol{X}}_{t|t} = \boldsymbol{X}_{t|t} - \hat{\boldsymbol{x}}_{t|t}\mathbb{1}_M^\top$.

Outputs: *$\hat{\boldsymbol{x}}_{t|t}$, $\boldsymbol{C}_{t|t}$ and the ensemble $\boldsymbol{X}_{t|t} = \left[\boldsymbol{x}_{t|t}^i\right]$, for $1 \le i \le M$.*

The EnKF can be seen as a sequential Monte Carlo (SMC) method (in the vein of the algorithms to be described in Section 2.3 below) that targets only the first ($\hat{\boldsymbol{x}}_{t|t}$) and second ($\boldsymbol{C}_{t|t}$) order moments of the posterior pdf. It is particularly efficient when the observation Eq. (2.2) is linear, as the computations reduce to a (nearly) classical Kalman update for the members of the ensemble. This is outlined in Fig. 2.2.

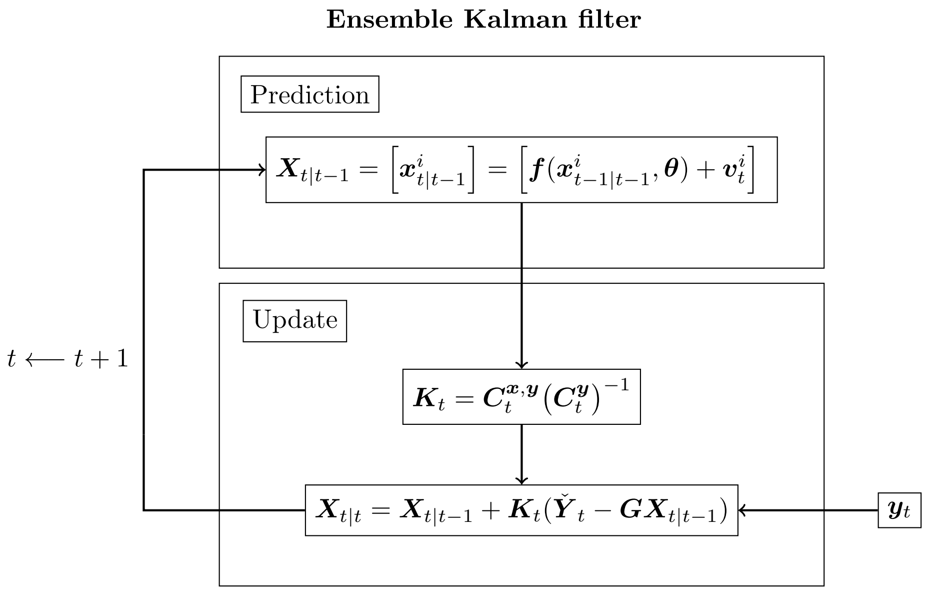

Ensemble Kalman filter
Prediction
$\boldsymbol{X}_{t|t-1} = \left[\boldsymbol{x}_{t|t-1}^i\right] = \left[\boldsymbol{f}(\boldsymbol{x}_{t-1|t-1}^i, \boldsymbol{\theta}) + \boldsymbol{v}_t^i\right]$
Update
$t \longleftarrow t+1$
$\boldsymbol{K}_t = \boldsymbol{C}_t^{\boldsymbol{x},\boldsymbol{y}}\left(\boldsymbol{C}_t^{\boldsymbol{y}}\right)^{-1}$
$\boldsymbol{X}_{t|t} = \boldsymbol{X}_{t|t-1} + \boldsymbol{K}_t(\check{\boldsymbol{Y}}_t - \boldsymbol{G}\boldsymbol{X}_{t|t-1})$
$\boldsymbol{y}_t$

Figure 2.2: Schematic description of the EnKF.

2.3 Sequential Monte Carlo methods

Sequential Monte Carlo methods or PFs are a collection of Monte Carlo algorithms that, based on Bayesian theory, aim at computing recursively the posterior probability distributions generated by the state space model [31, 33, 35, 42]. In particular, they solve the filtering problem by approximating the posterior probability distributions with discrete random measures composed of samples of the unknown states $\{\boldsymbol{x}_t^i\}_{1\leq i\leq M}$ (called *particles*) and associated weights $\{w_t^i\}_{1\leq i\leq M}$.

The set of particles and weights, that resembles the sigma-points of the UKF and the ensemble of the EnKF, is used to approximate integrals of a function $h(\boldsymbol{x})$ w.r.t. the posterior probability of $\boldsymbol{x}_t$, namely

$$\mathbb{E}[h(\boldsymbol{x}_t)|\boldsymbol{y}_{1:t}] \approx \sum_{i=1}^{M} w_t^i h(\boldsymbol{x}_t^i) \tag{2.44}$$

This approximation converges with rate $M^{-\frac{1}{2}}$ for a broad class of integrable test functions $h(\cdot)$. For example, at time t and with the weighted particle set $\{\boldsymbol{x}_{t|t}^i, w_t^i\}_{1\leq i\leq M}$, we can compute the estimates

$$\hat{\boldsymbol{x}}_{t|t} = \sum_{i=1}^{M} w_t^i \boldsymbol{x}_{t|t}^i, \tag{2.45}$$

$$\boldsymbol{C}_{t|t} = \sum_{i=1}^{M} w_t^i (\boldsymbol{x}_{t|t}^i - \hat{\boldsymbol{x}}_{t|t})(\boldsymbol{x}_{t|t}^i - \hat{\boldsymbol{x}}_{t|t})^\top, \tag{2.46}$$

where $\hat{\boldsymbol{x}}_{t|t}$ is the mean and $\boldsymbol{C}_{t|t}$ is the covariance matrix of $\boldsymbol{x}_t$ given $\boldsymbol{y}_{1:t}$.

Although the extensions of the Kalman filter described in Section 2.2 are designed for nonlinear state-space models and even some of these methods use samples or sigma-points to integrate w.r.t. a pdf, they make the assumption that all posterior probability distributions are Gaussian[2] and so restrict the computations to the mean and covariance of $\boldsymbol{x}_t$. Particle filters are an alternative to these algorithms, avoiding linearizations around current estimates and providing accurate performance in nonlinear and non-Gaussian problems. Also, there are many variations of this methodology that have been proposed to deal with different types of problems, such as the auxiliary particle filter (APF) [86], the sequential importance resampling (SIR) algorithm [63] and the Bao-Blackwellised PF [35]. Moreover, PFs offer other advantages such as their easy implementation and the possibility of parallelization [24, 76, 97]. Its computational cost is the main disadvantage, though. It is often necessary to generate a large number of particles M in order to obtain accurate results (specially for high-dimensional systems).

In Algorithm 4 we describe the bootstrap filter [42] (see also [34]) for the state estimation of the dynamical system (again, assuming known parameters $\boldsymbol{\theta}$). First, the state is initialized from the prior pdf $p(\boldsymbol{x}_0)$ in step 1, drawing the samples $\{\boldsymbol{x}_0^i\}_{1\leq i\leq M}$. For $t \geq 1$, the filter works recursively using importance sampling and resampling. The particles are propagated through the state Eq. (2.1) in step 2a, to yield a set o predictive samples $\{\boldsymbol{x}_{t|t-1}^i\}_{1\leq i\leq M}$. With a new observation $\boldsymbol{y}_t$ available in step 2b, we can compute the pdf $p(\boldsymbol{y}_t|\boldsymbol{x}_{t|t-1}^i)$ as well as the weights $\tilde{w}_t^i$. Finally, with the normalized weights w_t^i, the set $\{\boldsymbol{x}_{t|t-1}^i\}_{1\leq i\leq M}$ can be replaced by $\{\boldsymbol{x}_{t|t}^i\}_{1\leq i\leq M}$ in step 2c by resampling [61]. Figure 2.3 shows a schematic of the bootstrap filtering algorithm.

[2]Or can be well approximated by Gaussians, at any rate.

Algorithm 4 *Bootstrap filter*

```
***Inputs:***
- A priori pdf $p(\boldsymbol{x}_0)$.
- Known parameters $\boldsymbol{\theta}$.
- The number of particles $M$.
***Procedure:***
1. Initialization
   Draw a set of $M$ particles $\{\boldsymbol{x}_0^i\}_{1\leq i\leq M}$ from the prior $p(\boldsymbol{x}_0)$.
2. Recursive step
   (a) Draw $M$ particles $\boldsymbol{x}_{t|t-1}^i \sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1|t-1}^i)$, i.e., propagate the particles through the state Eq.
          $\boldsymbol{x}_{t|t-1}^i = \boldsymbol{f}(\boldsymbol{x}_{t-1|t-1}^i, \boldsymbol{\theta}) + \boldsymbol{v}_t^i,$   (2.47)
       for $i = 1, \ldots, M$.
   (b) For $i = 1, \ldots, M$, with a new observation $\boldsymbol{y}_t$, compute the weights
          $\tilde{w}_t^i \propto p(\boldsymbol{y}_t|\boldsymbol{x}_{t|t-1}^i),$   (2.48)
       and normalize them as
          $w_t^i = \frac{\tilde{w}_t^i}{\sum_{i=1}^M \tilde{w}_t^i}.$   (2.49)
   (c) Resampling: set $\boldsymbol{x}_{t|t}^m = \boldsymbol{x}_{t|t-1}^i$ with probability $w_t^i$ for each $m = 1, \ldots, M$,
***Outputs:*** $\{\boldsymbol{x}_{t|t}^i\}_{1\leq i\leq M}$.
```

2.4 Nested particle filter (NPF)

The NPF [25] aims at approximating the posterior probability distribution of the static parameters $\boldsymbol{\theta}$ and the state variables $\boldsymbol{x}_t$ based on the PF of Section 2.3. Different from the previous sections in this chapter, the fixed parameters are unknown and the approximation of the posterior $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ is the main goal of this algorithm.

Figure 2.4 displays a schematic of the NPF. This method has two layers of particle filters, or SMC methods, one inside the other forming a nested structure. The algorithm works recursively, approximating the posterior probability distribution of the parameters $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:n})$ in the first layer, and the posterior probability distribution of the dynamical state variables conditional on the parameters $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}, \boldsymbol{\theta})$ in the second layer.

Algorithm 5 describes the NPF for the joint estimation of the static parameters $\boldsymbol{\theta}$ and the dynamical state variables $\boldsymbol{x}_t$. First, we initialize the filter in step 1 drawing N parameter samples or *particles* from the

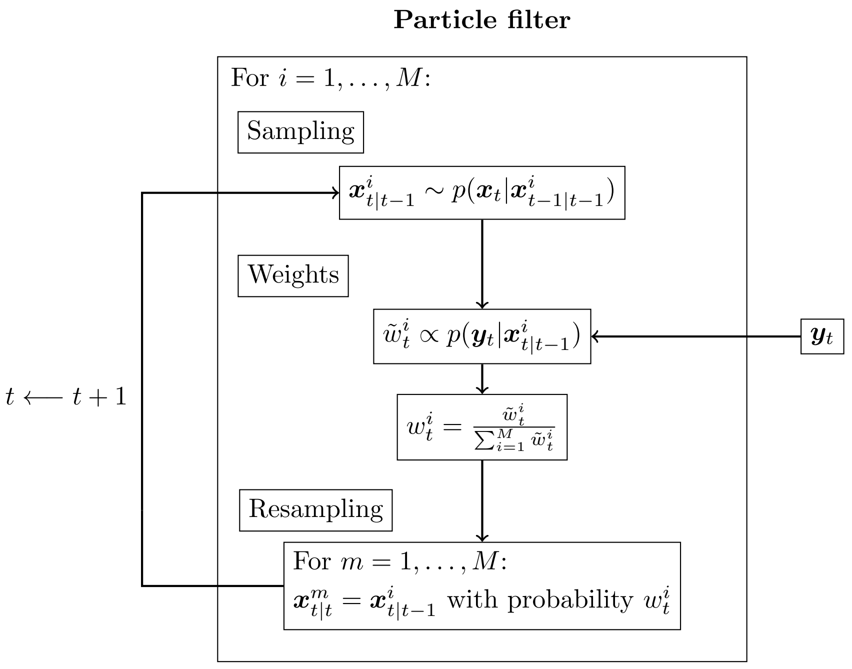

Particle filter
For $i = 1, \dots, M$:
Sampling
$\boldsymbol{x}^i_{t|t-1} \sim p(\boldsymbol{x}_t|\boldsymbol{x}^i_{t-1|t-1})$
Weights
$\tilde{w}^i_t \propto p(\boldsymbol{y}_t|\boldsymbol{x}^i_{t|t-1})$
$\boldsymbol{y}_t$
$w^i_t = \frac{\tilde{w}^i_t}{\sum_{i=1}^M \tilde{w}^i_t}$
$t \longleftarrow t+1$
Resampling
For $m = 1, \dots, M$:
$\boldsymbol{x}^m_{t|t} = \boldsymbol{x}^i_{t|t-1}$ with probability w^i_t

Figure 2.3: Schematic description of the PF.

prior pdf $p(\boldsymbol{\theta})$ and $N \times M$ state samples from the prior pdf $p(\boldsymbol{x}_0)$ to form the set $\{\boldsymbol{\theta}^i_0, \{\boldsymbol{x}^{i,j}_0\}_{1\le j\le M}\}_{1\le i\le N}$. For $t \geq 1$ and for $i = 1, \dots, N$, we perform a jittering (step 2(b)i) where particles $\tilde{\boldsymbol{\theta}}^i_t$ are drawn from the kernel $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}^i_{t-1})$. With a new observation $\boldsymbol{y}_t$ (step 2(b)ii), we can compute the pdf $p(\boldsymbol{y}_t|\boldsymbol{x}^{i,j}_{t|t-1}, \tilde{\boldsymbol{\theta}}^i_t)$ and approximate the weights of the second layer $\tilde{w}^{i,j}_t$. Also, we can approximate the pdf $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \tilde{\boldsymbol{\theta}}^i_t)$ and hence the weights of the first layer $\tilde{u}^i_t$. After normalizing the weights in step 2(b)iii , we can use them in the resampling step in order to obtain the set of state particles $\{\boldsymbol{x}^{i,j}_{t|t}\}_{1\le j\le M}$ (step 2(b)iv). Finally, in the first layer of the filter, the normalised weights $\{u^i_t\}_{1\le i\le N}$ are obtained in step 2c. We resample the parameter particles as well as the sets of state particles attached to them, obtaining the whole set $\{\boldsymbol{\theta}^i_0, \{\boldsymbol{x}^{i,j}_{t|t}\}_{1\le j\le M}\}_{1\le i\le N}$ in step 2d.

A key ingredient of the NPF is the jittering step 2(b)i.As opposed to the state $\boldsymbol{x}_t$, the parameters $\boldsymbol{\theta}$ does not evolve through a transition function and are not directly observed either. A naive approach to the approximation of $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ by particle filtering would be to actually fix the particles on the parameter space by making $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1}$. This leads to degeneracy, because the diversity of the values of the particles is drastically reduced after a few resampling steps. In order to avoid this poor approach, some artificial dynamics (or *jittering*) are introduced in the parameter $\boldsymbol{\theta}$. At every time step we can generate new particles $\{\tilde{\boldsymbol{\theta}}^i_t \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}^i_{t-1})\}_{1\le i\le N}$, where $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}')$ is a Markov kernel, i.e., a probability distribution for $\boldsymbol{\theta}$ conditional on $\boldsymbol{\theta}'$. With this kernel we can jitter a few particles of the set with an arbitrary variance or jitter all of them with a controlled variance. The selection of the kernel and the variance must be done carefully [25]. Using N and M particles in the first and second layers respectively, we can obtain a convergence error of $\mathcal{O}(N^{-\frac{1}{2}} \vee M^{-\frac{1}{2}})$ [25].

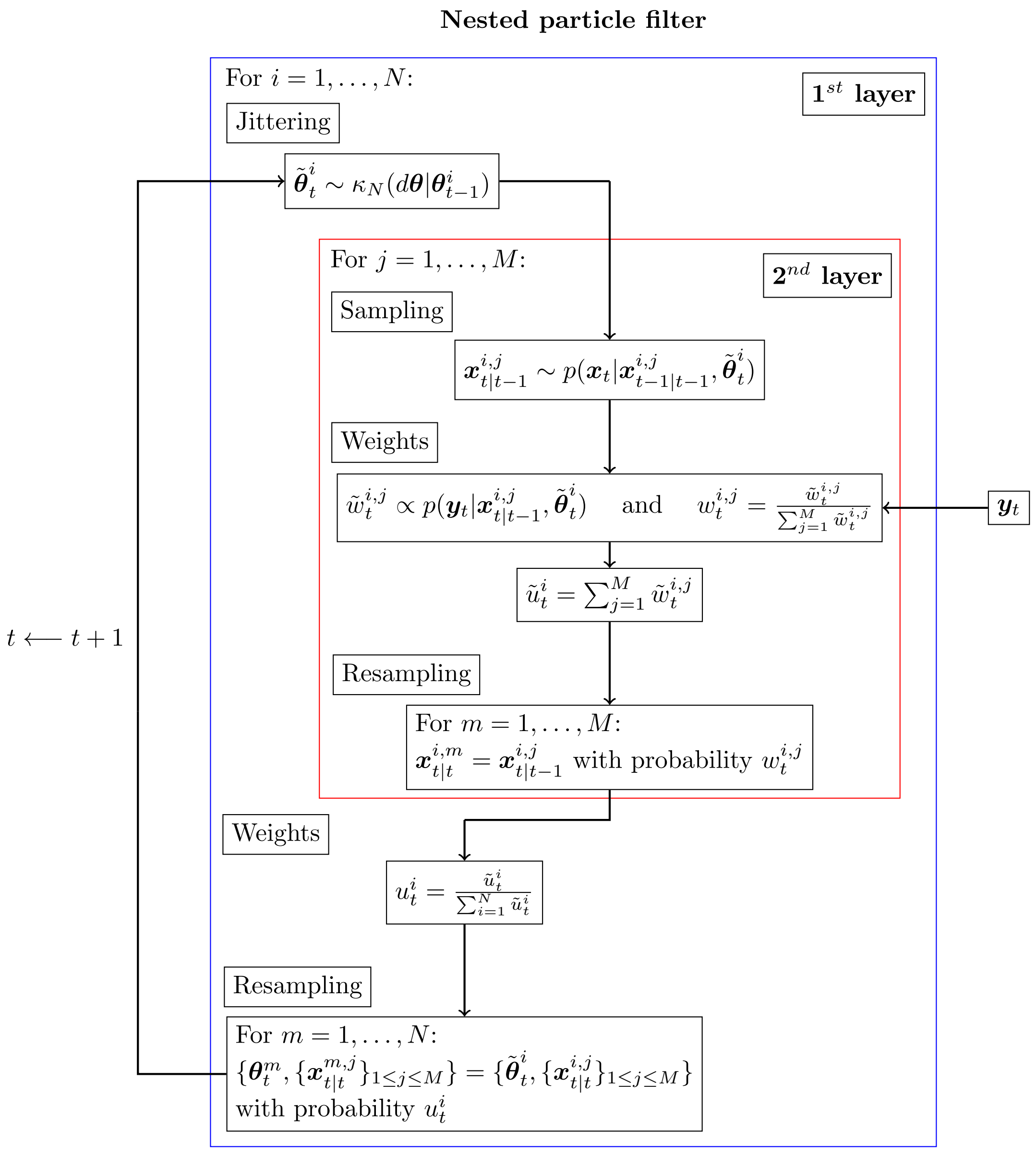

Nested particle filter
For $i = 1, \dots, N$:
$\mathbf{1}^{st}$ layer
Jittering
$\tilde{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i)$
For $j = 1, \dots, M$:
$\mathbf{2}^{nd}$ layer
Sampling
$\boldsymbol{x}_{t|t-1}^{i,j} \sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1|t-1}^{i,j}, \tilde{\boldsymbol{\theta}}_t^i)$
Weights
$\tilde{w}_t^{i,j} \propto p(\boldsymbol{y}_t|\boldsymbol{x}_{t|t-1}^{i,j}, \tilde{\boldsymbol{\theta}}_t^i)$ and $w_t^{i,j} = \frac{\tilde{w}_t^{i,j}}{\sum_{j=1}^M \tilde{w}_t^{i,j}}$
$\boldsymbol{y}_t$
$\tilde{u}_t^i = \sum_{j=1}^M \tilde{w}_t^{i,j}$
$t \longleftarrow t+1$
Resampling
For $m = 1, \dots, M$:
$\boldsymbol{x}_{t|t}^{i,m} = \boldsymbol{x}_{t|t-1}^{i,j}$ with probability $w_t^{i,j}$
Weights
$u_t^i = \frac{\tilde{u}_t^i}{\sum_{i=1}^N \tilde{u}_t^i}$
Resampling
For $m = 1, \dots, N$:
$\{\boldsymbol{\theta}_t^m, \{\boldsymbol{x}_{t|t}^{m,j}\}_{1 \le j \le M}\} = \{\tilde{\boldsymbol{\theta}}_t^i, \{\boldsymbol{x}_{t|t}^{i,j}\}_{1 \le j \le M}\}$
with probability u_t^i

Figure 2.4: Schematic description of the NPF.

Algorithm 5 *Nested particle filter (NPF)*

```
Inputs:
  - A priori pdfs $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{\theta}_0)$.
  - The number of particles $N$ (for the parameters) and $M$ (for the state).
Procedure:
  1. Initialization.
     Draw a set of $N$ particles $\{\boldsymbol{\theta}_0^i\}_{1\le i\le N}$ from the prior $p(\boldsymbol{\theta})$, and per each one of these particles, draw a set of $M$ particles $\{\boldsymbol{x}_0^{i,j}\}_{1\le j\le M}$ from the prior $p(\boldsymbol{x}_0)$. This creates the whole set $\{\boldsymbol{\theta}_0^i, \{\boldsymbol{x}_0^{i,j}\}_{1\le j\le M}\}_{1\le i\le N}$.
  2. Recursive step
     (a) Draw $\tilde{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i)$ for $i = 1, \ldots, N$, where $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}')$ is a Markov kernel.
     (b) For $i = 1, \ldots, N$:
         i. Draw $M$ particles $\boldsymbol{x}_{t|t-1}^{i,j} \sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1|t-1}^{i,j}, \tilde{\boldsymbol{\theta}}_t^i)$, i.e., for $j = 1, \ldots, M$ propagate the particles through the transition function
            $$\boldsymbol{x}_{t|t-1}^{i,j} = \boldsymbol{f}(\boldsymbol{x}_{t-1|t-1}^{i,j}, \tilde{\boldsymbol{\theta}}_t^i) + \boldsymbol{v}_t^{i,j}. \tag{2.50}$$
         ii. For $j = 1, \ldots, M$, with a new observation $\boldsymbol{y}_t$, compute the weights $\tilde{w}_t^{i,j}$ and the approximate likelihood $\tilde{u}_t^i$ as
            $$\tilde{w}_t^{i,j} \propto p(\boldsymbol{y}_t|\boldsymbol{x}_{t|t-1}^{i,j}, \tilde{\boldsymbol{\theta}}_t^i) \quad \text{and} \tag{2.51}$$
            $$\tilde{u}_t^i = \sum_{j=1}^{M} \tilde{w}_t^{i,j} \propto p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \tilde{\boldsymbol{\theta}}_t^i). \tag{2.52}$$
         iii. Normalize weights as
            $$w_t^{i,j} = \frac{\tilde{w}_t^{i,j}}{\sum_{i=1}^{M} \tilde{w}_t^{i,j}}. \tag{2.53}$$
         iv. For each $m = 1, \ldots, M$, set $\boldsymbol{x}_{t|t}^{i,m} = \boldsymbol{x}_{t|t-1}^{i,j}$ with probability $w_t^{i,j}$.
     (c) Normalize the approximate likelihood
            $$u_t^i = \frac{\tilde{u}_t^i}{\sum_{i=1}^{N} \tilde{u}_t^i}, \quad \text{for } i = 1, \ldots, N. \tag{2.54}$$
     (d) For each $m = 1, \ldots, N$, set $\{\boldsymbol{\theta}_t^m, \{\boldsymbol{x}_{t|t}^{m,j}\}_{1\le j\le M}\} = \{\tilde{\boldsymbol{\theta}}_t^i, \{\boldsymbol{x}_{t|t}^{i,j}\}_{1\le j\le M}\}$ with probability $u_t^i$.
Outputs: The set $\{\boldsymbol{\theta}_t^i, \{\boldsymbol{x}_{t|t}^{i,j}\}_{1\le j\le M}\}_{1\le i\le N}$.
```

3 Nested Hybrid Filters

This chapter is dedicated to the description of a class of nested hybrid filters (NHFs), that estimate the posterior probability distribution of the unknown static parameters and the dynamical state variables of a state-space system by using a structure of two intertwined layers of filtering methods. To be specific, within this framework we introduce the use of different types of Monte Carlo methods, namely sequential Monte Carlo (SMC) and sequential quasi-Monte Carlo (SQMC) in the first layer of the algorithm, while non-Monte Carlo methods are applied in the second layer including extended Kalman filters (EKFs) and ensemble Kalman filters (EnKFs). A detailed description of the proposed methodology is outlined in Section 3.2, after the introduction to the class of (stochastic) dynamical systems of interest in Section 3.1. An asymptotic convergence theorem is stated and discussed in Section 3.3. In Section 3.4, the stochastic Lorenz 96 model which is used in the simulations is described, and some illustrative numerical results are presented in Section 3.5. Finally, the conclusions are stated in Section 3.6. Table 3.1 summarizes the notation of this chapter.

3.1 Dynamical model and problem statement

The goal is to design methods for the recursive estimation of both the static parameters $\boldsymbol{\theta}$ and the states $\boldsymbol{x}_t$, for $t \in \mathbb{N}$, of the state-space model described by Eqs. (2.3)-(2.6), reproduced below for convenience:

$$\begin{aligned} \boldsymbol{x}_0 &\sim p(\boldsymbol{x}_0), && (3.1)\\ \boldsymbol{\theta} &\sim p(\boldsymbol{\theta}), && (3.2)\\ \boldsymbol{x}_t &\sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}), && (3.3)\\ \boldsymbol{y}_t &\sim p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}). && (3.4) \end{aligned}$$

Table 3.1: Notation of Chapter 3.

a	Vector of coefficients that characterize the polynomial $\ell(x, \mathsf{a})$
B, C, F and H	Parameters of the Lorenz 96 model
d_v and d_w	State noise dimension (for $\boldsymbol{x}$ and $\boldsymbol{z}$ respectively)
d_x, d_y and d_θ	State, observation and parameter dimension
$\boldsymbol{f}(\cdot)$	State function (continuous time)
$\boldsymbol{F}(\cdot)$	State function (discrete time)
$\boldsymbol{F}^m(\cdot)$	State function $\boldsymbol{F}$ applied m consecutive times
$\boldsymbol{g}(\cdot)$	Observation function
$\mathsf{h}(\cdot)$	Inverse of Hilbert curve
n	Discrete-time step in the time scale of the states ($\tau = n\Delta$, $n \in \mathbb{N}$)
ℓ	Polynomial of degree 2 of x
m	Number of discrete-time steps between each new observation
M	Number of samples (or ensemble members) in the second layer
N	Number of particles in the first layer of the NHF
$\boldsymbol{P}_t^i$	Estimate of the posterior covariance matrix of the state $\boldsymbol{x}_t$ given the parameters $\boldsymbol{\theta}_t^i$
$\boldsymbol{P}_t^N$	Estimate of the posterior covariance matrix of the parameters
q	Order of the discretization method
$\boldsymbol{r}_t$	Observation noise at time step t
t	Discrete-time step in the time scale of the observations ($\tau = tm\Delta$, $t \in \mathbb{N}$)
u_t	Likelihood $p(\boldsymbol{y}_t \vert \boldsymbol{\theta}, \boldsymbol{y}_{1:t-1})$
$\bar{\boldsymbol{v}}_n$ and $\boldsymbol{v}_t$	State ($\boldsymbol{x}$) noise at discrete-time n and t, respectively
w_t^i	Normalised importance weights at time step t
$\bar{\boldsymbol{w}}_n$ and $\boldsymbol{w}_t$	State ($\boldsymbol{z}$) noise at discrete-time n and t, respectively
$\bar{\boldsymbol{x}}_n$ and $\boldsymbol{x}_t$	State vector at discrete-time n and t, respectively
$\boldsymbol{y}_t$	Observation vector at discrete-time t
$\bar{\boldsymbol{z}}_n$ and $\boldsymbol{z}_t$	Fast state vector at discrete-time n and t, respectively
$\delta_{\boldsymbol{\theta}'}$	Dirac delta allocating probability 1 at point $\boldsymbol{\theta}'$
Δ	Fixed integration step
$\boldsymbol{\theta}_t^i$	i-th particle or parameter sample at time step t
κ_N	Markov kernel of the jittering step
μ_t	Posterior probability distribution of the parameter vector $\boldsymbol{\theta}$ at time step t
μ_t^N	Monte Carlo approximation of μ_t
π_t	Joint probability distribution of $\boldsymbol{\theta}$ and $\boldsymbol{x}$
$\boldsymbol{\rho}_t^i$	i-th vector of quasi Monte Carlo (QMC) samples at time step t
σ_o	Scale parameter that controls the stochastic perturbations of the observation
τ	Continuous time
$\psi(\cdot)$	Discrepancy-preserving bijection map

In this model, $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{\theta})$ are, respectively, the *a priori* pdfs of the system state and the parameter vector at time $t = 0$ ($\tau = 0$ as well), $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ is the conditional pdf of $\boldsymbol{x}_t$ given the state $\boldsymbol{x}_{t-1}$ and the parameters in $\boldsymbol{\theta}$, and $p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta})$ is the conditional pdf of the observation given the state and the parameters.

We note that:

- The priors $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{\theta})$ can be understood as a probabilistic characterization of uncertainty regarding the system initial condition. If the initial condition $\boldsymbol{x}_0$ were known exactly, $p(\boldsymbol{x}_0)$ could be replaced by a Dirac delta allocating probability 1 at that point.

- The pdf $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ may not have, in general, a closed-form expression. However, it is usually straightforward to simulate $\boldsymbol{x}_t$ given $\boldsymbol{x}_{t-1}$ and $\boldsymbol{\theta}$ and this is sufficient for many methods to work.

- The observations are conditionally independent given the states and the parameters. If the observational noise $\boldsymbol{r}_t$ is Gaussian, then $p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{g}(\boldsymbol{x}_t, \boldsymbol{\theta}), \sigma_o^2 \boldsymbol{I}_{d_y})$.

From a Bayesian perspective, all the information relevant for the characterization of $\boldsymbol{\theta}$ and $\boldsymbol{x}_t$ at discrete time t is contained in the joint posterior pdf $p(\boldsymbol{\theta}, \boldsymbol{x}_t|\boldsymbol{y}_{1:t})$, where $\boldsymbol{y}_{1:t} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_t\}$. The latter density cannot be computed exactly in general and the goal of this paper is to describe a class of flexible and efficient recursive methods for its approximation.

We will show that one way to attain this goal is to tackle the approximation of the sequence of posterior pdfs of the parameters, $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$, $n \in \mathbb{N}$, in the vein of the nested particle filter (NPF) of [25] (see Section 2.4). This yields, in a natural way, approximations for $p(\boldsymbol{\theta}, \boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ and $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ for each t.

3.2 Nested Hybrid Filtering

3.2.1 Importance sampling for parameter estimation

In order to introduce the proposed scheme of nested hybrid filters, let us consider the approximation of the t-th posterior probability distribution of the parameters, with pdf $p(\boldsymbol{\theta}_t|\boldsymbol{y}_{1:t})$, using classical importance sampling [89]. In particular, let $q_t(\boldsymbol{\theta})$ be an arbitrary proposal pdf for the parameter vector $\boldsymbol{\theta}$ and assume that $q_t(\boldsymbol{\theta}) > 0$ whenever $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) > 0$.

Assume that the posterior at time $t-1$, $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})$, is available. Then the posterior pdf at time t can be expressed, via Bayes' theorem, as

$$p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) \propto p(\boldsymbol{y}_t|\boldsymbol{\theta}, \boldsymbol{y}_{1:t-1}) p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1}), \tag{3.5}$$

where the proportionality constant, $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})$, is independent of $\boldsymbol{\theta}$. Equation (3.5) enables the application of the importance sampling method to approximate integrals w.r.t. the posterior pdf $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ (i.e., to approximate the statistics of this probability distribution). Specifically, if we

- draw N independent and identically distributed (i.i.d.) samples from $q_t(\boldsymbol{\theta})$, denoted $\boldsymbol{\theta}_t^i$, $i = 1, \ldots, N$,

- compute importance weights of the form

$$\tilde{w}_t^i = \frac{p(\boldsymbol{y}_t|\boldsymbol{\theta}_t^i, \boldsymbol{y}_{1:t-1}) p(\boldsymbol{\theta}_t^i|\boldsymbol{y}_{1:t-1})}{q_t(\boldsymbol{\theta}_t^i)},$$

and normalize them to obtain

$$w_t^i = \frac{\tilde{w}_t^i}{\sum_{k=1}^N \tilde{w}_t^k}, \quad i = 1, \ldots, N,$$

then it can be proved [89] that

$$\lim_{N\to\infty} \sum_{i=1}^N w_t^i f(\boldsymbol{\theta}_t^i) = \int f(\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) d\boldsymbol{\theta} \quad \text{almost surely (a.s.)} \tag{3.6}$$

for any integrable function $f : \mathbb{R}^{d_\theta} \to \mathbb{R}$ under mild regularity assumptions. In this way one could estimate the value of $\boldsymbol{\theta}$, e.g.,

$$\boldsymbol{\theta}_t^N = \sum_{i=1}^N w_t^i \boldsymbol{\theta}_t^i \approx \int \boldsymbol{\theta} p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) d\boldsymbol{\theta} =: \mathbb{E}[\boldsymbol{\theta}|\boldsymbol{y}_{1:t}],$$

where $\mathbb{E}[\boldsymbol{\theta}|\boldsymbol{y}_{1:t}]$ denotes the expected value of $\boldsymbol{\theta}$ conditional on the observations $\boldsymbol{y}_{1:t}$. We could also estimate the mean square error (MSE) of this estimator, as

$$\text{MSE}_t^N = \sum_{i=1}^N w_t^i \|\boldsymbol{\theta}_t^i - \hat{\boldsymbol{\theta}}_t\|^2 \approx \int \|\boldsymbol{\theta}_t^i - \mathbb{E}[\boldsymbol{\theta}|\boldsymbol{y}_{1:t}]\|^2 p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) d\boldsymbol{\theta}. \tag{3.7}$$

The choice of $q_t(\boldsymbol{\theta})$ is, of course, key to the complexity and the performance of importance sampling schemes. One particularly simple choice is $q_t(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})$, which reduces the importance sampling algorithm to

1. drawing N i.i.d. samples $\boldsymbol{\theta}_t^i$, $i = 1, \ldots, N$, from $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})$, and
2. computing normalized importance weights $w_t^i \propto p(\boldsymbol{y}_t|\boldsymbol{\theta}_t^i, \boldsymbol{y}_{1:t-1}), \quad i = 1, ..., N$.

Unfortunately, this method is not practical because

- it is not possible to draw exactly from $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$, since this pdf is unknown, and
- the likelihood function $p(\boldsymbol{y}_t|\boldsymbol{\theta}_t^i, \boldsymbol{y}_{1:t-1})$ cannot be evaluated exactly either.

In the sequel we tackle the two issues above and, in doing so, we obtain a general scheme for the approximation of the posterior distribution of the parameter vector $\boldsymbol{\theta}$ *and* the state vector $\boldsymbol{x}_t$, i.e., the distribution with pdf $p(\boldsymbol{\theta}, \boldsymbol{x}_t|\boldsymbol{y}_{1:t})$.

3.2.2 Sequential Monte Carlo hybrid filter

It is well known that the likelihood $u_t(\boldsymbol{\theta}) := p(\boldsymbol{y}_t|\boldsymbol{\theta}, \boldsymbol{y}_{1:t-1})$ can be approximated using filtering algorithms [7, 57]. To be specific, function $u_t(\boldsymbol{\theta})$ can be written as the integral

$$u_t(\boldsymbol{\theta}) = \int p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) p(\boldsymbol{x}_t|\boldsymbol{\theta}, \boldsymbol{y}_{1:t-1}) d\boldsymbol{\theta} \tag{3.8}$$

where, in turn, the predictive density $p(\boldsymbol{x}_t|\boldsymbol{\theta}, \boldsymbol{y}_{1:t-1})$ is

$$p(\boldsymbol{x}_t|\boldsymbol{\theta}, \boldsymbol{y}_{1:t-1}) = \int p(\boldsymbol{x}_t|\boldsymbol{\theta}, \boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}, \boldsymbol{y}_{1:t-1}) d\boldsymbol{x}_{t-1} \tag{3.9}$$

and

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{\theta},\boldsymbol{y}_{1:t-1}) \propto p(\boldsymbol{y}_{t-1}|\boldsymbol{\theta},\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{t-1}|\boldsymbol{\theta},\boldsymbol{y}_{1:t-2}). \tag{3.10}$$

Given a fixed parameter vector $\boldsymbol{\theta}$ and a prior pdf $p(\boldsymbol{x}_0|\boldsymbol{\theta})$, the sequence of likelihoods $u_t(\boldsymbol{\theta})$ can be computed by recursively applying Eqs. (3.8), (3.9) and (3.10) for $t = 1, 2, \ldots$.

Let us now assume that we are given a sequence of parameter vectors $\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{k-1}, \boldsymbol{\theta}_k$ and we are interested in computing the likelihood of the last vector, $\boldsymbol{\theta}_k = \boldsymbol{\theta}'$. Following [25], one can compute a sequence of *approximate* likelihoods $\hat{u}_t(\boldsymbol{\theta}_t)$, $t = 1, \ldots, k$, using the recursion

$$\begin{aligned}
\hat{p}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}_{t-1},\boldsymbol{y}_{1:t-1}) &\propto p(\boldsymbol{y}_{t-1}|\boldsymbol{\theta}_{t-1},\boldsymbol{x}_{t-1})\hat{p}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}_{t-1},\boldsymbol{y}_{1:t-2}) && (3.11)\\
\hat{p}(\boldsymbol{x}_t|\boldsymbol{\theta}_t,\boldsymbol{y}_{1:t-1}) &:= \int p(\boldsymbol{x}_t|\boldsymbol{\theta}_t,\boldsymbol{x}_{t-1})\hat{p}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}_{t-1},\boldsymbol{y}_{1:t-1})d\boldsymbol{x}_{t-1} && (3.12)\\
\hat{u}_t(\boldsymbol{\theta}_t) &:= \int p(\boldsymbol{y}_t|\boldsymbol{\theta}_t,\boldsymbol{x}_t)\hat{p}(\boldsymbol{x}_t|\boldsymbol{\theta}_t,\boldsymbol{y}_{1:t-1})d\boldsymbol{x}_t && (3.13)
\end{aligned}$$

which starts with the initial density $\hat{p}(\boldsymbol{x}_0|\boldsymbol{\theta}_0) := p(\boldsymbol{x}_0|\boldsymbol{\theta}_0)$. It can be proved, using the same type of continuity arguments in [25], that the approximation error

$$|u_k(\boldsymbol{\theta}') - \hat{u}_k(\boldsymbol{\theta}')|, \tag{3.14}$$

can be kept bounded, for any k, provided some simple assumptions on the state space model and the sequence $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t$ are satisfied. Note that, in expression of Eq. (3.14), $u_k(\boldsymbol{\theta}')$ is the actual likelihood calculated by iterating Eqs. (3.8), (3.9) and (3.10) for $t = 1, ..., k$, while $\hat{u}_k(\boldsymbol{\theta}')$ is the approximation computed using the sequence $\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{k-1}, \boldsymbol{\theta}_k = \boldsymbol{\theta}'$ and recursion of Eqs. (3.11)–(3.13).

The recursive approximation scheme for $\hat{u}_t(\boldsymbol{\theta})$ can be combined with the "naive" IS procedure of Section 3.2.1 to yield a general (and practical) method for the approximation of the sequence of *a posteriori* probability distributions of the parameter vector $\boldsymbol{\theta}$, hereafter denoted as

$$\mu_t(d\boldsymbol{\theta}) := p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})d\boldsymbol{\theta}.$$

We refer to the proposed scheme as an NHF and provide a detailed outline in Algorithm 6.

Algorithm 6 is essentially a SMC method, often known as a particle filter [28, 42, 63]. At each time step t, the output of the algorithm is an estimate of the posterior probability distribution $\mu_t(d\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})d\boldsymbol{\theta}$. Specifically we construct the discrete and random probability measure

$$\mu_t^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum \delta_{\boldsymbol{\theta}_t^i}(d\boldsymbol{\theta}) \tag{3.15}$$

that can be used to approximate any integrals w.r.t. the true probability measure $\mu_t(d\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})d\boldsymbol{\theta}$. For example, one can estimate any posterior expectations of the parameter vector $\boldsymbol{\theta}$ given the observations $\boldsymbol{y}_{1:t}$, namely

$$\mathbb{E}[\boldsymbol{\theta}|\boldsymbol{y}_{1:t}] = \int \boldsymbol{\theta}\mu_t(d\boldsymbol{\theta}) \approx \int \boldsymbol{\theta}\mu_t^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_i \boldsymbol{\theta}_t^i =: \boldsymbol{\theta}_t^N. \tag{3.16}$$

Since we have constructed a complete distribution, statistical errors can be estimated as well. The *a posteriori* covariance matrix of vector $\boldsymbol{\theta}$ can be approximated as

$$\mathbb{E}\left[(\boldsymbol{\theta} - \mathbb{E}[\boldsymbol{\theta}|\boldsymbol{y}_{1:t}])(\boldsymbol{\theta} - \mathbb{E}[\boldsymbol{\theta}|\boldsymbol{y}_{1:t}])^\top |\boldsymbol{y}_{1:t}\right] \approx \frac{1}{N}\sum_{i=1}^N \left(\boldsymbol{\theta}_t^i - \boldsymbol{\theta}_t^N\right)\left(\boldsymbol{\theta}_t^i - \boldsymbol{\theta}_t^N\right)^\top =: \boldsymbol{P}_t^N. \tag{3.17}$$

```
Algorithm 6 Nested hybrid filter (NHF)
  Inputs:
  - $N$ (number of Monte Carlo samples to generate).
  - A priori pdfs $p(\boldsymbol{\theta})$ and $p(\boldsymbol{x}_0)$.
  - A Markov kernel $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}')$ which, given $\boldsymbol{\theta}' \in D$, generates jittered parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$.
  Procedure:
  1. Initialization
     Draw $\boldsymbol{\theta}_0^i, i = 1, \ldots, N$, i.i.d. samples from $\mu_0(d\boldsymbol{\theta}) = p(\boldsymbol{\theta})d\boldsymbol{\theta}$.
  2. Recursive step
     (a) For $i = 1, \ldots, N$:
         i. Draw $\bar{\boldsymbol{\theta}}_t^i$ from $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i)$.
         ii. Approximate $\hat{p}(\boldsymbol{x}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{y}_{1:t-1})$ using a filtering algorithm.
         iii. Use this approximation to compute the estimate
              $$\hat{u}_t(\bar{\boldsymbol{\theta}}_t^i) = \int p(\boldsymbol{y}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{x}_t)\hat{p}(\boldsymbol{x}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{y}_{1:t-1})d\boldsymbol{x}_t \qquad (3.18)$$
              and let $w_t^i \propto \hat{u}_t(\bar{\boldsymbol{\theta}}_t^i)$ be the normalized weight of $\bar{\boldsymbol{\theta}}_t^i$.
     (b) Resample the discrete distribution
         $$\tilde{\mu}_t^N(d\boldsymbol{\theta}) = \sum_{i=1}^N w_t^i \delta_{\bar{\boldsymbol{\theta}}_t^i}(d\boldsymbol{\theta}) \qquad (3.19)$$
         $N$ times with replacement in order to obtain the particle set $\{\boldsymbol{\theta}_t^i\}_{i=1}^N$ and the approximate probability measure $\mu_t^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_t^i}(d\boldsymbol{\theta})$.
  Outputs: A set of particles $\{\boldsymbol{\theta}_t^i\}_{i=1}^N$ and a probability measure $\mu_t^N(d\boldsymbol{\theta})$.
```

As a byproduct, Algorithm 6 also yields an approximate predictive pdf for $\boldsymbol{x}_t$, namely

$$\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) = \sum_{i=1}^N w_t^i \hat{p}(\boldsymbol{x}_t|\boldsymbol{\theta}_t^i, \boldsymbol{y}_{1:t-1}).$$

Given the approximate filters $\hat{p}(\boldsymbol{x}_t|\boldsymbol{\theta}_t^i, \boldsymbol{y}_{1:t-1})$, the joint probability distribution of $\boldsymbol{\theta}$ and $\boldsymbol{x}_t$ conditioned on $\boldsymbol{y}_{1:t}$ (denoted $\pi_t(d\boldsymbol{\theta} \times d\boldsymbol{x}_t)$) can be approximated as

$$\pi_t^N(d\boldsymbol{\theta} \times d\boldsymbol{x}_t) = \sum_{i=1}^N w_t^i \hat{p}(\boldsymbol{x}_t|\boldsymbol{\theta}, \boldsymbol{y}_{1:t})\delta_{\bar{\boldsymbol{\theta}}_t^i}(d\boldsymbol{\theta})d\boldsymbol{x}_t.$$

The scheme of Algorithm 6 is referred to as *nested* because the SMC algorithm generates, at each time step t, a set of samples $\{\boldsymbol{\theta}_t^1, \ldots, \boldsymbol{\theta}_t^N\}$ and, for each sample $\boldsymbol{\theta}_t^i$, we embed a filter in the state space $\mathbb{R}^{d_x}$ in order to compute the pdf $\hat{p}(\boldsymbol{x}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{y}_{1:t-1})$ and the approximate likelihood $\hat{u}_t(\boldsymbol{\theta}_t^i)$. The term *hybrid* is used because the embedded filters need not be Monte Carlo methods –a variety of techniques can be used and in this work we focus on Gaussian filters, which are attractive because of their (relative) computational simplicity. A scheme with nested particle filters was introduced and analysed in [23, 25].

Let us finally remark that the NHF scheme relies on two approximations:

- *Jittering of the parameters*: The difficulty of drawing samples from $\mu_{t-1}(d\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})d\boldsymbol{\theta}$ can be circumvented if we content ourselves with an approximate sampling step. In particular, if we compute a Monte Carlo approximation $\mu_{t-1}^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_{t-1}^i}(d\boldsymbol{\theta})$ at time $t-1$ (with some of the samples replicated because of the resampling step) then we can generate new particles $\bar{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i)$, $i = 1, \ldots, N$, where $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}')$ is a Markov kernel, i.e., a probability distribution for $\boldsymbol{\theta}$ conditional on $\boldsymbol{\theta}'$. See Section 3.3 for guidelines on the selection of this kernel. Intuitively, we can either jitter a few particles with arbitrary variance (while leaving most of them unperturbed) or jitter all particles with a controlled variance that decreases as N increases.

- *Estimation of likelihoods*: The sequential approximation of Eqs. (3.11)–(3.13) yields biased estimates of the likelihoods $u_t(\boldsymbol{\theta}_t)$ [25]. This is discussed in Section 3.3. In Appendix A we provide details on the computation of the estimates $\hat{p}(\boldsymbol{x}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{y}_{1:t-1})$ and $\hat{u}_t(\boldsymbol{\theta}_t)$ using both the EnKF and the EKF. Other techniques (e.g., particle filters as in [25] or sigma-point Kalman filters [4, 10]) can be used as well.

3.2.3 Sequential quasi Monte Carlo hybrid filter

The SMC method in the first layer of Algorithm 6 can be replaced by other schemes that rely on the point-mass representation of the posterior probability distribution $\mu_t(d\boldsymbol{\theta})$. It is possible to devise procedures based, for instance, on an unscented Kalman filter [52] or other sigma-point Kalman methods [4, 10] to obtain a Gaussian approximation of $\mu_t(\boldsymbol{\theta})$. This approach, that is specifically exploited in Chapter 4, may bring computational savings but it is based on the assumption that the posterior pdf $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ is unimodal.

In this subsection, we describe an NHF method (hence, of the same class as Algorithm 6) where the SMC scheme is replaced by a SQMC procedure of the type introduced in [40]. The term QMC refers to a class of deterministic methods for numerical integration [80] that employ low-discrepancy point sets (e.g., Halton sequences [44] or Sobol sequences [18]), instead of random sample sets, for the approximation of multidimensional integrals. In the context of QMC, *discrepancy* is defined to quantify how uniformly the points in a sequence are distributed into an arbitrary set S. Hence, the lowest discrepancy is attained when these points are equi-distributed. The main advantage of (deterministic) QMC methods over (random) Monte Carlo schemes is that they can attain a faster rate of convergence relative to the number of points in the grid, N. Within an NHF, the use of QMC should lead to a better performance/complexity trade-off as long as the parameter dimension, d_θ, is relatively small. This is illustrated numerically for a stochastic two-scale Lorenz 96 model in Section 3.5.

```
Algorithm 7 Sequential quasi Monte Carlo nested hybrid filter
  Inputs:
  - $N$ (number of Monte Carlo samples to generate).
  - A priori pdfs $p(\boldsymbol{\theta})$ and $p(\boldsymbol{x}_0)$.
  - A Markov kernel $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}')$ which, given $\boldsymbol{\theta}'$, generates jittered parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$.
  Procedure:
  1. Initialization
     (a) Generate QMC uniform samples $\{\boldsymbol{\rho}_{-1}^i, \boldsymbol{\rho}_0^i\}_{i=1}^N$ in $[0,1)^{d_\theta}$. Draw $\boldsymbol{\theta}_0^i \sim p(\boldsymbol{\theta}|\boldsymbol{\rho}_{-1}^i)$, $i = 1, \dots, N$.
  2. Recursive step, $t \geq 1$.
     (a) For $i = 1, \dots, N$:
         i. If $t = 1$, then draw $\bar{\boldsymbol{\theta}}_1^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_0^i, \boldsymbol{\rho}_0^i)$, else draw $\bar{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i, \tilde{\boldsymbol{\rho}}_{t-1}^{\mathsf{c}(i)})$, for $t \geq 2$.
         ii. Approximate $\hat{p}(\boldsymbol{x}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{y}_{1:t-1})$.
         iii. Use this approximation to compute the estimate
              $$\hat{u}_t(\bar{\boldsymbol{\theta}}_t^i) = \int p(\boldsymbol{y}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{x}_t)\hat{p}(\boldsymbol{x}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{y}_{1:t-1})d\boldsymbol{x}_t. \quad (3.20)$$
              and let $w_t^i \propto \hat{u}_t(\bar{\boldsymbol{\theta}}_t^i)$ be the normalized weight of $\bar{\boldsymbol{\theta}}_t^i$.
     (b) Generate a QMC point set $\{\boldsymbol{\rho}_t^i\}_{i=1}^N$ in $[0,1)^{d_\theta+1}$; let $\boldsymbol{\rho}_t^i = (\rho_t^i, \tilde{\boldsymbol{\rho}}_t^i) \in [0,1) \times [0,1)^{d_\theta}$.
     (c) Hilbert sort: find a permutation $\mathsf{b}$ such that
         $$(\mathsf{h} \circ \psi)(\bar{\boldsymbol{\theta}}_t^{\mathsf{b}(1)}) \leq \ldots \leq (\mathsf{h} \circ \psi)(\bar{\boldsymbol{\theta}}_t^{\mathsf{b}(N)}), \quad \text{if } d_\theta \geq 2$$
         $$\bar{\boldsymbol{\theta}}_t^{\mathsf{b}(1)} \leq \ldots \leq \bar{\boldsymbol{\theta}}_t^{\mathsf{b}(N)}, \quad \text{if } d_\theta = 1.$$
     (d) Resampling: find a permutation $\mathsf{c}$ such that $\rho_t^{\mathsf{c}(1)} \leq \ldots \leq \rho_t^{\mathsf{c}(N)}$. For $i = 1, \dots, N$, set $\boldsymbol{\theta}_t^i = \bar{\boldsymbol{\theta}}_t^j$ if, and only if,
         $$\sum_{k=1}^{j-1} w_t^{\mathsf{b}(k)} < \rho_t^{\mathsf{c}(i)} \leq \sum_{k=1}^{j} w_t^{\mathsf{b}(k)}, \quad j \in \{1, \dots, N\}.$$
  Outputs: A set of particles $\{\boldsymbol{\theta}_t^i\}_{i=1}^N$ and a probability measure $\mu_t^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_t^i}(d\boldsymbol{\theta})$.
```

The NHF based on the SQMC methodology of [40] can be obtained from Algorithm 6 if we replace the sampling and resampling steps typical of the SMC schemes by the generation of low-discrepancy point sets. Let $\{\boldsymbol{\rho}_t^i\}_{i=1}^N$ be a Halton sequence of low-discrepancy (deterministic) uniform samples in the set $[0,1)^{d_\theta+1}$ [44]. These uniform samples can be used to generate low-discrepancy variates from other distributions via a number of methods[1]. For example, the Box-Muller transformation [16] can be used to generate pairs of independent, standard, normally distributed pseudo-random numbers. We explicitly indicate the use of low-discrepancy uniform numbers, $\boldsymbol{\rho}_t^i$, in the generation of samples with general distributions by conditioning on $\boldsymbol{\rho}_t^i$. Hence, drawing the i-th sample from the prior parameter pdf, $\boldsymbol{\theta}_0^i \sim p(\boldsymbol{\theta})$, is now replaced by $\boldsymbol{\theta}_0^i \sim p(\boldsymbol{\theta}|\boldsymbol{\rho}_0^i)$. In order to propagate the i-th sample at time $t-1$, $\boldsymbol{\theta}_{t-1}^i$, into time t, we draw from the kernel $\kappa_N(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}^i, \boldsymbol{\rho}_t^i)$. If sampling is needed in the second layer of filters (in order to compute the estimates $\hat{p}(\boldsymbol{x}_t|\bar{\boldsymbol{\theta}}_t^i, \boldsymbol{y}_{1:t-1})$ and $\hat{u}_t(\bar{\boldsymbol{\theta}}_t^i)$) we use additional Halton sequences in a similar way.

In order to keep the low-discrepancy property across the resampling step, we additionally introduce the following functions (see [40] for details).

- A discrepancy-preserving bijective map $\psi : \mathbb{R}^{d_\theta} \to [0,1]^{d_\theta}$. Several choices are possible for this function. Following [40], here we assume

$$\psi(\bar{\boldsymbol{\theta}}_t^i) = \left[1 + \exp\left(-\frac{\bar{\boldsymbol{\theta}}_t^i - \boldsymbol{\theta}_t^-}{\boldsymbol{\theta}_t^+ - \boldsymbol{\theta}_t^-}\right)\right]^{-1}, \tag{3.21}$$

  where $\boldsymbol{\theta}_t^-$ and $\boldsymbol{\theta}_t^+$ are the d_θ-dimensional vectors whose j-th components are, respectively,

$$[\boldsymbol{\theta}_t^-]_j = m_{t,j}^N - 2s_{t,j}^{2N} \quad \text{and} \quad [\boldsymbol{\theta}_t^+]_j = m_{t,j}^N + 2s_{t,j}^{2N},$$

  whereas $m_{t,j}^N = \sum_{i=1}^N w_t^i [\bar{\boldsymbol{\theta}}_n^i]_j$ and $s_{t,j}^{2N} = \sum_{i=1}^N w_t^i \left([\bar{\boldsymbol{\theta}}_t^i]_j - m_{t,j}^N\right)^2$, $j = 1, \ldots, d_\theta$, are component-wise means and variances.

- The inverse of the Hilbert curve, $\mathsf{h} : [0,1]^{d_\theta} \longrightarrow [0,1]$, which is a continuous fractal space-filling curve that provides a locality-preserving map between a 1-dimensional and a d_θ-dimensional space [45, 77].

The SQMC-based NHF is outlined in Algorithm 7.

3.3 Convergence analysis

The nested filtering schemes of Section 3.2 admit various implementations depending on how we choose to approximate the conditional pdf $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ which, in turn, is needed to estimate the likelihood function and compute the importance weights $w_t^i \propto \hat{u}(\bar{\boldsymbol{\theta}}_t^i) \approx u_t(\bar{\boldsymbol{\theta}}_t^i)$, $i = 1, \ldots, N$.

For each choice of approximation method, the estimate $\hat{u}_t(\boldsymbol{\theta})$ may behave differently and yield distinct convergence properties. Here we assume that $\hat{u}_t(\boldsymbol{\theta})$ is a r.v. with finite mean $\bar{u}_t(\boldsymbol{\theta}) = \mathbb{E}[\hat{u}_t(\boldsymbol{\theta})] < \infty$ and finite moments up to some prescribed order $p \geq 1$. Specifically, we make the following assumption.

A. 1 *Given $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$, the estimator $\hat{u}_t(\boldsymbol{\theta})$ is random and can be written as*

$$\hat{u}_t(\boldsymbol{\theta}) = \bar{u}_t(\boldsymbol{\theta}) + m_t(\boldsymbol{\theta}), \tag{3.22}$$

where $m_t(\boldsymbol{\theta})$ is a zero-mean r.v. satisfying $\mathbb{E}[m_t(\boldsymbol{\theta})^p] \leq \sigma^p < \infty$ for some prescribed $p \geq 1$. Furthermore, the mean $\bar{u}_t(\boldsymbol{\theta}) = \mathbb{E}\left[\hat{u}_t(\boldsymbol{\theta})\right]$ has the form

$$\bar{u}_t(\boldsymbol{\theta}) = u_t(\boldsymbol{\theta}) + b_t(\boldsymbol{\theta}), \tag{3.23}$$

where $b_t(\boldsymbol{\theta})$ is a deterministic and absolutely bounded bias function.

[1]One can use a number of techniques used to produce random samples from a given uniform source. See [69] for a comprehensive description of the field, both for single and multivariate distributions.

From here on, we use $D \subseteq \mathbb{R}^{d_\theta}$ to denote the support set of the parameter vector $\boldsymbol{\theta}$. Given a real function $a : D \to \mathbb{R}$, its absolute supremum is indicated as $\|a\|_\infty := \sup_{\boldsymbol{\theta} \in D} |a(\boldsymbol{\theta})|$. The set of absolutely bounded real functions on D is denoted $B(D)$, i.e., $B(D) := \{(a : D \to \mathbb{R}) : \|a\|_\infty < \infty\}$. For our analysis we assume that $u_t \in B(D)$ and, since we have also assumed the bias function b_t to be bounded, it follows that $\bar{u}_t \in B(D)$, i.e., $\|\bar{u}_t\|_\infty < \infty$. To be precise, we impose the following assumption.

A. 2 *Given a fixed sequence of observations $y_{1:t}$, the family of functions $\{\bar{u}_t(\boldsymbol{\theta}), \boldsymbol{\theta} \in D\}$ satisfies the following inequalities for each $t = 1, 2, ...$:*

1. *$\bar{u}_t \in B(D)$, and*
2. *$\bar{u}_t(\boldsymbol{\theta}) > 0$ for any $\boldsymbol{\theta} \in D$.*

Since $\|u_t\|_\infty < \infty$, A.2.1 follows from assumption A.1. Similarly, if $u_t(\boldsymbol{\theta}) > 0$ for all $\boldsymbol{\theta} \in D$ then A.2.2 is a natural assumption (since $\hat{u}(\boldsymbol{\theta})$ is an estimator of a positive magnitude).

We shall prove that, because of the bias $b_t(\boldsymbol{\theta})$, the approximation μ_t^N converges to the perturbed probability measure $\bar{\mu}_t$ induced by the mean function $\bar{u}_t$, instead of the true posterior probability measure μ_t induced by the true likelihood function u_t.

To be specific, the sequence of posterior measures μ_t, $t \geq 1$, can be constructed recursively, starting from a prior μ_0, by means of the projective product operation [13], denoted $\mu_t = u_t \cdot \mu_{t-1}$. When u is a positive and bounded function and μ is a probability measure, the new measure $u \cdot \mu$ is defined in terms of its integrals. In particular, if $a \in B(D)$ then

$$\int a(\boldsymbol{\theta})(u \cdot \mu)(d\boldsymbol{\theta}) := \frac{\int a(\boldsymbol{\theta}) u(\boldsymbol{\theta}) \mu(d\boldsymbol{\theta})}{\int u(\boldsymbol{\theta}) \mu(d\boldsymbol{\theta})}.$$

For conciseness, hereafter we use the shorthand

$$(a, \mu) := \int a(\boldsymbol{\theta}) \mu(d\boldsymbol{\theta})$$

for the integral of a function $a(\boldsymbol{\theta})$ w.r.t. a measure $\mu(d\boldsymbol{\theta})$. With this notation, we can write

$$(a, \mu_t) = (a, u_t \cdot \mu_{t-1}) = \frac{(a u_t, \mu_{t-1})}{(u_t, \mu_{t-1})}. \tag{3.24}$$

If, instead of the true likelihood u_t, we use the *biased* function $\bar{u}_t = u_t + b_t$ to update the posterior probability measure associated to the parameter vector $\boldsymbol{\theta}$ at each time t then we obtain the new sequence of measures

$$\bar{\mu}_0 = \mu_0, \quad \bar{\mu}_t = \bar{u}_t \cdot \bar{\mu}_{t-1}, \quad t = 1, 2, ...,$$

where, according to the definition of the projective product,

$$(a, \bar{\mu}_t) = \frac{(a \bar{u}_t, \bar{\mu}_{t-1})}{(\bar{u}_t, \bar{\mu}_{t-1})}$$

for any integrable function $a(\boldsymbol{\theta})$. Note that the two sequences, μ_t and $\bar{\mu}_t$, start from the same prior μ_0. Obviously, we recover the original sequence, i.e, $\bar{\mu}_t \to \mu_t$, when the bias vanishes, $b_t \to 0$.

In this section we prove that the approximation μ_t^N generated by a generic nested filter that satisfies A.1 and A.2 converges to $\bar{\mu}_t$ in L_p, for each $t = 1, 2, ...$, under an additional regularity assumption on the jittering kernel κ_N.

A. 3 *The kernel κ_N used in the jittering step satisfies the inequality*

$$\sup_{\boldsymbol{\theta}' \in D} \int |h(\boldsymbol{\theta}) - h(\boldsymbol{\theta}')| \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}') \leq \frac{c_\kappa \|h\|_\infty}{\sqrt{N}} \tag{3.25}$$

for every $h \in B(D)$ and some constant $c_\kappa < \infty$ independent of N.

A simple kernel that satisfies A.3 is [25]

$$\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}') = (1 - \epsilon_N)\delta_{\boldsymbol{\theta}'}(d\boldsymbol{\theta}) + \epsilon_N \kappa(d\boldsymbol{\theta}|\boldsymbol{\theta}'),$$

where $0 < \epsilon_N \leq \frac{1}{\sqrt{N}}$ and $\kappa(d\boldsymbol{\theta}|\boldsymbol{\theta}')$ is an arbitrary Markov kernel with mean $\boldsymbol{\theta}'$ and finite variance, for example $\kappa(d\boldsymbol{\theta}|\boldsymbol{\theta}') = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}', \tilde{\sigma}^2 \boldsymbol{I}_{d_\theta})$, where $\tilde{\sigma}^2 < \infty$ and $\boldsymbol{I}_{d_\theta}$ is the identity matrix. Intuitively, this kind of kernel changes each particle with probability ϵ_N and leaves it unmodified with probability $1 - \epsilon_N$.

Finally, we can state a general result on the convergence of Algorithm 6. For a real r.v. x and $p \geq 1$, let $\|x\|_p$ denote the L_p norm, i.e. $\|x\|_p := \mathbb{E}[|x|^p]^{\frac{1}{p}}$.

Theorem 1 *Let the sequence of observations $y_{1:t_o}$ be arbitrary but fixed, with $t_o < \infty$, and choose an arbitrary function $h \in B(D)$. If assumptions A.1, A.2 and A.3 hold, then*

$$\|(h, \mu_t^N) - (h, \bar{\mu}_t)\|_p \leq \frac{c_t \|h\|_\infty}{\sqrt{N}}, \quad \textit{for } t = 0, 1, \ldots, t_o \textit{ and any } p \geq 1, \tag{3.26}$$

where $\{c_t\}_{0 \leq t \leq t_o}$ is a sequence of finite constants independent of N.

Proof: See Appendix B. $\square$

We remark that Theorem 1 does *not* state that the approximate posterior probability measure output by Algorithm 6, μ_t^N, converges to the true posterior measure μ_t, but to the biased version $\bar{\mu}_t$. Moreover, the latter depends on the choice of filters used in the second layer of Algorithm 6 (i.e., on the estimator of the likelihood, $\hat{u}_t$). The value of this theorem is that it guarantees the numerical consistency of the nested hybrid filter: as we increase the computational effort (by increasing N), the random probability measure μ_t^N converges to a well defined limit (and so do any point estimators that we may derive from it, e.g., the posterior mean estimator $\boldsymbol{\theta}_t^N$). The connection between this limit measure, $\bar{\mu}_t$, and the true posterior measure μ_t is given by assumption A.1 and the projective product operation, namely,

$$\bar{\mu}_t = (u_t + b_t) \cdot \bar{\mu}_{t-1}, \quad \text{while} \quad \mu_t = u_t \cdot \mu_{t-1},$$

with both sequences starting with a common prior measure $\bar{\mu}_0 = \mu_0$. The practical performance of the proposed schemes (with finite N) is explored numerically in the sequel.

3.4 A stochastic Lorenz 96 model

In order to assess the proposed methods numerically, we have applied them to a stochastic, discrete-time version of the two-scale Lorenz 96 model [11, 43, 48]. The latter is a deterministic system of nonlinear differential equations that displays some key features of atmospheric dynamics (including chaotic behavior) in a relatively simple model of arbitrary dimension (the number d_x of dynamic variables can be scaled as needed). The model consists of two sets of dynamic variables, $\boldsymbol{x}$ and $\boldsymbol{z}$, which evolve according to the equations

$$\begin{aligned} d\boldsymbol{x} &= \boldsymbol{f}_1(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha})d\tau + \sigma_{\boldsymbol{x}} d\boldsymbol{v} \\ d\boldsymbol{z} &= \boldsymbol{f}_2(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha})d\tau + \sigma_{\boldsymbol{z}} d\boldsymbol{w} \end{aligned} \tag{3.27}$$

where τ denotes continuous time, $\boldsymbol{x}(\tau)$ and $\boldsymbol{z}(\tau)$ represent the *slow* and *fast* variables, respectively, $\boldsymbol{v}$ and $\boldsymbol{w}$ are Wiener processes, $\sigma_{\boldsymbol{x}}, \sigma_{\boldsymbol{z}} > 0$ are known scale parameters and $\boldsymbol{\alpha}$ is a parameter vector of dimension $d_\alpha = 4$. Let us assume there are d_x slow variables, x_j, $j = 0, \dots, d_x - 1$, and L fast variables per slow variable, i.e., z_l, $l = 0, ..., d_z - 1$, for $d_z = Ld_x$, overall. The maps $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are $\mathbb{R}^{d_x} \times \mathbb{R}^{d_z} \times \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_x}$ and $\mathbb{R}^{d_z} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_\alpha} \to \mathbb{R}^{L}$ functions, respectively, that can be written (skipping the time index τ) as

$$\begin{aligned}
\boldsymbol{f}_1 &= [f_{1,0}, \dots, f_{1,d_x-1}]^\top \quad \text{and} \quad f_{1,j}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) = -x_{j-1}(x_{j-2} - x_{j+1}) - x_j + F - \frac{HC}{B} \sum_{l=(j-1)L}^{Lj-1} z_l, \\
\boldsymbol{f}_2 &= [f_{2,0}, \dots, f_{2,d_xL-1}]^\top \quad \text{and} \quad f_{2,l}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\alpha}) = -CBz_{l+1}(z_{l+2} - z_{l-1}) - Cz_l + \frac{CF}{B} + \frac{HC}{B} x_{\lfloor \frac{l-1}{L} \rfloor},
\end{aligned} \tag{3.28}$$

where $\boldsymbol{\alpha}$ contains the parameters F,C,H and B. The forcing parameter F controls the turbulence of the chaotic flow, C determines the time scale of the fast variables $\{z_l\}_{l \geq 0}$, H controls the strength of the coupling between the fast and slow variables and B determines the amplitude of the fast variables [11]. The dynamic variables are assumed to be arranged on a circular structure, hence the operations on the j indices are modulo d_x and operations on the l indices are modulo L. This means that for any integer k, $j + k \equiv (j + k) \bmod d_x$ and $l + k \equiv (l + k) \bmod L$. Notation $\lfloor a \rfloor$ indicates the truncation of a positive real number a to the closest integer smaller than a.

We apply a discretization scheme with fixed step-size $\Delta > 0$ in order to obtain a discrete-time version of the two-scale Lorenz 96 model. To be specific, we numerically integrate Eq. (3.27) by means of the stochastic difference equations

$$\begin{aligned}
\bar{\boldsymbol{x}}_n &= \bar{\boldsymbol{x}}_{n-1} + \boldsymbol{F}_1\big(\bar{\boldsymbol{x}}_{n-1}, \bar{\boldsymbol{z}}_{n-1}, \boldsymbol{\alpha}, \Delta, \sigma_{\boldsymbol{x}} \bar{\boldsymbol{v}}_n\big), \\
\bar{\boldsymbol{z}}_n &= \bar{\boldsymbol{z}}_{n-1} + \boldsymbol{F}_2\big(\bar{\boldsymbol{x}}_{n-1}, \bar{\boldsymbol{z}}_{n-1}, \boldsymbol{\alpha}, \Delta, \sigma_{\boldsymbol{z}} \bar{\boldsymbol{w}}_n\big),
\end{aligned} \tag{3.29}$$

where $n \in \mathbb{N}$ denotes discrete time, $\bar{\boldsymbol{x}}_n \approx \boldsymbol{x}(n\Delta)$ is the system state at time $\tau = n\Delta$ and $\bar{\boldsymbol{v}}_n$ and $\bar{\boldsymbol{w}}_n$ are zero-mean Gaussian r.v.s of dimension $d_v \geq d_x$ and $d_w \geq d_z$ with covariance matrices $\Delta \boldsymbol{I}_{d_v}$ and $\Delta \boldsymbol{I}_{d_w}$, respectively. The functions $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ depend on the choice of discretization scheme. The simplest one is the Euler-Maruyama method, which yields [39]

$$\begin{aligned}
\bar{\boldsymbol{x}}_n &= \bar{\boldsymbol{x}}_{n-1} + \Delta \boldsymbol{f}_1(\bar{\boldsymbol{x}}_{n-1}, \bar{\boldsymbol{z}}_{n-1}, \boldsymbol{\alpha}) + \sigma_{\boldsymbol{x}} \bar{\boldsymbol{v}}_n, \\
\bar{\boldsymbol{z}}_n &= \bar{\boldsymbol{z}}_{n-1} + \Delta \boldsymbol{f}_2(\bar{\boldsymbol{x}}_{n-1}, \bar{\boldsymbol{z}}_{n-1}, \boldsymbol{\alpha}) + \sigma_{\boldsymbol{z}} \bar{\boldsymbol{w}}_n,
\end{aligned} \tag{3.30}$$

i.e., the noise is additive, with $d_v = d_x$ and $d_w = d_z$, and $\boldsymbol{F}_1(\bar{\boldsymbol{x}}_{n-1}, \bar{\boldsymbol{z}}_{n-1}, \boldsymbol{\alpha}, \Delta, \sigma_{\boldsymbol{x}} \bar{\boldsymbol{v}}_n) = \Delta \boldsymbol{f}_1(\bar{\boldsymbol{x}}_{n-1}, \bar{\boldsymbol{z}}_{n-1}, \boldsymbol{\alpha}) + \sigma_{\boldsymbol{x}} \bar{\boldsymbol{v}}_n$ and $\boldsymbol{F}_2(\bar{\boldsymbol{x}}_{n-1}, \bar{\boldsymbol{z}}_{n-1}, \boldsymbol{\alpha}, \Delta, \sigma_{\boldsymbol{z}} \bar{\boldsymbol{w}}_n) = \Delta \boldsymbol{f}_2(\bar{\boldsymbol{x}}_{n-1}, \bar{\boldsymbol{z}}_{n-1}, \boldsymbol{\alpha}) + \sigma_{\boldsymbol{z}} \bar{\boldsymbol{w}}_n$. For a Runge-Kutta method of order q, as a more sophisticated example, the functions $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ result from applying $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ q times, with a Gaussian perturbation passing through the nonlinearity at each of these intermediate steps. To be specific, we apply the 4th order Runge-Kutta (RK4) method [39] ($q = 4$). See [39] for details on various integration methods for stochastic differential equations (SDEs).

We assume that the system of Eq. (3.29) can be observed every m discrete-time steps (i.e., every $m\Delta$ continuous-time units). Therefore, we re-write the state equations in the time scale of the observations (i.e., discrete-time t rather than n) as

$$\begin{aligned}
\boldsymbol{x}_t &= \boldsymbol{x}_{t-1} + \boldsymbol{F}_1^m\big(\boldsymbol{x}_{t-1}, \boldsymbol{z}_{t-1}, \boldsymbol{\alpha}, \Delta, \sigma_{\boldsymbol{x}} \boldsymbol{v}_t\big), \\
\boldsymbol{z}_t &= \boldsymbol{z}_{t-1} + \boldsymbol{F}_2^m\big(\boldsymbol{x}_{t-1}, \boldsymbol{z}_{t-1}, \boldsymbol{\alpha}, \Delta, \sigma_{\boldsymbol{z}} \boldsymbol{w}_t\big),
\end{aligned} \tag{3.31}$$

$t \in \mathbb{N}$, where the notation $\boldsymbol{F}_1^m$ and $\boldsymbol{F}_2^m$ indicates that Eqs. in (3.29) are applied m consecutive times in order to move from $\boldsymbol{x}_{t-1} = \bar{\boldsymbol{x}}_{(t-1)m}$ to $\boldsymbol{x}_t = \bar{\boldsymbol{x}}_{tm}$ and $\boldsymbol{z}_{t-1} = \bar{\boldsymbol{z}}_{(t-1)m}$ to $\boldsymbol{z}_t = \bar{\boldsymbol{z}}_{tm}$.

We assume that the observations are linear but only 1 out of K slow variables can be observed. Therefore, the observation process has the form

$$\boldsymbol{y}_t = \begin{bmatrix} x_{K,t} \\ x_{2K,t} \\ \vdots \\ x_{d_y K,t} \end{bmatrix} + \boldsymbol{r}_t, \tag{3.32}$$

where $t = 1, 2, ...$ and $\boldsymbol{r}_t$ is a sequence of i.i.d. r.v.s with common pdf $\mathcal{N}(\boldsymbol{r}_t|0, \sigma_o^2 \mathbf{I}_{d_y})$. Note that fast variables are not observed.

In our computer experiments, system of Eq. (3.31) is often employed to generate both ground-truth values for the slow variables $\{\boldsymbol{x}_t\}_{t\geq 0}$ and synthetic observations, $\{\boldsymbol{y}_t\}_{t\geq 1}$. However, since in real world problems models are inherently imperfect (modelling errors always exist to some extent), a different and simpler version of the Lorenz 96 model is used in order to implement the NHFs. In this simpler version, the contribution of the fast variables to the j-th equation is substituted by a polynomial function of the slow variable x_j. To be specific, the set of functions $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ is replaced by the map

$$\bar{\boldsymbol{f}} = [\bar{f}_0, \dots, \bar{f}_{d_x-1}]^\top \quad \text{and} \quad \bar{f}_j(\boldsymbol{x}, \boldsymbol{\theta}) = [-x_{j-1}(x_{j-2} - x_{j+1}) - x_j + F - \ell(x_j, \mathsf{a})], \quad j = 0, ..., d_x - 1, \tag{3.33}$$

where $\mathsf{a} = [\mathsf{a}_1, \mathsf{a}_2]^\top$ is a (constant) parameter vector, $\boldsymbol{\theta} = [F, \mathsf{a}^\top]^\top$ contains all the parameters in the simplified model, and the function $\ell(x_j, \mathsf{a}) \in \mathbb{R}$ is a polynomial ansatz for the coupling term $\frac{HC}{B} \sum_{l=(j-1)L}^{Lj-1} z_l$ in Eq. (3.28). Note that in this simplified model we have removed the fast variables completely. In this paper we assume that $\ell(x_j, \mathsf{a})$ is a polynomial in x_j of degree 2, characterized by the coefficients a_1 and a_2 as

$$\ell(x_j, \mathsf{a}) = \mathsf{a}_1 x_j^2 + \mathsf{a}_2 x_j.$$

Then, the system of Eq. (3.31) can be replaced by

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \bar{\boldsymbol{F}}^m(\boldsymbol{x}_{t-1}, \boldsymbol{\theta}, \Delta, \sigma_{\boldsymbol{x}} \boldsymbol{v}_t) \tag{3.34}$$

where $\bar{\boldsymbol{F}}^m$ is the RK4 approximation of the function $\bar{\boldsymbol{f}} = [\bar{f}_0, \dots, \bar{f}_{d_x-1}]^\top$ in Eq. (3.33). Assuming $\boldsymbol{r}_t$ is a sequence of i.i.d. noise terms with Gaussian probability distribution, $p(\boldsymbol{r}) = \mathcal{N}(\boldsymbol{r}|0, \sigma_o^2 \boldsymbol{I}_{d_y})$, then

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{x}_t, \sigma_o^2 \boldsymbol{I}_{d_y}) \tag{3.35}$$

which denotes a d_y-dimensional Gaussian density with mean $\boldsymbol{x}_t$ and covariance matrix $\sigma_o^2 \boldsymbol{I}_{d_y}$, where $\boldsymbol{I}_{d_y}$ is the $d_y \times d_y$ identity matrix.

3.5 Numerical results

We have conducted computer simulations to illustrate the performance of the proposed NHF methods. In particular, we have carried out computer experiments for six different schemes: the NPF of [25], the two-stage filter of [92] and four NHFs that rely on the SQMC and the SMC, both in combination with EKFs or EnKFs. Then, two different versions of Algorithm 6 (SMC-EKF, SMC-EnKF) and Algorithm 7 (SQMC-EKF, SQMC-EnKF) are simulated. The simulation setup is described below, followed by the discussion of our numerical results in Section 3.5.

3.5.1 Simulation setup

For our computer experiments we have used the two-scale Lorenz 96 model of Eq. (3.27), in order to generate

- reference signals $\tilde{\boldsymbol{x}}_n$, $n = 0, 1, \ldots$, used as ground truth for the assessment of the estimators, and
- sequences of observations, $\boldsymbol{y}_t$, $t = 1, 2, \ldots$ as in Eq. 3.21.

The model is integrated using the RK4 method with Gaussian perturbations [39] (as outlined in Eq. (3.31)). The integration step is set to $\Delta = 5 \times 10^{-3}$ continuous-time units through all experiments and the fixed model parameters are $F = 8$, $H = 0.75$, $C = 10$ and $B = 15$. For all experiments, we assume that there are $L = 10$ fast variables per slow variable, hence the total dimension of the model is $(L+1)d_x = 11d_x$ (with different values of d_x for different experiments). The noise scaling factors are $\sigma_{\boldsymbol{x}} = \frac{\Delta}{4} = 1.25 \times 10^{-3}$, $\sigma_{\boldsymbol{z}} = \frac{\Delta}{16} = 3.125 \times 10^{-4}$ and $\sigma_o = 4$, all assumed known. We assume that half of the slow variables are observed in Gaussian noise, i.e., $K = 2$.

We assess the accuracy of the estimation algorithms in terms of the MSE of the predictors of the dynamic variables. For the NHFs, these estimators take the form

$$\hat{\boldsymbol{x}}_t = \sum_{i=1}^{N} w_t^i \boldsymbol{x}_t^i, \tag{3.36}$$

where $\boldsymbol{x}_t^i$ is the posterior-mean estimate obtained from the approximate filter $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_t^i)$, that can be expressed as $\mathcal{N}(\boldsymbol{x}_t|\boldsymbol{x}_t^i, \boldsymbol{P}_t^i)$, since the approximation is Gaussian. In the plots, however, we show the empirical MSE per dimension resulting directly from the simulations,

$$\text{MSE}_t = \frac{1}{d_x} \parallel \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \parallel^2 . \tag{3.37}$$

averaged over 100 independent simulation runs, being all of them of 40 continuous-time units of duration, which amount to $\frac{40}{\Delta} = 8000$ discrete time steps.

The simulations presented below include running times for the different methods. They have been coded in Matlab R2016a and run on a computer with 64 GB of DRAM and equipped with two Intel Xeon E5-2680 processors (running at 2.80GHz) with 10 cores each and HyperThreading as well as an Intel Xeon Phi co-processor.

3.5.2 Results

Table 3.2 shows a comparison of the performance of the NPF, the two-stage filter and the four NHFs, based on the use of SMC, SQMC, EKF and EnKF schemes as described in Section 3.2, in terms of their running times and the MSE of the state estimators (averaged over time and dimensions). We have carried out this computer simulation for a model with dimension $d_x = 40$ and a gap between observations of $m\Delta = 0.05$ continuous-time units ($m = 10$). For all of the algorithms compared, N represents the number of particles or samples in the first layer of computation, while M is the number of samples in the second layer of the filter. All NHFs algorithms work with $N = 100$ particles for the approximation of the posterior distributions of the fixed parameters, using $M = d_x = 40$ samples per each EnKF in the second-layer. Despite the larger number of particles in the NPF ($N = M = 800$), this method achieves the highest error and takes the longest running time. The NPF is followed by the two-stage filter method[2], that can run three times

[2]Algorithm introduced in [92] that alternates the estimation of static parameters (conditional on a fixed state estimate) and the tracking of the dynamic variables (conditional on a fixed estimate of the static parameters).

faster but cannot outperform any of the NHFs in terms of error. Both NHFs using EKF attain the least MSE with the smallest running time. In order to improve the performance of the NPF, the numbers of particles M and N would have to be considerably increased, but this would increase the running times correspondingly (the complexity of the NPF is $\mathcal{O}(NM)$ [25]).

Algorithm	Running time (minutes)	MSE
NHF: SQMC-EKF	2.16	0.46
NHF: SMC-EKF	2.27	0.49
NHF: SQMC-EnKF	6.83	0.62
NHF: SMC-EnKF	7.12	0.95
Two-Stage Filter ($N = 600, M = 400$)	6.85	4.59
NPF ($N = M = 800$)	17.96	11.91

Table 3.2: Running times and average MSE (over time and state dimensions) for the NPF, the two-stage filter and four NHFs, based on the SQMC, the SMC, the EKF and the EnKF, respectively.

In the next experiment we assess the performance of the different NHFs depending on the number of particles used in the first-layer of the filter, in order to choose appropriately this number to carry out the following computer experiments. For this purpose, we consider a model with dimension $d_x = 100$, a gap between observations of $m\Delta = 0.05$ continuous-time units ($m = 10$) and a number of particles that ranges from 50 to 400. Figure 3.1 shows the numerical results for this experiment. We observe that the MSE for the four algorithms stabilizes quickly. At the sight of these results, we set $N = 100$ for all remaining experiments. Additionally, Fig. 3.1 also shows the difference between the NHFs. Specifically, we see that using SQMC in the first-layer we can slightly improve the performance. For this reason, in the next experiments we only simulate NHFs that rely on SQMC. Moreover, it is easy to observe that the filters that use EKFs (instead of EnKFs) in the second-layer obtain better results.

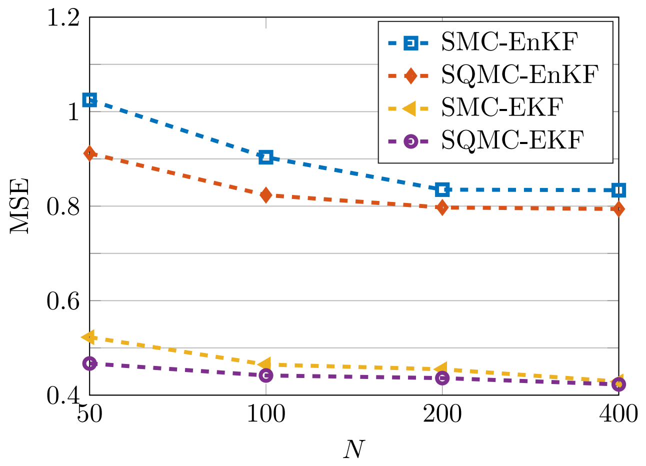


Figure 3.1: MSE of the different NHFs depending on the number of particles N used in the first-layer of the filter.

In the next set of computer experiments we compare the SQMC-EKF and the SQMC-EnKF methods in terms of their average MSE and their running times for different values of the state dimension d_x and the gap between consecutive observations m (in discrete time steps). For each combination of d_x and m we have carried out 100 independent simulation runs. The number of particles in the parameter space is fixed, $N = 100$, for all simulations, but the size of the ensemble in the EnKFs is adjusted to the dimension. In particular, we set $M = d_x$.

Figure 3.2 shows (a) the running times and (b) the average MSE attained by the two SQMC NHFs when the state dimension d_x ranges from 100 to 800. The gap between observations is fixed to $m = 20$ (i.e., 0.1 time units versus 0.05 in Figure 3.1). We observe that the SQMC-EKF method attains significantly lower running times compared to the SQMC-EnKF, since the former increases linearly with dimension while the latter increases its cost exponentially. However, the SQMC-EKF obtains an MSE that increases with the dimension d_x, while the values of MSE for the SQMC-EnKF method are steady w.r.t. d_x.

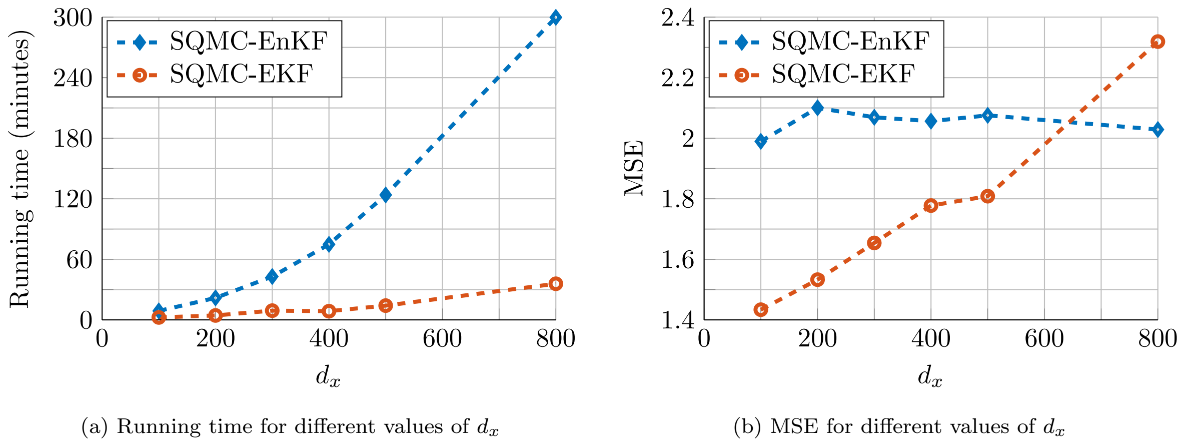
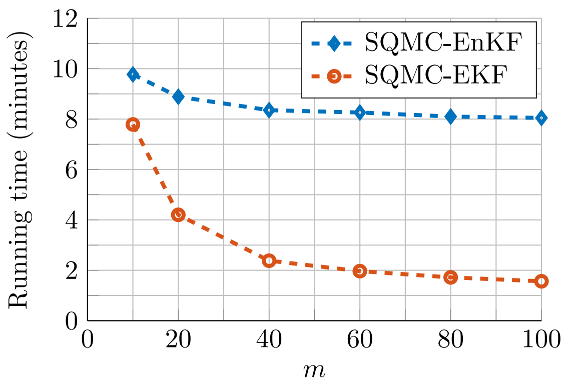


Figure 3.2: Comparison of the SQMC-EKF (red lines) and SQMC-EnKF (blue lines) in terms of their running time (plot 3.2a) and their MSE (plot 3.2b) as the state dimension d_x increases, with a fixed gap between observations of $m = 20$ discrete time steps.

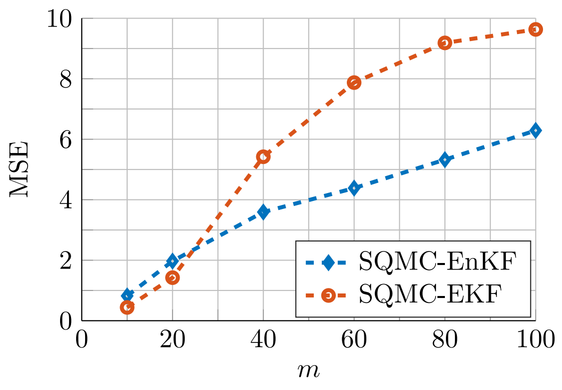
Next, Fig. 3.3 displays the running times and the average MSEs attained by the two NHFs as we increase the gap between observations from $m = 10$ to $m = 100$ (hence, from $m\Delta = 0.05$ to $m\Delta = 0.50$ continuous time units). The dimension of the state for this experiment is fixed to $d_x = 100$. Note that, as the gap m increases, less data points are effectively available for the estimation of both the parameters and the states. We observe, again, that the SQMC-EnKF is computationally more costly than the SQMC-EKF, however it attains a consistently smaller MSE when the gap between observations increases, suggesting that it may be a more efficient algorithm in data-poor scenarios.

Finally, we show results for a computer experiment in which we have used the SQMC-EnKF method to estimate the parameters F and a and track the state variables of the two-scale Lorenz 96 system with dimension $d_x = 5,000$ and a gap between consecutive observations of $m\Delta = 0.05$ continuous-time units ($m = 10$). As in the rest of computer simulations, the number of particles used to approximate the sequence of parameter posterior distributions is $N = 100$.

Figure 3.4 shows the true state trajectories, together with their estimates, for the first two slow state variables of the two-scale Lorenz 96 model. We note that the first variable, $x_1(\tau)$, is observed in Gaussian noise (with $\sigma_o = 4$) while the second variable, $x_2(\tau)$, is not observed. The accuracy of the estimation is similar, though, over the 20 continuous-time units of the simulation run (corresponding to $\frac{20}{\Delta} = 4000$ discrete time steps), achieving and MSE ≈ 0.87. Taking into account the steadiness of MSE w.r.t. dimension



(a) Running time for different values of m

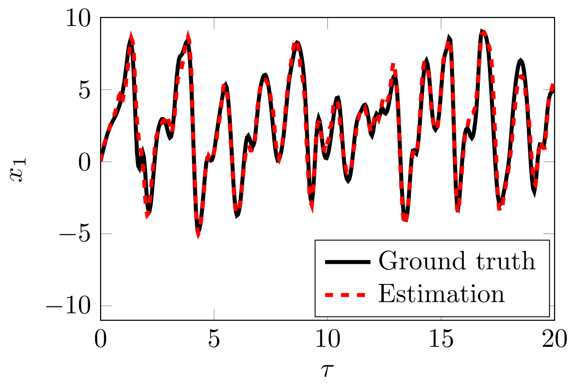


(b) MSE for different values of m

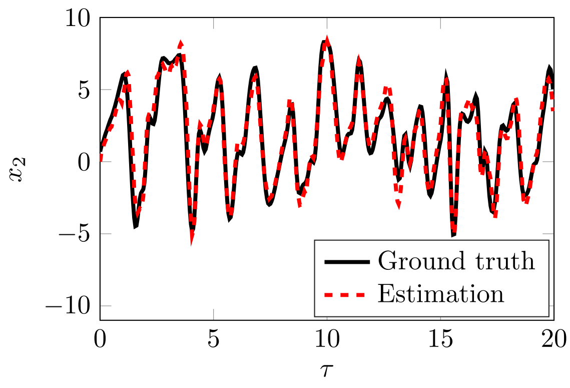
Figure 3.3: Comparison of the SQMC-EKF (red lines) and SQMC-EnKF (blue lines) in terms of their running time (plot 3.3a) and their MSE (plot 3.3b) as the gap between observations m increases, with fixed state dimension $d_x = 100$.

of SQMC-EnKF in Figure 3.2b and the values of MSE shown in Figure 3.3b for the gap selected in this experiment ($m = 10$), the results obtained are within the expected range.

In Figure 3.5 we observe the estimated posterior pdfs of the fixed parameters F, a_1 and a_2, together with the reference values. Note that the value $F = 8$ is ground truth, but the values of a_1 and a_2 are genie-aided least-squares estimates obtained by observing directly the fast variables of the two-scale model. Figure 3.5a displays the approximate posterior pdf of the parameter F (red dashed line) together with the true value $F = 8$ (vertical black line). We observe that nearly all probability mass is allocated close to the true value. In Fig. 3.5b we compare the approximate pdf of the coefficients $\mathsf{a} = [\mathsf{a}_1, \mathsf{a}_2]^\top$ produced by the NHF (dashed contour lines) with a kernel density estimator computed from the least-squares genie-aided estimates obtained from 100 independent simulations with the same setting (solid contour lines). The modes of the two pdfs are slightly shifted but the two functions are otherwise similar. The genie-aided estimate of a is located in a high probability region of the density function computed by the NHF.

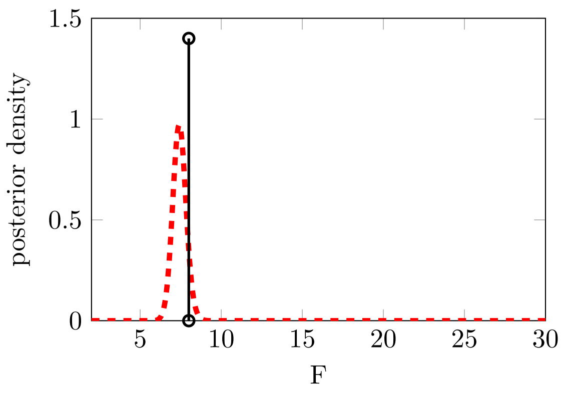


(a) Time sequence of x_1

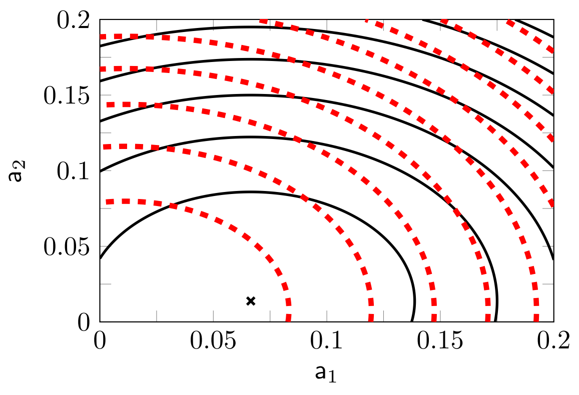


(b) Time sequence of x_2

Figure 3.4: Sequences of state values (black line) and estimates (dashed red line) in x_1 (plot 3.4a) and x_2 (plot 3.4b) over time. Variable x_1 is observed (in Gaussian noise), while x_2 is unobserved.



(a) Posterior pdf of F



(b) Posterior pdf of $\mathsf{a} = [\mathsf{a}_1, \mathsf{a}_2]^\top$

Figure 3.5: Posterior density of the parameters $\mathsf{a} = [\mathsf{a}_1, \mathsf{a}_2]^\top$ and F at $\tau = 5$ in a 5,000-dimensional Lorenz 96 model (red dashed lines). The true value of F (in plot 3.5a) is indicated by a black vertical line, while the location of reference values of a (in plot 3.5b) is marked by a black star. Note that there is no ground truth for the parameters in a.

3.6 Conclusions

We have introduced a nested filtering methodology to recursively estimate the static parameters and the dynamic variables of nonlinear dynamical systems. The proposed framework combines a recursive Monte Carlo approximation method to compute the posterior probability distribution of the static parameters with a variety of filtering techniques to estimate the posterior distribution of the state variables of the system. In particular, we have investigated the use of Gaussian filters in the second layer of the nested structure, as they admit fast implementations that can be well suited to high dimensional systems. As a result, we have proposed a class of nested hybrid filters that combine Monte Carlo and quasi Monte Carlo schemes for the (moderate dimensional) unknown static parameters of the dynamical system with either EKFs or EnKFs for the (higher dimensional) time-varying states. Additionally, when SMC is applied in the first layer of the NHF scheme, we have proved that the algorithm converges with rate $\mathcal{O}\left(N^{-\frac{1}{2}}\right)$ to a well defined limit distribution. We have presented numerical results for a two-scale stochastic Lorenz 96 system, a model commonly used for the assessment of data assimilation methods in geophysics. We illustrate the average performance of the methods in terms of estimation errors and running times, and show numerical results for a 5,000-dimensional system.

4 Nested Gaussian Filters

This chapter is devoted to the extension of the nested hybrid filters (NHFs), introduced in Chapter 3. To be specific, we devise methods that employ non-Monte Carlo schemes in both layers of the nested filtering structure. This results in a new set of algorithms that we refer to as nested Gaussian filters (NGFs). The problem statement and the state space models are introduced in Section 4.1, followed by the detailed description of the NGFs in Section 4.2. In Sections 4.3 and 4.4, we present numerical results for the stochastic Lorenz 63 model and a stochastic volatility model, respectively. Conclusions are explained in Section 4.5. The key notation for this chapter is summarized in Table 4.1.

4.1 Problem Statement

We tackle the same inference problem as in Chapter 3. In particular, we look into state-space dynamical systems with additive noise that can be described by the pair of equations

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) + \boldsymbol{v}_t, \tag{4.1}$$

$$\boldsymbol{y}_t = g(\boldsymbol{x}_t, \boldsymbol{\theta}) + \boldsymbol{r}_t, \tag{4.2}$$

where $t \in \mathbb{N}$ denotes discrete time, $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$ is the d_x-dimensional system state, $f\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \longrightarrow \mathbb{R}^{d_x}$ and $g\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \longrightarrow \mathbb{R}^{d_y}$, $d_x \geq d_y$, are possibly nonlinear functions parameterized by a (random but fixed) vector of unknown parameters, $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$, $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$ is the observation vector at time t and $\boldsymbol{v}_t$ and $\boldsymbol{r}_t$ are zero-mean random vectors playing the roles of state and observation noises.

Table 4.1: Notation of Chapter 4.

$\hat{\boldsymbol{C}}^{\boldsymbol{x}}_{t\|t-1,\boldsymbol{\theta}'}$	Predictive state covariance matrix estimate conditional on $\boldsymbol{\theta}'$ at time t
$\hat{\boldsymbol{C}}^{\boldsymbol{x}}_{t\|t,\boldsymbol{\theta}'}$	Posterior state covariance matrix estimate conditional on $\boldsymbol{\theta}'$ at time t
$\hat{\boldsymbol{C}}^{\boldsymbol{\theta}}_{t}$	Parameter covariance matrix estimate at time t
d_x	State dimension
d_y	Observation dimension
d_θ	Parameter dimension
f and g	State transition and observation function, respectively
$\boldsymbol{J}_{f,\boldsymbol{x}'}$	Jacobian matrix of function f evaluated at point $\boldsymbol{x}'$
$\boldsymbol{J}_{g,\boldsymbol{x}'}$	Jacobian matrix of function g evaluated at point $\boldsymbol{x}'$
k_o	Parameter of the observation equation of the Lorenz 63 model
M	Number of reference points or samples (in the first layer)
M_o	Number of times steps between each observation
$\boldsymbol{r}_t$	Observation noise vector
$\boldsymbol{R}$	Observation noise covariance matrix
t	Discrete-time steps
T	Length of the simulation run in continuous-time
$\boldsymbol{v}_t$	State noise vector
$\boldsymbol{V}$	State noise covariance matrix
$\boldsymbol{x}_t$	State vector
$\hat{\boldsymbol{x}}_{t\|t-1,\boldsymbol{\theta}'}$	Predictive mean state estimate conditional on the parameter $\boldsymbol{\theta}'$ at time t
$\hat{\boldsymbol{x}}_{t\|t,\boldsymbol{\theta}'}$	Posterior mean state estimate conditional on the parameter $\boldsymbol{\theta}'$ at time t
$\boldsymbol{y}_t$	Observation vector
Δ	Integration step
$\boldsymbol{\theta}$	Parameter vector
$\hat{\boldsymbol{\theta}}_t$	Mean parameter estimate
$\boldsymbol{\theta}^i_t$	Sample or reference point of the parameters
λ	Threshold for activate the iterative mode of the algorithm
τ	Continuous time

The same as in previous chapters, we also resort, when needed, to a specification of the state space model in terms of the relevant pdfs, namely

$$\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \quad \boldsymbol{\theta} \sim p(\boldsymbol{\theta}), \tag{4.3}$$

$$\boldsymbol{x}_t \sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \tag{4.4}$$

$$\boldsymbol{y}_t \sim p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}), \tag{4.5}$$

where $p(\boldsymbol{\theta})$ and $p(\boldsymbol{x}_0)$ are the a *priori* pdfs of the parameters and the state, respectively, $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ is the conditional density of the state $\boldsymbol{x}_t$ given $\boldsymbol{x}_{t-1}$ and the parameter vector $\boldsymbol{\theta}$, and $p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta})$ is the conditional pdf of the observation $\boldsymbol{y}_t$ given $\boldsymbol{x}_t$ and $\boldsymbol{\theta}$. We assume that $\boldsymbol{y}_t$ is conditionally independent of all other observations (given $\boldsymbol{x}_t$ and $\boldsymbol{\theta}$) and the prior pdfs of the state, $p(\boldsymbol{x}_0)$, and the parameters, $p(\boldsymbol{\theta})$, are known and the corresponding probability distributions are independent.

4.1.1 Model inference

The key difficulty in this class of models is the same as in the nested particle filter (NPF) [25] (see Section 2.4): the Bayesian estimation of the parameter vector $\boldsymbol{\theta}$. Therefore, we aim at computing the posterior pdf $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$, that can be written as

$$p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) = \int p(\boldsymbol{\theta}, \boldsymbol{x}_t|\boldsymbol{y}_{1:t})d\boldsymbol{x}_t, \tag{4.6}$$

leading naturally to approximations for $p(\boldsymbol{\theta}, \boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ for each t. This means that when computing $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ we may not only estimate the parameter vector $\boldsymbol{\theta}$, but we may also implicitly track the state dynamical variables. The main aim in this chapter is to obtain a Gaussian approximation of $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ within a nested Gaussian filtering scheme, whose second layer of filters will provide, in addition, Gaussian approximations for $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$.

4.2 Nested Gaussian filters

In this section, we introduce a class of nested filter for state-space models with unknown parameters that combine different types of Gaussian approximations in the inner and outer layers. We outline the methodology used to obtain the Gaussian approximations of $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ (in the outer layer) and $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ (in the inner layer).

We keep using $p(\cdot)$ to denote the actual pdfs. We aim, however, at constructing Gaussian approximations of the posterior pdfs induced by the state-space model of Eqs. (4.3)-(4.5) and the sequence of observations. In particular, we show how to recursively compute approximations $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) \approx \mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_t, \hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}})$, $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}}, \hat{\boldsymbol{C}}_{t|t,\boldsymbol{\theta}}^{\boldsymbol{x}})$ and $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) \approx \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{C}}_t^{\boldsymbol{x}})$.

4.2.1 Sequential Gaussian approximation

Let us aim at computing expectations of the form $\mathbb{E}[f(\boldsymbol{\theta})|\boldsymbol{y}_{1:t}] = \int f(\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})d\boldsymbol{\theta}$ for some test function of the parameters, $f(\boldsymbol{\theta})$. Different choices of $f(\cdot)$ enable us to compute different statistics, e.g., for $f(\boldsymbol{\theta}) = \boldsymbol{\theta}$ we obtain the posterior mean $\bar{\boldsymbol{\theta}}_t = \mathbb{E}[\boldsymbol{\theta}|\boldsymbol{y}_{1:t}]$ while if we let

$$f(\boldsymbol{\theta}) = \left(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}_t\right)^\top \left(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}_t\right)$$

then we obtain the posterior covariance matrix of $\boldsymbol{\theta}$. Using Bayes' rule, we can express the posterior pdf of $\boldsymbol{\theta}$ as

$$p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) = \frac{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})} \times p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1}), \tag{4.7}$$

hence, we can rewrite the posterior expectation as

$$\mathbb{E}[f(\boldsymbol{\theta})|\boldsymbol{y}_{1:t}] = \int \psi(\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})d\boldsymbol{\theta}, \tag{4.8}$$

where the function $\psi(\boldsymbol{\theta})$ is constructed as

$$\psi(\boldsymbol{\theta}) := \frac{f(\boldsymbol{\theta})p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})}. \tag{4.9}$$

If we assume that $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})$ is Gaussian, then we can approximate the integral of Eq. (4.8) using cubature rules [10] or the unscented transform (UT) [90]. Specifically, a Gaussian approximation $\mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_{t-1}, \boldsymbol{C}_{t-1}^{\boldsymbol{\theta}}) \approx$

$p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})$ can be represented at time t by a set of reference points and weights, $\{\boldsymbol{\theta}_{t-1}^i, w_{t-1}^i\}_{0\leq i\leq M-1}$, $M = 2d_\theta + 1$ (see Section 2.2.3), which in turn we may use to approximate the integral in Eq. (4.8) as

$$\int \psi(\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})d\boldsymbol{\theta} \approx \sum_{i=0}^{M-1} \psi(\boldsymbol{\theta}_{t-1}^i)w_{t-1}^i. \tag{4.10}$$

On the other hand, the pdf in the denominator of Eq. (4.9), $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})$, can be written as

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}) = \int p(\boldsymbol{y}_t, \boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})d\boldsymbol{\theta}, \tag{4.11}$$

where the joint pdf of $\boldsymbol{y}_t$ and $\boldsymbol{\theta}$ given all previous observations can be decomposed as

$$p(\boldsymbol{y}_t, \boldsymbol{\theta}|\boldsymbol{y}_{1:t-1}) = p(\boldsymbol{y}_t|\boldsymbol{\theta}, \boldsymbol{y}_{1:t-1})p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1}). \tag{4.12}$$

Then, the integral in Eq. (4.11) can also be approximated using the same set of reference points and weights as

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}) \approx \sum_{i=0}^{M-1} p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)w_{t-1}^i. \tag{4.13}$$

Finally, we can approximate the pdf $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$, $i = 0, \ldots, M-1$, using a bank of M Gaussian filters placed in the second layer of the nested filter as shown in Chapter 3. Once these densities are computed, we can approximate $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})$ as in Eq. (4.13).

The argument above enables us to approximate any integral $\int f(\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})d\boldsymbol{\theta}$. We can compute the mean vector and covariance matrix of a pdf $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) \approx \mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_t, \hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}})$ by taking $f(\boldsymbol{\theta}) = \boldsymbol{\theta}$ and $f(\boldsymbol{\theta}) = (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_t)^\top$, respectively, where

$$\hat{\boldsymbol{\theta}}_t = \int \boldsymbol{\theta} p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})d\boldsymbol{\theta}. \tag{4.14}$$

Specifically, we obtain the formulation for approximating the mean parameter vector, $\hat{\boldsymbol{\theta}}_t$, and its covariance matrix, $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$, sequentially as

$$\hat{\boldsymbol{\theta}}_t \approx \sum_{i=0}^{M-1} \boldsymbol{\theta}_{t-1}^i \frac{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})} w_{t-1}^i \quad \text{and} \tag{4.15}$$

$$\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}} \approx \sum_{i=0}^{M-1} (\boldsymbol{\theta}_{t-1}^i - \hat{\boldsymbol{\theta}}_t)(\boldsymbol{\theta}_{t-1}^i - \hat{\boldsymbol{\theta}}_t)^\top \frac{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})} w_{t-1}^i. \tag{4.16}$$

We outline the procedure for the sequential computation of the Gaussian approximations $\mathcal{N}(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}_t, \hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}) \approx p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$, $t = 1, 2, \ldots$, in Algorithm 8. The calculations done in the second layer of filters are summarized in step 2a. Notice that, at any time $t \geq 1$, we update the reference points $\boldsymbol{\theta}_{t-1}^i$, $i = 0, \ldots, M-1$, and, therefore, we need to run the M Gaussian filters in the second layer from scratch (i.e., from $n = 0$ to $n = t$) in order to (approximately) evaluate the densities $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$. Thus, Algorithm 8 is sequential but not recursive and, as a consequence, not well suited to handle long sequences of observations.

Algorithm 8 *Nested Gaussian filters.*

Inputs:

- *Prior pdfs $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{\theta})$. Assume that either $p(\boldsymbol{x}_0)$ is Gaussian or a Gaussian approximation is available.*

Procedure:

1. *Initialization*
   (a) *Generate M reference points, $\boldsymbol{\theta}_0^i$, from $p(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_0, \boldsymbol{C}_0^{\boldsymbol{\theta}})$ for $i = 0, \dots, M-1$, with weights w_0^i.*
2. *Sequential step, $t \geq 1$.*
   (a) *For each $i = 0, \dots, M-1$, use a Gaussian filter to approximately compute $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$.*
   (b) *Compute $\hat{\boldsymbol{\theta}}_t$ and $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$ via Eqs.* (4.15) *and* (4.16).
   (c) *Generate new reference points $\boldsymbol{\theta}_t^i$ and weights w_t^i, $i = 0, \dots, M-1$, from $\hat{\boldsymbol{\theta}}_t$ and $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$.*

Outputs: $\hat{\boldsymbol{\theta}}_t$ *and* $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$.

4.2.2 Recursive algorithm

For every new observation vector $\boldsymbol{y}_t$, in Algorithm 8 the pdfs $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ are computed by running the nested filters from time 0 until the current time t, which makes the computational cost increase with t^2. However, the entries of the covariance matrix, $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$, also tend to stabilize over time, which makes the difference between consecutive reference points, $\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i$, decrease. If we also assume that the function $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ is continuous in $\boldsymbol{\theta}$, then we can make the computation recursive by assuming that $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) \approx p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-2}^i)$ when $\boldsymbol{\theta}_{t-1}^i \approx \boldsymbol{\theta}_{t-2}^i$. For the sake of clarity we summarize the steps for computing $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ in Algorithm 9, relying on a bank of extended Kalman filters (EKFs) for the second layer of the scheme. Let us remark that this second layer can be implemented using a variety of filters, e.g., particle filters as in [25] or Gaussian filters as in 3, including unscented Kalman filters (UKFs) as we have done for the first layer. We choose a bank of EKFs simply because it is the computationally less demanding alternative.

Algorithm 10 outlines a recursive nested Gaussian filter with a UKF/cubature Kalman filter (CKF) in the first layer and EKFs in the second layer. It can be seen as a recursive and explicit implementation of Algorithm 8. The initialization remains the same (step 1a), computing M reference points $\boldsymbol{\theta}_0^i$ and weights w_0^i, $i = 0, \dots, M-1$, from the prior $p(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_0, \boldsymbol{C}_0^{\boldsymbol{\theta}})$. Also, we initialize the state and its covariance matrix in every Gaussian filter of the second layer (step 1b) by setting $\hat{\boldsymbol{x}}_0^i = \hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{C}}_0^{\boldsymbol{x},i} = \boldsymbol{C}_0^{\boldsymbol{x}}$, $i = 0, \dots, M-1$, from the prior $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0|\hat{\boldsymbol{x}}_0, \boldsymbol{C}_0^{\boldsymbol{x}})$.

Algorithm 9 *Extended Kalman filter conditional on $\boldsymbol{\theta}_{t-1}^i$, used in the second layer of the nested filter.*

Inputs:

- *Prior pdf $p(\boldsymbol{x}_0)$ and parameter vector $\boldsymbol{\theta}_{t-1}^i$.*
- *State-space model described in Eqs. (4.1) and (4.2). In particular, $f(\cdot)$ denotes the transition function in the state Eq. (4.1) and $g(\cdot)$ is the observation function in Eq. (4.2). The covariance of the state noise is denoted $\boldsymbol{V}$ and the covariance of the observation noise is denoted $\boldsymbol{R}$.*

Procedure:

1. *Initialization*

   (a) Assume $p(\boldsymbol{x}_0)$ is Gaussian with mean $\hat{\boldsymbol{x}}_0$ and covariance $\hat{\boldsymbol{C}}_0^{\boldsymbol{x}}$, i.e., $p(\boldsymbol{x}_0) \approx \mathcal{N}(\boldsymbol{x}_0 | \hat{\boldsymbol{x}}_0, \hat{\boldsymbol{C}}_0^{\boldsymbol{x}})$.

2. *Sequential step, $t \geq 1$.*

   (a) ***Prediction step.*** *Compute*

$$\begin{aligned} \hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i} &= f(\hat{\boldsymbol{x}}_{t-1|t-1,\boldsymbol{\theta}_{t-1}^i}, \boldsymbol{\theta}_{t-1}^i), \\ \hat{\boldsymbol{C}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}} &= \boldsymbol{J}_{f,\hat{\boldsymbol{x}}_{t-1|t-1,\boldsymbol{\theta}_{t-1}^i}} \hat{\boldsymbol{C}}_{t-1|t-1,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}} \boldsymbol{J}_{f,\hat{\boldsymbol{x}}_{t-1|t-1,\boldsymbol{\theta}_{t-1}^i}}^\top + \boldsymbol{V}, \end{aligned} \tag{4.17}$$

   where $\boldsymbol{J}_{f,\hat{\boldsymbol{x}}_{t-1|t-1,\boldsymbol{\theta}_{t-1}^i}}$ is the Jacobian matrix of $f(\cdot)$ evaluated at $\hat{\boldsymbol{x}}_{t-1|t-1,\boldsymbol{\theta}_{t-1}^i}$.

   (b) Approximate $p(\boldsymbol{x}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) \approx \mathcal{N}(\boldsymbol{x}_t | \hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}, \hat{\boldsymbol{C}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}})$ and compute

$$\begin{aligned} p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) &= \int p(\boldsymbol{y}_t | \boldsymbol{x}_t, \boldsymbol{\theta}_{t-1}^i) p(\boldsymbol{x}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) d\boldsymbol{x}_t \\ &\approx \int p(\boldsymbol{y}_t | \boldsymbol{x}_t, \boldsymbol{\theta}_{t-1}^i) \mathcal{N}(\boldsymbol{x}_t | \hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}, \hat{\boldsymbol{C}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}) d\boldsymbol{x}_t. \end{aligned} \tag{4.18}$$

   (c) ***Update step.*** *Compute*

$$\begin{aligned} \hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i} &= \hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i} + \boldsymbol{K}_t (\boldsymbol{y}_t - g(\hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}, \boldsymbol{\theta}_{t-1}^i)), && (4.19) \\ \hat{\boldsymbol{C}}_{t|t,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}} &= (\boldsymbol{I}_{d_x} - \boldsymbol{K}_t \boldsymbol{J}_{g,\hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}}) \hat{\boldsymbol{C}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}}, && (4.20) \\ \boldsymbol{K}_t &= \hat{\boldsymbol{C}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}} \boldsymbol{J}_{g,\hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}}^\top (\boldsymbol{J}_{g,\hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}} \hat{\boldsymbol{C}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}} \boldsymbol{J}_{g,\hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}}^\top + \boldsymbol{R})^{-1}, \end{aligned}$$

   where $\boldsymbol{J}_{g,\hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}}$ is the Jacobian matrix of $g(\cdot)$ evaluated at $\hat{\boldsymbol{x}}_{t|t-1,\boldsymbol{\theta}_{t-1}^i}$. Approximate $p(\boldsymbol{x}_t | \boldsymbol{y}_{1:t}, \boldsymbol{\theta}_{t-1}^i) \approx \mathcal{N}(\boldsymbol{x}_t | \hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i}, \hat{\boldsymbol{C}}_{t|t,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}})$.

Outputs: *$\hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i}$, $\hat{\boldsymbol{C}}_{t|t,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}}$ and $p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$.*

Algorithm 10 *Recursive nested Gaussian filters.*

```
Inputs:
    - Prior pdfs $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{\theta})$.
    - A fixed threshold $\lambda > 0$.
Procedure:
    1. Initialization
        (a) Generate $M$ reference points, $\boldsymbol{\theta}_0^i$, for $p(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta}_0, \boldsymbol{C}_0^{\boldsymbol{\theta}})$, $i = 0, \ldots, M-1$, with weights $w_0^i$.
        (b) If $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0 | \hat{\boldsymbol{x}}_0, \boldsymbol{C}_0^{\boldsymbol{x}})$, then set $\hat{\boldsymbol{x}}_0^i = \hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{C}}_0^{\boldsymbol{x},i} = \boldsymbol{C}_0^{\boldsymbol{x}}$ for $i = 0, \ldots, M-1$.
    2. Sequential step, $t \geq 1$.
        (a) For $i = 0, \ldots, M-1$:
            i. If $\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_p < \lambda \|\boldsymbol{\theta}_{t-2}^i\|_p$, then compute $p(\boldsymbol{x}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ from $p(\boldsymbol{x}_{t-1} | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) \approx p(\boldsymbol{x}_{t-1} | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-2}^i)$, where $p(\boldsymbol{x}_{t-1} | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-2}^i) \approx \mathcal{N}(\boldsymbol{x}_{t-1} | \hat{\boldsymbol{x}}_{t-1|t-1,\boldsymbol{\theta}_{t-2}^i}, \hat{\boldsymbol{C}}_{t-1|t-1,\boldsymbol{\theta}_{t-2}^i}^{\boldsymbol{x}})$.
               Else, approximate $p(\boldsymbol{x}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ from the prior $p(\boldsymbol{x}_0)$.
            ii. Use $p(\boldsymbol{x}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ to compute $p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$.
        (b) Compute $\hat{\boldsymbol{\theta}}_t$, $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$, $\hat{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{C}}_t^{\boldsymbol{x}}$ from Eqs. (4.15), (4.16), (4.27) and (4.28), respectively.
        (c) Generate reference points $\boldsymbol{\theta}_t^i$ and weights $w_t^i$ from $\hat{\boldsymbol{\theta}}_t$ and $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$ for $i = 0, \ldots, M-1$.
Outputs: $\hat{\boldsymbol{x}}_t$, $\hat{\boldsymbol{\theta}}_t$, $\hat{\boldsymbol{C}}_t^{\boldsymbol{x}}$ and $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$.
```

The sequential procedure starts by approximating $p(\boldsymbol{x}_t | \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ with the second layer of Gaussian filters (step 2(a)i). This is done differently depending on whether we assume $\boldsymbol{\theta}_{t-1}^i \approx \boldsymbol{\theta}_{t-2}^i$ or not. To be specific, the norm[1] $\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_p$ is computed and compared against a prescribed relative threshold $\lambda > 0$ in order to determine whether the prediction and update steps in the second layer of filters can be performed recursively or not. Specifically:

[1]Although other metrics $d(\boldsymbol{\theta}_{t-1}^i, \boldsymbol{\theta}_{t-2}^i)$ could be used, we adopt p-norms of the difference $\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i$ for this work. This is a flexible setup that admits several variants, e.g.,

$$\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_1 = \sum_{j=1}^{d_\theta} |\boldsymbol{\theta}_{t-1,j}^i - \boldsymbol{\theta}_{t-2,j}^i|, \tag{4.21}$$

$$\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_2 = \sqrt{\sum_{j=1}^{d_\theta} (\boldsymbol{\theta}_{t-1,j}^i - \boldsymbol{\theta}_{t-2,j}^i)^2} \quad \text{and} \tag{4.22}$$

$$\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_\infty = \max_{1 \leq j \leq d_\theta} |\boldsymbol{\theta}_{t-1,j}^i - \boldsymbol{\theta}_{t-2,j}^i|; \tag{4.23}$$

i.e., the taxicab norm or Manhattan norm ($p = 1$), the Euclidean norm ($p = 2$) and the maximum norm ($p = \infty$) respectively.

- If $\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_p < \lambda\|\boldsymbol{\theta}_{t-2}^i\|_p$ is not satisfied for $\boldsymbol{\theta}_{t-1}^i$, the i-th filter runs from scratch following the scheme in Algorithm 9.
- When $\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_p < \lambda\|\boldsymbol{\theta}_{t-2}^i\|_p$ is satisfied for $\boldsymbol{\theta}_{t-1}^i$, only one prediction and update step (from time $t-2$ to time $t-1$) is needed. In particular, we make the approximation $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) \approx p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-2}^i)$.

In either case, we use $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ in order to compute $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ as in step 2b of Algorithm 9. Finally, we can compute the mean vector $\hat{\boldsymbol{\theta}}_t$ and the covariance matrix $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta},i}$ at time t in step 2b, by using Eqs. (4.15) and (4.16). We prepare the new reference points $\boldsymbol{\theta}_t^i$ and their weights w_t^i from $\mathcal{N}(\hat{\boldsymbol{\theta}}_t, \hat{\boldsymbol{C}}_t^{\boldsymbol{\theta},i})$ for the next time step.

4.2.3 State tracking

We can take advantage of the filters in the second layer in order to provide state estimates as well. Let us write the expectation of $\boldsymbol{x}_t$ as

$$\mathbb{E}[\boldsymbol{x}_t|\boldsymbol{y}_{1:t}] = \int_D \Big[\int_{\mathcal{X}} \boldsymbol{x}_t p(\boldsymbol{x}_t|\boldsymbol{\theta}, \boldsymbol{y}_{1:t}) d\boldsymbol{x}_t\Big] p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) d\boldsymbol{\theta}, \tag{4.24}$$

where $D \subseteq \mathbb{R}^{d_\theta}$ and $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ denote the support sets of the parameter $\boldsymbol{\theta}$ and the state $\boldsymbol{x}$, respectively, and the integral in square brackets can be approximated by the M Gaussian filters of the second layer. In this case, we assume they are the EKFs of Algorithm 9 conditional on $\boldsymbol{\theta} = \boldsymbol{\theta}_{t-1}^i$. This yields a Gaussian approximation $p(\boldsymbol{x}_t|\boldsymbol{\theta}_{t-1}^i, \boldsymbol{y}_{1:t}) \approx \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i}, \hat{\boldsymbol{C}}_{t|t,\boldsymbol{\theta}_{t-1}^i})$, where

$$\hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i} \approx \mathbb{E}[\boldsymbol{x}_t|\boldsymbol{\theta}_{t-1}^i, \boldsymbol{y}_{1:t}] \quad \text{and} \tag{4.25}$$

$$\hat{\boldsymbol{C}}_{t|t,\boldsymbol{\theta}_{t-1}^i}^{\boldsymbol{x}} \approx \mathbb{E}[(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i})(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i})^\top|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_{t-1}^i]. \tag{4.26}$$

Then, a Gaussian approximation $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) \approx \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{C}}_t^{\boldsymbol{x}})$ can be constructed, where

$$\hat{\boldsymbol{x}}_t \approx \sum_{i=0}^{M-1} \hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i} \frac{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})} w_{t-1}^i \quad \text{and} \tag{4.27}$$

$$\hat{\boldsymbol{C}}_t^{\boldsymbol{x}} \approx \sum_{i=0}^{M-1} (\hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i} - \hat{\boldsymbol{x}}_t)(\hat{\boldsymbol{x}}_{t|t,\boldsymbol{\theta}_{t-1}^i} - \hat{\boldsymbol{x}}_t)^\top \frac{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})} w_{t-1}^i. \tag{4.28}$$

4.2.4 Continuity of the conditional filter pdf

The key to keep Algorithm 10 recursive is the test in step 2a, which sets

$$p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) \approx \mathcal{N}(\boldsymbol{x}_{t-1}|\hat{\boldsymbol{x}}_{t-1|t-1,\boldsymbol{\theta}_{t-2}^i}, \hat{\boldsymbol{C}}_{t-1|t-1,\boldsymbol{\theta}_{t-2}^i}^{\boldsymbol{x}}) \tag{4.29}$$

when $\frac{\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_p}{\|\boldsymbol{\theta}_{t-2}^i\|_p} < \lambda$ for some prescribed threshold $\lambda > 0$. This step relies on the assumption that $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \approx p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}')$ when $\boldsymbol{\theta} \approx \boldsymbol{\theta}'$, i.e., we are assuming that the conditional filtering pdf $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ is a continuous function of the parameter $\boldsymbol{\theta}$. In this section we state sufficient conditions for the conditional filter $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ to be Lipschitz-continuous.

For conciseness, let us denote

$$\pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) := p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t},\boldsymbol{\theta}), \tag{4.30}$$

$$\xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) := p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta}), \quad \text{and} \tag{4.31}$$

$$\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta}) := p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta}). \tag{4.32}$$

Hereafter we assume that the observation sequence $\{\boldsymbol{y}_t, t \geq 1\}$ is arbitrary but fixed (i.e., deterministic). Additionally, we impose the following regularity assumptions:

A. 4 *The conditional pdfs $\pi_t(\boldsymbol{x}_t|\boldsymbol{\theta})$, $\xi_t(\boldsymbol{x}_t|\boldsymbol{\theta})$ and $\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})$ exist for every $t \geq 1$, every $\boldsymbol{x}_t \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ and every parameter vector $\boldsymbol{\theta} \in D \subseteq \mathbb{R}^{d_\theta}$, where D denotes the parameter space.*

A. 5 *The transition pdf $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{\theta})$ is Lipschitz w.r.t. $\boldsymbol{\theta}$, i.e., there exists a constant $0 < L < \infty$ such that*

$$\sup_{\boldsymbol{x}_{t-1}\in\mathcal{X}} \int |p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{\theta}) - p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{\theta}')|d\boldsymbol{x}_t < L\|\boldsymbol{\theta}-\boldsymbol{\theta}'\| \tag{4.33}$$

for every $t \geq 1$ and every pair $(\boldsymbol{\theta},\boldsymbol{\theta}') \in D \times D$.

Remark 1 *In Assumption 5, we denote $\|\boldsymbol{\theta}-\boldsymbol{\theta}'\| = \sqrt{\sum_{i=1}^{d_\theta}(\theta_i - \theta_i')^2}$, the Euclidean distance between $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$.*

A. 6 *The conditional pdfs $p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta})$ are strictly positive and uniformly Lipschitz w.r.t. $\boldsymbol{\theta}$. In particular, $p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta}) > 0$ and*

$$\sup_{\boldsymbol{x}_t\in\mathcal{X}} \frac{|p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta}) - p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta}')|}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} < G_t\|\boldsymbol{\theta}-\boldsymbol{\theta}'\| \tag{4.34}$$

for some positive $G_t < \infty$.

A. 7 *The ratio $\frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta})}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})}$ is bounded. Specifically, there exist finite constants $0 < M_t < \infty$ such that*

$$\sup_{\substack{\boldsymbol{\theta}\in D\\ \boldsymbol{x}_{t-1}\in\mathcal{X}}} \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta})}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} < M_t. \tag{4.35}$$

Assumptions 4 and 5 are rather mild and easy to check for a given state-space model. Assumptions 6 and 7, on the other hand, may be restrictive in some problems. We note, however, that for fixed $\boldsymbol{y}_t$, $t \geq 1$, and a compact parameter support $D \subset \mathbb{R}^{d_\theta}$, the factor $\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})$ can often be bounded away from 0, while $p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta})$ is typically upper bounded. In any case, Assumptions 4-7 lead to the result below regarding the continuity of the filter, $\pi_t(\boldsymbol{x}_t|\boldsymbol{\theta})$, and predictive, $\xi_t(\boldsymbol{x}_t|\boldsymbol{\theta})$, pdfs w.r.t. the parameter vector $\boldsymbol{\theta}$.

Proposition 1 *: If Assumptions 4 to 7 hold, there exist sequences of finite constants $\tilde{L}_t$ and L_t such that, for $t \geq 1$,*

$$\int |\xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')|d\boldsymbol{x}_t \leq \tilde{L}_t\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|, \quad \textit{and} \tag{4.36}$$

$$\int |\pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')|d\boldsymbol{x}_t \leq L_t\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|. \tag{4.37}$$

Proof: See Appendix D. □

4.3 Example: the stochastic Lorenz 63 model

4.3.1 Stochastic Lorenz 63 model

Consider the 3-dimensional continuous-time stochastic process $\boldsymbol{x}(\tau) = [x_1(\tau), x_2(\tau), x_3(\tau)]^\top$, where $\tau \in (0, \infty)$ denotes continuous time, taking values on $\mathbb{R}^3$, whose dynamics are described by the system of stochastic differential equations (SDEs)

$$dx_1 = -S(x_1 - x_2) + \sigma dv_1, \tag{4.38}$$

$$dx_2 = Rx_1 - x_2 - x_1x_3 + \sigma dv_2, \tag{4.39}$$

$$dx_3 = x_1x_2 - Bx_3 + \sigma dv_3, \tag{4.40}$$

where the v_i's are independent 1-dimensional Wiener processes, $\sigma > 0$ is a known scale parameter and $S, R, B \in \mathbb{R}$ are unknown static model parameters. Eqs. 4.38-4.40 are a stochastic version of the Lorenz 63 model [67, 73]. Using the Euler-Maruyama scheme in order to integrate the SDEs of Eqs. (4.38)–(4.40), it is straightforward to cast them into the discrete-time state equation

$$\boldsymbol{x}_{t+1} = f_\Delta(\boldsymbol{x}_t, \boldsymbol{\theta}) + \sqrt{\Delta}\sigma \boldsymbol{v}_t, \quad t = 1, 2, \dots \tag{4.41}$$

where $f_\Delta : \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_x}$ ($d_x = d_\theta = 3$) is the function defined by

$$\begin{aligned} f_{1,\Delta}(\boldsymbol{x}_t, \boldsymbol{\theta}) &= x_{1,t} - \Delta S(x_{1,t} - x_{2,t}), \\ f_{2,\Delta}(\boldsymbol{x}_t, \boldsymbol{\theta}) &= x_{2,t} + \Delta[(R - x_{3,t})x_{1,t} - x_{2,t}], \\ f_{3,\Delta}(\boldsymbol{x}_t, \boldsymbol{\theta}) &= x_{3,t} + \Delta(x_{1,t}x_{2,t} - Bx_{3,t}), \end{aligned}$$

Δ is the integration time-step (given in continuous-time units), $\boldsymbol{\theta} = (S, R, B)^\top$ is the 3×1 vector of unknown parameters and $\boldsymbol{v}_t$ is a sequence of 3-dimensional Gaussian independent random vectors with zero mean and covariance matrix $\boldsymbol{I}_3$ (with $\boldsymbol{I}_d$ denoting the $d \times d$ identity matrix). Hence, the state transition density $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ is Gaussian and can be written down as $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_t|f_\Delta(\boldsymbol{x}_{t-1}, \boldsymbol{\theta}), \sigma^2\Delta\boldsymbol{I}_{d_x})$. This function is Lipschitz on $\boldsymbol{\theta}$.

In order to complete the specification of a state space model, we need to characterize the observations. For our simulation setup we assume linear observations of the form

$$\boldsymbol{y}_t = k_o \begin{bmatrix} x_{1,t} \\ x_{3,t} \end{bmatrix} + \boldsymbol{r}_t, \tag{4.42}$$

where k_o is a fixed parameter and $\boldsymbol{r}_t \sim \mathcal{N}(\boldsymbol{r}_t|\boldsymbol{0}, \sigma_y^2\boldsymbol{I}_2)$ is a 2-dimensional additive noise with zero mean and covariance function $\sigma_y^2\boldsymbol{I}_2$. Therefore, the conditional part of the observations (and hence the likelihood function) is also Gaussian and can be written as $p(\boldsymbol{y}_t|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{G}\boldsymbol{x}_t, \sigma_y^2\boldsymbol{I}_2)$, where $\boldsymbol{G} = \begin{bmatrix} k_o & 0 & 0 \\ 0 & 0 & k_o \end{bmatrix}$ is the observation matrix. This function has a finite upper bound independent of $\boldsymbol{\theta}$.

Observations are not collected at every time t. Instead we assume that an observation vector is received every M_o steps of the state Eq. (4.41) (every $\Delta \times M_o$ continuous-time units). Therefore, we can rewrite the observations as

$$\boldsymbol{y}_t = \begin{cases} k_o \begin{bmatrix} x_{1,t} \\ x_{3,t} \end{bmatrix} + \boldsymbol{r}_t & \text{if } t = kM_o,\ k \in \mathbb{Z} \\ \boldsymbol{r}_t & \text{if } t \neq kM_o \end{cases} . \tag{4.43}$$

4.3.2 Simulation setup

For our computer experiments we have used the stochastic Lorenz 63 model outlined in Eqs. (4.41) and (4.42) in order to generate signals $\boldsymbol{x}_t$ and $\boldsymbol{y}_t, t = \{0, 1, \ldots\}$, used as the ground truth and the data, respectively, for the assessment of the algorithm. We integrate the model with the step-size $\Delta = 2 \times 10^{-4}$ continuous-time units. The true parameters for the generation of the signal and data are $S = 10$, $R = 28$ and $B = \frac{8}{3}$ (which yield underlying chaotic dynamics); while the initial state is Gaussian with mean[2] $\hat{\boldsymbol{x}}_0 = [-6, -5.5, -24.5]^\top$ and covariance matrix $\boldsymbol{I}_3$, i.e., $p(\boldsymbol{x}_0|\hat{\boldsymbol{x}}_0, \boldsymbol{I}_3)$. The noise scale factors, $\sigma^2 = 0.1$ and $\sigma_y^2 = 1$, as well as the constant $k_o = 5$, are assumed known.

For the estimation task we use Algorithm 10. We assume a Gaussian prior distribution for the unknown parameters, namely $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_\theta, \sigma_\theta^2 \boldsymbol{I}_3)$, where the a priori mean $\boldsymbol{\mu}_\theta$ is drawn at random from a uniform distribution $\mathcal{U}(\boldsymbol{\theta}_\star - \boldsymbol{\epsilon}, \boldsymbol{\theta}_\star + \boldsymbol{\epsilon})$ for each independent simulation and $\sigma_\theta^2 = 1$. $\boldsymbol{\theta}_\star = [10, 28, \frac{8}{3}]^\top$ is the true parameter vector and the offset vector is $\boldsymbol{\epsilon} = [3, 1, 0.5]^\top$. The algorithm does not collect an observation at every time step, but every $M_o = 5$ discrete-time steps (i.e., every $M_o \times \Delta = 10^{-3}$ continuous-time units). Hence, the prediction step of the state variables at the second layer of the nested filter corresponds to $M_o = 5$ discrete-time steps of the difference Eq. (4.41). When an observation $\boldsymbol{y}_t$ (at discrete time $t = kM_o$, for some $k \in \mathbb{Z}$) arrives, both the state and parameter distributions are updated. The length of each simulation run is $T = 40$ continuous-time units, which amounts to $\frac{T}{\Delta} = 2 \times 10^5$ discrete-time steps of the state Eq. (4.41).

We have assessed the ability of several Bayesian computation algorithms to jointly track the state $\boldsymbol{x}_t$ and estimate the parameters $\boldsymbol{\theta} = (S, R, B)^\top$ of this model. To be specific, we have coded and run the following schemes:

- The proposed Algorithm 10 using a UKF in the first layer and a bank of EKFs in the second layer.
- A UKF [90] algorithm with state augmentation [46, 65] where the parameters are added to the state vector.
- An ensemble Kalman filter (EnKF) [36] algorithm with state augmentation as well.
- An NHF [91] with a sequential Monte Carlo (SMC) algorithm in the first layer and a bank of EKFs in the second layer.

Table 4.2 summarizes the computational complexity of the different algorithms that we have compared in our study. For the comparison, we have assumed a "roll-back probability" q that corresponds to the probability of the event

$$\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_p \geq \lambda \|\boldsymbol{\theta}_{t-2}^i\|_p$$

in step 2(a)i of Algorithm 10 for large t. In other words, q represents the average probability of Algorithm 10 to have to approximate the density $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i)$ starting from the prior $p(\boldsymbol{x}_0)$ at time t (an action which has a cost of order t^2). In practice, our simulations show that $q << 1$ for sufficiently large t.

The accuracy of the various algorithms is compared in terms of the normalized mean square error (NMSE) of the predictor of the state and the predictor of the parameters. We assess the empirical NMSE

[2]In order to obtain the initial vector $\hat{\boldsymbol{x}}_0$, we simulate a deterministic version of the Lorenz 63 model of Eq. (4.41) ($\sigma = 0$) for 20 continuous-time units. We set the initial state as the values of the variable $\boldsymbol{x}$ at the last time step of this simulation. This initialization is used in all simulations of this computer experiment in order to generate the "ground truth" sequence of $\boldsymbol{x}_t$ and the associated sequences of observations $\boldsymbol{y}_t$.

Algorithm	Computational complexity
UKF with state augmentation	$\mathcal{O}((d_\theta + d_x)^3 d_y^3 t)$
EnKF with state augmentation	$\mathcal{O}((d_\theta + d_x) d_y^3 M t)$
NPF	$\mathcal{O}(MNt)$
NHF (SMC-EKF)	$\mathcal{O}(d_x^2 d_y^3 N t)$
Algorithm 10 (UKF-EKF)	$\mathcal{O}\left(d_\theta^3 d_x^2 d_y^3 \left[qt^2 + (1-q)t\right]\right)$

Table 4.2: Computational complexity of the algorithms compared in the numerical study. For the EnKF with state-augmentation, M is the number of ensemble elements. For the NPF, N is the number of particles in the first layer and M is the number of particles in the second layer. For the SMC-EKF scheme, N is the number of particles in the first layer. For the UKF-EKF algorithm, q is the average "roll-back" probability in step 2(a)i of Algorithm 10.

resulting directly from the simulations, namely,

$$\text{NMSE}_{\boldsymbol{x},t} = \frac{\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\|^2}{\|\boldsymbol{x}_t\|^2}, \quad \text{NMSE}_{\boldsymbol{\theta},t} = \frac{\|\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_t\|^2}{\|\boldsymbol{\theta}_t\|^2}, \tag{4.44}$$

as well as the averages

$$\text{NMSE}_{\boldsymbol{x}} = \frac{\Delta}{T} \sum_{t=0}^{\frac{T}{\Delta}-1} \text{NMSE}_{x,t} \quad \text{and} \quad \text{NMSE}_{\boldsymbol{\theta}} = \frac{\Delta}{T} \sum_{t=0}^{\frac{T}{\Delta}-1} \text{NMSE}_{\boldsymbol{\theta},t}.$$

Finally, Table 4.3 presents a summary of the model and algorithm parameters, and their values, as needed to reproduce the simulation results in this section.

Parameter	Value
S	10
R	28
B	$\frac{8}{3}$
Δ	2×10^{-4}
T	40
M_o	5
k_o	5
$\boldsymbol{\theta}_\star$	$[10, 28, \frac{8}{3}]^\top$
$\hat{\boldsymbol{x}}_0$	$[-6, -5.5, -24.5]^\top$
σ^2	0.1
σ_y^2	1
σ_θ^2	1
$\boldsymbol{\epsilon}$	$[3, 1, 0.5]^\top$
λ	10^{-3}
M	10
N	120

Table 4.3: Model and algorithm parameters for the simulation setup of Section 4.3.3.

4.3.3 Numerical Results

In the first computer experiments we study the choice of norm $\|\boldsymbol{\theta}_{t-1}^i - \boldsymbol{\theta}_{t-2}^i\|_p$ in step 2(a)i of Algorithm 10. Specifically, we have considered a setup where the model parameters $\boldsymbol{\theta} = (S, R, B)^\top$ are assumed known and the goal is to track the state $\boldsymbol{x}_t$ using an EKF. We first generate a sequence of observations $\boldsymbol{y}_{1:T}$ from the model with parameters $\boldsymbol{\theta} = (10, 28, \frac{8}{3})^\top$. Then, for this sequence, the EKF runs with a perturbed set of parameters of the form $\boldsymbol{\theta}' = \boldsymbol{\theta} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_e^2)$ is a zero-mean Gaussian perturbation. We carry out 100 independent simulations for each value of σ_e^2 for $\sigma_e^2 = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

Fig. 4.1 summarizes the outcome of this experiment. In particular, it displays the NMSE in the tracking of $\boldsymbol{x}_t$, averaged over all 100 simulation runs, versus the average norms $\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_2$ and $\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_\infty$. The plot illustrates that:

1. The NMSE$_{\boldsymbol{x}}$ is an increasing magnitude w.r.t. the perturbation $\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_p$, both with Euclidean or maximum norms. The NMSE$_{\boldsymbol{x}}$ remains below 10^{-4} when $\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_p$ is approximately below 10^{-2}.

2. The NMSE$_{\boldsymbol{x}}$ is slightly higher when the parameter perturbation is given in terms of the norm $\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|_\infty$.

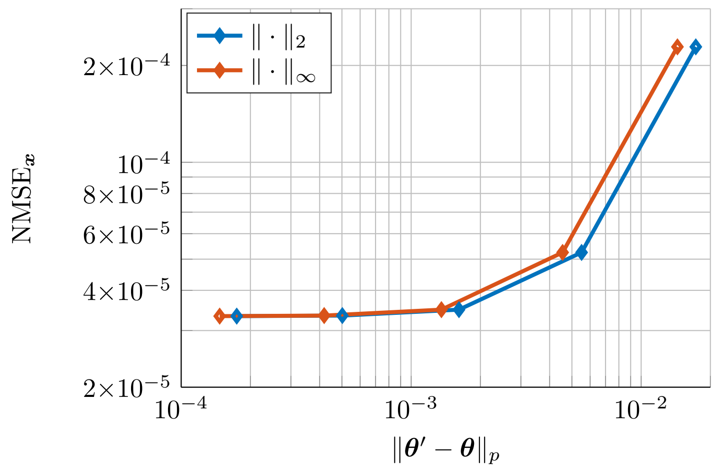
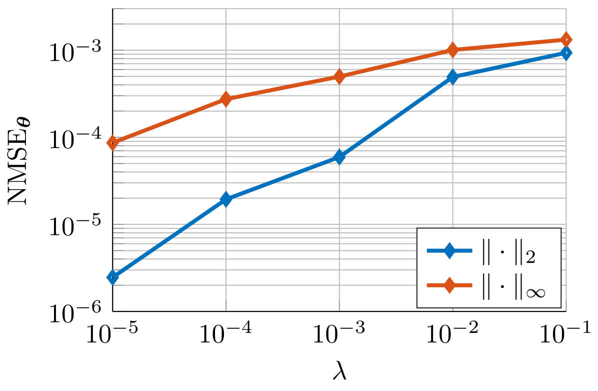


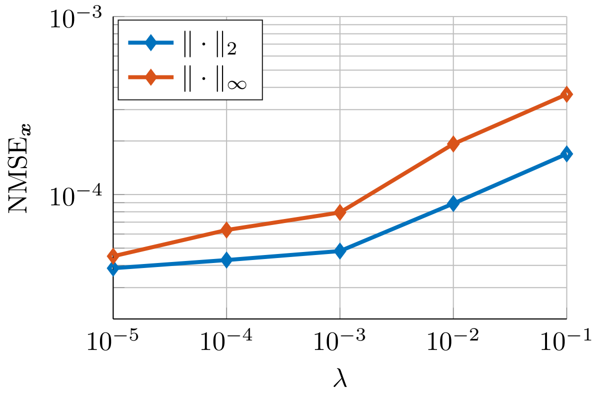
Figure 4.1: EKF performance with known parameters, $\boldsymbol{\theta}'$, that are obtained by modifying the true parameters, $\boldsymbol{\theta}$. In the abscissa axis, we represent the average distance of the simulation runs to the ground truh.

In a second experiment, Fig. 4.2 shows the results of using Algorithm 10 with both $\|\cdot\|_2$ and $\|\cdot\|_\infty$ norms for several values of λ. Again, each point of the graphs represents the average of 80 independent simulation runs. We display NMSE$_{\boldsymbol{\theta}}$, NMSE$_{\boldsymbol{x}}$ and run-times in minutes[3] in Figs. 4.2a, 4.2b and 4.2c, respectively. In Fig. 4.2a, we see that NMSE$_{\boldsymbol{\theta}}$ increases with λ. This is as expected because the larger λ, the worse the approximation $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) \sim p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-2}^i)$. We also see that the Euclidean norm $\|\cdot\|_2$ yields a smaller error. However, in the results obtained for NMSE$_{\boldsymbol{x}}$ in Fig. 4.2b, we observe that below $\lambda = 10^{-3}$ there is almost no improvement in the error, and the curve is similar to the one in Fig. 4.1. Finally, Fig. 4.2c shows that the runtime of the nested filtering Algorithm 10 increases significantly when $\lambda < 10^{-3}$ (because the algorithm takes longer to become strictly recursive). The comparison of the

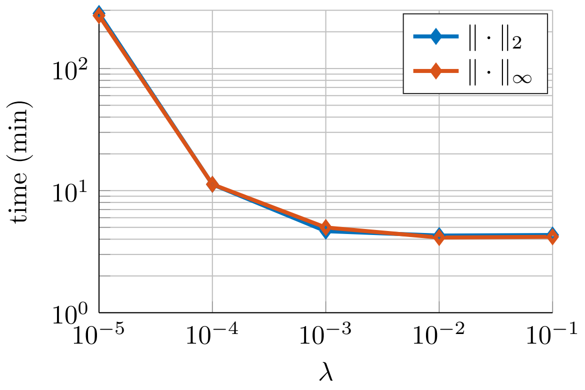
[3]The algorithms have been coded in MATLAB R2017a and run on a computer with 128 GB of DRAM and equipped with two Intel Xeon Gold 5115 10-Core CPU processors (running at 2.40 GHz).



(a) NMSE$_{\boldsymbol{\theta}}$ for different values of λ.



(b) NMSE$_{\boldsymbol{x}}$ for different values of λ.



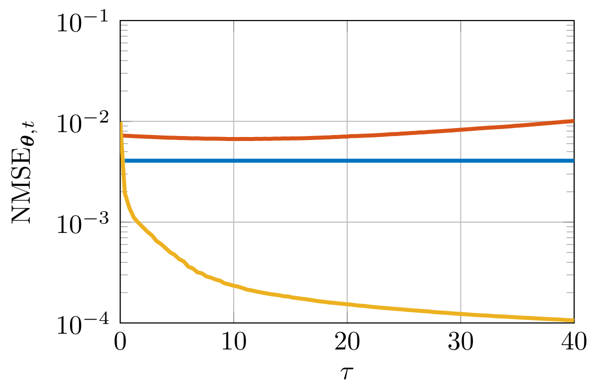
(c) Average simulation runtime.

Figure 4.2: The average NMSE and average time of simulation in minutes over 80 simulation runs for different values of λ, using the Euclidean norm $\|\cdot\|_2$ (in blue) and the maximum norm $\|\cdot\|_\infty$ (in red).

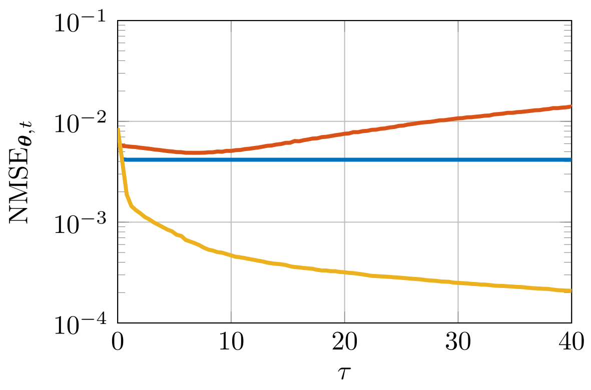
NMSE values in Figs. 4.2a and 4.2b with the run-time in Fig. 4.2c enables us to select the value of the threshold λ in order to attain a certain trade-off between computational cost and accuracy of the estimates provided by Algorithm 10. For the rest of the experiments in this section we set $\lambda = 10^{-3}$. When applied to different models, the effect of λ on the performance of Algorithm 10 can be different and adequate trade-offs may be attained for different values of the threshold. In a practical application, one may run a computer experiment with synthetic data in order to reproduce Figs. 4.2a, 4.2b and 4.2c for the model of interest and then choose the suitable value of λ to be used with real data.

In the next experiment we compare the proposed nested Gaussian filters (Algorithm 10) with two classical methods: the unscented Kalman filter (UKF) [52] and the ensemble Kalman filter (EnKF) [36], both relying on the state-augmentation technique [8, 64] to incorporate the unknown parameters. To be specific, this approach implies that the system state $\boldsymbol{x}_t$ is extended with the parameter vector to obtain the augmented state $\tilde{\boldsymbol{x}}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{\theta} \end{bmatrix}$. The UKF and EnKF algorithms are used to track $\tilde{\boldsymbol{x}}_t$ instead of $\boldsymbol{x}_t$.

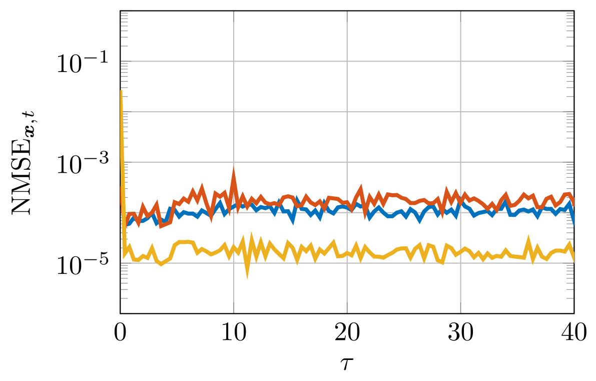
We have carried out two sets of computer simulations. In the first one we assume that the observation vectors are of the form $\boldsymbol{y}_t = k_o \boldsymbol{x}_t + \boldsymbol{r}_t$, i.e., all the state variables are observed in Gaussian noise. The results are displayed in Figs. 4.3a and 4.3c, which show the NMSE for the parameters $\boldsymbol{\theta}$ and the state $\boldsymbol{x}_t$ over time,



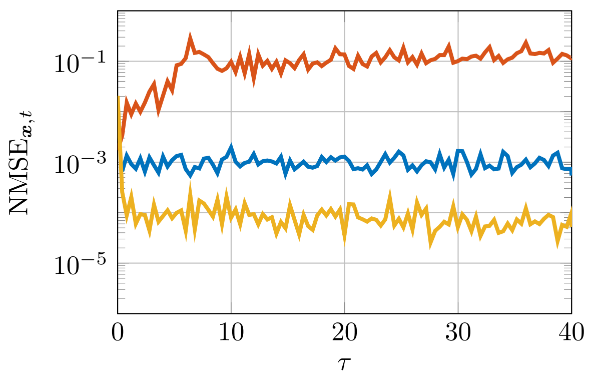
(a) $\mathrm{NMSE}_{\boldsymbol{\theta},t}$ observing $[x_1, x_2, x_3]^\top$.



(b) $\mathrm{NMSE}_{\boldsymbol{\theta},t}$ observing $[x_1, x_3]^\top$.



(c) $\mathrm{NMSE}_{\boldsymbol{x},t}$ observing $[x_1, x_2, x_3]^\top$.

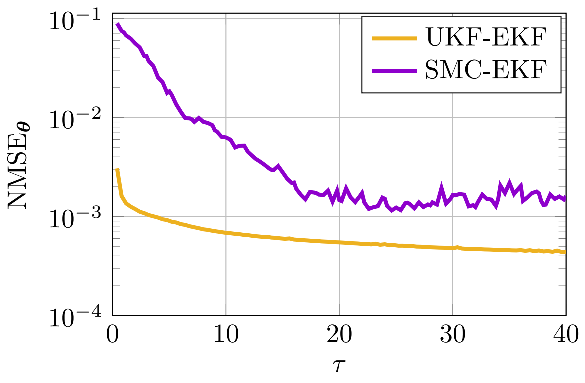


(d) $\mathrm{NMSE}_{\boldsymbol{x},t}$ observing $[x_1, x_3]^\top$.

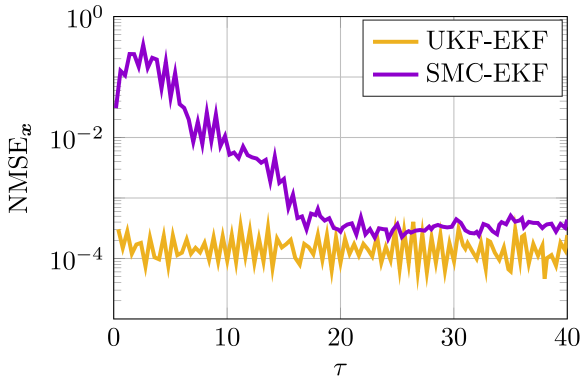
Figure 4.3: Performance of UKF (red), EnKF (blue) and UKF-EKFs (yellow) for two different setups, averaged over 50 independent simulation runs. Figs. 4.3a and 4.3c show $\mathrm{NMSE}_{\boldsymbol{\theta},t}$ and $\mathrm{NMSE}_{\boldsymbol{x},t}$ respectively, where the whole state vector is observed. In Figs. 4.3b and 4.3d, the error is plotted for a setup where only the first and third components of the state (x_1 and x_3) are observed.

respectively, for the three competing algorithms. The nested scheme outperforms the augmented-state methods clearly in terms of parameter estimation (Fig. 4.3a) and by a smaller margin in terms of state tracking (Fig. 4.3c). When the observations are reduced to two state variables $\boldsymbol{y}_t = k_o \begin{bmatrix} x_{1,t} \\ x_{3,t} \end{bmatrix} + \boldsymbol{r}_t$, in Gaussian noise, the advantage of the nested scheme becomes larger, as shown in Figs. 4.3b and 4.3d.

Next, for the same simulation setup of Figs. 4.3b and 4.3d, we compare the performance of the UKF-EKF nested filter (Algorithm 10) with one of the nested hybrid filters in [91]. The latter method consists of a SMC filter with $N = 120$ particles for the first layer and a bank of EKFs for the second layer. Figs. 4.4a and 4.4b show the $\mathrm{NMSE}_{\boldsymbol{\theta},t}$ and the $\mathrm{NMSE}_{\boldsymbol{x},t}$ respectively, for both the SMC-EKF (violet line) and the UKF-EKF (yellow line) methods. Although the time of convergence of the SMC-EKF scheme can be reduced, the UKF-EKF algorithm converges clearly faster. Also, once it converges, the estimation error for both parameters and states is slightly lower for the UKF-EKF method. However, the greatest improvement is related to the computational cost. For this experiment the UKF-EKF algorithm is three times faster (4.5 minutes run-time versus 14.8) than the SMC-EKF scheme. Therefore, it considerably reduces the computational cost while obtaining similar or slightly better results in estimation error.



(a) Averaged NMSE$_{\boldsymbol{\theta},t}$.



(b) Averaged NMSE$_{\boldsymbol{x},t}$.

Figure 4.4: Performance of a SMC-EKF (violet) and a UKF-EKF (yellow) averaged over 100 simulation runs. The SMC scheme propagates $N = 120$ samples over time. Fig. 4.4a shows NMSE$_{\boldsymbol{\theta},t}$ and Fig. 4.4b shows NMSE$_{\boldsymbol{x},t}$.

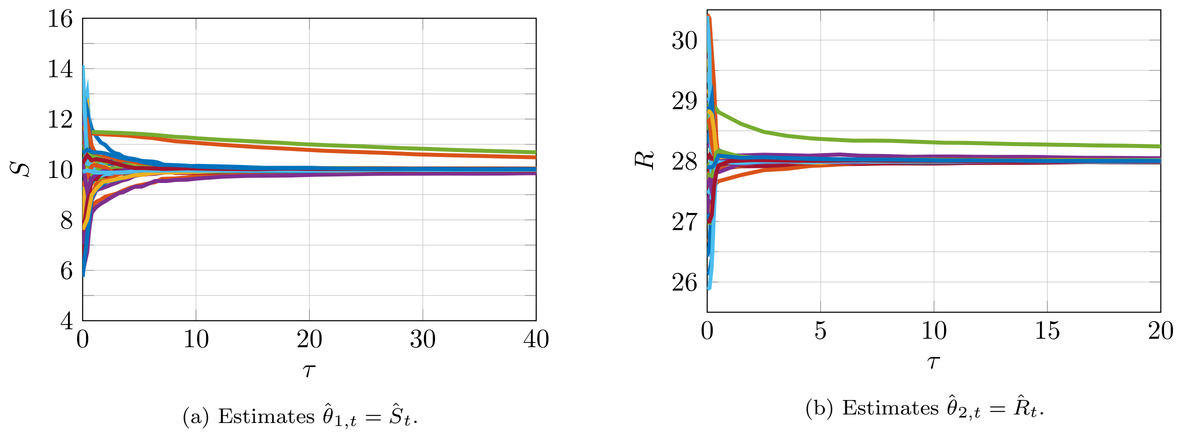
For the next computer experiment, Fig. 4.5 shows the parameter estimates obtained by running 50 independent simulations of the proposed UKF-EKF nested filter. The three dimensions of $\hat{\boldsymbol{\theta}}_t$ are displayed over time (Figs. 4.5a–4.5c) in order to illustrate how they converge as observations are collected. Although the length of the simulations is $T = 40$ continuous-time units, we have plotted just the intervals of time where the estimates converge. The interval varies from one plot to another because the time of convergence is not the same for all parameters (having shorter times for B and longer times for S). In spite of that, this figure shows how all parameters converge to the true values for different initializations. In the same vein, Fig. 4.6 illustrates the evolution of the three entries ($\sigma^2_{S,t}$,$\sigma^2_{R,t}$ and $\sigma^2_{B,t}$) on the diagonal of the covariance matrix $\hat{\boldsymbol{C}}^{\boldsymbol{\theta}}_t$ over time (the plots are an average of 50 independent simulation runs). The same as the mean values shown in Fig. 4.5, the estimated variances of the parameters S, R and B stabilize over time at similar rates, albeit with different steady-state values.

Fig. 4.7a, on the other hand, illustrates the accuracy of state estimates, $\hat{\boldsymbol{x}}_t$, by averaging the NMSE$_{\boldsymbol{x},t}$ obtained for the same set of 50 simulation runs as in Fig. 4.5. The error NMSE$_{\boldsymbol{x},t}$ decreases with time as the parameter estimates get closer to their true values, and its value stabilizes around $t = 5$. By that time, all parameter estimates in Fig. 4.5 have already converged (or at least got closer to their steady values) and, consequently, the state estimates become reliable.

In Figs. 4.7b, 4.7c and 4.7d, the estimated marginal pdfs of each element in $\hat{\boldsymbol{\theta}}_t$ at time $t = 40$ are plotted for a typical simulation run. These plot illustrates the uncertainty associated to each parameter. The means of these Gaussian pdfs are close to the true parameters, in agreement with results seen in Fig. 4.5. In addition, the variances are small, hence all the probability distributions are tightly packed around the ground truth.

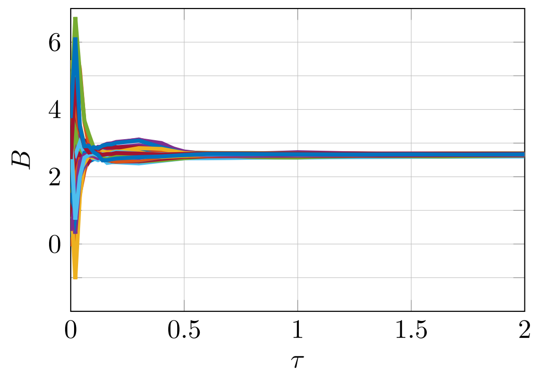
Fig. 4.8 displays the average performance of the UKF-EKF nested filter for different observation noise variances, σ^2_y. While all the previous experiments are done with $\sigma^2_y = 1$, in Fig. 4.8a we obtain similar results of NMSE$_{\boldsymbol{\theta}}$ for $\sigma^2_y = 2$ and slightly worse errors for $\sigma^2_y = 4$ and $\sigma^2_y = 10$. Although the errors increase for values of σ^2_y greater than one, the general performance of the algorithm is still accurate for larger values of the variance in the observation noise.

Finally, Fig. 4.9 illustrates the average performance of the proposed nested filter (UKF-EKF) in terms of NMSE$_{\boldsymbol{\theta}}$ and NMSE$_{\boldsymbol{x}}$ given different prior distributions of the form $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_\theta, \sigma^2_\theta \boldsymbol{I}_3)$. In Figs. 4.9a



(a) Estimates $\hat{\theta}_{1,t} = \hat{S}_t$.

(b) Estimates $\hat{\theta}_{2,t} = \hat{R}_t$.



(c) Estimates $\hat{\theta}_{3,t} = \hat{B}_t$.

Figure 4.5: Sequences of posterior-mean estimates, $\hat{\boldsymbol{\theta}}_t$, over time obtained from 50 independent simulation runs.

and 4.9b we depict the NMSE$_{\boldsymbol{\theta}}$ and NMSE$_{\boldsymbol{x}}$, respectively, as the prior variance σ_θ^2 is increased and the a priori mean $\boldsymbol{\mu}_\theta$ is drawn at random from a uniform distribution $\mathcal{U}(\boldsymbol{\theta}_\star - \boldsymbol{\epsilon}, \boldsymbol{\theta}_\star + \boldsymbol{\epsilon})$ for each independent simulation, where $\boldsymbol{\epsilon} = [3, 1, 0.5]^\top$. Changes in the variance of the prior distribution $p(\boldsymbol{\theta})$ slightly degrade the performance of the nested filter in terms of NMSE$_{\boldsymbol{\theta}}$ and NMSE$_{\boldsymbol{x}}$, obtaining comparable results for values of σ_θ^2 from 1 to 3. Similarly, in Figs. 4.9c and 4.9d the variance is fixed ($\sigma_\theta^2 = 1$) but the priori mean $\boldsymbol{\mu}_\theta$ is drawn from a uniform distribution of the form $\mathcal{U}(\boldsymbol{\theta}_\star - k\boldsymbol{\epsilon}, \boldsymbol{\theta}_\star + k\boldsymbol{\epsilon})$ with increasing values of k, while keeping $\boldsymbol{\epsilon} = [3, 1, 0.5]^\top$. The NMSEs obtained in this way increase approximately linearly with k, to yield an NMSE$_{\boldsymbol{\theta}}$ of almost one order of magnitude higher for $k = 3$ (but still below 10^{-2}).

4.4 Example: a stochastic volatility model with real data

In this section, we assess the performance of the proposed algorithm (UKF-EKF) for the task of estimating the parameters of a stochastic-volatility model using real-world time-series data (namely, euro-USD exchange rates between December 2014 and December 2016).

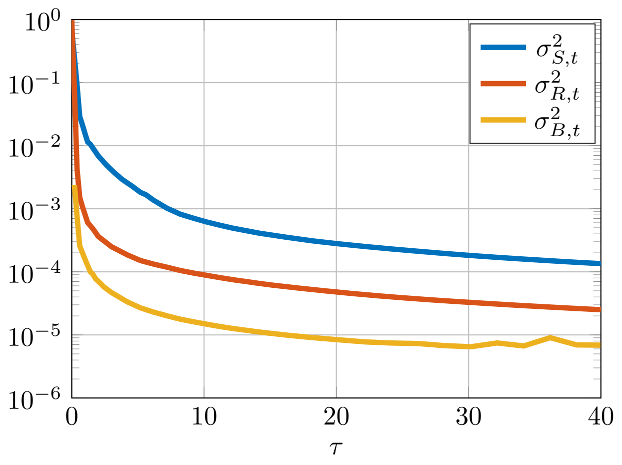


Figure 4.6: Values on the diagonal of the posterior covariance matrix, $\hat{\boldsymbol{C}}_t^{\boldsymbol{\theta}}$, over time. The plot is averaged over 50 independent simulation runs.

4.4.1 Stochastic Volatility model

We assume the stochastic volatility state-space model [54]

$$\begin{aligned} x_0 &= \mu + \sqrt{\frac{\sigma_v^2}{1-\phi^2}} v_0, & (4.45)\\ x_t &= \mu + \phi(x_{t-1} - \mu) + \sigma_v v_t, & (4.46)\\ y_t &= x_t + \sqrt{\omega} r_t, & (4.47)\end{aligned}$$

where $\boldsymbol{\theta} = [\mu, \sigma_v^2, \phi]^\top \in \mathbb{R} \times \mathbb{R}_+ \times [-1, 1]$ is the vector of static unknown parameters and the state x_t represents the log-volatility of the time-series of observations y_t, $t = 0, \dots, T$. The noise series v_t and r_t are i.i.d. Gaussian sequences and the variance of the observation noise $\omega = \frac{\pi^2}{2}$ is assumed known.

Given the historical EUR-USD exchange rate data sequence $s_0, \dots, s_T$ from 2014-12-31 to 2016-12-31 (obtained from www.quandl.com), we generate the time series of log-returns y_t following the same procedure as in [3, 26, 95]. To be specific, at time t we compute

$$\bar{y}_t = 100 \log\left(\frac{s_t}{s_{t-1}}\right), \tag{4.48}$$

for $1 \leq t \leq T$. Then, we further transform the log-returns into

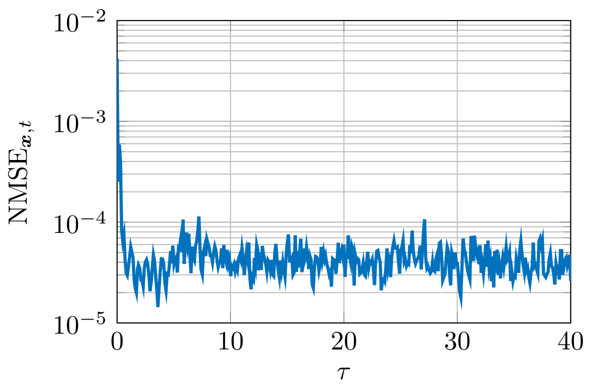
$$y_t = \log(\bar{y}_t^2) + 1.27, \tag{4.49}$$

in order to obtain the observations as described in Eq. (4.47) [54].

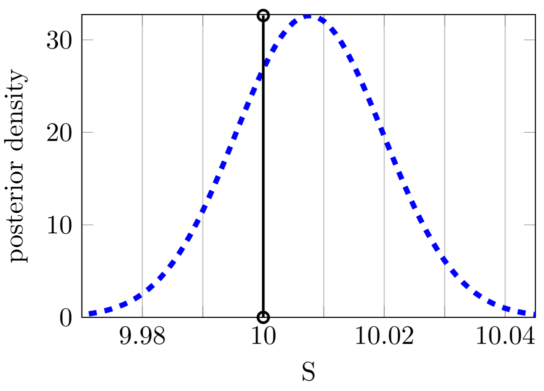
4.4.2 Simulation setup

Similar to Section 4.3, we have assessed several algorithms for the estimation of the parameters $\boldsymbol{\theta} = [\mu, \sigma_v^2, \phi]^\top$ and the state x_t of the stochastic volatility model. In particular, we have implemented the following methods:

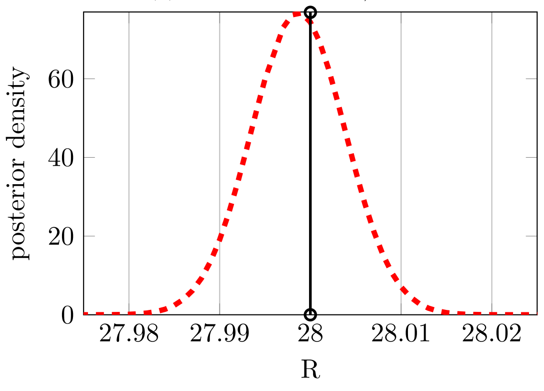
- The proposed Algorithm 10, using a UKF and a bank of EKFs in the first and second layers of the filter, respectively.



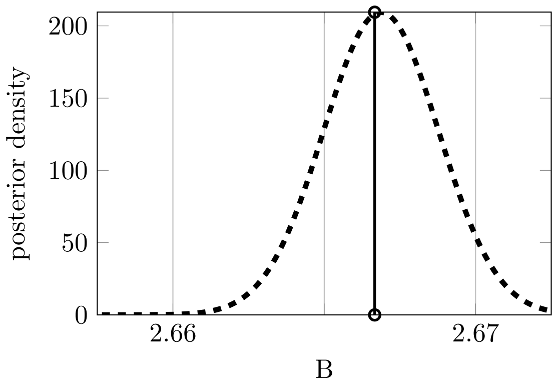
(a) Averaged $\mathrm{NMSE}_{\boldsymbol{x},t}$.



(b) Posterior density of $\hat{\boldsymbol{\theta}}_{1,T} = \hat{S}_T$.



(c) Posterior density of $\hat{\boldsymbol{\theta}}_{2,T} = \hat{R}_T$.



(d) Posterior density of $\hat{\boldsymbol{\theta}}_{3,T} = \hat{B}_T$.

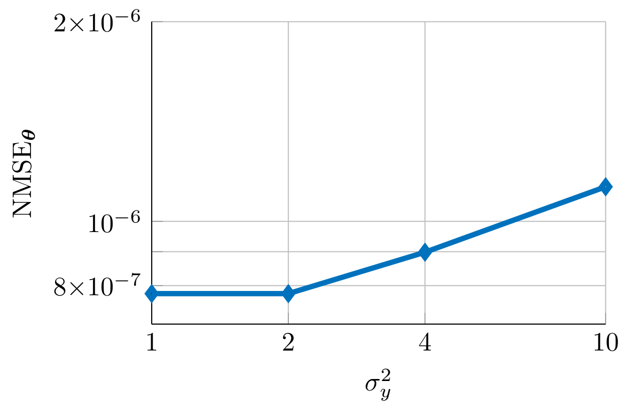
Figure 4.7: The mean $\mathrm{NMSE}_{\boldsymbol{x},t}$ of 50 simulation runs over time is plotted in Fig. 4.7a. Figs. 4.7b, 4.7c and 4.7d show the posterior density of parameters (dashed lines) at time $t = T$ and their true values (black vertical lines).

- An EnKF algorithm [36] with state augmentation (running $N = 100$ ensembles).
- An NHF [91] using a SMC scheme in the first layer (with $N = 100$ particles) and a bank of EKFs in the second layer.
- An NPF [25] with $N = 100$ particles in the SMC scheme of each layer (i.e., $10,000$ particles overall).

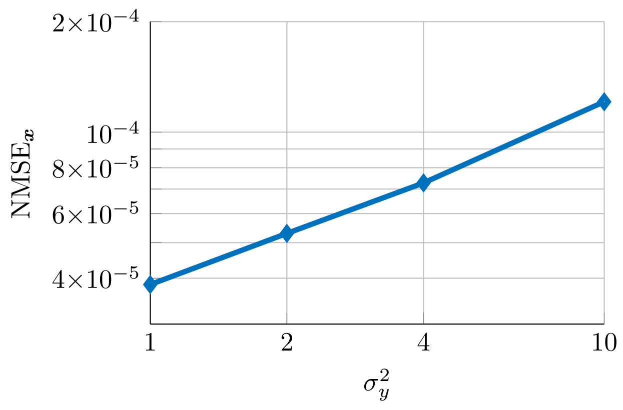
We recall that the computational complexity of these algorithms is outlined in Table 4.2.

For the UKF-EKF scheme we assume a Gaussian prior distribution for the unknown parameters, namely $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, 10^{-2}\boldsymbol{I}_3)$, where the a priori mean is drawn at random from a normal distribution $\boldsymbol{\mu}_{\boldsymbol{\theta}} \sim \mathcal{N}(\mu_{\boldsymbol{\theta}}|[0.85, 0.05, 0]^\top, 10^{-2}\boldsymbol{I}_3)$. For the rest of the methods, the prior distribution for the unknown parameters is $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|[0.9, 0.2, 0]^\top, 10^{-1}\boldsymbol{I}_3)$. The prior distribution for the state variable x_t is given conditional on the parameters, namely $p(x_0) = \mathcal{N}\left(x_0|\mu, \frac{\sigma_v^2}{1-\phi^2}\right)$ for all the different methods used in this section.

We have assessed the performance of the algorithms in terms of the marginal log-likelihood (MLL) estimated for the model, since there is no available ground truth for either the state or the parameters. We also illustrate the performance of the algorithms with the marginal pdfs of the three parameters at the final discrete time step T. For all methods, the marginal densities are averaged over 40 simulation runs.



(a) NMSE$_{\boldsymbol{\theta}}$ for different σ_y^2.



(b) NMSE$_{\boldsymbol{x}}$ for different σ_y^2.

Figure 4.8: NMSE$_{\boldsymbol{\theta}}$ (4.8a) and NMSE$_{\boldsymbol{x}}$ (4.8b) of UKF-EKF, averaged over 50 simulation runs, for different values of the noise variance σ_y^2.

4.4.3 Numerical results

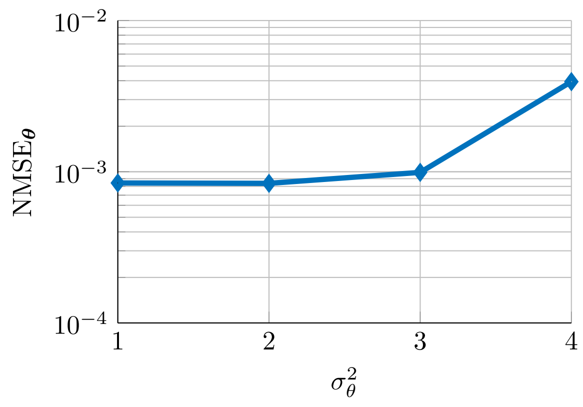
Fig. 4.10 shows the averaged marginal pdfs of the parameters at the end of the simulation runs (at discrete time $T = 513$) for the different algorithms compared here (NPF in blue, SMC-EKF in red, the proposed UKF-EKF in yellow and EnKF in violet). In general, all the algorithms obtain pdfs that overlap significantly, showing that the estimation task yields coherent similar results independently of the method applied. However, it is seen from Fig. 4.10c that the NPF and NHF methods yield pdfs which are non-Gaussian. Moreover, the estimates of the marginal pdfs output by the UKF-EKF method are narrower than the densities estimated by the other methods. This suggests that the UKF-EKF scheme may yield rather accurate point estimates for the parameters but possibly underestimate their variance.

Fig. 4.11 shows the marginal log-likelihood (MLL) as estimated using the same four methods. The smaller estimated variance for the parameters of the UKF-EKF algorithms reflects on a higher confidence on the model (i.e., a larger MLL). As for the dispersion on the MLL estimates, we should note that the UKF-EKF scheme is a deterministic method, hence the only source of variance is the randomness in the prior that we have introduced for the simulations. All other algorithms are stochastic and so display a larger variability of the estimates.

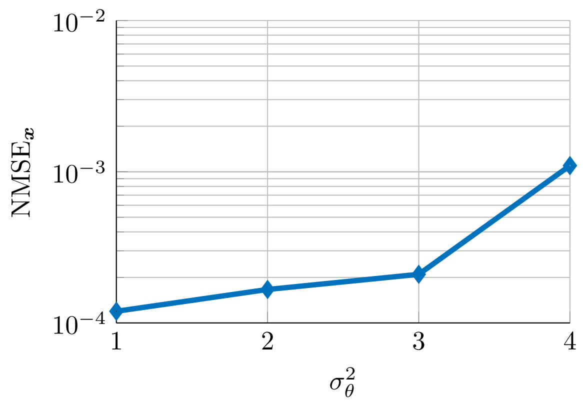
Finally, Fig. 4.12 illustrates the run-times averaged over 40 simulation runs. The use of Gaussian filters reduces considerably the computational cost, since the NPF is 2 orders of magnitude slower than any of the other algorithms. These run-times have been obtained running MATLAB R2018 on a MacBook Pro laptop computer with a 2,3 GHz Dual-Core Intel Core i5processor and 16 GB 2133 MHz LPDDR3 of RAM.

4.5 Conclusions

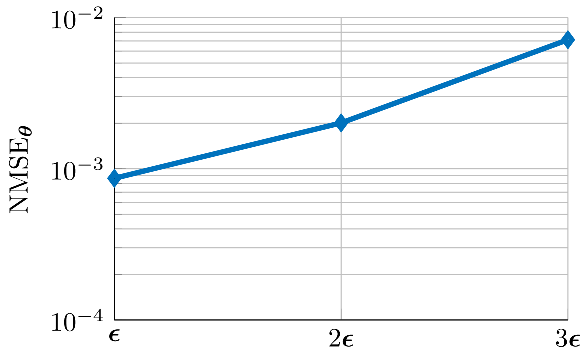
We have introduced a generalization of the NHF methodology of [91] that, using long sequences of observations collected over time, estimates the static parameters and tracks the stochastic dynamical variables of a state space model. This scheme combines two layers of filters, one inside the other, in order to compute the joint posterior probability distribution of the parameters and the states. In this generalization of the methodology, we introduce the use of deterministic sampling techniques in the first layer of the



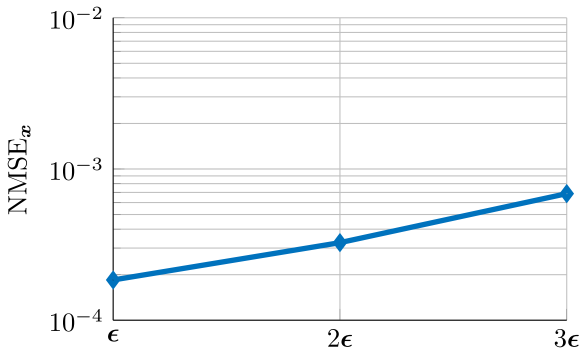
(a) $\text{NMSE}_{\boldsymbol{\theta}}$ for different σ_θ^2.



(b) $\text{NMSE}_{\boldsymbol{x}}$ for different σ_θ^2.



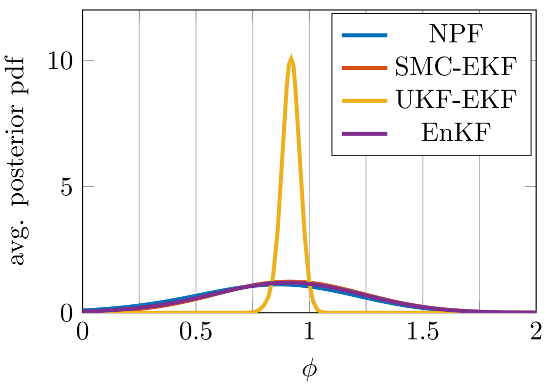
(c) $\text{NMSE}_{\boldsymbol{\theta}}$ for different $\boldsymbol{\epsilon}$.



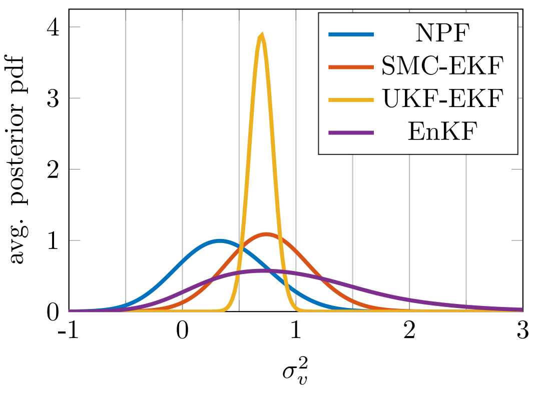
(d) $\text{NMSE}_{\boldsymbol{x}}$ for different $\boldsymbol{\epsilon}$.

Figure 4.9: $\text{NMSE}_{\boldsymbol{\theta}}$ (4.9a) and $\text{NMSE}_{\boldsymbol{x}}$ (4.9b) of UKF-EKF, averaged over 60 simulation runs, for different values of the σ_θ^2. Also, $\text{NMSE}_{\boldsymbol{\theta}}$ (4.9c) and $\text{NMSE}_{\boldsymbol{x}}$ (4.9d) for greater values of the $\boldsymbol{\epsilon}$ and fixed variance (σ_θ^2).

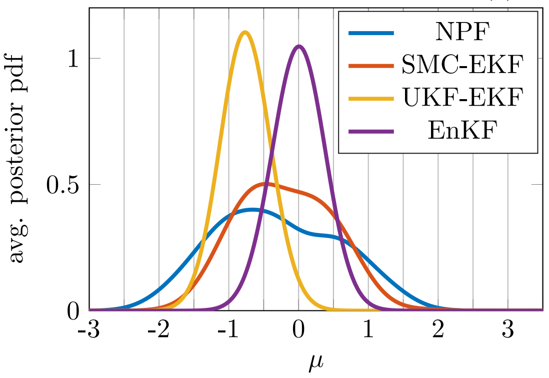
algorithm (the cubature Kalman filter (CKF) or the unscented Kalman filter (UKF)), instead of Monte Carlo methods, describing in detail how the algorithms can work sequentially and recursively. We have presented numerical results for a stochastic Lorenz 63 model with synthetic data and for a stochastic volatility model with real-world data, using a scheme with a UKF for the parameters in the first layer, and EKFs for the time-varying state variables in the second layer. We have introduced and assessed the values of a relative threshold that enables the algorithm to work recursively, and we have evaluated the performance of the algorithm in terms of the normalized mean square errors for the parameters and the dynamic state variables. We have also compared these results with other algorithms, such as the EnKF or the UKF, that implement state augmentation (i.e., an extended state that includes both parameters and state), and also with an NHF (with a SMC in the first layer and EKFs in the second layer) and an NPF. The use of Gaussian filters in the two layers of the algorithm not only leads to a significant reduction in computational complexity compared to Monte Carlo-based implementations but also increases the accuracy compared to state-augmented Gaussian filters. Additional research is still needed for an analytical characterization of the performance, possibly under suitable regularity conditions that enable a theoretical study.



(a) Averaged pdfs of ϕ.



(b) Averaged pdfs of σ_v^2.



(c) Averaged pdfs of μ.

Figure 4.10: Pdfs of the unknown parameters $\boldsymbol{\theta} = (\phi, \sigma_v^2, \mu)^\top$ at time $T = 513$, averaged over 40 simulation runs. We have assessed four different algorithms: the NPF (blue), the SMC-EKF (red), the EnKF (violet) and the UKF-EKF (yellow).

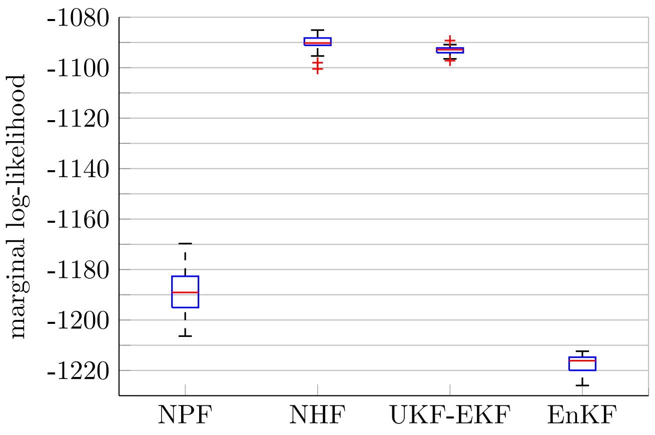


Figure 4.11: Marginal log-likelihood at time T, obtained from 40 simulation runs, for the NPF, the SMC-EKF, the UKF-EKF and the EnKF with state augmentation.

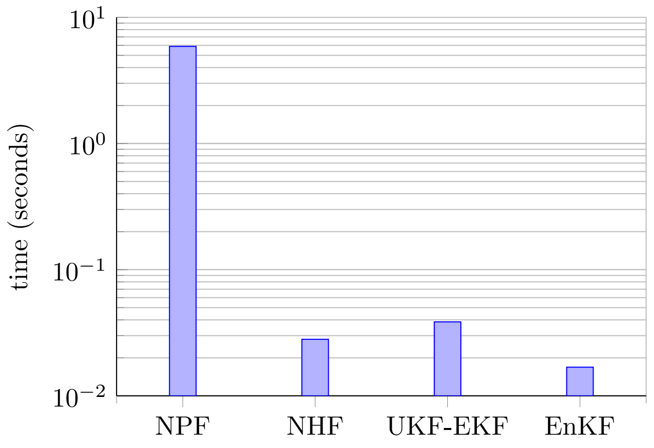


Figure 4.12: Running time in seconds, averaged over 40 simulation runs, for the NPF, the SMC-EKF, the UKF-EKF and the EnKF with state augmentation.

5 Multi-Scale Nested Filters

This chapter is dedicated to a further generalization of the nested hybrid filters (NHFs); we aim at performing recursive Bayesian inference for a class of heterogeneous multi-scale state-space models [1]. The new scheme can be described as a three-layer nested smoother that approximates, in a recursive manner, the posterior probability distributions of the parameters and two sets of state variables given the sequence of available observations. To be specific, in a first layer of computation we approximate the posterior probability distribution of the parameters, in a second layer we approximate the posterior probability distribution of the slow state variables, and the posterior probability distribution of the fast state variables is approximated in a third layer. The inference techniques used in each layer can vary, leading to different computational costs and degrees of accuracy. In Section 5.1 we state the problem to be addressed, introducing heterogeneous multi-scale state-space models. In Section 5.2, we describe the optimal smoother for multi-scale systems with static parameters and two sets of dynamic state variables. Two specific methods derived from the general methodology are introduced in Section 5.3. Finally, numerical results for the stochastic two-scale Lorenz 96 model are shown in Section 5.4. The notation for this chapter is summarized in Table 5.1.

5.1 Problem Statement

5.1.1 State space models

In this chapter we place our attention on state space models that result from the analysis of physical systems that display (intertwined) dynamical features at different time scales. To be specific, let us consider the class of multidimensional stochastic differential equations (SDEs) that can be written as

$$d\boldsymbol{x} = f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{\theta})d\tau + g_{\boldsymbol{x}}(\boldsymbol{z},\boldsymbol{\theta})d\tau + \boldsymbol{Q}_x d\boldsymbol{v}, \tag{5.1}$$

$$d\boldsymbol{z} = f_{\boldsymbol{z}}(\boldsymbol{x},\boldsymbol{\theta})d\tau + g_{\boldsymbol{z}}(\boldsymbol{z},\boldsymbol{\theta})d\tau + \boldsymbol{Q}_z d\boldsymbol{w}, \tag{5.2}$$

where

- τ denotes continuous time,
- $\boldsymbol{x}(\tau) \in \mathbb{R}^{d_x}$ and $\boldsymbol{z}(\tau) \in \mathbb{R}^{d_z}$ are the slow and fast states of the system, respectively,
- $f_{\boldsymbol{x}}\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_x}$, $g_{\boldsymbol{x}}\colon \mathbb{R}^{d_z} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_x}$, $f_{\boldsymbol{z}}\colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_z}$ and $g_{\boldsymbol{z}}\colon \mathbb{R}^{d_z} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_z}$ are (possibly nonlinear) transition functions parameterized by a fixed vector of unknown parameters, $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$,
- $\boldsymbol{Q}_x$ and $\boldsymbol{Q}_z$ are known scaling matrices that control the intensity and covariance of the stochastic perturbations,
- and $\boldsymbol{v}(\tau)$ and $\boldsymbol{w}(\tau)$ are vectors of independent standard Wiener processes with dimension d_x and d_z, respectively.

Equations (5.1)–(5.2) do not have closed form solutions for general nonlinear functions $f_{\boldsymbol{x}}$, $f_{\boldsymbol{z}}$, $g_{\boldsymbol{x}}$ and $g_{\boldsymbol{z}}$ and they have to be discretized for their numerical integration. In order to handle the slow and fast time scales, we apply a macro-micro solver [96, 102] that runs an Euler-Maruyama scheme for each set of state variables, albeit with different integration steps. To be specific, we use Δ_z as the integration step of $\boldsymbol{z}$ while $\Delta_x \gg \Delta_z$ is the integration step of $\boldsymbol{x}$. Then, we can simulate $\boldsymbol{x}$ and $\boldsymbol{z}$ using the pair of difference equations

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \Delta_x(f_{\boldsymbol{x}}(\boldsymbol{x}_{t-1},\boldsymbol{\theta}) + g_{\boldsymbol{x}}(\bar{\boldsymbol{z}}_t,\boldsymbol{\theta})) + \sqrt{\Delta_x}\boldsymbol{Q}_x\boldsymbol{v}_t, \tag{5.3}$$

$$\boldsymbol{z}_n = \boldsymbol{z}_{n-1} + \Delta_z(f_{\boldsymbol{z}}(\boldsymbol{x}_{\lfloor\frac{n-1}{h}\rfloor},\boldsymbol{\theta}) + g_{\boldsymbol{z}}(\boldsymbol{z}_{n-1},\boldsymbol{\theta})) + \sqrt{\Delta_z}\boldsymbol{Q}_z\boldsymbol{w}_n, \tag{5.4}$$

where $\boldsymbol{x}_t \approx \boldsymbol{x}(t\Delta_x)$ and $\boldsymbol{z}_n \approx \boldsymbol{z}(n\Delta_z)$ are the state signals, $t \in \mathbb{N}$ denotes discrete time in the time scale of the slow variables, $n \in \mathbb{N}$ denotes discrete time in the fast time scale, $h = \frac{\Delta_x}{\Delta_z} \in \mathbb{Z}^+$ is the number of fast steps (in the scale of $\boldsymbol{z}$) per slow step (in the scale of $\boldsymbol{x}$), $\boldsymbol{v}_t$ and $\boldsymbol{w}_n$ are Gaussian r.v.s of zero mean and covariance matrices $\boldsymbol{I}_{d_x}$ and $\boldsymbol{I}_{d_z}$ respectively, and $\bar{\boldsymbol{z}}_t$ is an average of the fast signal computed as

$$\bar{\boldsymbol{z}}_t = \frac{1}{h}\sum_{i=h(t-1)+1}^{ht} \boldsymbol{z}_i. \tag{5.5}$$

We assume that the available observations may be directly related to both sets of state variables $\boldsymbol{x}_t$ and $\boldsymbol{z}_n$, but only in the (slow) time scale of $\boldsymbol{x}$. To be specific, the t-th observation is a d_y-dimensional r.v., $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$, which we model as

$$\boldsymbol{y}_t = l(\boldsymbol{z}_{ht},\boldsymbol{x}_t,\boldsymbol{\theta}) + \boldsymbol{r}_t, \tag{5.6}$$

where $l\colon \mathbb{R}^{d_z} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_y}$ is a transformation that maps the states into the observation space, and $\boldsymbol{r}_t$ is a zero-mean observational-noise vector with covariance matrix $\boldsymbol{R}$.

Table 5.1: Notation of Chapter 5.

$\boldsymbol{C}_t^{i,j}(\boldsymbol{y})$ and $\check{\boldsymbol{C}}_t^{i,j}(\boldsymbol{x})$	Predictive covariance matrices of the observation and the slow state, respectively
$\boldsymbol{C}_t^{i,j}(\boldsymbol{x},\boldsymbol{y})$	Cross-covariance matrix of the slow state and the observation
$\boldsymbol{C}_t^{i,j}(\boldsymbol{z},\boldsymbol{y})$	Cross-covariance matrix of the fast state and the observation
$\check{\boldsymbol{C}}_n^{i,j}(\boldsymbol{z})$ and $\hat{\boldsymbol{C}}_n^{i,j}(\boldsymbol{z})$	Predictive and posterior covariance matrices of the fast state, respectively
d_x, d_y, d_z, d_θ	Slow state, observation, fast state and parameter dimension, respectively
$f_{\boldsymbol{x}}(\cdot)$ and $g_{\boldsymbol{x}}(\cdot)$	Slow state transition functions
$f_{\boldsymbol{z}}(\cdot)$ and $g_{\boldsymbol{z}}(\cdot)$	Fast state transition function
h	Number of fast state steps per slow state steps
$\boldsymbol{H}_t^{i,j}(\boldsymbol{z})$ and $\boldsymbol{J}_n^{i,j}(\boldsymbol{z})$	Jacobian matrices of the observation function and the fast state transition function, respectively
J	Number of particles or ensemble members in the second layer
$\boldsymbol{K}_t(\boldsymbol{x})$ and $\boldsymbol{K}_t(\boldsymbol{z})$	Kalman gain for the slow and fast state respectively
$l(\cdot)$	Observation function
L	Number of sigma-points in the third layer of the algorithm
n	Discrete-time steps in the time scale of $\boldsymbol{z}$
N	Number of particles in the first layer of the algorithm
$\boldsymbol{r}_t$	Observation noise vector at time step t
$\boldsymbol{R}$	Covariance matrix of the observation noise
t	Discrete-time steps in the time scale of $\boldsymbol{x}$
$\tilde{u}_t^{i,j}$ and $\tilde{v}_t^i$	Non-normalized weights of the second and first layers, respectively
$\boldsymbol{v}_t$ and $\boldsymbol{w}_n$	Slow and fast state noise vectors at time steps t and n, respectively
$\tilde{w}_t^{i,j,l}$	Approximate likelihood computed in the third layer at time step t
$\boldsymbol{x}_t$	Slow state vector at time step t
$\boldsymbol{x}_t^{i,j}$	Posterior samples of the slow state at time step t
$\check{\boldsymbol{x}}_t^{i,j}$ and $\hat{\boldsymbol{x}}_t^{i,j}$	Predictive and posterior mean of the slow state at time step t
$\bar{\boldsymbol{x}}_t^{i,j}$ and $\tilde{\boldsymbol{x}}_t^{i,j,l}$	Predictive samples of the slow state in the second and third layers
$\bar{\boldsymbol{X}}_t^i$ and $\hat{\boldsymbol{X}}_t^i$	Predictive and posterior ensemble with J samples of the slow state
$\boldsymbol{y}_t$	Observation vector
$\hat{\boldsymbol{y}}_t$	Predictive mean of the observation vector
$\tilde{\boldsymbol{y}}_t^{i,j}$ and $\tilde{\boldsymbol{y}}_t^{i,j,l}$	State samples or sigma-points projected into the observation space
$\boldsymbol{Y}_t^i$	Observation ensemble
$\boldsymbol{z}_n$	Fast state vector at time step n
$\bar{\boldsymbol{z}}_t$	Average of the fast state variables from $n = h(t-1)+1$ to $n = ht$
$\check{\boldsymbol{z}}_n^{i,j}$ and $\hat{\boldsymbol{z}}_n^{i,j}$	Predictive and posterior mean of the fast state
$\breve{\boldsymbol{z}}_n^{i,j,l}$ and $\boldsymbol{z}_n^{i,j,l}$	Predictive and posterior fast state sigma-points at time step n
Δ_x and Δ_z	Integration steps of $\boldsymbol{x}$ and $\boldsymbol{z}$ respectively
$\boldsymbol{\theta}_t^i$	Posterior i-th particle or parameter sample at time step t
$\kappa_N^{\boldsymbol{\theta}'}$	Markov kernel of the jittering step
$\lambda_n^{i,j,l}$	Weights of the sigma-points in the UKF of the third layer
τ	Continuous time

5.1.2 Model inference

Similar to Chapter 4, we aim at performing *joint* Bayesian estimation of the parameters, $\boldsymbol{\theta}$, and all states, $\boldsymbol{x}$ and $\boldsymbol{z}$, for the state-space model described by Eqs. (5.3)-(5.4) and (5.6). Typically, the three vectors of unknowns are tightly coupled. The estimation of the fixed parameters is necessary to track both sets of state variables and, at the same time, tracking the slow state variables is needed for predicting the time evolution of the fast states and vice versa.

While in many practical applications one is typically interested in filtering, i.e., the computation of the posterior pdf of $\boldsymbol{\theta}$, $\boldsymbol{x}_t$ and $\boldsymbol{z}_n$ (for $n = ht$) given the data sequence $\boldsymbol{y}_{1:t} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_t\}$, we find more convenient to tackle the smoothing pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{0:t}, \boldsymbol{\theta}|\boldsymbol{y}_{1:t})$. Using the chain rule, we can factorize the latter density as

$$p(\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{0:t}, \boldsymbol{\theta}|\boldsymbol{y}_{1:t}) = p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}), \tag{5.7}$$

where we identify the three key conditional distributions that we seek to compute (or approximate). Each one of these pdfs can be handled in a different *layer* of computation. Hence, we aim at designing a nested inference algorithm (in the vein of [91]) with three layers. In the first layer we compute $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$, in the second one we obtain $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$, and in the third layer we tackle $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$.

Hereafter we describe the methodology for the optimal (yet impractical) calculation of the posterior pdf in Eq. (5.7) as well as two approximate numerical solutions that admit feasible computational implementations.

5.2 Optimal nested smoother

We introduce the optimal nested smoothing algorithm, consisting of three layers, that computes each of the pdfs in Eq. (5.7). The scheme is summarized in Fig. 5.1. As a result, we obtain the posterior smoothing density $p(\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{0:t}, \boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ which, in turn, can be used to compute the optimal filtering pdf, $p(\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta}|\boldsymbol{y}_{1:t})$, by marginalization if necessary. When the exact computations demanded by this algorithm are not feasible (for general nonlinear and/or non-Gaussian dynamical systems) it serves as a template for approximate numerical schemes, as shown in Section 5.3.

5.2.1 First layer: static parameters

The aim of this layer is to compute the posterior pdf of the parameters, $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$, recursively. We assume that the *a priori* density $p(\boldsymbol{\theta})$ is known.

At time t, assume that $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})$ has been calculated. When a new observation, $\boldsymbol{y}_t$, is obtained, we need to compute the likelihood $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ in order to obtain the posterior pdf of $\boldsymbol{\theta}$ at time t as

$$p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) \propto p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1}). \tag{5.8}$$

However, the parameter likelihood $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ cannot be computed directly. Instead, we decompose it as

$$p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) = \int p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})d\boldsymbol{x}_{0:t}, \tag{5.9}$$

where $p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ and $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ are the likelihood and the predictive pdf of the state sequence $\boldsymbol{x}_{0:t}$, respectively, conditional on the previous observations and the parameters. These pdfs are computed in the second layer of the algorithm.

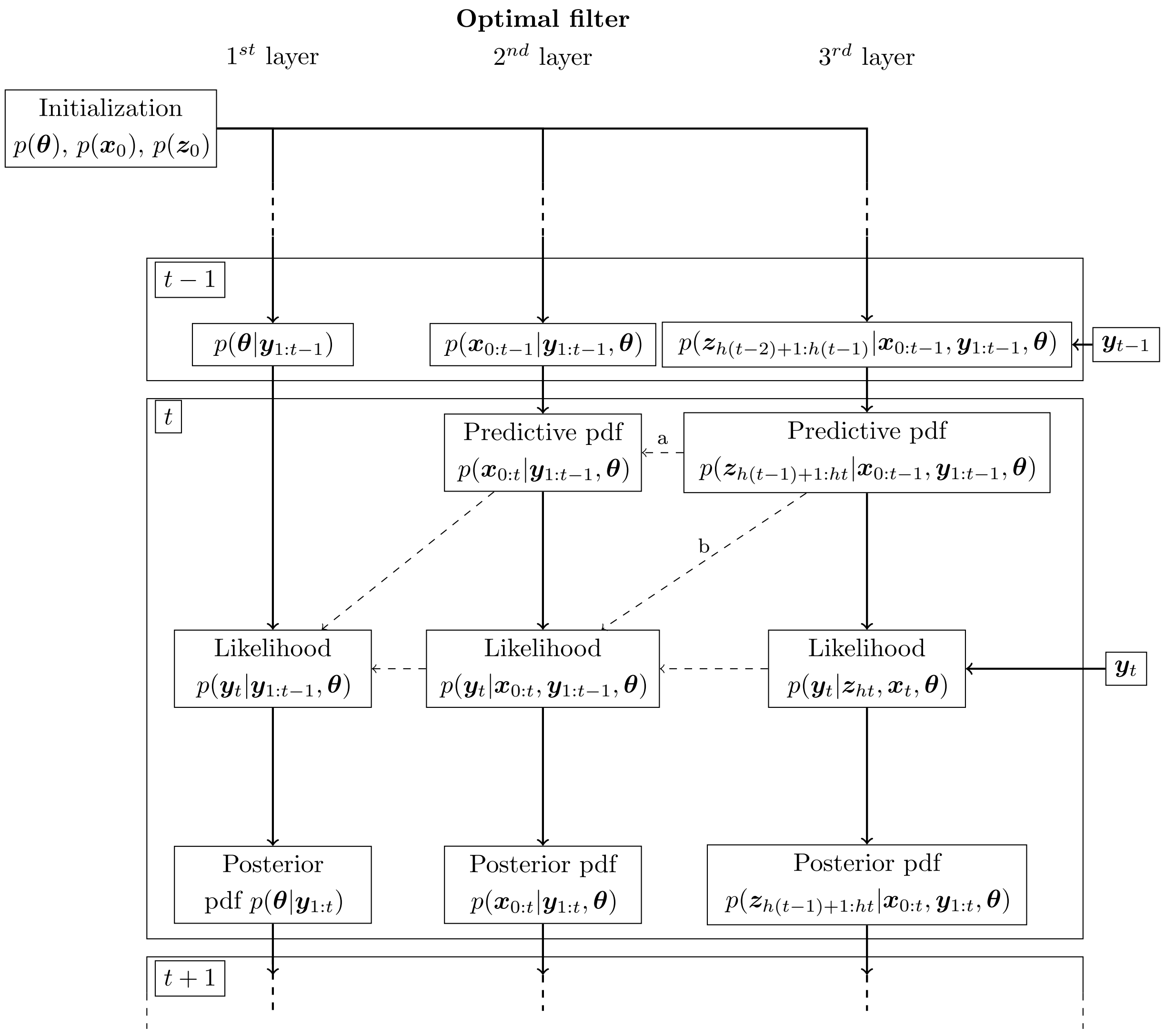

Optimal filter
1^{st} layer
2^{nd} layer
3^{rd} layer
Initialization $p(\boldsymbol{\theta})$, $p(\boldsymbol{x}_0)$, $p(\boldsymbol{z}_0)$
$t-1$
$p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t-1})$
$p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$
$p(\boldsymbol{z}_{h(t-2)+1:h(t-1)}|\boldsymbol{x}_{0:t-1},\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$
$\boldsymbol{y}_{t-1}$
t
Predictive pdf $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$
a
Predictive pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1},\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$
b
Likelihood $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$
Likelihood $p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t},\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$
Likelihood $p(\boldsymbol{y}_t|\boldsymbol{z}_{ht},\boldsymbol{x}_t,\boldsymbol{\theta})$
$\boldsymbol{y}_t$
Posterior pdf $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$
Posterior pdf $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t},\boldsymbol{\theta})$
Posterior pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t},\boldsymbol{y}_{1:t},\boldsymbol{\theta})$
$t+1$

Figure 5.1: Schematic depiction of the optimal smoother. Each column represents a layer of computation and the dependencies among pdfs are indicated by arrows. The dashed arrows are used to show relations among different layers while the solid arrows represent dependencies in the same layer. Arrows a and b indicate that some intermediate computations are needed to relate both pdfs.

5.2.2 Second layer: slow states

Computations in this layer are conditional on the parameter vector $\boldsymbol{\theta}$. We seek to compute the smoothing posterior $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t},\boldsymbol{\theta})$ as well as the predictive density $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$ and the likelihood $p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t},\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$, which are needed in the first layer –see Eq. (5.9). We assume that the prior density $p(\boldsymbol{x}_0)$ is known and the posterior pdf of the slow states at time $t-1$ (conditional on $\boldsymbol{\theta}$), $p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$, is available at time t.

We first seek the predictive density of $\boldsymbol{x}_{0:t}$, namely,

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta}) = p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1},\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta}), \quad (5.10)$$

which is obtained recursively from the posterior at time $t-1$, $p(\boldsymbol{x}_{0:t-1}|\boldsymbol{y}_{1:t-1},\boldsymbol{\theta})$, but requires the evaluation of the marginal density $p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$. The latter is not directly available. It has to be computed as an integral w.r.t. the fast state variables, in particular

$$\begin{aligned} p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) &= \int p(\boldsymbol{x}_t|\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \times \\ &\qquad p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) d\boldsymbol{z}_{h(t-1)+1:ht}. \end{aligned} \tag{5.11}$$

The two densities in the integrand of Eq. (5.11), which involve the fast state variables $\boldsymbol{z}_{h(t-1)}, \dots, \boldsymbol{z}_{ht}$, are calculated in the third layer. Recall that h is the number of discrete-time steps of the fast states per each single time step of the slow variables (i.e., the $\boldsymbol{z}_n$'s are h time faster than the $\boldsymbol{x}_t$'s).

As for the likelihood, when $\boldsymbol{y}_t$ becomes available we update the posterior density of $\boldsymbol{x}_{0:t}$ (conditional on $\boldsymbol{\theta}$) as

$$p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}) \propto p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}). \tag{5.12}$$

In the equation above, the likelihood $p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ can be computed as an integral w.r.t. the fast state variables, specifically,

$$\begin{aligned} p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) &= \int p(\boldsymbol{y}_t|\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) d\boldsymbol{z}_{h(t-1)+1:ht} \\ &= \int p(\boldsymbol{y}_t|\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta}) p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) d\boldsymbol{z}_{h(t-1)+1:ht}. \end{aligned} \tag{5.13}$$

The likelihood function $p(\boldsymbol{y}_t|\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta})$ can be obtained directly from the state-space model described by Eqs. (5.3)-(5.6), while the conditional pdf of the subsequence $\boldsymbol{z}_{h(t-1)+1:ht}$ can be further decomposed as

$$p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{x}_t|\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta})}{p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})} \times p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}). \tag{5.14}$$

Both the likelihood $p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ of Eq. (5.13) and the predictive density $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ of Eq. (5.14) are explicitly computed in the third layer.

5.2.3 Third layer: fast states

Computations on this layer are conditional on the parameter vector $\boldsymbol{\theta}$ and the sequence of slow states $\boldsymbol{x}_{0:t}$. In particular, we seek to compute the conditional posterior pdfs of $\boldsymbol{z}_{h(t-1)+1:ht}$, including the predictive densities,

$$p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \quad \text{and} \quad p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}),$$

as well as the updated density $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$. We also evaluate the plain likelihood function $p(\boldsymbol{y}_t|\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta})$. We assume that the prior pdf of the fast states, $p(\boldsymbol{z})$, is known and the posterior up to time $t-1$, $p(\boldsymbol{z}_{h(t-2)+1:h(t-1)}|\boldsymbol{x}_{t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$, is available to enable recursive computations.

The first predictive pdf is computed recursively from the posterior up to time $t-1$ as the integral

$$\begin{aligned} p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) &= \int p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{z}_{h(t-2)+1:h(t-1)}, \boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \times \\ &\qquad \times p(\boldsymbol{z}_{h(t-2)+1:h(t-1)}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) d\boldsymbol{z}_{h(t-2)+1:h(t-1)} \\ &= \int p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{z}_{h(t-1)}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta}) \times \\ &\qquad \times p(\boldsymbol{z}_{h(t-2)+1:h(t-1)}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) d\boldsymbol{z}_{h(t-2)+1:h(t-1)}, \end{aligned} \tag{5.15}$$

where the transition pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{z}_{h(t-1)}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ is obtained immediately by iterating Eq. (5.3) in the state-space model h times and $p(\boldsymbol{z}_{h(t-2)+1:h(t-1)}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})d\boldsymbol{z}_{h(t-2)+1:h(t-1)}$ is the posterior pdf of the fast states in the previous time step. Besides, the second predictive density, $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$, is obtained by substituting the first predictive pdf of Eq. (5.15) into Eq. (5.14)[1].

Finally, when the observation $\boldsymbol{y}_t$ becomes available, we compute the plain likelihood $p(\boldsymbol{y}_t|\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta})$ (from Eq. (5.6) in the state-space model) and then update the conditional posterior pdf of the fast state variables, namely,

$$p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{y}_t|\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta})p(\boldsymbol{x}_t|\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta})}{p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})} \times p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$$

or, simply,

$$p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta}) \propto p(\boldsymbol{y}_t|\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta})p(\boldsymbol{x}_t|\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta})p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) \tag{5.16}$$

if we skip the normalization constant that is typically not needed explicitly for numerical implementations.

5.2.4 Outline of the optimal nested smoother

The optimal nested smoother uses each layer of computation to track a subset of r.v.s that evolve over their own time scale, by computing the corresponding predictive and updated pdfs (when observations are collected), as well as the necessary likelihoods. To be specific:

- The third layer tracks the fast state variables, $\boldsymbol{z}_n$, and computes the predictive pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ of Eq. (5.15) and the likelihood $p(\boldsymbol{y}_t|\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta})$. They are used to track the conditional posterior density $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ of Eq. (5.16).

- The second layer takes the pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ and the likelihood $p(\boldsymbol{y}_t|\boldsymbol{z}_{ht}, \boldsymbol{x}_t, \boldsymbol{\theta})$ in order to compute the predictive pdf $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ in Eq. (5.10) and the likelihood $p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ in Eq. (5.13). These are used to track the posterior pdf of the slow state, $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$, of Eq. (5.12).

- The first layer takes the pdfs $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ and $p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ to track the posterior pdf of the parameters, $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$, of Eq. (5.8).

Finally, the three conditional posterior pdfs are needed to compute the joint smoothing density $p(\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{0:t}, \boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ in Eq. (5.7).

Figure 5.1 is a schematic representation of the optimal smoother, which displays each layer in a different column. Most of the pdfs that need to be computed are included in this scheme, showing the dependencies among them with arrows. These relations are direct except for the arrows labeled a and b, which require one or more intermediate computations. In the case of arrow a, the predictive pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ is used to compute $p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ in Eq. (5.11), that is necessary to calculate the predictive density $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ of the second layer in Eq. (5.10). As for the arrow labeled b, the predictive density $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ is used to compute the pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ in Eq. (5.14), that is used, in turn, to obtain the likelihood $p(\boldsymbol{y}_t|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ in Eq. (5.13).

[1]Note that, in Eq. (5.14), the density $p(\boldsymbol{x}_t|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$ is the normalization constant for the conditional pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}, \boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$, while $p(\boldsymbol{x}_t|\boldsymbol{z}_{h(t-1)+1:ht}, \boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ results from the iteration of Eq. (5.3).

5.3 Approximate smoothing algorithms

The optimal algorithm described in Section 5.2 cannot be implemented exactly for most practical models. Instead, one needs to devise suitable approximations that can be implemented numerically in an efficient way. One possible approach is a full-blown sequential Monte Carlo (SMC) implementation that extends the nested particle filter of [25]. However, such a scheme with three layers of computation results in a prohibitive computational cost. Instead, we introduce herein two different algorithms that combine SMC and Gaussian approximations at the different layers. The resulting algorithms can be implemented numerically in a more efficient manner and are suitable for parallelization, which leads to very fast runtimes.

The first method involves using SMC schemes both at the first and second layer, together with a bank of unscented Kalman filters (UKFs) [52, 71] to approximate (as Gaussians) the conditional densities to be computed at the third layer. This implementation has great potential for parallelization, but it is computationally costly nevertheless. Hence, we also introduce a second, less demanding scheme that utilizes the same SMC scheme at the first layer but employs ensemble Kalman filters (EnKFs) [36] at the second layer and simple extended Kalman filters (EKFs) [5] to approximate the densities needed in the second and third layer, respectively. A numerical study of performance is carried out in Section 5.4 for both implementations.

5.3.1 First scheme

We introduce a numerical approximate smoother where the probability measures $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})d\boldsymbol{\theta}$ and $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t},\boldsymbol{\theta})d\boldsymbol{x}_{0:t}$ are approximated using SMC while we replace the conditional smoothing pdf $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t},\boldsymbol{y}_{1:t},\boldsymbol{\theta})$ by a sequence of Gaussian approximations of the densities $p(\boldsymbol{z}_n|\boldsymbol{x}_{0:t},\boldsymbol{y}_{1:t},\boldsymbol{\theta})$, for $n = h(t-1)+1,\dots,ht$, computed using a bank of UKFs.

First layer Algorithm 11 describes the first layer of the nested smoother, which aims at the approximation of the posterior distribution of the parameters. It receives as inputs the prior pdfs of the parameters, $p(\boldsymbol{\theta})$, and the two subsets of state variables, $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{z}_0)$. In the initialization step, they are used to generate starting Monte Carlo particles (for the SMC schemes) and sigma-points (for the UKFs) needed at each layer. Specifically, we generate N parameter samples $\{\boldsymbol{\theta}_0^i\}_{1\le i\le N}$, J slow state particles per each parameter sample, $\{\boldsymbol{x}_0^{i,j}\}_{1\le j\le J}$, and L sigma-points of the fast state per each slow state sample, $\{\boldsymbol{z}_0^{i,j,l}\}_{0\le l\le L-1}$, to obtain a set of the form $\{\boldsymbol{\theta}_0^i,\{\boldsymbol{x}_0^{i,j},\{\boldsymbol{z}_0^{i,j,l}\}_{0\le l\le L-1}\}_{1\le j\le J}\}_{1\le i\le N}$. All particles are independent at time $t = n = 0$, provided the priors are independent.

Additionally, a Markov kernel $\kappa_N^{\boldsymbol{\theta}'}(d\boldsymbol{\theta})$ is needed for the jittering of parameter samples [25], i.e., to draw a new set of particles, $\{\bar{\boldsymbol{\theta}}_t^i\}_{1\le i\le N}$, at each discrete-time step. This is needed to preserve the diversity of the particles, otherwise after a few resampling steps the parameter particles would be reduced to just a few distinct values and the filter would collapse. The choice of this kernel has been discussed in Chapter 3.

At every time step t (in the slow time scale), we compute the approximate likelihood for each particle $\bar{\boldsymbol{\theta}}_t^i$, namely

$$\hat{p}^{J,L}(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1},\bar{\boldsymbol{\theta}}_t^i) \approx p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1},\bar{\boldsymbol{\theta}}_t^i),$$

in order to obtain the non-normalized weights $\{\tilde{v}_t^i\}_{1\le i\le N}$. The superscripts J and L indicate the dependence of the approximation on the number of particles generated for the second layer (J) and the number of sigma-points employed by the UKFs in the third layer (L). The states $\{\boldsymbol{x}_{t-1}^{i,j},\{\boldsymbol{z}_{h(t-1)}^{i,j,l}\}_{0\le l\le L-1}\}_{1\le j\le J}$ are propagated to time t in the nested layers of filtering in step 2b. Finally, we normalize the weights in order to resample not only the parameter particles $\bar{\boldsymbol{\theta}}_t^i$, but also their associated sets of state variables.

Algorithm 11 *SMC approximation of $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ in the first method*

```
***Inputs***

- Prior distributions $p(\boldsymbol{\theta})$, $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{z}_0)$.
- A Markov kernel $\kappa_N^{\boldsymbol{\theta}'}(d\boldsymbol{\theta})$ which, given $\boldsymbol{\theta}'$, generates jittered parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$.

***Initialization:*** this is a joint initialization for all three layers.

- Draw $N$ i.i.d. sample $\boldsymbol{\theta}_0^i$, $i = 1, \ldots, N$ from the prior distribution $p(\boldsymbol{\theta})$.
- Draw $J$ i.i.d. samples $\boldsymbol{x}_0^{i,j}$, $i = 1, \ldots, N$, $j = 1, \ldots, J$, from the prior distribution $p(\boldsymbol{x}_0)$.
- Compute $L = 2d_z+1$ sigma-points, $\boldsymbol{z}_0^{i,j,l}$, with their respective weights, $\lambda_0^{i,j,l}$, $i = 1, \ldots, N$, $j = 1, \ldots, J$,
  $l = 0, \ldots, L-1$, from the prior distribution $p(\boldsymbol{z}_0|\hat{\boldsymbol{z}}_0, \boldsymbol{C}_0(\boldsymbol{z}))$ as
```

$$
\begin{aligned}
\boldsymbol{z}_0^{i,j,0} &= \hat{\boldsymbol{z}}_0, & \lambda_0^{i,j,0} &= \frac{1}{1+d_z}, \\
\boldsymbol{z}_0^{i,j,l} &= \hat{\boldsymbol{z}}_0 + \boldsymbol{S}_l, & \lambda_0^{i,j,l} &= \frac{1-\lambda_0^{i,j,0}}{2d_z}, \\
\boldsymbol{z}_0^{i,j,l+d_z} &= \hat{\boldsymbol{z}}_0 - \boldsymbol{S}_l, & \lambda_0^{i,j,l+d_z} &= \frac{1-\lambda_0^{i,j,0}}{2d_z},
\end{aligned}
$$

```
  for $l = 1, \ldots, d_z$, where $\boldsymbol{S}_l$ is the $l$-th row or column of the matrix square root of $\frac{d_z}{1-\lambda_0^{i,j,0}}\boldsymbol{C}_0(\boldsymbol{z})$.

***Procedure*** For $t \geq 0$:

1. Draw $N$ i.i.d samples $\bar{\boldsymbol{\theta}}_t^i$ from $\kappa_N^{\boldsymbol{\theta}_{t-1}^i}(d\boldsymbol{\theta})$.
2. For $i = 1, \ldots, N$:
    (a) Compute
            $\tilde{v}_t^i = \hat{p}^{J,L}(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$,        (5.17)
        where the approximate likelihood is evaluated at layer 2.
    (b) Obtain new particles $\{\boldsymbol{x}_t^{i,j}, \{\boldsymbol{z}_{ht}^{i,j,l}\}_{0\leq l\leq L-1}\}_{1\leq j\leq J}$ at time $t$ (from layers 2 and 3).
    (c) Normalize the weights
            $v_t^i = \frac{\tilde{v}_t^i}{\sum_{i=1}^N \tilde{v}_t^i}$.        (5.18)
3. Resample: set for each $m = 1, \ldots, N$ and with probability $v_t^i$
        $\{\boldsymbol{\theta}_t^m, \{\boldsymbol{x}_t^{(m,j)}, \{\boldsymbol{z}_{ht}^{m,j,l}\}_{0\leq l\leq L-1}\}_{1\leq j\leq J}\} = \{\bar{\boldsymbol{\theta}}_t^i, \{\boldsymbol{x}_t^{(i,j)}, \{\boldsymbol{z}_{ht}^{i,j,l}\}_{0\leq l\leq L-1}\}_{1\leq j\leq J}\}$.        (5.19)

***Outputs:*** $\{\boldsymbol{\theta}_t^i, \{\boldsymbol{x}_t^{(i,j)}, \{\boldsymbol{z}_{ht}^{i,j,l}\}\}_{1\leq j\leq J}\}_{1\leq i\leq N}$.
```

Second layer Algorithm 12 describes the implementation of a bank of conditional SMC schemes in the second layer of the multi-scale nested smoother, one for each particle $\bar{\boldsymbol{\theta}}_t^i$, $i = 1, \ldots, N$. In this second layer we approximate the posterior distribution with density $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i)$. The procedure is similar to the one in Algorithm 11, starting with the generation of the particles $\bar{\boldsymbol{x}}_t^{i,j}$, $j = 1, \ldots, J$, the computation of the approximate likelihood

$$\hat{p}^L(\boldsymbol{y}_t|\bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i) \approx p(\boldsymbol{y}_t|\bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$$

and the non-normalized weights $\{\tilde{u}_t^{i,j}\}_{1\le j\le J}$ in step 1a. By averaging the latter weights we can obtain $\tilde{v}_t^i$ for its use in the first layer[2]. After propagating the fast state variables in the third layer (as described below), one can resample the set $\{\boldsymbol{x}_{0:t}^{i,j}, \{\boldsymbol{z}_{ht}^{i,j,l}\}_{0\le l\le L-1}\}_{1\le j\le J}$ using the normalized weights $\{u_t^{i,j}\}_{j=1}^J$ obtained in step 1c.

Third layer Algorithm 13 outlines the implementation of a bank of UKFs [52] conditional on each parameter sample $\bar{\boldsymbol{\theta}}_t^i$ and the set of slow states $\{\bar{\boldsymbol{x}}_t^{i,j}\} \cup \boldsymbol{x}_{0:t-1}^{i,j}$. If we follow the template of the optimal smoother, then we should seek an approximation of the density $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$. However, performing this calculation with a UKF-like scheme implies that the dimension of the filter should be $d_z \times h$, in order to include the whole subsequence of states $\boldsymbol{z}_{h(t-1)+1:ht}$. Such approach would demand $2d_zh + 1$ sigma-points for each conditional UKF algorithm, and the computation of NJL covariance matrices with dimension $2d_zh \times 2d_zh$ each, which is impractical even for moderate d_z and h. For simplicity, in order to avoid operations with large matrices, we choose to compute Gaussian approximations of the marginal predictive densities $p(\boldsymbol{z}_q|\boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$, for $q = h(t-1)+1, \ldots, ht$, and then use these marginals to estimate the average of the fast states $\bar{\boldsymbol{z}}_t$ which is necessary in the micro-macro solver of Eq. (5.3). The complete procedure is outlined in Algorithm 13, with further details below.

Step 1 of Algorithm 13 generates new sigma-points in the space of fast states, $\tilde{\boldsymbol{z}}_q^{i,j,l}$, for $q = h(t-1) + 1, \ldots, ht$ and $l = 0, \ldots, L-1$, conditional on the parameters $\bar{\boldsymbol{\theta}}_t^i$ and slow variables $\boldsymbol{x}_{t-1}^{i,j}$. At each time q, we compute a predictive mean and a covariance matrix as

$$\check{\boldsymbol{z}}_q^{i,j} = \sum_{l=0}^{L-1} \lambda_{q-1}^{i,j,l} \tilde{\boldsymbol{z}}_q^{i,j,l} \quad \text{and} \tag{5.20}$$

$$\check{\boldsymbol{C}}_q^{i,j}(\boldsymbol{z}) = \sum_{l=0}^{L-1} \lambda_{q-1}^{i,j,l} (\tilde{\boldsymbol{z}}_q^{i,j,l} - \check{\boldsymbol{z}}_q^{i,j})(\tilde{\boldsymbol{z}}_q^{i,j,l} - \check{\boldsymbol{z}}_q^{i,j})^\top + \Delta_z \boldsymbol{Q}_z, \tag{5.21}$$

where the $\lambda_{q-1}^{i,j,l}$'s are the weights[3] of the sigma points $\check{\boldsymbol{z}}_{q-1}^{i,j,l}$ and $\Delta_z \boldsymbol{Q}_z$ is the covariance matrix of the noise in Eq. (5.4). The mean in Eq. (5.20) and the covariance in Eq. (5.21) yield the approximation

$$\mathcal{N}(\boldsymbol{z}_q|\check{\boldsymbol{z}}_q^{i,j}, \check{\boldsymbol{C}}_q^{i,j}(\boldsymbol{z})) \approx p(\boldsymbol{z}_q|\boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i) \tag{5.22}$$

and we compute a new weighted set of sigma-points $\{\check{\boldsymbol{z}}_q^{i,j,l}, \lambda_q^{i,j,l}\}$ to represent the Gaussian density in Eq. (5.22).

[2]The average $\tilde{v}_t^i = \frac{1}{J}\sum_{j=1}^J \tilde{u}_t^{i,j}$ is an approximation of the integral of Eq. (5.9).

[3]These weights are deterministic and can be computed a priori in different ways. See [71] for a survey of methods.

Algorithm 12 *SMC approximation of* $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$

```
**Inputs**

- Known parameter $\bar{\boldsymbol{\theta}}_t^i$ and known initial states, $\boldsymbol{x}_{t-1}^{i,j}$ and $\boldsymbol{z}_{h(t-1)}^{i,j,l}$, for $j = 1, \ldots, J$ and $l = 0, \ldots, L-1$.

**Procedure** For $t \geq 0$:

1. For $j = 1, \ldots, J$:
    (a) Compute
        $\tilde{u}_t^{i,j} = \hat{p}^L(\boldsymbol{y}_t|\bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i),$    (5.23)
        $\tilde{v}_t^i = \frac{1}{J}\sum_{j=1}^{J} \tilde{u}_t^{i,j},$    (5.24)
        where the new particle $\bar{\boldsymbol{x}}_t^{i,j}$ is generated, and the approximate likelihood is evaluated, at layer 3.
    (b) Obtain new particles $\{\boldsymbol{z}_{ht}^{i,j,l}\}_{0 \leq l \leq L-1}$ at time t, from layer 3.
    (c) Normalize the weights
        $u_t^{i,j} = \frac{\tilde{u}_t^{i,j}}{\sum_{j=1}^{J} \tilde{u}_t^{i,j}}.$    (5.25)

2. Resample: set
    $\{\boldsymbol{x}_t^{i,m}, \{\boldsymbol{z}_{ht}^{i,m,l}\}_{0 \leq l \leq L-1}\} = \{\bar{\boldsymbol{x}}_t^{i,j}, \{\boldsymbol{z}_{ht}^{i,j,l}\}_{0 \leq l \leq L-1}\}$    (5.26)
    with probability $u_t^{i,j}$ for each $m = 1, \ldots, J$.

**Outputs:** $\{\boldsymbol{x}_t^{i,j}, \{\boldsymbol{z}_{ht}^{i,j,l}\}_{0 \leq l \leq L-1}\}_{1 \leq j \leq J}$ and $\tilde{v}_t^i$.
```

Algorithm 13 *UKF approximation of* $p(\boldsymbol{z}_{h(t-1)+1:ht}|\boldsymbol{x}_{0:t}, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$

```
**Inputs**

- Integration steps $\Delta_x$, $\Delta_z$ and time scale ratio $h = \frac{\Delta_x}{\Delta_z} \in \mathbb{Z}^+$.
- Known parameter vector $\bar{\boldsymbol{\theta}}_t^i$ and initial slow state $\boldsymbol{x}_{t-1}^{i,j}$. Weighted sigma-points for the fast state at time $h(t-1)$, denoted $\{\boldsymbol{z}_{h(t-1)}^{i,j,l}, \lambda_{h(t-1)}^{i,j,l}\}_{l=0}^{L-1}$.

**Procedure** For $t > 0$:

1. Set $\check{\boldsymbol{z}}_{h(t-1)}^{i,j,l} = \boldsymbol{z}_{h(t-1)}^{i,j,l}$. For $l = 0, \ldots, L-1$ and for $q = h(t-1)+1, ..., ht$:
```

(a) *Integrate with step* Δ_z

$$\breve{\boldsymbol{z}}_q^{i,j,l} = \breve{\boldsymbol{z}}_{q-1}^{i,j,l} + \Delta_z(f_{\boldsymbol{z}}(\breve{\boldsymbol{z}}_{q-1}^{i,j,l}, \bar{\boldsymbol{\theta}}_t^i) + g_{\boldsymbol{z}}(\boldsymbol{x}_{t-1}^{i,j}, \bar{\boldsymbol{\theta}}_t^i)), \tag{5.27}$$

and compute the predictive mean, $\breve{\boldsymbol{z}}_q^{i,j}$, *and the predictive covariance matrix,* $\breve{\boldsymbol{C}}_q^{i,j}(\boldsymbol{z})$, *using Eqs.* (5.20) *and* (5.21).

(b) *Approximate the predictive pdf of the fast states as Gaussian density,*

$$p(\boldsymbol{z}_q|\boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i) \approx \mathcal{N}(\boldsymbol{z}_q|\breve{\boldsymbol{z}}_q^{i,j}, \breve{\boldsymbol{C}}_q^{i,j}(\boldsymbol{z})).$$

Represent this Gaussian distribution by a set of weighted sigma-points denoted $\{\breve{\boldsymbol{z}}_q^{i,j,l}, \lambda_q^{i,j,l}\}_{l=0}^{L-1}$.

2. *In the space of the slow state variables:*

(a) *For* $l = 0, \ldots, L-1$, *project the sigma-points* $\breve{\boldsymbol{z}}_{h(t-1)+1:ht}^{i,j,l}$ *to obtain sigma-points in the space of the slow states,*

$$\tilde{\boldsymbol{x}}_t^{i,j,l} = \boldsymbol{x}_{t-1}^{i,j} + \Delta_x(f_{\boldsymbol{x}}(\boldsymbol{x}_{t-1}^{i,j}, \bar{\boldsymbol{\theta}}_t^i) + g_{\boldsymbol{x}}(\bar{\boldsymbol{z}}_t^{i,j,l}, \bar{\boldsymbol{\theta}}_t^i)), \tag{5.28}$$

where $\bar{\boldsymbol{z}}_t^{i,j,l} = \frac{1}{h}\sum_{q=h(t-1)+1}^{ht} \breve{\boldsymbol{z}}_q^{i,j,l}$. *Then, compute a mean vector* $\breve{\boldsymbol{x}}_t^{i,j}$ *and a covariance matrix* $\breve{\boldsymbol{C}}_t^{i,j}(\boldsymbol{x})$ *using Eqs.* (5.34) *and* (5.35).

(b) *Sample* $\bar{\boldsymbol{x}}_t^{i,j} \sim \mathcal{N}(\boldsymbol{x}_t|\breve{\boldsymbol{x}}_t^{i,j}, \breve{\boldsymbol{C}}_t^{i,j}(\boldsymbol{x}))$.

3. *Once we collect a new observation* $\boldsymbol{y}_t$,

(a) *For* $l = 0, \ldots, L-1$, *project the sigma-points* $\breve{\boldsymbol{z}}_{ht}^{i,j,l}$ *and the new sample* $\bar{\boldsymbol{x}}_t^{i,j}$ *into the observation space,*

$$\tilde{\boldsymbol{y}}_t^{i,j,l} = l(\breve{\boldsymbol{z}}_{ht}^{i,j,l}, \bar{\boldsymbol{x}}_t^{i,j}, \bar{\boldsymbol{\theta}}_t^i), \tag{5.29}$$

then compute the mean vector $\hat{\boldsymbol{y}}_{ht}^{i,j}$ *and the covariance matrix* $\boldsymbol{C}_t^{i,j}(\boldsymbol{y})$ *using Eqs.* (5.36) *and* (5.37).

(b) *Compute* $\tilde{w}_t^{i,j,l} = p(\boldsymbol{y}_t|\breve{\boldsymbol{z}}_{ht}^{i,j,l}, \bar{\boldsymbol{x}}_t^{i,j}, \bar{\boldsymbol{\theta}}_t^i)p(\bar{\boldsymbol{x}}_t^{i,j}|\breve{\boldsymbol{z}}_{h(t-1)+1:ht}^{i,j,l}, \boldsymbol{x}_{t-1}^{i,j}, \bar{\boldsymbol{\theta}}_t^i)$ *and the weights for the second layer*

$$\tilde{u}_t^{i,j} = \sum_{l=0}^{L-1} \lambda_{ht}^{i,j,l} \tilde{w}_t^{i,j,l}. \tag{5.30}$$

4. *Update the mean and the covariance matrix of the fast variables*

$$\boldsymbol{K}_t(\boldsymbol{z}) = \boldsymbol{C}_t^{i,j}(\boldsymbol{z}, \boldsymbol{y})\big(\boldsymbol{C}_t^{i,j}(\boldsymbol{y})\big)^{-1}, \tag{5.31}$$

$$\hat{\boldsymbol{z}}_{ht}^{i,j} = \breve{\boldsymbol{z}}_{ht}^{i,j} + \boldsymbol{K}_t(\boldsymbol{z})\big(\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t^{i,j}\big) \quad \textit{and} \tag{5.32}$$

$$\hat{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z}) = \breve{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z}) + \boldsymbol{K}_t(\boldsymbol{z})\boldsymbol{C}_t^{i,j}(\boldsymbol{y})\big(\boldsymbol{K}_t(\boldsymbol{z})\big)^\top, \tag{5.33}$$

where $\boldsymbol{C}_t^{i,j}(\boldsymbol{z}, \boldsymbol{y})$ *is the cross-covariance matrix computed in Eq.* (5.38).

5. *From the new Gaussian pdf* $\mathcal{N}(\boldsymbol{z}_{ht}|\hat{\boldsymbol{z}}_{ht}^{i,j}, \hat{\boldsymbol{C}}_t^{i,j}(\boldsymbol{z}))$, *generate* $L = 2d_z + 1$ *sigma-points and weights* $\{\boldsymbol{z}_{ht}^{i,j,l}, \lambda_{ht}^{i,j,l}\}_{0 \le l \le L-1}$.

Outputs: $\{\boldsymbol{z}_{ht}^{i,j,l}\}_{0 \le l \le L-1}$, $\bar{\boldsymbol{x}}_t^{i,j}$ *and* $\tilde{u}_t^{i,j}$.

In step 2 of Algorithm 13 we use the sigma-points at time $q = ht$ to generate new particles for the slow states at time t. Specifically, we project the $\breve{\boldsymbol{z}}_{ht}^{i,j,l}$'s through the state equation of the slow state variables to obtain sigma-points in the space of the slow variables, denoted $\tilde{\boldsymbol{x}}_t^{i,j,l}$. From these sigma-points, we obtain a mean vector and a covariance matrix, respectively,

$$\breve{\boldsymbol{x}}_t^{i,j} = \sum_{l=0}^{L-1} \lambda_{ht}^{i,j,l} \tilde{\boldsymbol{x}}_t^{i,j,l} \quad \text{and} \tag{5.34}$$

$$\breve{\boldsymbol{C}}_t^{i,j}(\boldsymbol{x}) = \sum_{l=0}^{L-1} \lambda_{ht}^{i,j,l} (\tilde{\boldsymbol{x}}_t^{i,j,l} - \breve{\boldsymbol{x}}_t^{i,j})(\tilde{\boldsymbol{x}}_t^{i,j,l} - \breve{\boldsymbol{x}}_t^{i,j})^\top + \Delta_x \boldsymbol{Q}_x, \tag{5.35}$$

where $\Delta_x \boldsymbol{Q}_x$ is the covariance matrix of the noise in Eq. (5.3). Eqs. (5.34) and (5.35) yield a Gaussian approximation of the predictive pdf of the slow states, namely,

$$p(\boldsymbol{x}_t | \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{0:t-1}, \bar{\boldsymbol{\theta}}_t^i) \approx \mathcal{N}(\boldsymbol{x} | \breve{\boldsymbol{x}}_t^{i,j}, \breve{\boldsymbol{C}}_t^{i,j}(\boldsymbol{x})).$$

We generate the new particle $\bar{\boldsymbol{x}}_t^{i,j}$ from this Gaussian density.

In step 3 of the algorithm we propagate the sigma-points $\breve{\boldsymbol{z}}_{ht}^{i,j,l}$ and the particle $\bar{\boldsymbol{x}}_t^{i,j}$ through the observation function $l(\cdot)$ to obtain projected sigma-points (on the observation space) $\{\tilde{\boldsymbol{y}}_t^{i,j,l}\}_{0 \le l \le L-1}$. We use these projected sigma-points to obtain a predictive mean and covariance matrix for the observation $\boldsymbol{y}_t$, namely,

$$\hat{\boldsymbol{y}}_t^{i,j} = \sum_{l=0}^{L-1} \lambda_{ht}^{i,j,l} \tilde{\boldsymbol{y}}_t^{i,j,l} \quad \text{and} \tag{5.36}$$

$$\boldsymbol{C}_t^{i,j}(\boldsymbol{y}) = \sum_{l=0}^{L-1} \lambda_{ht}^{i,j,l} (\tilde{\boldsymbol{y}}_t^{i,j,l} - \hat{\boldsymbol{y}}_t^{i,j})(\tilde{\boldsymbol{y}}_t^{i,j,l} - \hat{\boldsymbol{y}}_t^{i,j})^\top + \boldsymbol{R}, \tag{5.37}$$

where $\boldsymbol{R}$ is the covariance matrix of the noise in the observation equation. At this step we also compute the non-normalized importance weight $\tilde{u}_t^{i,j}$ which is output to layer 2.

In step 4, we compute the Kalman gain using the observation covariance matrix of Eq. (5.37) and the cross-covariance matrix

$$\boldsymbol{C}_t^{i,j}(\boldsymbol{z}, \boldsymbol{y}) = \sum_{l=0}^{L-1} \lambda_{ht}^{i,j,l} (\breve{\boldsymbol{z}}_{ht}^{i,j,l} - \breve{\boldsymbol{z}}_{ht}^{i,j})(\tilde{\boldsymbol{y}}_t^{i,j,l} - \hat{\boldsymbol{y}}_t^{i,j})^\top. \tag{5.38}$$

Then we update the mean, $\hat{\boldsymbol{z}}_{ht}^{i,j}$, and covariance matrix, $\hat{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z})$, of the fast state variables to obtain the approximation

$$p(\boldsymbol{z}_{ht} | \bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i) \approx \mathcal{N}(\boldsymbol{z}_{ht} | \hat{\boldsymbol{z}}_{ht}^{i,j}, \hat{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z})). \tag{5.39}$$

Finally, in step 5 we generate new weighted sigma-points to characterize the Gaussian pdf in Eq. (5.39).

5.3.2 Second scheme

The method in Section 5.3.1 may still have a prohibitive computational cost, as it generates a total of $N \times J \times L$ particles in the joint space of the parameters, the slow states and the fast states (if we count sigma-points as deterministic particles). In this section we describe a computationally-lighter procedure that replaces the SMC procedure in layer 2 by an EnKF [36] and the UKF in layer 3 by a simpler EKF [5]. The complete procedure is described below.

First layer We describe the use of a SMC algorithm for the first layer of the nested algorithm in order to approximate $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$. This is the same as in Algorithm 11, except that we need to take into account the initializations needed for layers 2 and 3. Algorithm 14 receives as inputs the prior distributions of the parameters, $p(\boldsymbol{\theta})$, and both subsets of state variables, $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{z}_0)$. They are used to generate

- the initial particles from $p(\boldsymbol{\theta})$ for the SMC scheme, denoted $\{\boldsymbol{\theta}_0^i\}_{i=1}^N$,
- the samples from $p(\boldsymbol{x}_0)$ utilized to build the ensembles $\boldsymbol{X}_0^i$, $i = 1, \dots, N$, for the EnKFs in the second layer,
- and the mean and covariance matrix of $\boldsymbol{z}$ for the EKFs in the third layer, denoted $\boldsymbol{z}_0^{i,j} = \mathbb{E}[\boldsymbol{z}_0]$ and $\boldsymbol{C}_0^{i,j}(\boldsymbol{z}) = \text{Cov}(\boldsymbol{z}_0)$, respectively (note that they are the same for all i and j).

The rest of the procedure is the same as in Algorithm 11.

Second layer In Algorithm 15 we employ an EnKF to obtain ensemble approximations of $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$ and $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i)$. The ensembles are denoted $\bar{\boldsymbol{X}}_t^i$ and $\boldsymbol{X}_t^i$, respectively, and they are used to approximate the computations that involved the joint pdfs $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$ and $p(\boldsymbol{x}_{0:t}|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i)$ in the optimal smoother. Note that all calculations are conditional on the i-th parameter particle, $\bar{\boldsymbol{\theta}}_t^i$.

The scheme is similar to Algorithm 12. At step 1a, we retrieve the new samples $\bar{\boldsymbol{x}}_t^{i,j}$ and the approximate likelihood $\tilde{u}_t^{i,j}$ from layer 3, and compute the non-normalized importance weight $\tilde{v}_t^i$ which is output to layer 1.

At steps 2 and 3 we generate the predictive ensemble for the slow states, $\bar{\boldsymbol{X}}_t^i$, and the observations, $\boldsymbol{Y}_t^i$, respectively. These ensembles are then used, when the new observation $\boldsymbol{y}_t$ is collected, to compute an updated ensemble $\boldsymbol{X}_t^i$ which yields non-weighted particle approximation of the distribution with pdf $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i)$. The update step of the EnKF can be implemented in different ways. Here we follow the scheme in [66] (as explained in Section 2.2.4) which avoids the direct computation of the inverse of the covariance observation matrix ($d_y \times d_y$), being better suited for high-dimensional systems.

Third layer In Algorithm 16 we describe how to use an EKF to obtain Gaussian approximations $\mathcal{N}(\boldsymbol{z}_q|\check{\boldsymbol{z}}_q, \check{\boldsymbol{C}}_q(\boldsymbol{z})) \approx p(\boldsymbol{z}_q|\boldsymbol{x}_{t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$, $q = h(t-1)+1, \dots, ht$, and the updated pdf $\mathcal{N}(\boldsymbol{z}_{ht}|\hat{\boldsymbol{z}}_{ht}^{i,j}, \hat{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z})) \approx p(\boldsymbol{z}_{ht}|\bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{t-1}^{i,j}, \boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i)$. We also generate the new slow states at time t, denoted $\bar{\boldsymbol{x}}_t^{i,j}$, and the likelihood estimates $\tilde{u}_t^{i,j} \approx p(\boldsymbol{y}_t|\bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$. Note that all computations in this layer are conditional on $\boldsymbol{x}_{t-1}^{i,j}$ and $\bar{\boldsymbol{\theta}}_t^i$.

In step 1, the algorithm propagates the mean, $\hat{\boldsymbol{z}}_{h(t-1)}^{i,j}$, and the covariance matrix, $\hat{\boldsymbol{C}}_{h(t-1)}^{i,j}(\boldsymbol{z})$, for $q = h(t-1)+1, \dots, ht$, conditional on the parameters $\bar{\boldsymbol{\theta}}_t^i$ and slow variables $\boldsymbol{x}_{t-1}^{i,j}$. At each time step q, we obtain a predictive mean, $\check{\boldsymbol{z}}_q^{i,j}$, and the predictive covariance matrix, $\check{\boldsymbol{C}}_q^{i,j}(\boldsymbol{z})$. The average of the predictive means, $\bar{\boldsymbol{z}}_t^{i,j} = \frac{1}{h}\sum_{q=h(t-1)+1}^{ht} \check{\boldsymbol{z}}_q^{i,j}$, is then used to propagate the slow state $\boldsymbol{x}_{t-1}^{i,j}$ and generate the new sample $\bar{\boldsymbol{x}}_t^{i,j}$ from the Gaussian approximation

$$\mathcal{N}(\boldsymbol{x}_t|\check{\boldsymbol{x}}_t^{i,j}, \Delta_x \boldsymbol{Q}_x) \approx p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$$

at step 2. Note that the covariance of the $\boldsymbol{z}_q^{i,j}$'s is neglected for simplicity in this computation and we use just the covariance matrix $\Delta_x \boldsymbol{Q}_x$ of the slow state in Eq. (5.3).

In step 3 we project the predictive mean $\check{\boldsymbol{z}}_{ht}^{i,j}$ and the sample $\bar{\boldsymbol{x}}_t^{i,j}$ into the observation space to obtain the predictive observation $\hat{\boldsymbol{y}}_t^{i,j}$. When the *actual* observation $\boldsymbol{y}_t$ is available we also estimate the likelihood $p(\boldsymbol{y}_t|\bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$ as

$$\tilde{u}_t^{i,j} = p(\boldsymbol{y}_t|\check{\boldsymbol{z}}_{ht}^{i,j}, \bar{\boldsymbol{x}}_t^{i,j}, \bar{\boldsymbol{\theta}}_t^i)p(\boldsymbol{x}_t|\check{\boldsymbol{z}}_{h(t-1)+1:ht}^{i,j}, \boldsymbol{x}_{t-1}^{i,j}, \bar{\boldsymbol{\theta}}_t^i).$$

Note that we use the predictive means $\check{\boldsymbol{z}}_{h(t-1):ht}^{i,j}$ for simplicity, instead of actually integrating w.r.t. the random states $\boldsymbol{z}_{h(t-1):ht}$.

Finally, in step 4, we compute the Kalman gain using the predictive covariance matrix of Eq. (5.50) and the Jacobian matrix $\boldsymbol{H}_t^{i,j}(\boldsymbol{z})$. Then, we update the mean, $\hat{\boldsymbol{z}}_{ht}^{i,j}$, and covariance matrix, $\hat{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z})$, of the fast variables for the next time step.

Algorithm 14 *SMC approximation of $p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t})$ in the second method* ***Inputs***

- *Prior distributions $p(\boldsymbol{\theta})$, $p(\boldsymbol{x}_0)$ and $p(\boldsymbol{z}_0)$.*
- *A Markov kernel $\kappa_N^{\boldsymbol{\theta}'}(d\boldsymbol{\theta})$ which, given $\boldsymbol{\theta}'$, generates jittered parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$.*

Initialization: *this is a joint initialization for all three layers.*

- *Draw N i.i.d. sample $\boldsymbol{\theta}_0^i$, $i = 1, \dots, N$ from the prior distribution $p(\boldsymbol{\theta})$.*
- *Draw NJ i.i.d. samples $\boldsymbol{x}_0^{i,j}$, $i = 1, \dots, N$, $j = 1, \dots, J$, from the prior distribution $p(\boldsymbol{x}_0)$, and build the ensembles $\boldsymbol{X}_0^i$, $i = 1, \dots, N$, as*

$$\boldsymbol{X}_0^i = [\boldsymbol{x}_0^{i,1}, \dots, \boldsymbol{x}_0^{i,J}]. \tag{5.40}$$

- *Set $\hat{\boldsymbol{z}}_0^{i,j} = \bar{\boldsymbol{z}}_0$ and $\hat{\boldsymbol{C}}_0^{i,j}(\boldsymbol{z}) = \boldsymbol{C}_0$, for $i = 1, \dots, N$, $j = 1, \dots, J$, where $\bar{\boldsymbol{z}}_0$ and $\boldsymbol{C}_0$ are the prior mean and prior covariance of $\boldsymbol{z}_0$, respectively, obtained from the prior density $p(\boldsymbol{z}_0)$.*

Procedure *For $t \geq 0$:*

1. *Draw N i.i.d samples $\bar{\boldsymbol{\theta}}_t^i$ from $\kappa_N^{\boldsymbol{\theta}_{t-1}^i}(d\boldsymbol{\theta})$.*
2. *For $i = 1, \dots, N$:*
   (a) *Retrieve*
   $$\tilde{v}_t^i = \hat{p}^J(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i), \tag{5.41}$$
   where the approximate likelihood is evaluated at layer 2.
   (b) *Obtain new particles $\{\hat{\boldsymbol{X}}_t^i, \{\hat{\boldsymbol{z}}_{ht}^{i,j}, \hat{\boldsymbol{C}}_t^{i,j}(\boldsymbol{z})\}_{1\leq j\leq J}\}$ at time t (from layers 2 and 3).*
   (c) *Normalize the weights*
   $$v_t^i = \frac{\tilde{v}_t^i}{\sum_{i=1}^N \tilde{v}_t^i}. \tag{5.42}$$
3. *Resample: set for each $m = 1, \dots, N$*
$$\{\boldsymbol{\theta}_t^m, \boldsymbol{X}_t^m, \{\hat{\boldsymbol{z}}_{ht}^{m,j}, \hat{\boldsymbol{C}}_t^{m,j}(\boldsymbol{z})\}_{j=1}^J\} = \{\bar{\boldsymbol{\theta}}_t^i, \hat{\boldsymbol{X}}_t^i, \{\hat{\boldsymbol{z}}_{ht}^{i,j}, \hat{\boldsymbol{C}}_t^{i,j}(\boldsymbol{z})\}_{j=1}^J\} \tag{5.43}$$
with probability v_t^i.

Outputs: $\{\boldsymbol{\theta}_t^i, \boldsymbol{X}_t^i, \{\hat{\boldsymbol{z}}_{ht}^{i,j}, \hat{\boldsymbol{C}}_t^{i,j}(\boldsymbol{z})\}_{j=1}^J\}_{i=1}^N$.

Algorithm 15 *EnKF approximation of* $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$

Inputs

- *Parameter vector* $\bar{\boldsymbol{\theta}}_t^i$; *ensemble of slow states,* $\boldsymbol{X}_{t-1}^i = [\boldsymbol{x}_{t-1}^{i,1}, \dots, \boldsymbol{x}_{t-1}^{i,J}]$; *mean vector* $\hat{\boldsymbol{z}}_{h(t-1)}^{i,j}$ *and the covariance matrix* $\hat{\boldsymbol{C}}_{t-1}^{i,j}(\boldsymbol{z})$, *for* $j = 1, \dots, J$.

Procedure *For* $t \geq 0$:

1. *For* $j = 1, \dots, J$:

   (a) *Retrieve the new sample* $\bar{\boldsymbol{x}}_t^{i,j}$ *and the likelihood estimate* $\tilde{u}_t^{i,j}$ *from layer 3 and compute the non-normalized importance weight*

   $$\tilde{v}_t^i = \frac{1}{J} \sum_{j=1}^{J} \tilde{u}_t^{i,j}, \tag{5.44}$$

   (b) *Obtain the new mean* $\hat{\boldsymbol{z}}_{ht}^{i,j}$ *and covariance matrix* $\hat{\boldsymbol{C}}_t^{i,j}(\boldsymbol{z})$ *at time* t, *from layer 3.*

2. *Compute the predictive mean* $\check{\boldsymbol{x}}_t^i$ *and construct the predictive ensemble* $\bar{\boldsymbol{X}}_t^i$ *as*

$$\check{\boldsymbol{x}}_t^i = \frac{1}{J} \sum_{j=1}^{J} \bar{\boldsymbol{x}}_t^{i,j} \quad \text{and} \quad \bar{\boldsymbol{X}}_t^i = [\bar{\boldsymbol{x}}_t^{i,1}, \dots, \bar{\boldsymbol{x}}_t^{i,J}]. \tag{5.45}$$

3. *Obtain predictive observations* $\tilde{\boldsymbol{y}}_t^{i,j}$ *from layer 3, then compute the mean* $\hat{\boldsymbol{y}}_t^i$ *and the ensemble* $\boldsymbol{Y}_t^i$ *as*

$$\hat{\boldsymbol{y}}_t^i = \frac{1}{J} \sum_{j=1}^{J} \tilde{\boldsymbol{y}}_t^{i,j} \quad \text{and} \quad \boldsymbol{Y}_t^i = [\tilde{\boldsymbol{y}}_t^{i,1}, \dots, \tilde{\boldsymbol{y}}_t^{i,J}], \quad \text{for } j = 1, \dots, J. \tag{5.46}$$

4. *Update the ensemble of slow variables*

$$\begin{aligned}
\boldsymbol{C}_t^i(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{J} \tilde{\boldsymbol{X}}_t^i (\tilde{\boldsymbol{Y}}_t^i)^\top, \\
\big(\boldsymbol{C}_t^i(\boldsymbol{y})\big)^{-1} &= \boldsymbol{R}^{-1} - \boldsymbol{R}^{-1} \frac{1}{J} \tilde{\boldsymbol{Y}}_t^i \left(\boldsymbol{I}_J + (\tilde{\boldsymbol{Y}}_t^i)^\top \boldsymbol{R}^{-1} \frac{1}{J} \tilde{\boldsymbol{Y}}_t^i \right)^{-1} (\tilde{\boldsymbol{Y}}_t^i)^\top \boldsymbol{R}^{-1}, \\
\boldsymbol{K}_t(\boldsymbol{x}) &= \boldsymbol{C}_t^i(\boldsymbol{x}, \boldsymbol{y}) \big(\boldsymbol{C}_t^i(\boldsymbol{y})\big)^{-1}, && (5.47) \\
\hat{\boldsymbol{X}}_t^i &= \bar{\boldsymbol{X}}_t^i + \boldsymbol{K}_t(\boldsymbol{x}) \big(\boldsymbol{y}_t \mathbb{1}_{d_y \times J} + \boldsymbol{T}_t^i - \boldsymbol{Y}_t^i\big), && (5.48)
\end{aligned}$$

where $\tilde{\boldsymbol{X}}_t^i = \bar{\boldsymbol{X}}_t^i - \check{\boldsymbol{x}}_t^i \mathbb{1}_{d_x \times J}$ *and* $\tilde{\boldsymbol{Y}}_t^i = \boldsymbol{Y}_t^i - \hat{\boldsymbol{y}}_t^i \mathbb{1}_{d_y \times J}$, $\boldsymbol{C}_t^i(\boldsymbol{x}, \boldsymbol{y})$ *is the cross covariance matrix,* $\boldsymbol{C}_t^i(\boldsymbol{y})$ *is the covariance matrix of the observation,* $\boldsymbol{R}$ *is the covariance matrix of the noise in the observation equation,* $\mathbb{1}_{a \times b}$ *is a* $a \times b$ *matrix of ones and* $\boldsymbol{T}_t^i = \big[\boldsymbol{r}_t^1, \dots, \boldsymbol{r}_t^J\big]$ *with* $\boldsymbol{r}_t^j \sim \mathcal{N}(\boldsymbol{r}_t|\boldsymbol{0}, \boldsymbol{R})$ *is a matrix of Gaussian perturbations.*

Outputs: $\hat{\boldsymbol{X}}_t^i$, $\{\hat{\boldsymbol{z}}_{ht}^{i,j}, \hat{\boldsymbol{C}}_t^{i,j}(\boldsymbol{z})\}_{j=1}^J$ *and* $\tilde{v}_t^i$.

Algorithm 16 *EKF approximation of $p(\boldsymbol{z}_{ht}|\boldsymbol{x}_t, \boldsymbol{y}_{1:t}, \boldsymbol{\theta})$*

Inputs

- *Integration steps Δ_x, Δ_z and time scale ratio $h = \frac{\Delta_x}{\Delta_z} \in \mathbb{Z}^+$.*
- *Parameter vector $\bar{\boldsymbol{\theta}}_t^i$; slow states $\boldsymbol{x}_{t-1}^{i,j}$ (i.e. the j-th column of $\boldsymbol{X}_{t-1}^i$) and mean fast state $\hat{\boldsymbol{z}}_{h(t-1)}^{i,j}$ with covariance matrix $\hat{\boldsymbol{C}}_{t-1}^{i,j}(\boldsymbol{z})$.*

Procedure *For $t \geq 0$:*

1. *Set $\check{\boldsymbol{z}}_{h(t-1)}^{i,j} = \boldsymbol{z}_{h(t-1)}^{i,j}$ and $\check{\boldsymbol{C}}_q^{i,j}(\boldsymbol{z}) = \boldsymbol{C}_{h(t-1)}^{i,j}(\boldsymbol{z})$.*
   For $q = h(t-1)+1, ..., ht$, compute

$$\check{\boldsymbol{z}}_q^{i,j} = \check{\boldsymbol{z}}_{q-1}^{i,j} + \Delta_z(f_{\boldsymbol{z}}(\check{\boldsymbol{z}}_{q-1}^{i,j}, \bar{\boldsymbol{\theta}}_t^i) + g_{\boldsymbol{z}}(\boldsymbol{x}_{t-1}^{i,j}, \bar{\boldsymbol{\theta}}_t^i)), \tag{5.49}$$

$$\check{\boldsymbol{C}}_q^{i,j}(\boldsymbol{z}) = \boldsymbol{J}_q^{i,j}(\boldsymbol{z})\check{\boldsymbol{C}}_{q-1}^{i,j}(\boldsymbol{z})\big(\boldsymbol{J}_q^{i,j}(\boldsymbol{z})\big)^\top + \Delta_z \boldsymbol{Q}_z, \tag{5.50}$$

   where $\boldsymbol{J}_q^{i,j}(\boldsymbol{z})$ is the Jacobian matrix of the transition function of $\boldsymbol{z}$ and $\Delta_z\boldsymbol{Q}_z$ is the covariance matrix of the noise in Eq. (5.4).

2. *In the space of the slow state variables:*
   (a) *Project the predictive means $\check{\boldsymbol{z}}_{h(t-1)+1:ht}^{i,j}$ into the space of slow variables,*

$$\check{\boldsymbol{x}}_t^{i,j} = \boldsymbol{x}_{t-1}^{i,j} + \Delta_x(f_{\boldsymbol{x}}(\boldsymbol{x}_{t-1}^{i,j}, \bar{\boldsymbol{\theta}}_t^i) + g_{\boldsymbol{x}}(\bar{\boldsymbol{z}}_t^{i,j}, \bar{\boldsymbol{\theta}}_t^i)), \tag{5.51}$$

   where $\bar{\boldsymbol{z}}_t^{i,j} = \frac{1}{h}\sum_{q=h(t-1)+1}^{ht} \check{\boldsymbol{z}}_q^{i,j}$.
   (b) *Sample $\bar{\boldsymbol{x}}_t^{i,j} \sim \mathcal{N}(\boldsymbol{x}_t|\check{\boldsymbol{x}}_t^{i,j}, \Delta_x\boldsymbol{Q}_x)$, where $\Delta_x\boldsymbol{Q}_x$ is the covariance matrix of the noise in Eq. (5.3).*

3. *When a new observation $\boldsymbol{y}_t$ is collected:*
   (a) *Project the predictive mean $\check{\boldsymbol{z}}_{ht}^{i,j}$ and the slow state $\bar{\boldsymbol{x}}_t^{i,j}$ into the observation space*

$$\tilde{\boldsymbol{y}}_t^{i,j} = l(\check{\boldsymbol{z}}_{ht}^{i,j}, \bar{\boldsymbol{x}}_t^{i,j}, \bar{\boldsymbol{\theta}}_t^i). \tag{5.52}$$

   (b) *Compute $\tilde{u}_t^{i,j} = p(\boldsymbol{y}_t|\check{\boldsymbol{z}}_{ht}^{i,j}, \bar{\boldsymbol{x}}_t^{i,j}, \bar{\boldsymbol{\theta}}_t^i)p(\boldsymbol{x}_t|\check{\boldsymbol{z}}_{h(t-1)+1:ht}^{i,j}, \boldsymbol{x}_{t-1}^{i,j}, \bar{\boldsymbol{\theta}}_t^i)$. This is an estimate of the likelihood $p(\boldsymbol{y}_t|\bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$.*

4. *Update the mean and the covariance matrix of the fast variables*

$$\boldsymbol{K}_t(\boldsymbol{z}) = \check{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z})\boldsymbol{H}_t^{i,j}(\boldsymbol{z})^\top\left(\boldsymbol{H}_t^{i,j}(\boldsymbol{z})\check{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z})\boldsymbol{H}_t^{i,j}(\boldsymbol{z})^\top + \boldsymbol{R}\right)^{-1}, \tag{5.53}$$

$$\hat{\boldsymbol{z}}_{ht}^{i,j} = \check{\boldsymbol{z}}_{ht}^{i,j} + \boldsymbol{K}_t(\boldsymbol{z})\big(\boldsymbol{y}_t - \tilde{\boldsymbol{y}}_t^{i,j}\big) \quad and \tag{5.54}$$

$$\hat{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z}) = \left(\boldsymbol{I}_{d_z} - \boldsymbol{K}_t(\boldsymbol{z})\boldsymbol{H}_t^{i,j}(\boldsymbol{z})\right)\check{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z}), \tag{5.55}$$

   where $\boldsymbol{H}_t^{i,j}(\boldsymbol{z})$ is the Jacobian matrix of function $l(\cdot, \bar{\boldsymbol{x}}_t^{i,j}, \bar{\boldsymbol{\theta}}_t^i)$ w.r.t. $\check{\boldsymbol{z}}_{ht}^{i,j}$ and $\boldsymbol{R}$ is the covariance matrix of the noise in the observation equation. We obtain the approximation $\mathcal{N}(\boldsymbol{z}_{ht}|\hat{\boldsymbol{z}}_{ht}^{i,j}, \hat{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z})) \approx p(\boldsymbol{z}_{ht}|\bar{\boldsymbol{x}}_t^{i,j}, \boldsymbol{x}_{0:t-1}^{i,j}, \boldsymbol{y}_{0:t}, \bar{\boldsymbol{\theta}}_t^i)$.

Outputs: *$\hat{\boldsymbol{z}}_{ht}^{i,j}$,$\hat{\boldsymbol{C}}_{ht}^{i,j}(\boldsymbol{z})$, $\bar{\boldsymbol{x}}_t^{i,j}$ and $\tilde{u}_t^{i,j}$.*

5.4 Example

5.4.1 Stochastic two-scale Lorenz 96 model

In order to illustrate the application of the methods described in Section 5.3, we consider a stochastic version of the two-scale Lorenz 96 model [11], which depends on a set of fixed parameters, a set of fast variables and a set of slow variables[4]. The slow variables are represented by a d_x-dimensional vector, $\boldsymbol{x}$, while the fast variables, $\boldsymbol{z}$, are d_z-dimensional. Let us assume there are R fast variables per slow variable, therefore $d_z = Rd_x$. The system is described, in continuous-time τ, by the SDEs

$$\begin{aligned} dx_j &= \Big[- x_{j-1}(x_{j-2} - x_{j+1}) - x_j + F - \tfrac{HC}{B} \textstyle\sum_{l=(j-1)R}^{Rj-1} z_l \Big] d\tau + \sigma_x dv_j, & (5.56)\\ dz_l &= \Big[- CBz_{l+1}(z_{l+2} - z_{l-1}) - Cz_l + \tfrac{CF}{B} + \tfrac{HC}{B} x_{\lfloor (l-1)/R \rfloor} \Big] d\tau + \sigma_z dw_l, & (5.57) \end{aligned}$$

where $j = 0, \ldots, d_x - 1$, $l = 0, \ldots, d_z - 1$; $\boldsymbol{v} = (v_0, \ldots, v_{d_x-1})^\top$ and $\boldsymbol{w} = (w_0, \ldots, w_{d_z-1})^\top$ are, respectively, d_x- and d_z-dimensional vectors of independent standard Wiener processes; $\sigma_x > 0$ and $\sigma_z > 0$ are known scale parameters and $\boldsymbol{\theta} = (F, H, C, B)^\top \in \mathbb{R}$ are static model parameters. Using a micro-macro solver [96, 102] that runs an Euler-Maruyama scheme at each time-scale to integrate Eqs. (5.56)–(5.57), the discrete-time state equation can be written as

$$\begin{aligned} x_{t+1,j} &= x_{t,j} + \Delta_x(f_{\boldsymbol{x},j}(\boldsymbol{x}_t, \boldsymbol{\theta}) + g_{\boldsymbol{x},j}(\bar{\boldsymbol{z}}_{t+1}, \boldsymbol{\theta})) + \sqrt{\Delta_x}\sigma_x v_{t+1,j}, & (5.58)\\ z_{n+1,l} &= z_{n,l} + \Delta_z(f_{\boldsymbol{z},l}(\boldsymbol{x}_{\lfloor \frac{n}{h} \rfloor}, \boldsymbol{\theta}) + g_{\boldsymbol{z},l}(\boldsymbol{z}_n, \boldsymbol{\theta})) + \sqrt{\Delta_z}\sigma_z w_{n+1,l}, & (5.59) \end{aligned}$$

where

$$\boldsymbol{x}_t = (x_{t,0}, \ldots, x_{t,d_x-1})^\top \quad \text{and} \quad \boldsymbol{z}_n = (z_{n,0}, \ldots, z_{n,d_z-1})^\top$$

are the discrete-time slow and fast variables, respectively; $\bar{\boldsymbol{z}}_t$ is the time-average

$$\bar{\boldsymbol{z}}_t = \frac{1}{h} \sum_{n=h(t-1)+1}^{ht} \boldsymbol{z}_n$$

and we denote $\bar{\boldsymbol{z}}_t = (\bar{z}_{t,0}, \ldots, \bar{z}_{t,d_z-1})^\top$; the terms $v_{t,j}$ and $w_{n,l}$ are independent Gaussian variables with identical $\mathcal{N}(\cdot|0,1)$ pdf for all t, j, n and l, and the functions

$$\begin{aligned} f_{\boldsymbol{x},j} &: \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_x}, \\ g_{\boldsymbol{x},j} &: \mathbb{R}^{d_z} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_x}, \\ f_{\boldsymbol{z},l} &: \mathbb{R}^{d_x} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_z} \quad \text{and} \\ g_{\boldsymbol{z},l} &: \mathbb{R}^{d_z} \times \mathbb{R}^{d_\theta} \to \mathbb{R}^{d_z} \end{aligned}$$

can be expressed as

$$\begin{aligned} & f_{\boldsymbol{x},j}(\boldsymbol{x}_t, \boldsymbol{\theta}) = -x_{t,j-1}(x_{t,j-2} - x_{t,j+1}) - x_{t,j} + F, \\ & g_{\boldsymbol{x},j}(\bar{\boldsymbol{z}}_t, \boldsymbol{\theta}) = -\frac{HC}{B} \sum_{l=(j-1)R}^{Rj-1} \bar{z}_{t,l}, \end{aligned}$$

[4]This is the same set of SDEs in the simulations of Chapter 3. The way in which fast variables are handled is totally different, though.

$$f_{\boldsymbol{z},l}(\boldsymbol{x}_t, \boldsymbol{\theta}) = \frac{HC}{B} x_{t,\lfloor (l-1)/R \rfloor} \quad \text{and}$$
$$g_{\boldsymbol{z},l}(\boldsymbol{z}_n, \boldsymbol{\theta}) = -CBz_{n,l+1}(z_{n,l+2} - z_{n,l-1}) - Cz_{n,l} + \frac{CF}{B}.$$

We assume that the observations are linear and Gaussian, namely,

$$\boldsymbol{y}_t = \boldsymbol{A}_t \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{z}_{ht} \end{bmatrix} + \boldsymbol{r}_t, \tag{5.60}$$

where $\boldsymbol{A}_t$ is a known $d_y \times (d_x + d_z)$ matrix and $\boldsymbol{r}_t$ is a d_y-dimensional Gaussian random vector with known covariance matrix

$$\boldsymbol{R} = \begin{bmatrix} \sigma_{y,x}^2 \boldsymbol{I}_{d_x} & \boldsymbol{0} \\ \boldsymbol{0} & \sigma_{y,z}^2 \boldsymbol{I}_{d_z} \end{bmatrix}, \tag{5.61}$$

and $\sigma_{y,x}^2, \sigma_{y,z}^2 > 0$ are known variances.

5.4.2 Numerical results

We have run simulations for the two-scale Lorenz 96 model of Section 5.4.1, with dimensions $d_x = 10$ and $d_z = 50$. The time steps for the Euler-Maruyama integrators are $\Delta_x = 10^{-3}$ and $\Delta_z = 10^{-4}$ continuous-time units. We set the fixed parameters as $F = 8$, $H = 0.75$, $C = 10$ and $B = 15$. In order to obtain the initial states $\boldsymbol{x}_0$ and $\boldsymbol{z}_0$, we simulate a deterministic version of Eqs. (5.58)–(5.59) ($\sigma_x = \sigma_z = 0$) for 20 continuous-time units. We set the initial states as the values of variables $\boldsymbol{x}$ and $\boldsymbol{z}$ at the last time step of this simulation. This initialization is used in all simulations of this computer experiment in order to generate both "ground truth" sequences of $\boldsymbol{x}_t$ and $\boldsymbol{z}_n$ and the associated sequences of observations $\boldsymbol{y}_t$. We set the matrix $\boldsymbol{A}_t = \boldsymbol{I}_{d_y}$, for $d_y = d_x + d_z$.

In the experiments, we compare the performance of both methods proposed (the first one of Section 5.3.1 and the second one of Section 5.3.2). We experiment with different number of samples N and J in the first and second layers of the former methods. Additionally, for the first method (SMC-SMC-UKF) we run the multi-scale hybrid filter with $L = 2d_z + 1 = 101$ sigma-points for the UKF in the third layer. We need to estimate $\boldsymbol{\theta} = [F, C, H, B]^\top$ (hence, $d_\theta = 4$). The prior for the unknown parameters is uniform, namely $p(\boldsymbol{\theta}) = \mathcal{U}([2, 20]^2)$, while the priors used in the filtering algorithm for both unknown state variables are Gaussian, namely $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0, 0.1\boldsymbol{I}_{d_x})$ and $p(\boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_0, 10\boldsymbol{I}_{d_z})$. The noise scaling factors, $\sigma_x = \frac{1}{2}$, $\sigma_z = \frac{1}{16}$, $\sigma_{y,x} = 10^{-1}$ and $\sigma_{y,z} = 10^{-3}$, are known. The jittering kernel is $\kappa_N^{\boldsymbol{\theta}'}(d\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}', \tilde{\sigma}^2 \boldsymbol{I}_{d_\theta})$, where $\tilde{\sigma}^2 = \frac{0.05}{\sqrt{N^3}}$ is selected following [25].

We assess the accuracy of the algorithms in terms of the normalized mean square error (NMSE) of the estimators of the parameters, the slow state variables and the fast state variables. In the plots, we show the NMSEs computed at time t,

$$\text{NMSE}_{\boldsymbol{\theta},t} = \frac{\|\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_t\|^2}{\|\boldsymbol{\theta}_t\|^2}, \tag{5.62}$$

$$\text{NMSE}_{\boldsymbol{x},t} = \frac{\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\|^2}{\|\boldsymbol{x}_t\|^2}, \tag{5.63}$$

$$\text{NMSE}_{\boldsymbol{z},t} = \frac{\|\boldsymbol{z}_{ht} - \hat{\boldsymbol{z}}_{ht}\|^2}{\|\boldsymbol{z}_{ht}\|^2}, \tag{5.64}$$

averaged over 50 independent simulation runs of 20 continuous-time units each, where the estimators take the form

$$\hat{\boldsymbol{\theta}}_t = \sum_{i=1}^{N} v_t^i \boldsymbol{\theta}^i, \tag{5.65}$$

$$\hat{\boldsymbol{x}}_t = \sum_{i=1}^{N}\sum_{j=1}^{J} v_t^i u_t^{i,j} \boldsymbol{x}_t^{i,j} \quad \text{and} \tag{5.66}$$

$$\hat{\boldsymbol{z}}_{ht} = \sum_{i=1}^{N}\sum_{j=1}^{J}\sum_{l=0}^{L} v_t^i u_t^{i,j} \lambda_{ht}^{i,j,l} \boldsymbol{z}_{ht}^{i,j,l}, \tag{5.67}$$

for the first method. For the second method the estimators of the state variables are

$$\hat{\boldsymbol{x}}_t = \frac{1}{J}\sum_{i=1}^{N}\sum_{j=1}^{J} v_t^i \boldsymbol{x}_t^{i,j} \quad \text{and} \tag{5.68}$$

$$\hat{\boldsymbol{z}}_{ht} = \frac{1}{J}\sum_{i=1}^{N}\sum_{j=1}^{J} v_t^i \boldsymbol{z}_{ht}^{i,j}, \tag{5.69}$$

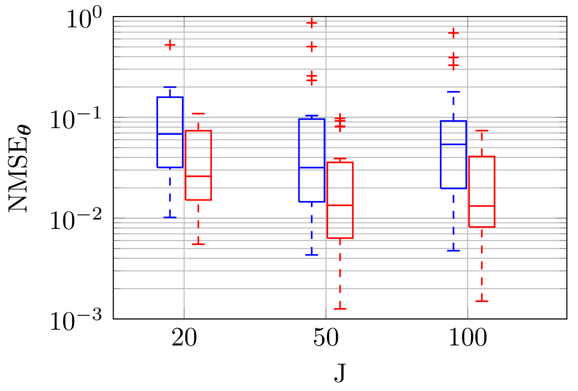
where $\boldsymbol{x}_t^{i,j}$ is the j-th member of the ensemble $\boldsymbol{X}_t^i$ in Algorithm 15.

Figure 5.2 shows the performance of the proposed methods for different values of J (number of samples in the second layer) and $N = 20$. This is evaluated in terms of averaged NMSE$_{\boldsymbol{\theta}}$, NMSE$_{\boldsymbol{x}}$ and NMSE$_{\boldsymbol{z}}$ together with the running time in hours. The first method (SMC-SMC-UKF) shows an improvement in the accuracy as the number of samples J increases, although this improvement is only significant for the slow state (Fig. 5.2b). The second method (SMC-EnKF-EKF) remains stable with J. The second method outperforms the first one in accuracy of the parameter estimation (Fig. 5.2a) as well as the slow state estimation (Fig. 5.2b). However, the first method obtains a better NMSE$_{\boldsymbol{z}}$. Additionally, the second method runs faster since the computational cost is considerably lower.

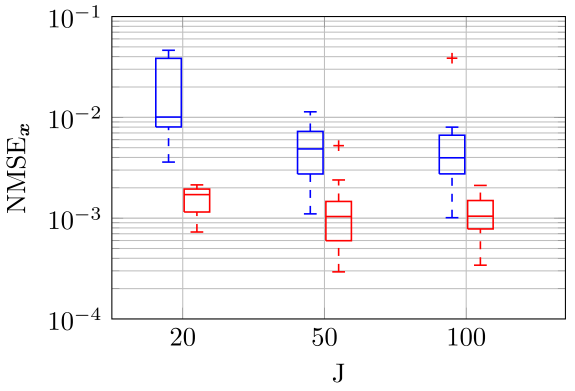
Figure 5.3 compares the performance of the proposed methods and the EnKF for different values of N (number of samples in the first layer) and $J = 50$. This is shown with the averaged NMSE$_{\boldsymbol{\theta}}$, NMSE$_{\boldsymbol{x}}$ and NMSE$_{\boldsymbol{z}}$ together with the running time in hours. Similar to the previous figure, the first method (SMC-SMC-UKF) shows a slight improvement in the accuracy of the slow state estimation as the number of samples J increases (Fig. 5.3b). The second method (SMC-EnKF-EKF) remains stable with N. The second method outperforms the first one in accuracy of the parameter estimation (Fig. 5.3a) and the slow state estimation (Fig. 5.3b), but not for the fast state estimation (Fig. 5.3c). Again, the second method runs faster since the computational cost is considerably lower.

Finally, we show results for a computer experiment in which we have used the SMC-EnKF-EKF method to estimate the parameters F, C, B and H and track the state variables of the two-scale Lorenz 96 system with dimension $d_x = 10$ and $d_z = 50$. The number of particles used to approximate the sequence of parameter posterior distributions is $N = 50$ and the number of samples in the ensembles of the second layer is $J = 50$.

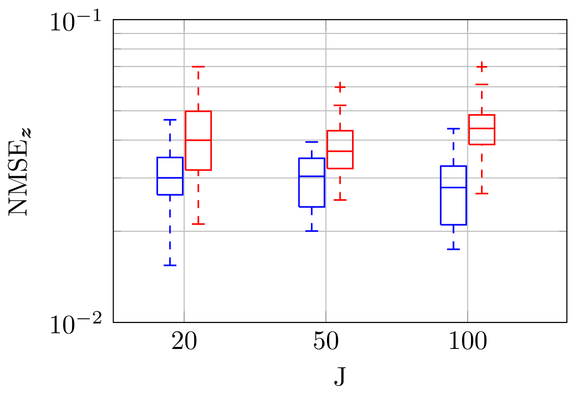
Figure 5.4 shows the true state trajectories, together with their estimates, for the first slow state variable (x_1) and the first fast state variable (z_1) of the two-scale Lorenz 96 model. We note that although the accuracy of the estimation of the fast variable is similar throughout the whole simulation run (over 20 continuous-time units), we only show the last 2 continuous-time units of the simulation.



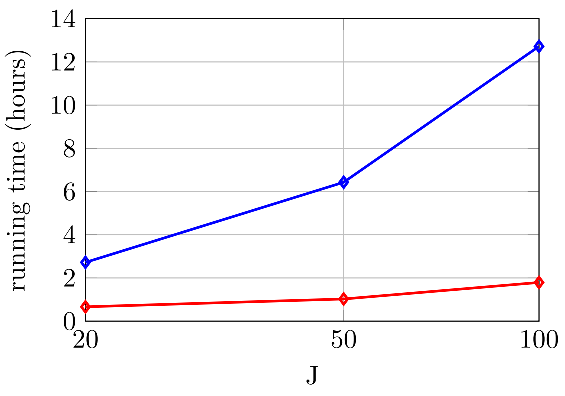
(a) NMSE$_{\boldsymbol{\theta}}$ for different number of particles/ensembles (J) in the second layer of the filter.



(b) NMSE$_{\boldsymbol{x}}$ for different number of particles/ensembles (J) in the second layer of the filter.



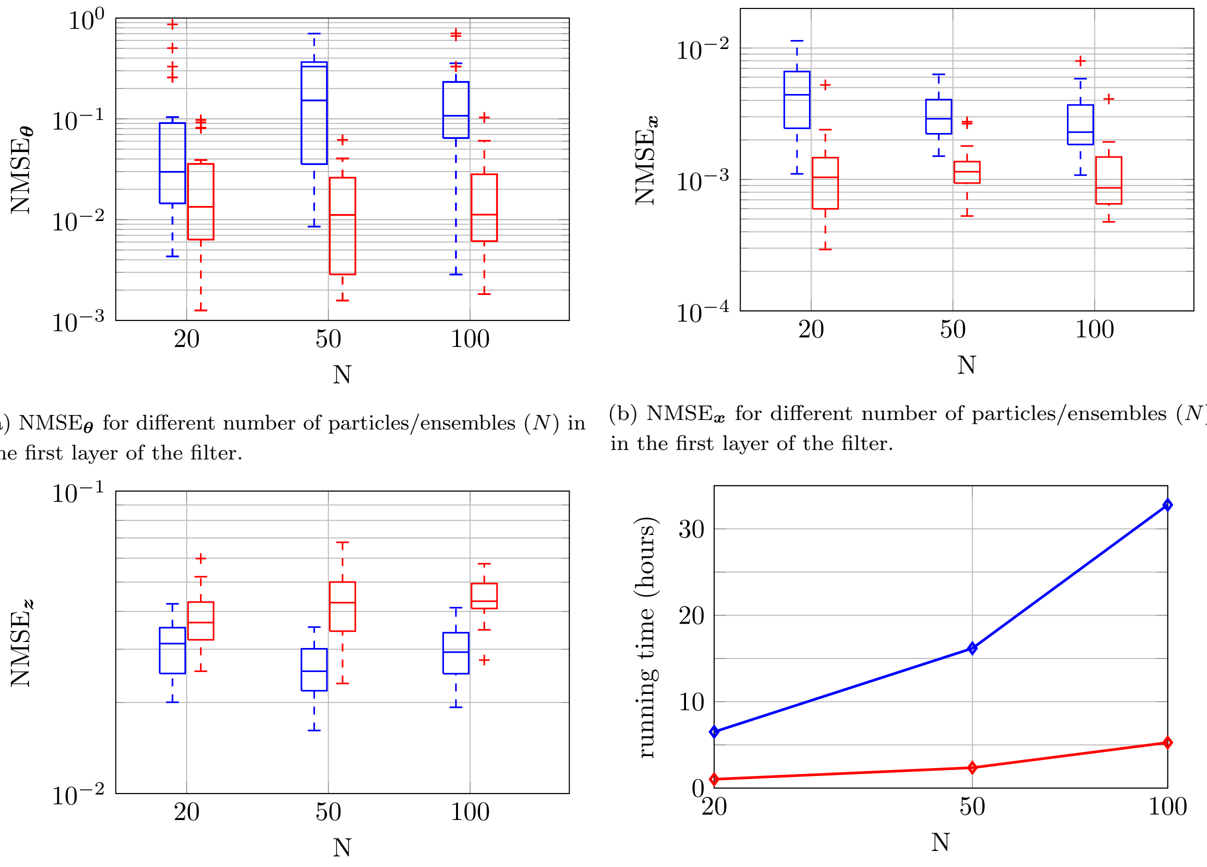
(c) NMSE$_{\boldsymbol{z}}$ for different number of particles/ensembles (J) in the second layer of the filter.



(d) Running time for different number of particles/ensembles (J) in the second layer of the filter.

Figure 5.2: Averaged NMSE$_{\boldsymbol{\theta}}$ (5.2a), NMSE$_{\boldsymbol{x}}$ (5.2b), NMSE$_{\boldsymbol{z}}$ (5.2c) and average running time (5.2d) of SMC-SMC-UKF (in blue) and SMC-EnKF-EKF (in red), averaged over 50 simulation runs. The number of particles of the first layer (SMC) is set to $N = 20$.

In Fig. 5.5 we observe the estimated posterior pdfs of the fixed parameters F, C, B and H, together with the ground truth values. Figure 5.5a displays the approximate posterior pdf of the parameter F (red dashed line) together with the true value $F = 8$ (vertical black line), Fig. 5.5b displays the approximate posterior pdf of the parameter C (blue dashed line) together with the true value $C = 10$ (vertical black line), Fig. 5.5c displays the approximate posterior pdf of the parameter B (green dashed line) together with the true value $B = 15$ (vertical black line) and Fig. 5.5d displays the approximate posterior pdf of the parameter H (magenta dashed line) together with the true value $H = 0.75$ (vertical black line). We observe that for all the pdfs, nearly all probability mass is allocated close to the true values, except for the parameter B (Fig. 5.5c). In this case, the pdf is slightly shifted w.r.t. the true value.



(a) NMSE$_{\boldsymbol{\theta}}$ for different number of particles/ensembles (N) in the first layer of the filter.

(b) NMSE$_{\boldsymbol{x}}$ for different number of particles/ensembles (N) in the first layer of the filter.

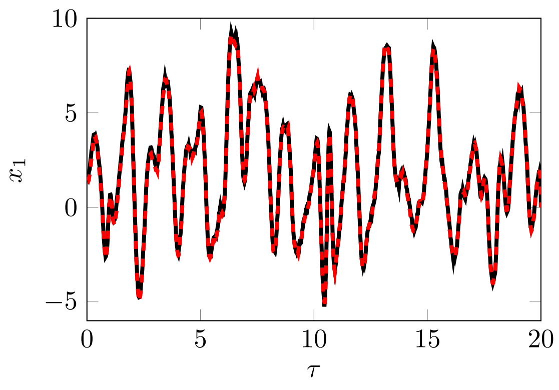
(c) NMSE$_{\boldsymbol{z}}$ for different number of particles/ensembles (N) in the first layer of the filter.

(d) Running time for different number of particles/ensembles (N) in the first layer of the filter.

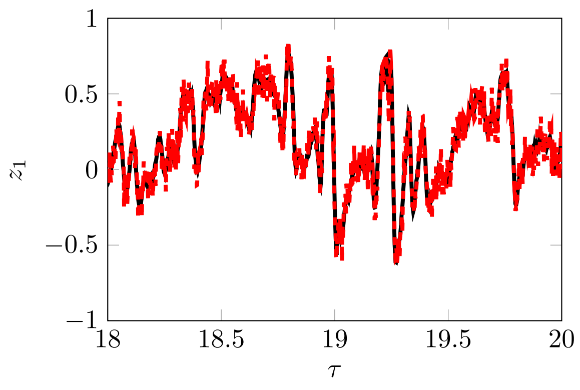
Figure 5.3: Averaged NMSE$_{\boldsymbol{\theta}}$ (5.2a), NMSE$_{\boldsymbol{x}}$ (5.2b), NMSE$_{\boldsymbol{z}}$ (5.2c) and average running time (5.2d) of SMC-SMC-UKF (in blue) and SMC-EnKF-EKF (in red), averaged over 50 simulation runs. The number of particles/ensembles of the second layer (SMC for the first method and EnKF for the second method) is set to $J = 50$.

5.5 Conclusions

We have introduced a further generalization of the NHF methodology of [91] that, using long sequences of observations collected over time, estimates the static parameters and the stochastic dynamical variables of a class of heterogeneous multi-scale state-space models [1]. This scheme combines three layers of filters, one inside the other. It approximates recursively the posterior probability distributions of the parameters and the two sets of state variables given the sequence of available observations. In a first layer of computation we approximate the posterior probability distribution of the parameters, in a second layer we approximate the posterior probability distribution of the slow state variables, and the posterior probability distribution of the fast state variables is approximated in a third layer. The inference techniques used in each layer can vary, leading to different computational costs and degrees of accuracy. To be specific, we describe two possible algorithms that derive from this scheme, combining Monte Carlo methods and Gaussian filters at different layers. The first method involves using sequential Monte Carlo (SMC) methods in both first and second layers, together with a bank of unscented Kalman filter (UKFs) in the third layer (i.e., the

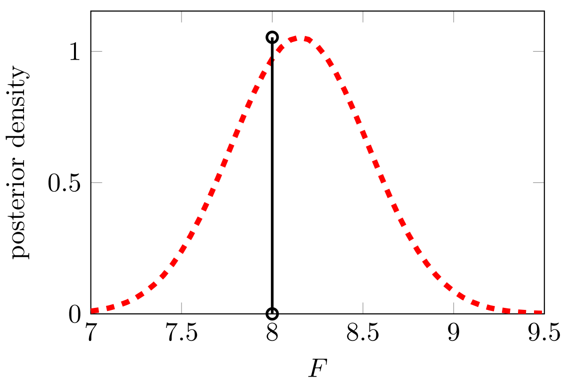


(a) Time sequence of x_1

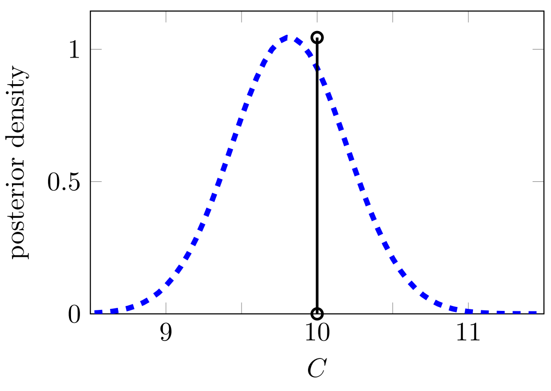


(b) Time sequence of x_2

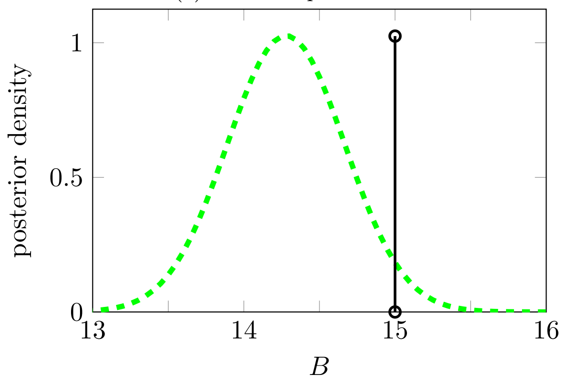
Figure 5.4: Sequences of state values (black line) and estimates (dashed red line) in x_1 (plot 5.4a) and z_1 (plot 5.4b) over time.



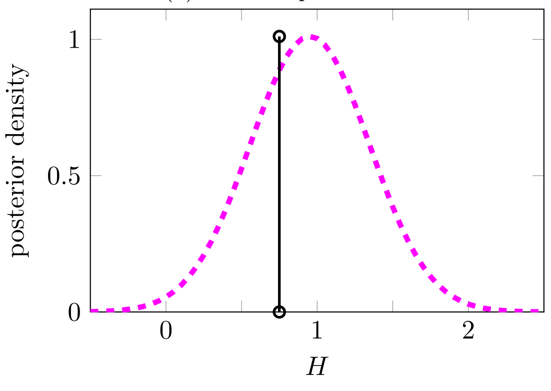
(a) Posterior pdf of F



(b) Posterior pdf of C



(c) Posterior pdf of B



(d) Posterior pdf of H

Figure 5.5: Posterior density of the parameters (dashed lines) at time $\tau = 20$. The true values are indicated by a black vertical line.

SMC-SMC-UKF). The second method employs a SMC in the first layer, ensemble Kalman filters (EnKFs) at the second layer and introduces the use of a bank of extended Kalman filters (EKFs) in the third layer (i.e., the SMC-EnKF-EKF). We have presented numerical results for a two-scale stochastic Lorenz 96 model with synthetic data and we have evaluated the performance of the algorithm in terms of the normalized mean square errors (NMSEs) for the parameters and the dynamic (slow and fast) state variables. The proposed implementations (both of them) obtain good results in terms of accuracy, having a considerably reduction in running time (i.e., the computational cost) with the second method. Further research is still needed, studying the stability of the multi-layer structure when the sequence of observations are rare and/or few. Moreover, we can compare the proposed algorithms with other methods, using the Lorenz 96 system but also other models.

6 Conclusions and Future Research

This chapter is dedicated to the summary of the contributions introduced in this dissertation as well as the discussion of future research avenues. In Section 6.1 we outline the general methodology that has been developed over Chapters 3, 4 and 5, and discuss the various feasible implementations and their relative merits. The basic (continuity) assumptions on which the methodology relies are summed up in Section 6.2 and, in Section 6.3, we comment on the results obtained from the various computer experiments. Finally, some promising paths for future research are discussed in Section 6.4.

6.1 The methodology

The motivation for this thesis has been the investigation of algorithms for parameter estimation and state tracking in state space models. Traditionally, both inferential tasks have been addressed separately, and there are few methods that calculate the *full* posterior probability distribution of all the unknown variables and parameters of the model. In the last years some methods that accomplish this task have been proposed. Among these techniques, the nested particle filter (NPF) stands out because this algorithm is not a batch procedure, but a purely recursive (online) scheme. In other words, every time a new observation arrives, the whole sequence of observations does *not* have to to be re-processed from scratch in order to update the estimates. Instead, the updated estimates are computed from the previous ones and the new observation alone. However, the use of two layers of intertwined sequential Monte Carlo (SMC) algorithms makes its computational cost prohibitive in high-dimensional problems.

This algorithm was the starting point of the research of this thesis, since the first objective was to simplify the NPF in order to create more efficient algorithms. In particular, we have studied different ways of replacing the SMC schemes in the NPF. The first solution we have proposed is the replacement of the SMC modules of the second layer of the algorithm by a bank of Kalman filters. We have chosen Kalman filters because they are one of the simplest known filtering techniques and, in addition, we have modified the second layer of the nested scheme because the implementation in the state tracking layer is less complex. Specifically, in Chapter 3 we have introduced a class of nested hybrid algorithms, that use Monte Carlo in the first layer while they run Kalman filters, i.e., extended Kalman filters (EKFs) and ensemble Kalman filters (EnKFs), in the second layer. We have also introduced a modification in the first layer of the filter, by replacing the Monte Carlo samples by quasi Monte Carlo samples, i.e., we have substituted the SMC by a sequential quasi-Monte Carlo (SQMC) procedure. Surprisingly, these new methods not only reduce the computational cost compared to the NPF but they also obtained better performance in some computer experiments.

The use of Kalman filters in the second layer of the method worked satisfactorily and this led us to the introduction of Kalman-based filters in the first layer of the nested schemes as well. This idea is introduced in Chapter 4, where we use an unscented Kalman filter (UKF) in the first layer while there is a bank of EKFs in the second layer. Unfortunately, the use of non-Monte Carlo methods in the first layer leads to a non-straightforward problem. The key difficulty is to keep the algorithm recursive. The reason for this is that the jittering procedure (used both in the NPF and the methods of Chapter 3) cannot be employed anymore. The jittering step consists in drawing a new set of parameter particles at each discrete-time step even if the parameters are static. This is done with a Markov kernel, which either perturbs (jitters) a few particles with arbitrary variance (while leaving most of them unperturbed) or jitters all particles with a controlled variance that decreases as the number of samples increases. Without this step, the diversity of the values of the parameter particles decreases sharply after a few resampling steps, leading to poor approximations of the parameter posterior probability distributions.

Jittering cannot be extended to Gaussian filters in a practical way. Instead, we have made the update of the filter in the outer layer dependent on a distance defined on the parameter space. When the distance between consecutive parameter estimates falls below a prescribed threshold the algorithm operates in a purely recursive manner. However, the selection of this threshold is not trivial and could vary from one problem to another. A poor choice of the threshold may lead to two possible scenarios. First, when we set a threshold with too small a value, the algorithm operates non-recursively more often than it should. This increases drastically the computational cost of the resulting method. Second, when the threshold is too high, the computational cost of the algorithm decreases in exchange for greater errors in the approximations of the pdfs of the parameters and the state variables. We have carried out a specific study of the choice of the threshold and shown, as a result, that the algorithm of Chapter 4 can be more efficient than the previous nested algorithms.

An alternative way to think of the unknown parameters and the dynamic variables in a state space model is tracking it as two sets of state variables that evolve over different time scales: the unknown parameters are slow variables (so slow that we can work with them as if they were genuinely static) while dynamic variables are rapidly time-varying in comparison. Following this argument, it makes sense to extend the nested filtering methodology to general multi-scale state space models where different subsets of state variables evolve over different time scales. This realization has led to the methods described in Chapter 5, which apply the nested hybrid filtering methodology to a class of heterogeneous multi-scale models by relating each relevant time scale to a layer of computation in the nested structure. In particular, we have described a three-layer nested smoother that approximates, in a recursive manner, the posterior probability distributions of the parameters and two sets of state variables (fast and slow ones) given the sequence of available observations. The computations on the second layer are conditional on the candidate parameter values generated on the first layer, while the calculations on the third layer are conditional on the candidates drawn at the first and second layers. As in the previous methodologies, the inference techniques used in each layer can vary, which leads to different algorithms.

The methods and algorithms introduced in Chapters 3, 4 and 5 as well as the NPF, belong to the same class of methods. Therefore, this thesis describes a generalised methodology that comprises all of them.

6.2 Basic assumptions and analytical results

The key difference between the methods that we have introduce and other Bayesian schemes for inference in state space models , such as the particle Markov chain Monte Carlo (PMCMC) and the sequential Monte Carlo square (SMC^2), is that the nested algorithms presented in this thesis are recursive. Therefore, it is not necessary to reprocess the whole sequence of observations every time a new observation arrives in order to update the estimates, leading to algorithms that are better suited for online implementations.

This can be achieved only when the target posterior distributions (namely the posterior pdf of the state variables conditional on the parameters) are continuous, e.g., as implied by Assumption 3 in Chapter 3 and Assumptions 5 and 6 in Chapter 4. The continuity of these pdfs enables the introduction of the jittering of the parameters (as in Chapter 3) and the recursive implementation of the UKF (as in Chapter 4) in the first layer of the scheme without resetting the second-layer filters. This in turn enables the resulting algorithms to switch from a batch technique (with a quadratic increase of the computational effort over time) to a recursive filter (with linear cost over time).

The continuity of the posterior pdfs conditional on the parameters can be guaranteed when the basic elements of the state space model (i.e., the transition pdf and the likelihood) satisfy themselves certain continuity conditions, as made explicit by the assumptions in the analysis of Chapter 4. Intuitively, if the transition density and the likelihood are “sufficiently” continuous with respect to the parameters, then the posterior pdf of the state variables is continuous as well and this enables us to introduce small perturbations to the parameter samples (i.e., to jitter them) in order to keep their diversity without having to reprocess the whole set of observations as, e.g., the SMC^2 method does. For this reason, in Chapter 3 the Markov kernel of the jittering procedure needs a controlled variance to draw a whole set of new particles (or, instead, just evolve a few particles with an arbitrary variance). In the same vein, the method of Chapter 4 can work recursively when the distance between the new set of parameters and the previous set is below a prescribed threshold.

Under such continuity assumptions, it is possible to obtain theoretical guarantees on the convergence of, at least, the family of nested hybrid filters that employ Monte Carlo (or quasi Monte Carlo) in the first layer of the nested scheme. In particular, we have proved that the approximate posterior distribution of the parameters output by the first layer of the nested filter converges to a certain limit distribution that depends on the algorithm used in the second layer. Note that this is not a guarantee of convergence towards the true posterior distribution of the parameters, but possibly towards a biased probability distribution. The bias of the latter depends on the choice of filters used in the second layer of the algorithm, being the resulting algorithm biased if the filters in the second layer are so (as it is the case in general with approximate Gaussian filters). However, it guarantees that a limit distribution does exist and the convergence rate is the classical Monte Carlo rate of $N^{-\frac{1}{2}}$.

The price to pay for recursivity is a performance loss compared to comparable batch methods. This has been shown in [25] for nested particle filters. We provide numerical evidence of the errors due to the recursive implementation in Chapter 4. In particular, we show the performance of the nested Gaussian filter (NGF) deteriorates quickly when the distance between the new set of parameters and the previous set surpasses a specific threshold.

6.3 Computer experiments

In this dissertation we have presented numerical results for several nonlinear models such as the stochastic two-scale stochastic Lorenz 96 system [11] and the Lorenz 63 system [67, 73]. These models display underlying chaotic dynamics and are commonly used for the assessment of data assimilation methods in geophysics [22, 27, 58]. We have also presented results for a stochastic volatility model with real-world data (namely, euro-to-USD exchange rates between 2014 and 2016) [3, 26, 95] in Chapter 4.

In Chapter 3 we have used the two-scale stochastic Lorenz 96 system in order to assess the performance of four nested hybrid filters that combine Monte Carlo, quasi-Monte Carlo, EKF and EnKF schemes in different ways. Specifically, we have assessed four algorithms: SMC-EnKF, SMC-EKF, SQMC-EnKF and SQMC-EKF. These two-layer schemes estimate the static unknown parameters as well as the slow dynamical state variables, while the contribution of the fast variables is replaced by a polynomial ansatz which has to be fitted as well. Within this framework, we see that the resulting algorithms outperform other methods such as the two-stage filter [92] and the NPF. The use of Gaussian filters in the second layer of the nested hybrid filters not only leads to a significant reduction in computational complexity compared to the NPF, but this is attained without a significant loss of accuracy. The selection of filtering techniques in each layer depends on the computational cost one can afford, obtaining better performance with the most computationally complex methods. Therefore, the proposed framework enables a trade-off between accuracy and computational cost.

In Chapter 4, we describe the use of a UKF [90] in the outer layer of the nested filtering scheme while using EKFs in the inner layer (for simplicity). For this algorithm, we present numerical results for a stochastic Lorenz 63 model using synthetic data, as well as for a stochastic volatility model with real-world data (namely, euro-to-USD exchange rates between 2014 and 2016). We have introduced and assessed the values of a relative threshold that enables the algorithm to work recursively, and we have evaluated the performance of the algorithm in terms of the normalized mean square errors for the parameters and the dynamic state variables. We have also compared these results with other algorithms, such as the EnKF or the UKF, that implement state augmentation (i.e., an extended state that includes both parameters and dynamical variables), an NPF, and also with a nested hybrid filter that incorporates a SMC scheme in the first layer and EKFs in the second layer. The introduction of Gaussian techniques in both the first and second layers of the algorithm entails another improvement in the performance of the nested methodology, since we obtain similar errors as the NPF and the nested hybrid filters (NHFs), but with a further reduction in the computational cost. Also, the accuracy of the nested Gaussian filters is considerably increased compared to Gaussian filters that implement state augmentation.

The two-scale stochastic Lorenz 96 system has also been used for assessment in Chapter 5, although in this case we estimate the static unknown parameters as well as both the slow and fast dynamical state variables. In that chapter, we have studied the performance of two different algorithms with three layers of inference. Specifically, we implement SMC-SMC-UKF and SMC-EnKF-EKF schemes. The computational cost increases considerably compared with any algorithm of two layers, but in exchange we obtain estimates for all the dynamical variables. The proposed implementations perform the inference task effectively but additional research is needed to optimise the algorithms and compare them with alternative techniques.

6.4 Future research

The nested methodology that we have introduced within this thesis can be extended in various ways. To be specific, we can readily identify the following research topics:

- Improvement of the multi-scale filters in Chapter 5.

  In Chapter 5 we have studied the performance of two different algorithms with three layers of inference. Specifically, we have implemented SMC-SMC-UKF and SMC-EnKF-EKF schemes. Both algorithms are evaluated for a two scale Lorenz 96 model with static parameters and a set of slow and fast state dynamic variables. Although we obtain good results in terms of accuracy, they are still preliminary. The methodology and the proposed methods need further analysis, e.g., a more detailed study of the performance of the resulting algorithms compared to other existing methods. Also, it would be useful to assess the use of the methodology in different scenarios (e.g., with very noisy observations, with few states observed or with a a greater number of parameters to estimate) and using different models (i.e., other heterogeneous multi-scale systems).

- Cubature schemes for NGFs.

  In Chapter 4, we have introduced the class of nested Gaussian filters (NGFs). To be specific, we have described the use of a UKF [90] in the outer layer of the nested filtering scheme while using EKFs in the inner layer (for simplicity). However, the methodology not only enables the use of the UKF in the first layer of the filter, but any deterministic sampling technique. In particular, we can incorporate different cubature rules for the first layer of the NGFs and study and compare the performance of the new algorithms.

- Recursive maximum likelihood methods.

  In Chapter 4, we have introduced the NGFs, where we use deterministic sampling techniques in the first layer of the algorithm instead of Monte Carlo methods. The motivation for this approach is the reduction of the computational complexity of the resulting nested filters. Another way to reduce the computational cost of the nested methods may be the introduction of recursive maximum likelihood (RML) methods [8, 9, 29, 53, 94] in the first layer of the algorithm, since considerably less samples would be needed. This may yield a further generalization of the proposed methodology, producing a broader class of nested algorithms.

- Extended convergence analysis.

  In Chapter 3, we obtain theoretical guarantees on the convergence of, at least, the family of nested hybrid filters that employ Monte Carlo (or quasi Monte Carlo) in the first layer of the nested scheme. In particular, we have proved that the approximate posterior distribution of the parameters output by the first layer of the nested filter converges to a certain (possibly biased) limit distribution that depends on the algorithm used in the second layer. Since there are recent developments in the state-of-the-art error analysis of the EnKF, it would be useful to study more precisely how the convergence is affected by the use of a bank of EnKFs in the second layer of the algorithm (i.e., study the bias generated by the EnKF). Also, further analysis would be possible if the EnKF is placed in the first layer of the filter.

- Application to high-dimensional problems.

  The proposed methodology has been numerically assessed in Chapters 3, 4 and 5. In most computer experiments, we have evaluated the numerical performance for the Lorenz 63 model and the two-scale Lorenz 96 model. As this may limit the study of the performance, it is necessary to extend the evaluation of the proposed algorithms to other significant systems. To be specific, we would start by applying the methodology to large-scale models coming from partial differential equations (PDEs). These models not only describe the evolution of the state with time but also with space. This type of models can be found quite often in geophysics, meteorology, biology and chemistry, and a good performance in this framework can lead to solving a wide range of problems.

A
Summary of NHF algorithms

In this Appendix we provide details of the four versions of the nested hybrid filter (NHF) used in the experiments of Section 3.5 (Chapter 3). The differences between the specific methods depend on whether the posterior distribution of the parameters is approximated using either a sequential Monte Carlo (SMC) or a sequential quasi-Monte Carlo (SQMC) scheme and whether the posterior (approximate) pdfs $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i)$, $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i)$ and $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_t^i)$, are computed using either an extended Kalman filter (EKF) or an ensemble Kalman filter (EnKF) method.

In order to implement the EKF scheme in the NHF, the state and the observation functions (and $\boldsymbol{g}$) are assumed nonlinear and differentiable and, therefore, the mean $\boldsymbol{x}_t^i$ and the covariance matrix $\boldsymbol{P}_t^i$ of the approximate pdf $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{x}_t^i, \boldsymbol{P}_t^i)$ can be directly calculated. This can be done by computing their respective Jacobian matrices ($\boldsymbol{J}_{\boldsymbol{f},\boldsymbol{x},\boldsymbol{\theta}}$ and $\boldsymbol{J}_{\boldsymbol{g},\boldsymbol{x},\boldsymbol{\theta}}$) evaluated at the point $\boldsymbol{x}$ in the state space and $\boldsymbol{\theta}$ in the parameter space. In the NHF scheme that includes an EnKF [36] in the second layer, the approximate filter $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_t^i)$ is represented by an ensemble of M Monte Carlo particles or ensemble members $\{\boldsymbol{x}_t^{i,j}\}_{j=1,\dots,M}$, which can be combined to yield an empirical mean $\bar{\boldsymbol{x}}_t^i$ and covariance matrix $\bar{\boldsymbol{P}}_t^i$. Hence, there is no need to assume that the state or observation functions are differentiable.

Each ensemble member can be stored in a $d_x \times M$ matrix $\boldsymbol{X}_t^i = [\boldsymbol{x}_t^{i,1}, \boldsymbol{x}_t^{i,2}, \dots, \boldsymbol{x}_t^{i,M}] = \left[\boldsymbol{x}_t^{i,j}\right]$, for $j = 1, \dots, M$. The i-th mean and the i-th covariance matrix can be computed as

$$\bar{\boldsymbol{x}}_t^i = \frac{1}{M}\boldsymbol{X}_t^i \mathbb{1}_M \quad \text{and} \quad \bar{\boldsymbol{P}}_t^i = \frac{1}{M}\tilde{\boldsymbol{X}}_t^i(\tilde{\boldsymbol{X}}_t^i)^\top,$$

respectively, where $\mathbb{1}_M = [1, \dots, 1]^\top$ is an M-dimensional column vector and $\tilde{\boldsymbol{X}}_t^i = \boldsymbol{X}_t^i - \bar{\boldsymbol{x}}_t^i \mathbb{1}_M^\top$ is an ensemble of deviations from $\bar{\boldsymbol{x}}_t^i$. We hence write $\mathcal{N}(\boldsymbol{x}_t|\boldsymbol{X}_t^i)$ as a shorthand for the Gaussian pdf $\mathcal{N}(\boldsymbol{x}_t|\bar{\boldsymbol{x}}_t^i, \bar{\boldsymbol{P}}_t^i)$.

We assume that the prior pdf of the state is Gaussian with known mean and covariance matrix, namely

$$p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0|\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{P}}_0). \tag{A.1}$$

The noise terms in the state space model are also assumed Gaussian, with zero mean and known covariance matrices, specifically

$$\boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{v}_t|\boldsymbol{0}, \boldsymbol{V}) \quad \text{and} \quad \boldsymbol{r}_t \sim \mathcal{N}(\boldsymbol{r}_t|\boldsymbol{0}, \boldsymbol{R}) \tag{A.2}$$

represent the state and observation noise respectively.

Below, we describe algorithms that rely on either Monte Carlo (MC) or quasi Monte Carlo (QMC) sampling schemes. Our simulations show that QMC can attain better performance, but this is only true if this sampling scheme is used both in the parameter space and the state space. To be specific, the methods that we have assessed are the following:

- SMC-EKF (Algorithm 17) relies on sequential MC sampling for the parameters and uses EKFs for the weight computation. There is no need for sampling in the EKF algorithm.
- SMC-EnKF (Algorithm 18) uses a sequential MC sampling procedure in the parameter space and a bank of EnKFs for the computation of the weights. Sampling inside each EnKF is standard MC.
- SQMC-EKF (Algorithm 19) performs sequential QMC sampling in the parameter space and computes weights using EKFs.
- SQMC-EnKF (Algorithm 20) uses sequential QMC sampling in the parameter space an QMC inside the EnKFs.

The notation used in this Appendix is summarized in Tables 3.1 and A.1.

Table A.1: Notation of Appendix A.

$\boldsymbol{C}_t^i(\boldsymbol{y})$	Observation covariance matrix given $\bar{\boldsymbol{\theta}}_t^i$
$\boldsymbol{C}_t^i(\boldsymbol{x}, \boldsymbol{y})$	Cross-covariance matrix of $\boldsymbol{x}$ and $\boldsymbol{y}$ given $\bar{\boldsymbol{\theta}}_t^i$
$\boldsymbol{G}$	Function of the EKF that computes the predictive state covariance matrix in the time scale of n
$\boldsymbol{G}^m$	Function $\boldsymbol{G}$ applied m consecutive times
$\boldsymbol{J}_{\boldsymbol{F},\boldsymbol{x}',\boldsymbol{\theta}'}$	Jacobian matrix of the state function $\boldsymbol{g}$ evaluated at the point $\boldsymbol{x}'$ in the state space and $\boldsymbol{\theta}'$ in the parameter space
$\boldsymbol{J}_{\boldsymbol{g},\boldsymbol{x}',\boldsymbol{\theta}'}$	Jacobian matrix of the state function $\boldsymbol{F}$ evaluated at the point $\boldsymbol{x}'$ in the state space and $\boldsymbol{\theta}'$ in the parameter space
$\boldsymbol{K}_t^i$	Kalman gain at time step t conditional on $\bar{\boldsymbol{\theta}}_t^i$
$\boldsymbol{R}$	Observation noise covariance matrix
$\boldsymbol{x}_t^i$	Posterior mean estimate of $\boldsymbol{x}_t$ given $\boldsymbol{\theta}_t^i$ (in NHF with EKFs)
$\bar{\boldsymbol{x}}_t^i$	Posterior mean estimate of $\boldsymbol{x}_t$ given $\boldsymbol{\theta}_t^i$ (in NHF with EnKFs)
$\hat{\boldsymbol{x}}_t^i$ and $\check{\boldsymbol{x}}_t^i$	Predictive and updated mean estimates of $\boldsymbol{x}_t$ given $\bar{\boldsymbol{\theta}}_t^i$
$\boldsymbol{X}_t^i$	Ensemble with the posterior M estimates of $\boldsymbol{x}_t$ given $\boldsymbol{\theta}_t^i$
$\hat{\boldsymbol{X}}_t^i$	Ensemble with the predictive M estimates of $\boldsymbol{x}_t$ given $\bar{\boldsymbol{\theta}}_t^i$
$\check{\boldsymbol{X}}_t^i$	Ensemble with the updated M estimates of $\boldsymbol{x}_t$ given $\bar{\boldsymbol{\theta}}_t^i$
$\boldsymbol{V}$	State noise covariance matrix

Algorithm 17 *SMC-EKF*

1. *Initialization: draw N i.i.d. particles $\boldsymbol{\theta}_0^i \sim \mu_0(d\boldsymbol{\theta})$ and $\boldsymbol{x}_0^i \sim p(\boldsymbol{x}_0)$, for $i = 1, \ldots, N$. Let $\boldsymbol{P}_0^i = \boldsymbol{P}_0$.*

2. *Recursive step: at time $t-1$, we have obtained $\mu_{t-1}^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_{t-1}^i}(d\boldsymbol{\theta})$ and, for each $i = 1, \ldots, N$, $\hat{p}(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_{t-1}^i, \boldsymbol{P}_{t-1}^i)$.*

   (a) *Prediction step:*

      i. *Draw $\bar{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i)$, for $i = 1, \ldots, N$.*

      ii. *For each $i = 1, \ldots, N$ compute*

$$\hat{\boldsymbol{x}}_t^i = \boldsymbol{x}_{t-1}^i + \boldsymbol{F}^m(\boldsymbol{x}_{t-1}^i, \bar{\boldsymbol{\theta}}_t^i, \Delta) \tag{A.3}$$

$$\hat{\boldsymbol{P}}_t^i = \boldsymbol{G}^m(\boldsymbol{P}_{t-1}^i, \sigma_{\boldsymbol{x}}, \boldsymbol{V}) \tag{A.4}$$

      where $\boldsymbol{V}$ is the covariance matrix of the state and $\boldsymbol{G}^m$ denotes the composition of function $\boldsymbol{G}$ m times ($\boldsymbol{G} \circ \ldots \circ \boldsymbol{G}$). Function $\boldsymbol{G}$ in turn, is described as

$$\boldsymbol{G}(\boldsymbol{P}_{n-1}^i, \sigma_{\boldsymbol{x}}, \boldsymbol{V}) = \boldsymbol{J}_{\boldsymbol{F}, \hat{\boldsymbol{x}}_{n-1}^i, \bar{\boldsymbol{\theta}}_t^i} \boldsymbol{P}_{n-1}^i \boldsymbol{J}_{\boldsymbol{F}, \hat{\boldsymbol{x}}_{n-1}^i, \bar{\boldsymbol{\theta}}_t^i}^\top + \sigma_{\boldsymbol{x}}^2 \boldsymbol{V}. \tag{A.5}$$

      iii. *Set $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_t^i, \hat{\boldsymbol{P}}_t^i)$.*

   (b) *Update step:*

      i. *For $i = 1, \ldots, N$, compute*

$$\boldsymbol{C}_t^i(\boldsymbol{y}) = \boldsymbol{J}_{\boldsymbol{g}, \hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i} \hat{\boldsymbol{P}}_t^i \boldsymbol{J}_{\boldsymbol{g}, \hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i}^\top + \boldsymbol{R} \tag{A.6}$$

$$\boldsymbol{K}_t^i = \hat{\boldsymbol{P}}_t^i \boldsymbol{J}_{\boldsymbol{g}, \hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i}^\top \left(\boldsymbol{C}_t^i(\boldsymbol{y})\right)^{-1} \tag{A.7}$$

$$\check{\boldsymbol{x}}_t^i = \hat{\boldsymbol{x}}_t^i + \boldsymbol{K}_t^i\left(\boldsymbol{y}_t - \boldsymbol{g}(\hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i)\right) \tag{A.8}$$

$$\check{\boldsymbol{P}}_t^i = (\boldsymbol{I}_{d_x} - \boldsymbol{K}_t^i \boldsymbol{J}_{\boldsymbol{g}, \hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i}) \hat{\boldsymbol{P}}_t^i \tag{A.9}$$

      where $\boldsymbol{R} = \sigma_o^2 \boldsymbol{I}_{d_y}$ is the measurement noise covariance.

      ii. *Compute $\hat{u}(\bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}\left(\boldsymbol{y}_t|\boldsymbol{g}(\hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i), \boldsymbol{C}_t^i(\boldsymbol{y})\right)$ and obtain the normalized weights,*

$$w_t^i = \frac{\hat{u}(\bar{\boldsymbol{\theta}}_t^i)}{\sum_{j=1}^N \hat{u}_t(\bar{\boldsymbol{\theta}}_t^j)}, \quad i = 1, \ldots, N. \tag{A.10}$$

      iii. *Set the filter approximation $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\check{\boldsymbol{x}}_t^i, \check{\boldsymbol{P}}_t^i)$.*

   (c) *Resampling: draw indices $j_1, \ldots, j_N$ from the multinomial distribution with probabilities $w_t^1, \ldots, w_t^N$, then set*

$$\boldsymbol{\theta}_t^i = \bar{\boldsymbol{\theta}}_t^{j_i}, \quad \boldsymbol{x}_t^i = \check{\boldsymbol{x}}_t^{j_i} \quad \textit{and} \quad \boldsymbol{P}_t^i = \check{\boldsymbol{P}}_t^{j_i} \tag{A.11}$$

   for $i = 1, \ldots, N$. Hence $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{x}_t^i, \boldsymbol{P}_t^i)$ and $\mu_t^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_t^i}(d\boldsymbol{\theta})$.

Algorithm 18 *SMC-EnKF*

```
1. Initialization: draw $N$ i.i.d. particles $\boldsymbol{\theta}_0^i \sim \mu_0(d\boldsymbol{\theta})$ and $\{\boldsymbol{x}_0^{i,j}\} \sim p(\boldsymbol{x}_0)$, for $i = 1, \dots, N$, $j = 1, \dots, M$.
   Let $\boldsymbol{X}_0^i = \left[\boldsymbol{x}_0^{i,j}\right]$, for $i = 1, \dots, N$ and $j = 1, \dots, M$.
2. Recursive step: at time $t-1$, we have obtained $\mu_{t-1}^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_{t-1}^i}(d\boldsymbol{\theta})$ and, for each $i = 1, \dots, N$, $\hat{p}(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boldsymbol{X}_{t-1}^i)$.
   (a) Prediction step:
      i. Draw $\bar{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i)$, $i = 1, \dots, N$.
      ii. For each $i = 1, \dots, N$ compute
          $$\hat{\boldsymbol{X}}_t^i = \boldsymbol{X}_{t-1}^i + \boldsymbol{F}^m(\boldsymbol{X}_{t-1}^i, \bar{\boldsymbol{\theta}}_t^i, \Delta, \sigma_{\boldsymbol{x}} \boldsymbol{V}_t^i) \quad (A.12)$$
          where $\boldsymbol{V}_t^i = [\boldsymbol{v}_t^{i,1}, \dots, \boldsymbol{v}_t^{i,M}]$, $i = 1, \dots, N$, is a $mqd_x \times M$ matrix of Gaussian perturbations ($q$ denotes the order of the underlying RK integrator). This is done by generating $mqd_x \times M$ QMC random variates[1] and then applying the inversion method [69] to generate the Gaussian variates (one per uniform sample).
      iii. Set $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{X}}_t^i)$.
   (b) Update step:
      i. For $i = 1, \dots, N$, compute
          $$\boldsymbol{C}_t^i(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{M-1}\tilde{\boldsymbol{X}}_t^i(\tilde{\boldsymbol{Y}}_t^i)^\top \quad (A.13)$$
          $$\boldsymbol{C}_t^i(\boldsymbol{y}) = \frac{1}{M-1}\tilde{\boldsymbol{Y}}_t^i(\tilde{\boldsymbol{Y}}_t^i)^\top + \boldsymbol{R} \quad (A.14)$$
          $$\boldsymbol{K}_t^i = \boldsymbol{C}_t^i(\boldsymbol{x}, \boldsymbol{y})\big(\boldsymbol{C}_t^i(\boldsymbol{y})\big)^{-1} \quad (A.15)$$
          $$\check{\boldsymbol{X}}_t^i = \hat{\boldsymbol{X}}_t^i + \boldsymbol{K}_t^i\big(\boldsymbol{y}_t\mathbb{1}_M^\top + \boldsymbol{T}_t^i - \boldsymbol{Y}_t^i\big) \quad (A.16)$$
          where $\boldsymbol{R} = \sigma_o^2 \boldsymbol{I}_{d_y}$ is the measurement noise covariance, $\boldsymbol{T}_t^i = \left[\boldsymbol{r}_t^1, \dots, \boldsymbol{r}_t^M\right]$ with $\boldsymbol{r}_t^j \sim \mathcal{N}(\boldsymbol{r}_t|\boldsymbol{0}, \boldsymbol{R})$ is a matrix of Gaussian perturbations, $\boldsymbol{Y}_t^i = \boldsymbol{g}(\hat{\boldsymbol{X}}_t^i, \bar{\boldsymbol{\theta}}_t^i)$, and $\tilde{\boldsymbol{X}}_t^i$ and $\tilde{\boldsymbol{Y}}_t^i$ are calculated as
          $$\tilde{\boldsymbol{X}}_t^i = \hat{\boldsymbol{X}}_t^i - \bar{\boldsymbol{x}}_t^i\mathbb{1}_M^\top \quad (A.17)$$
          $$\tilde{\boldsymbol{Y}}_t^i = \boldsymbol{Y}_t^i - \bar{\boldsymbol{y}}_t^i\mathbb{1}_M^\top, \quad (A.18)$$
          where $\bar{\boldsymbol{x}}_t^i = \frac{1}{M}\hat{\boldsymbol{X}}_t^i\mathbb{1}_M$ and $\bar{\boldsymbol{y}}_t = \frac{1}{M}\boldsymbol{Y}_t^i\mathbb{1}_M$.
      ii. Compute $\hat{u}(\bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}\big(\boldsymbol{y}_t|\bar{\boldsymbol{y}}_t^i, \boldsymbol{C}_t^i(\boldsymbol{y})\big)$ and obtain the normalized weights,
          $$w_t^i = \frac{\hat{u}(\bar{\boldsymbol{\theta}}_t^i)}{\sum_{j=1}^N \hat{u}_t(\bar{\boldsymbol{\theta}}_t^j)}, \quad i = 1, \dots, N. \quad (A.19)$$
      iii. Set the filter approximation $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\check{\boldsymbol{X}}_t^i)$.
```

[1]Using functions haltonset, scramble and net in Matlab you can easily obtain a Halton quasi-random point set.

(c) *Resampling: draw indices* $j_1, \ldots, j_N$ *from the multinomial distribution with probabilities* $w_t^1, \ldots, w_t^N$*, then set*

$$\boldsymbol{\theta}_t^i = \bar{\boldsymbol{\theta}}_t^{j_i}, \quad \text{and} \quad \boldsymbol{X}_t^i = \check{\boldsymbol{X}}_t^{j_i} \tag{A.20}$$

for $i = 1, \ldots, N$*. Hence*

$$\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{X}_t^i) \quad \text{and} \quad \mu_t^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_t^i}(d\boldsymbol{\theta}).$$

Algorithm 19 *SQMC-EKF*

1. *Initialization: generate QMC uniform samples* $\{\boldsymbol{\rho}_{-1}^i, \boldsymbol{\rho}_0^i\}$ *in* $[0,1)^{d_\theta}$ *and* $\bar{\boldsymbol{\rho}}_{-1}^i$ *in* $[0,1)^{d_x}$*. Draw* $\boldsymbol{\theta}_0^i \sim \mu_0(d\boldsymbol{\theta}|\boldsymbol{\rho}_{-1}^i)$ *and* $\boldsymbol{x}_0^i \sim p(\boldsymbol{x}_0|\bar{\boldsymbol{\rho}}_{-1}^i)$*, for* $i = 1, \ldots, N$*. Let* $\boldsymbol{P}_0^i = \boldsymbol{P}_0$*.*

2. *Recursive step,* $t \geq 1$*. At time* $t-1$*, we have obtained* $\mu_{t-1}^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_{t-1}^i}(d\boldsymbol{\theta})$ *and, for each* $i = 1, \ldots, N$*,* $\hat{p}(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_{t-1}^i, \boldsymbol{P}_{t-1}^i)$*.*

   (a) *Prediction step:*

   i. *If* $t = 1$*, draw* $\bar{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i, \boldsymbol{\rho}_0^i)$*, else draw* $\bar{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i, \tilde{\boldsymbol{\rho}}_{t-1}^{\mathsf{c}(i)})$*,* $i = 1, \ldots, N$*, for* $t \geq 2$*.*

   ii. *For each* $i = 1, \ldots, N$ *compute*

$$\begin{aligned} \hat{\boldsymbol{x}}_t^i &= \boldsymbol{x}_{t-1}^i + \boldsymbol{F}^m(\boldsymbol{x}_{t-1}^i, \bar{\boldsymbol{\theta}}_t^i, \Delta), & \text{(A.21)} \\ \hat{\boldsymbol{P}}_t^i &= \boldsymbol{G}^m(\boldsymbol{P}_{t-1}^i, \sigma_{\boldsymbol{x}}, \boldsymbol{V}), & \text{(A.22)} \end{aligned}$$

   where $\boldsymbol{V}$ *is the covariance matrix of the state and* $\boldsymbol{G}^m$ *denotes the composition of function* $\boldsymbol{G}$ *m times (*$\boldsymbol{G} \circ \ldots \circ \boldsymbol{G}$*). Function* $\boldsymbol{G}$ *in turn, is described as*

$$\boldsymbol{G}(\boldsymbol{P}_{n-1}^i, \sigma_{\boldsymbol{x}}, \boldsymbol{V}) = \boldsymbol{J}_{\boldsymbol{F}, \hat{\boldsymbol{x}}_{n-1}^i, \bar{\boldsymbol{\theta}}_t^i} \boldsymbol{P}_{n-1}^i \boldsymbol{J}_{\boldsymbol{F}, \hat{\boldsymbol{x}}_{n-1}^i, \bar{\boldsymbol{\theta}}_t^i}^\top + \sigma_{\boldsymbol{x}}^2 \boldsymbol{V}. \tag{A.23}$$

   iii. *Set* $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{x}}_t^i, \hat{\boldsymbol{P}}_t^i)$*.*

   (b) *Update step:*

   i. *For* $i = 1, \ldots, N$*, compute*

$$\begin{aligned} \boldsymbol{C}_t^i(\boldsymbol{y}) &= \boldsymbol{J}_{\boldsymbol{g}, \hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i} \hat{\boldsymbol{P}}_t^i \boldsymbol{J}_{\boldsymbol{g}, \hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i}^\top + \boldsymbol{R}, & \text{(A.24)} \\ \boldsymbol{K}_t^i &= \hat{\boldsymbol{P}}_t^i \boldsymbol{J}_{\boldsymbol{g}, \hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i}^\top \big(\boldsymbol{C}_t^i(\boldsymbol{y})\big)^{-1}, & \text{(A.25)} \\ \check{\boldsymbol{x}}_t^i &= \hat{\boldsymbol{x}}_t^i + \boldsymbol{K}_t^i\big(\boldsymbol{y}_t - \boldsymbol{g}(\hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i)\big) & \text{(A.26)} \\ \check{\boldsymbol{P}}_t^i &= (\boldsymbol{I}_{d_x} - \boldsymbol{K}_t^i \boldsymbol{J}_{\boldsymbol{g}, \hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i}) \hat{\boldsymbol{P}}_t^i, & \text{(A.27)} \end{aligned}$$

   where $\boldsymbol{R} = \sigma_o^2 \boldsymbol{I}_{d_y}$ *is the measurement noise covariance.*

ii. *Compute* $\hat{u}(\bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}\big(\boldsymbol{y}_t|\boldsymbol{g}(\hat{\boldsymbol{x}}_t^i, \bar{\boldsymbol{\theta}}_t^i), \boldsymbol{C}_t^i(\boldsymbol{y})\big)$ *and obtain the normalized weights,*

$$w_t^i = \frac{\hat{u}(\bar{\boldsymbol{\theta}}_t^i)}{\sum_{j=1}^N \hat{u}_t(\bar{\boldsymbol{\theta}}_t^j)}, \quad i = 1, \ldots, N. \tag{A.28}$$

iii. *Set the filter approximation* $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\check{\boldsymbol{x}}_t^i, \check{\boldsymbol{P}}_t^i)$.

(c) *Generate a QMC point set* $\{\boldsymbol{\rho}_t^i\}_{i=1}^N$ *in* $[0,1)^{d_\theta+1}$; *let* $\boldsymbol{\rho}_t^i = (\rho_t^i, \tilde{\boldsymbol{\rho}}_t^i) \in [0,1) \times [0,1)^{d_\theta}$.

(d) *Hilbert sort: find a permutation* b *such that*

$$\begin{aligned} (\mathsf{h} \circ \psi)(\bar{\boldsymbol{\theta}}_t^{\mathsf{b}(1)}) \leq \ldots \leq (\mathsf{h} \circ \psi)(\bar{\boldsymbol{\theta}}_t^{\mathsf{b}(N)}), &\quad \textit{if } d_\theta \geq 2 \\ \bar{\boldsymbol{\theta}}_t^{\mathsf{b}(1)} \leq \ldots \leq \bar{\boldsymbol{\theta}}_t^{\mathsf{b}(N)}, &\quad \textit{if } d_\theta = 1. \end{aligned}$$

(e) *Resampling: find a permutation* c *such that* $\rho_t^{\mathsf{c}(1)} \leq \ldots \leq \rho_t^{\mathsf{c}(N)}$. *For* $i = 1, \ldots, N$ *set* $\boldsymbol{\theta}_t^i = \bar{\boldsymbol{\theta}}_t^j$, $\boldsymbol{x}_t^i = \check{\boldsymbol{x}}_t^j$ *and* $\boldsymbol{P}_t^i = \check{\boldsymbol{P}}_t^j$ *if, and only if,*

$$\sum_{k=1}^{j-1} w_t^{\mathsf{b}(k)} < \rho_t^{\mathsf{c}(i)} \leq \sum_{k=1}^{j} w_t^{\mathsf{b}(k)}, \quad j \in \{1, \ldots, N\}.$$

Hence $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{x}_t^i, \boldsymbol{P}_t^i)$ *and* $\mu_t^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_t^i}(d\boldsymbol{\theta})$.

Algorithm 20 *SQMC-EnKF*

1. *Initialization: generate QMC uniform samples* $\{\boldsymbol{\rho}_{-1}^i, \boldsymbol{\rho}_0^i\}$ *in* $[0,1)^{d_\theta}$ *and* $\bar{\boldsymbol{\rho}}_{-1}^i$ *in* $[0,1)^{d_x}$. *Draw* $\boldsymbol{\theta}_0^i \sim \mu_0(d\boldsymbol{\theta}|\boldsymbol{\rho}_{-1}^i)$ *and* $\{\boldsymbol{x}_0^{i,j}\} \sim p(\boldsymbol{x}_0|\bar{\boldsymbol{\rho}}_{-1}^i)$, *for* $i = 1, \ldots, N$, $j = 1, \ldots, M$. *Let* $\boldsymbol{X}_0^i = \left[\boldsymbol{x}_0^{i,j}\right]$, *for* $i = 1, \ldots, N$, $j = 1, \ldots, M$.

2. *Recursive step,* $t \geq 1$. *At time* $t-1$, *we have obtained* $\mu_{t-1}^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_{t-1}^i}(d\boldsymbol{\theta})$ *and, for each* $i = 1, \ldots, N$, $\hat{p}(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}_{t-1}^i) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boldsymbol{X}_{t-1}^i)$.

   (a) *Prediction step:*

   i. *If* $t = 1$, *draw* $\bar{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i, \boldsymbol{\rho}_0^i)$, *else draw* $\bar{\boldsymbol{\theta}}_t^i \sim \kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i, \tilde{\boldsymbol{\rho}}_{t-1}^{\mathsf{c}(i)})$, $i = 1, \ldots, N$, *for* $t \geq 2$.

   ii. *For each* $i = 1, \ldots, N$ *compute*

$$\hat{\boldsymbol{X}}_t^i = \boldsymbol{X}_{t-1}^i + \boldsymbol{F}^m(\boldsymbol{X}_{t-1}^i, \bar{\boldsymbol{\theta}}_t^i, \Delta, \sigma_{\boldsymbol{x}}\boldsymbol{V}_t^i) \tag{A.29}$$

   where $\boldsymbol{V}_t^i = [\boldsymbol{v}_t^{i,1}, \ldots, \boldsymbol{v}_t^{i,M}]$, $i = 1, \ldots, N$, *is a* $mqd_x \times M$ *matrix of Gaussian perturbations (q denotes the order of the underlying RK integrator). This is done by generating* $mqd_x \times M$ *QMC random variates*[2] *and then applying the inversion method [69] to generate the Gaussian variates (one per uniform sample).*

[2]Using functions haltonset, scramble and net in Matlab you can easily obtain a Halton quasi-random point set.

iii. Set $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}, \bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\hat{\boldsymbol{X}}_t^i)$.

(b) Update step:

i. For $i = 1, \dots, N$, *compute*

$$\begin{aligned} \boldsymbol{C}_t^i(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{M-1}\tilde{\boldsymbol{X}}_t^i(\tilde{\boldsymbol{Y}}_t^i)^\top && \text{(A.30)} \\ \boldsymbol{C}_t^i(\boldsymbol{y}) &= \frac{1}{M-1}\tilde{\boldsymbol{Y}}_t^i(\tilde{\boldsymbol{Y}}_t^i)^\top + \boldsymbol{R} && \text{(A.31)} \\ \boldsymbol{K}_t^i &= \boldsymbol{C}_t^i(\boldsymbol{x}, \boldsymbol{y})\big(\boldsymbol{C}_t^i(\boldsymbol{y})\big)^{-1} && \text{(A.32)} \\ \check{\boldsymbol{X}}_t^i &= \hat{\boldsymbol{X}}_t^i + \boldsymbol{K}_t^i\big(\boldsymbol{y}_t\mathbb{1}_M^\top + \boldsymbol{T}_t^i - \boldsymbol{Y}_t^i\big) && \text{(A.33)} \end{aligned}$$

where $\boldsymbol{R} = \sigma_o^2\boldsymbol{I}_{d_y}$ *is the measurement noise covariance,* $\boldsymbol{T}_t^i = \big[\boldsymbol{r}_t^1, \dots, \boldsymbol{r}_t^M\big]$ *with* $\boldsymbol{r}_t^j \sim \mathcal{N}(\boldsymbol{r}_t|\boldsymbol{0}, \boldsymbol{R})$ *is a matrix of Gaussian perturbations,* $\boldsymbol{Y}_t^i = \boldsymbol{g}(\hat{\boldsymbol{X}}_t^i, \bar{\boldsymbol{\theta}}_t^i)$, *and* $\tilde{\boldsymbol{X}}_t^i$ *and* $\tilde{\boldsymbol{Y}}_t^i$ *are calculated as*

$$\begin{aligned} \tilde{\boldsymbol{X}}_t^i &= \hat{\boldsymbol{X}}_t^i - \bar{\boldsymbol{x}}_t^i\mathbb{1}_M^\top && \text{(A.34)} \\ \tilde{\boldsymbol{Y}}_t^i &= \boldsymbol{Y}_t^i - \bar{\boldsymbol{y}}_t^i\mathbb{1}_M^\top, && \text{(A.35)} \end{aligned}$$

where $\bar{\boldsymbol{x}}_t^i = \frac{1}{M}\hat{\boldsymbol{X}}_t^i\mathbb{1}_M$ *and* $\bar{\boldsymbol{y}}_t = \frac{1}{M}\boldsymbol{Y}_t^i\mathbb{1}_M$.

ii. Compute $\hat{u}(\bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}\big(\boldsymbol{y}_t|\bar{\boldsymbol{y}}_t^i, \boldsymbol{C}_t^i(\boldsymbol{y})\big)$ *and obtain the normalized weights,*

$$w_t^i = \frac{\hat{u}(\bar{\boldsymbol{\theta}}_t^i)}{\sum_{j=1}^N \hat{u}_t(\bar{\boldsymbol{\theta}}_t^j)}, \quad i = 1, \dots, N. \tag{A.36}$$

iii. Set the filter approximation $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \bar{\boldsymbol{\theta}}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\check{\boldsymbol{X}}_t^i)$.

(c) Generate a QMC point set $\{\boldsymbol{\rho}_t^i\}_{i=1}^N$ *in* $[0,1)^{d_\theta+1}$; *let* $\boldsymbol{\rho}_t^i = (\rho_t^i, \tilde{\boldsymbol{\rho}}_t^i) \in [0,1) \times [0,1)^{d_\theta}$.

(d) Hilbert sort: find a permutation b *such that*

$$\begin{aligned} (\mathsf{h} \circ \psi)(\bar{\boldsymbol{\theta}}_t^{\mathsf{b}(1)}) \leq \dots \leq (\mathsf{h} \circ \psi)(\bar{\boldsymbol{\theta}}_t^{\mathsf{b}(N)}), &\quad \textit{if } d_\theta \geq 2 \\ \bar{\boldsymbol{\theta}}_t^{\mathsf{b}(1)} \leq \dots \leq \bar{\boldsymbol{\theta}}_t^{\mathsf{b}(N)}, &\quad \textit{if } d_\theta = 1. \end{aligned}$$

(e) Resampling: find a permutation c *such that* $\rho_t^{\mathsf{c}(1)} \leq \dots \leq \rho_t^{\mathsf{c}(N)}$. *For* $i = 1, \dots, N$ *set* $\boldsymbol{\theta}_t^i = \bar{\boldsymbol{\theta}}_t^j$ *and* $\boldsymbol{X}_t^i = \check{\boldsymbol{X}}_t^j$ *if, and only if,*

$$\sum_{k=1}^{j-1} w_t^{\mathsf{b}(k)} < \rho_t^{\mathsf{c}(i)} \leq \sum_{k=1}^{j} w_t^{\mathsf{b}(k)}, \quad j \in \{1, \dots, N\}.$$

Hence $\hat{p}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}_t^i) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{X}_t^i)$ *and* $\mu_t^N(d\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N \delta_{\boldsymbol{\theta}_t^i}(d\boldsymbol{\theta})$.

The computational complexity of sampling in the parameter space increases linearly with N using either the SMC or the SQMC methods. However, the complexity of SQMC increases also with $N \log N$ due to the resampling step since it requires the computation of the Hilbert inverse as well as two permutations in steps 2d and 2e, in Algorithms 19 and 20. The computational cost of the EKF method increases with rates d_y^3 and d_x^2. The former is due to the inversion of the d_y-dimensional matrix $\boldsymbol{C}_t^i(\boldsymbol{y})$ in Eq. (A.7) and Eq. (A.25) of Algorithms 17 and 19 respectively, while the product in Eq. (A.9) and Eq. (A.27), used to obtain the predictive state covariance matrix $\breve{\boldsymbol{P}}_t^i$, justifies the latter. Using the EnKF method, the complexity is also cubic with d_y because of the inversion of $\boldsymbol{C}_t^i(\boldsymbol{y})$ in Eq. (A.15) and Eq. (A.32) of Algorithms 18 and 20, and it increases linearly with M and d_x because Algorithms 18 and 20 do not require the computation of the predictive state covariance matrices. In Table A.2 the complexity is summarised depending on the filters we choose. In order to alleviate the computational cost of the inversion of the observation covariance matrix $\boldsymbol{C}_t^i(\boldsymbol{y})$, in practice we use the approximation described in Appendix C.

Algorithm	SMC	SQMC
EKF	$\mathcal{O}(d_x^2 d_y^3 N)$	$\mathcal{O}(d_x^2 d_y^3 N \log N)$
EnKF	$\mathcal{O}(d_x d_y^3 M N)$	$\mathcal{O}(d_x d_y^3 M N \log N)$

Table A.2: Summary of the complexity of the different NHFs.

B

Proof of Theorem 1

B.1 Outline of the proof

We need to prove that the approximation μ_t^N generated by a generic nested filter that satisfies assumptions A.1, A.2 and A.3 converges to $\bar{\mu}_t$ in L_p, for each $t = 1, 2, ..., n_0 < \infty$. We split the analysis of the nested filter in three steps: jittering, weight computation and resampling. The approximation μ_{t-1}^N of $\bar{\mu}_{t-1}$ is available at the beginning of the t-th time step. After jittering, we obtain a new approximation,

$$\breve{\mu}_{t-1}^N = \frac{1}{N}\sum_{i=1}^{N} \delta_{\bar{\boldsymbol{\theta}}_t^i}, \tag{B.1}$$

that can be proved to converge to $\bar{\mu}_{t-1}$ using an auxiliary result from [25]. After the computation of the weights, the measure

$$\tilde{\mu}_t^N = \sum_{i=1}^{N} w_t^i \delta_{\bar{\boldsymbol{\theta}}_t^i} \tag{B.2}$$

is obtained and its convergence towards $\bar{\mu}_t$ has to be established. Finally, after the resampling step, a standard piece of analysis proves the convergence of

$$\mu_t^N = \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{\theta}_t^i} \tag{B.3}$$

to $\bar{\mu}_t$. Below, we provide three lemmas for the conditional convergence of $\breve{\mu}_{t-1}^N$, $\tilde{\mu}_t^N$ and μ_t^N, respectively. Then we combine them in order to prove Theorem 1 by an induction argument.

B.2 Jittering

In the jittering step, a new cloud of particles $\{\bar{\boldsymbol{\theta}}_t^i\}_{i=1}^N$ is generated by propagating the existing samples across the kernels $\kappa_N(d\boldsymbol{\theta}|\boldsymbol{\theta}_{t-1}^i)$, $i = 1, \ldots, N$. This step has been analyzed in [25] in the context of the nested particle filter (NPF). Several types of kernels can be used. In general, there is a trade-off between the number of particles that are changed using this kernel and the "amount of perturbation" that can be applied to each particle. For this reason, we let the jittering kernel κ_N depend explicitly on N. For our analysis, assumption A.3 is sufficient.

The convergence results to be given in this appendix are presented in terms of upper bounds for the L_p norms of the approximation errors. For a random variable z, its L_p norm is $\|z\|_p = \mathbb{E}\left[|\boldsymbol{z}|^p\right]^{\frac{1}{p}}$. The approximate measures generated by the nested filter, e.g., μ_t^N, are measured-valued random variables. Therefore, integrals of the form (h, μ_t^N), for some $h \in B(D)$, are real random variables and it makes sense to evaluate the L_p norm of the random error $(h, \mu_t^N) - (h, \bar{\mu}_t)$. We start with the approximation $\breve{\mu}_{t-1}^N$ produced after the jittering step at time t.

Lemma 1 *Let the sequence of observations $y_{1:t}$ be arbitrary but fixed. If $h \in B(D)$, A.3 holds and*

$$\|(h, \mu_{t-1}^N) - (h, \bar{\mu}_{t-1})\|_p \leq \frac{c_{t-1}\|h\|_\infty}{\sqrt{N}} \tag{B.4}$$

for some $p \geq 1$ and a constant $c_{t-1} < \infty$ independent of N, then

$$\|(h, \breve{\mu}_{t-1}^N) - (h, \bar{\mu}_{t-1})\|_p \leq \frac{c_{1,t}\|h\|_\infty}{\sqrt{N}}, \tag{B.5}$$

where the constant $c_{1,t} < \infty$ is also independent of N.

Proof: The proof of this Lemma is identical to the proof of [25, Lemma 3]. □

B.3 Computation of the weights

In order to analyze the errors at the weight computation step we rely on assumption A.2. An upper bound for the error in the weight computation step is established next.

Lemma 2 *Let the sequence of observations $y_{1:t}$ be arbitrary but fixed, choose any $h \in B(D)$ and some $p \geq 1$. If assumptions A.1 and A.2 hold, and*

$$\|(h, \breve{\mu}_{t-1}^N) - (h, \bar{\mu}_{t-1})\|_p \leq \frac{c_{1,t}\|h\|_\infty}{\sqrt{N}} \tag{B.6}$$

for some constant $c_{1,t} < \infty$ independent of N, then

$$\|(h, \tilde{\mu}_t^N) - (h, \bar{\mu}_t)\|_p \leq \frac{c_{2,t}\|h\|_\infty}{\sqrt{N}}, \tag{B.7}$$

where the constant $c_{2,t} < \infty$ is independent of N.

Proof: We address the characterization of the weights and, therefore, of the approximate measure $\tilde{\mu}_t^N = \sum_{i=1}^N w_t^i \delta_{\bar{\theta}_t^i}$. From the definition of the projective product in (3.24), the integrals of h w.r.t. $\bar{\mu}_t$ and $\tilde{\mu}_t^N$ can be written as

$$(h, \bar{\mu}_t) = \frac{(\bar{u}_t h, \bar{\mu}_{t-1})}{(\bar{u}_t, \bar{\mu}_{t-1})}, \quad \text{and} \quad (h, \tilde{\mu}_t^N) = \frac{(\hat{u}_t h, \breve{\mu}_{t-1}^N)}{(\hat{u}_t, \breve{\mu}_{t-1}^N)}, \tag{B.8}$$

respectively. From (B.8) one can write the difference $(h,\tilde{\mu}_t^N)-(h,\bar{\mu}_t)$ as

$$(h,\tilde{\mu}_t^N)-(h,\bar{\mu}_t)=\frac{(h\hat{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\bar{\mu}_{t-1})}{(\bar{u}_t,\bar{\mu}_{t-1})}+(h,\tilde{\mu}_t^N)\frac{(\bar{u}_t,\bar{\mu}_{t-1})-(\hat{u}_t,\breve{\mu}_{t-1}^N)}{(\bar{u}_t,\bar{\mu}_{t-1})},$$

which readily yields the inequality

$$|(h,\tilde{\mu}_t^N)-(h,\bar{\mu}_{t-1})|\leq\frac{|(h\hat{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\bar{\mu}_{t-1})|}{(\bar{u}_t,\bar{\mu}_{t-1})}+\frac{\|h\|_\infty|(\hat{u}_t,\breve{\mu}_{t-1}^N)-(\bar{u}_t,\bar{\mu}_{t-1})|}{(\bar{u}_t,\bar{\mu}_{t-1})}\tag{B.9}$$

by simply noting that $|(h,\tilde{\mu}_t^N)|\leq\|h\|_\infty$, since $\tilde{\mu}_t^N$ is a probability measure. From (B.9) and Minkowski's inequality we easily obtain the bound

$$\begin{aligned}\|(h,\tilde{\mu}_t^N)-(h,\bar{\mu}_{t-1})\|_p &\leq \frac{1}{(\bar{u}_t,\bar{\mu}_{t-1})}\left[\|h\|_\infty\|(\hat{u}_t,\breve{\mu}_{t-1}^N)-(\bar{u}_t,\bar{\mu}_{t-1})\|_p\right.\\ &\quad \left.+\|(h\hat{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\bar{\mu}_{t-1})\|_p,\right]\end{aligned}\tag{B.10}$$

where $(\bar{u}_t,\bar{\mu}_{t-1})>0$ from assumption A.2.2.

We need to find upper bounds for the two terms on the right hand side of (B.10). Consider first the term $\|(h\hat{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\bar{\mu}_{t-1})\|_p$. A simple triangle inequality yields

$$\|(h\hat{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\bar{\mu}_{t-1})\|_p\leq\|(h\hat{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\breve{\mu}_{t-1}^N)\|_p+\|(h\bar{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\bar{\mu}_{t-1})\|_p.\tag{B.11}$$

On one hand, since $\sup_{\boldsymbol{\theta}\in D}|h(\boldsymbol{\theta})\bar{u}_t(\boldsymbol{\theta})|\leq\|h\|_\infty\|\bar{u}_t\|_\infty<\infty$ (see A.2.1), it follows from the assumption in Eq. (B.6) that

$$\|(h\bar{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\bar{\mu}_{t-1})\|_p\leq\frac{c_{1,t}\|h\|_\infty\|\bar{u}_t\|_\infty}{\sqrt{N}},\tag{B.12}$$

where $c_{1,t}<\infty$ is a constant independent of N.

On the other hand, we may note that

$$|(h\hat{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\breve{\mu}_{t-1}^N)|^p=\left|\frac{1}{N}\sum_{i=1}^N\left(h(\bar{\boldsymbol{\theta}}_t^i)\hat{u}_t(\bar{\boldsymbol{\theta}}_t^i)-h(\bar{\boldsymbol{\theta}}_t^i)\bar{u}_t(\bar{\boldsymbol{\theta}}_t^i)\right)\right|^p.\tag{B.13}$$

Let $\mathcal{G}_t$ be the σ-algebra generated by the random particles $\{\bar{\boldsymbol{\theta}}_{1:t-1}^i,\boldsymbol{\theta}_{0:t-1}^i\}_{1\leq i\leq N}$ and assume that p is even. Then we can apply conditional expectations on both sides of (B.13) to obtain

$$\mathbb{E}\left[\left|(h\hat{u}_t,\breve{\mu}_{t-1}^N)-(h\bar{u}_t,\breve{\mu}_{t-1}^N)\right|^p\Big|\mathcal{G}_t\right]\quad=\quad\mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^N h(\bar{\boldsymbol{\theta}}_t^i)m_t(\bar{\boldsymbol{\theta}}_t^i)\right)^p\Big|\mathcal{G}_t\right]$$

where the expression on the right hand side has been simplified by using the assumption $\hat{u}_t(\boldsymbol{\theta})=\bar{u}_t(\boldsymbol{\theta})+m_t(\boldsymbol{\theta})$ in A.1. Also from assumption A.1, the random variables $m_t(\bar{\boldsymbol{\theta}}_t^i)$ are conditionally independent (given $\mathcal{G}_t$), have zero mean and finite moments of order p, $\mathbb{E}[m_t(\bar{\boldsymbol{\theta}}_t^i)^p]\leq\sigma^p<\infty$. If we realise that

$$\mathbb{E}[h(\bar{\boldsymbol{\theta}}_t^i)m_t(\bar{\boldsymbol{\theta}}_t^i)|\mathcal{G}_t]=h(\bar{\boldsymbol{\theta}}_t^i)\mathbb{E}[m_t(\bar{\boldsymbol{\theta}}_t^i)|\mathcal{G}_t]=0$$

and bear in mind the conditional independence of the $m_t(\bar{\boldsymbol{\theta}}_t^i)$'s, then it is an exercise in combinatorics to show that the number of non-zero terms in

$$\mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^N h(\bar{\boldsymbol{\theta}}_t^i)m_t(\bar{\boldsymbol{\theta}}_t^i)\right)^p\Big|\mathcal{G}_t\right]=\sum_{i_1}\ldots\sum_{i_p}\mathbb{E}\left[h(\bar{\boldsymbol{\theta}}_t^{i_1})m_t(\bar{\boldsymbol{\theta}}_t^{i_1})\ldots h(\bar{\boldsymbol{\theta}}_t^{i_p})m_t(\bar{\boldsymbol{\theta}}_t^{i_p})\Big|\mathcal{G}_t\right]$$

is at most $\tilde{c}^p N^{\frac{p}{2}}$, for some constant $\tilde{c}^p < \infty$ independent of N and h. Since each of the non-zero terms is upper bounded by $\mathbb{E}\left[(h(\bar{\boldsymbol{\theta}}_t^i)m_t(\bar{\boldsymbol{\theta}}_t^i))^p \middle| \mathcal{G}_t\right] \leq \|h\|_\infty^p \sigma^p < \infty$ (using A.1 again), then it follows that

$$\mathbb{E}\left[\left|(h\hat{u}_t, \breve{\mu}_{t-1}^N) - (h\bar{u}_t, \breve{\mu}_{t-1}^N)\right|^p\right] = \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^N h(\bar{\boldsymbol{\theta}}_t^i)m_t(\bar{\boldsymbol{\theta}}_t^i)\right)^p \middle| \mathcal{G}_t\right] \leq \frac{\tilde{c}^p \sigma^p \|h\|_\infty^p}{N^{\frac{p}{2}}} \tag{B.14}$$

for even p. Given (B.14), it is straightforward to show that the same result holds for every $p \geq 1$ using Jensen's inequality. Finally, since the bound on the right hand side of (B.14) is independent of $\mathcal{G}_t$, we can take expectations on both sides of the inequality and obtain that

$$\|(h\hat{u}_t, \breve{\mu}_{t-1}^N) - (h\bar{u}_t, \breve{\mu}_{t-1}^N)\|_p \leq \frac{\tilde{c}\sigma\|h\|_\infty}{\sqrt{N}}. \tag{B.15}$$

Substituting (B.15) and (B.12) into (B.11) yields

$$\|(h\hat{u}_t, \breve{\mu}_{t-1}^N) - (h\bar{u}_t, \bar{\mu}_{t-1})\|_p \leq \frac{c_t'\|h\|_\infty^p\|\bar{u}_t\|_\infty}{\sqrt{N}}, \tag{B.16}$$

where $c_t' = c_{1,t} + \tilde{c}\sigma$ is a constant independent of N.

The same argument leading to the bound in (B.16) can be repeated, step by step, on the norm $\|(\hat{u}_t, \breve{\mu}_{t-1}^N) - (\bar{u}_t, \bar{\mu}_{t-1})\|_p$ (simply taking $h(\boldsymbol{\theta}) = 1$), to arrive at

$$\|(\hat{u}_t, \breve{\mu}_{t-1}^N) - (\bar{u}_t, \bar{\mu}_{t-1})\|_p \leq \frac{c_t'\|\bar{u}_t\|_\infty}{\sqrt{N}}. \tag{B.17}$$

To complete the proof, we substitute (B.16) and (B.17) back into (B.10) and so obtain

$$\|(h, \tilde{\mu}_t^N) - (h, \bar{\mu}_{t-1})\|_p \leq \frac{c_{2,t}\|h\|_\infty}{\sqrt{N}},$$

where the constant $c_{2,t} = \|\bar{u}_t\|_\infty (2c_t') / (\bar{u}_t, \bar{\mu}_{t-1}) < \infty$ is independent of N. □

B.4 Resampling

The quantification of the error in the resampling step of the nested filter is a standard piece of analysis, well known from the particle filtering literature (see, e.g., [13]). We can state the following result.

Lemma 3 *Let the sequence of observations $y_{1:t}$ be arbitrary but fixed. If $h \in B(D)$ and*

$$\|(h, \tilde{\mu}_t^N) - (h, \bar{\mu}_t)\|_p \leq \frac{c_{2,t}\|h\|_\infty}{\sqrt{N}} \tag{B.18}$$

for a constant $c_{2,t} < \infty$ independent of N, then

$$\|(h, \mu_t^N) - (h, \bar{\mu}_t)\|_p \leq \frac{c_{3,t}\|h\|_\infty}{\sqrt{N}},$$

where the constant $c_{3,t} < \infty$ is independent of N as well.

Proof: See, e.g., the proof of [74, Lemma 1]. □

B.5 An induction proof for Theorem 1

Finally, we can put Lemmas 1, 2 and 3 together in order to prove the inequality (3.26) by induction in t. At time $t = 0$, we draw $\boldsymbol{\theta}_0^i$, $i = 1, \ldots, N$, independently from the prior $\mu_0 = \bar{\mu}_0$ and it is straightforward to show that $\|(h, \mu_0^N) - (h, \bar{\mu}_0)\|_p \leq \frac{c_0 \|h\|_\infty}{\sqrt{N}}$, where c_0 does not depend on N.

Assume that, at time $t - 1$,

$$\|(h, \mu_{t-1}^N) - (h, \bar{\mu}_{t-1})\|_p \leq \frac{c_{t-1} \|h\|_\infty}{\sqrt{N}}$$

where $c_{t-1} < \infty$ is independent of N. Then, we simply apply Lemmas 1, 2 and 3 in sequence to obtain

$$\|(h, \mu_t^N) - (h, \bar{\mu}_t)\|_p \leq \frac{c_t \|h\|_\infty}{\sqrt{N}}$$

for a constant $c_t = c_{3,t} < \infty$ independent of N. $\square$

C
Simplification of the inverse

The predictive covariance of the observation vector $\boldsymbol{y}_t$ is a $d_y \times d_y$ matrix $\boldsymbol{C}_t(\boldsymbol{y})$. Inverting $\boldsymbol{C}_t(\boldsymbol{y})$ has a cost $\mathcal{O}(d_y^3)$, which can become intractable. Assuming that variables located "far away" in the circumference of the Lorenz 96 model have small correlation we can approximate $\boldsymbol{C}_t(\boldsymbol{y})$ as a block diagonal matrix, namely, $\hat{\boldsymbol{C}}_t(\boldsymbol{y}) = \boldsymbol{C}_t(\boldsymbol{y}) \odot \boldsymbol{M}$, where $\odot$ denotes element-wise product,

$$\boldsymbol{M} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{1} \end{bmatrix} \tag{C.1}$$

is a mask matrix and $\mathbf{0}$ and $\mathbf{1}$ are, respectively, matrices of zeros and ones of dimension $d_q \times d_q$. There are Q blocks in the diagonal of $\boldsymbol{M}$, hence $d_y = Qd_q$. The original matrix could contain some non-zero values where the zero blocks of $\boldsymbol{M}$ are placed, however their values are assumed close to zero. The resulting matrix,

$$\hat{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{C}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{C}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{C}_Q \end{bmatrix}, \quad \text{is easily inverted as} \quad \hat{\boldsymbol{C}}_t^{-1} = \begin{bmatrix} \boldsymbol{C}_1^{-1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{C}_2^{-1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{C}_Q^{-1} \end{bmatrix}$$

with a computational cost $\mathcal{O}(Qd_q^3) = \mathcal{O}(\frac{d_y^3}{Q^2})$.

D
Proof of Proposition 1

We proceed by induction in the time index t. For $t = 0$ we have $\pi_0(\boldsymbol{x}_0|\boldsymbol{\theta}) = p(\boldsymbol{x}_0)$ independently of $\boldsymbol{\theta}$, hence for any pair $(\boldsymbol{\theta}, \boldsymbol{\theta}') \in D \times D$ we obtain

$$\int \left|\pi_0(\boldsymbol{x}_0|\boldsymbol{\theta}) - \pi_0(\boldsymbol{x}_0|\boldsymbol{\theta}')\right| d\boldsymbol{x}_0 = \int \left|p(\boldsymbol{x}_0) - p(\boldsymbol{x}_0')\right| = L_0\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$$

for $L_0 = 0$.

For the induction step, assume that

$$\int \left|\pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}) - \pi_{t-1}(\boldsymbol{x}_{t-1}\middle|\boldsymbol{\theta}')\right| d\boldsymbol{x}_{t-1} < L_{t-1}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \tag{D.1}$$

for some $L_{t-1} < \infty$. Straightforward calculations yield

$$\begin{aligned}
&\int \left|\xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')\right| d\boldsymbol{x}_t = \\
&\quad = \int \bigg| \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta})\pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}) d\boldsymbol{x}_{t-1} - \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}')\pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}') d\boldsymbol{x}_{t-1} \\
&\quad \pm \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta})\pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}') d\boldsymbol{x}_{t-1} \bigg| d\boldsymbol{x}_t \\
&\quad \leq \int\int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) \left|\pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}) - \pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}')\right| d\boldsymbol{x}_{t-1} d\boldsymbol{x}_t \\
&\quad + \int\int \left|p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) - p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}')\right| \pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}') d\boldsymbol{x}_{t-1} d\boldsymbol{x}_t
\end{aligned}$$

and reordering the integrals we obtain

$$\int \left|\xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')\right| d\boldsymbol{x}_t \leq$$

$$\leq \int \left[\int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) d\boldsymbol{x}_t \right] \left| \pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}) - \pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}') \right| d\boldsymbol{x}_{t-1}$$
$$+ \int \left[\int \left| p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) - p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}') \right| d\boldsymbol{x}_t \right] \pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}') d\boldsymbol{x}_{t-1} \tag{D.2}$$

However, $\int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) d\boldsymbol{x}_t = 1$ for any $\boldsymbol{x}_{t-1}$ and any $\boldsymbol{\theta}$, while Assumption 5 yields $\int |p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}) - p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}')| d\boldsymbol{x}_t \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$. Therefore, (D.2) becomes

$$\int \left| \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') \right| d\boldsymbol{x}_t \leq \int \left| \pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}) - \pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}') \right| d\boldsymbol{x}_{t-1}$$
$$+ L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \int \pi_{t-1}(\boldsymbol{x}_{t-1}|\boldsymbol{\theta}') d\boldsymbol{x}_{t-1}$$
$$\leq (L_{t-1} + L)\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \tag{D.3}$$

where the second inequality follows from the induction hypothesis (D.1).

As for the difference between $\pi_t(\cdot|\boldsymbol{\theta})$ and $\pi_t(\cdot|\boldsymbol{\theta}')$, the Bayes theorem readily yields

$$\int \left| \pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') \right| d\boldsymbol{x}_t = \int \left| \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta})}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} - \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta}')} \right| d\boldsymbol{x}_t \tag{D.4}$$

and the absolute difference in the integrand of (D.4) can be rewritten as

$$\left| \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta})}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} - \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta}')} \right|$$
$$= \left| \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta})}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} \pm \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} - \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta}')} \right|$$
$$= \left| \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} + \pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') \frac{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta}') - \eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} \right| \tag{D.5}$$

where we have used the relationship $\pi_t(\boldsymbol{x}_t|\theta') = \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta})}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta}')}$ to obtain the second identity. Now, if we substitute (D.5) into (D.4) and then realize that

$$\left| \eta_t(\boldsymbol{y}_t|\boldsymbol{\theta}) - \eta_t(\boldsymbol{y}_t|\boldsymbol{\theta}') \right| \leq \int \left| p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') \right| d\boldsymbol{x}_t$$

and $\int \pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') d\boldsymbol{x}_t = 1$, we obtain the upper bounds

$$\int \left| \pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') \right| d\boldsymbol{x}_t$$
$$\leq \frac{2}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} \int \left| p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') \right| d\boldsymbol{x}_t \tag{D.6}$$
$$\leq \frac{2}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} \left[\int p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) \left| \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') \right| d\boldsymbol{x}_t \right.$$
$$\left. + \int \left| p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}) - p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta}') \right| \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') d\boldsymbol{x}_t \right] \tag{D.7}$$

where (D.7) is obtained by applying a triangular inequality in (D.6).

The first integral in (D.7) can be bounded using Assumption 7 and inequality (D.3), which together yield,

$$\int \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{\theta})}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})} \left| \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}') \right| d\boldsymbol{x}_t \leq M_t \int \left| \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) \right| d\boldsymbol{x}_t$$
$$\leq M_t(L_{t-1} + L)\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|, \tag{D.8}$$

while the second integral can be bounded using Assumption 6, which leads to

$$\begin{aligned}\frac{2}{\eta_t(\boldsymbol{y}_t|\boldsymbol{\theta})}\int \left|p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta}) - p(\boldsymbol{y}_t|\boldsymbol{x}_t,\boldsymbol{\theta}')\right|\xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')d\boldsymbol{x}_t &\le 2G_t\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|\int \xi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')d\boldsymbol{x}_t \\ &= 2G_t\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|. \end{aligned} \tag{D.9}$$

Plugging (D.8) and (D.9) into (D.7) yields

$$\begin{aligned}\int \left|\pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}) - \pi_t(\boldsymbol{x}_t|\boldsymbol{\theta}')\right| &\le M_t(L_{t-1}+L)\|\boldsymbol{\theta}-\boldsymbol{\theta}'\| + 2G_t\|\boldsymbol{\theta}-\boldsymbol{\theta}'\| \\ &\le L_t\|\boldsymbol{\theta}-\boldsymbol{\theta}'\|\end{aligned}$$

with $L_t = M_t(L_{t-1}+L) + 2G_t < \infty$. $\square$

References

[1] Assyr Abdulle, E Weinan, Björn Engquist, and Eric Vanden-Eijnden. The heterogeneous multiscale method. *Acta Numerica*, 21:1–87, 2012.

[2] A. Aksoy, F. Zhang, and J. W. Nielsen-Gammon. Ensemble-based simultaneous state and parameter estimation in a two-dimensional sea-breeze model. *Monthly Weather Review*, 134(10):2951–2970, 2006.

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

[4] Jaison Thomas Ambadan and Youmin Tang. Sigma-point Kalman filter data assimilation methods for strongly nonlinear systems. *Journal of the Atmospheric Sciences*, 66(2):261–285, 2009.

[5] B. D. O. Anderson and J. B. Moore. *Optimal Filtering.* Englewood Cliffs, 1979.

[6] Jeffrey L Anderson. Spatially and temporally varying adaptive covariance inflation for ensemble filters. *Tellus A*, 61(1):72–83, 2009.

[7] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society B*, 72:269–342, 2010.

[8] C. Andrieu, A. Doucet, S. S. Singh, and V. B. Tadić. Particle methods for change detection, system identification and control. *Proceedings of the IEEE*, 92(3):423–438, March 2004.

[9] Christophe Andrieu, Arnaud Doucet, and Vladislav B Tadić. One-line parameter estimation in general state-space models using a pseudo-likelihood approach. *IFAC Proceedings Volumes*, 45(16):500–505, 2012.

[10] I. Arasaratnam and S. Haykin. Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009.

[11] H. M. Arnold. *Stochastic parametrisation and model uncertainty.* PhD thesis, University of Oxford, 2013.

[12] Trevor T Ashley and Sean B Andersson. Method for simultaneous localization and parameter estimation in particle tracking experiments. *Physical Review E*, 92(5):052707, 2015.

[13] A. Bain and D. Crisan. *Fundamentals of Stochastic Filtering.* Springer, 2008.

[14] David Barber and Yali Wang. Gaussian processes for bayesian estimation in ordinary differential equations. In *International conference on machine learning*, pages 1485–1493. PMLR, 2014.

[15] 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.

[16] George EP Box, Mervin E Muller, et al. A note on the generation of random normal deviates. *The annals of mathematical statistics*, 29(2):610–611, 1958.

[17] Richard J Boys, Darren J Wilkinson, and Thomas BL Kirkwood. Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*, 18(2):125–135, 2008.

[18] Paul Bratley and Bennett L Fox. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1):88–100, 1988.

[19] Laure Buhry, Filippo Grassia, Audrey Giremus, Eric Grivel, Sylvie Renaud, and Sylvain Saïghi. Automated parameter estimation of the Hodgkin-Huxley model using the differential evolution algorithm: application to neuromimetic analog integrated circuits. *Neural computation*, 23(10):2599–2625, 2011.

[20] C. M. Carvalho, M. S. Johannes, H. F. Lopes, and N. G. Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2010.

[21] Nicolas Chopin, Pierre E Jacob, and Omiros Papaspiliopoulos. SMC^2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013.

[22] A. M. Clayton, A. Lorenc, and D. M. Barker. Operational implementation of a hybrid ensemble/4D-Var global data assimilation system at the Met Office. *Quarterly Journal of the Royal Meteorological Society*, 139(675):1445–1461, 2013.

[23] 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.

[24] D. Crisan, J. Miguez, and G. Ríos. On the performance of parallelisation schemes for particle filtering. *EURASIP Journal on Advances in Signal Processing*, in press, 2018.

[25] Dan Crisan, Joaquin Miguez, et al. Nested particle filters for online parameter estimation in discrete-time state-space Markov models. *Bernoulli*, 24(4A):3039–3086, 2018.

[26] Johan Dahlin and Thomas B Schön. Getting started with particle Metropolis-Hastings for inference in nonlinear dynamical models. *arXiv preprint arXiv:1511.01707*, 2015.

[27] D. P. Dee, S. M. Uppala, A. J. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M. A. Balmaseda, G. Balsamo, and P. Bauer. The ERA-interim reanalysis: Configuration and performance of the data assimilation system. *Quarterly Journal of the royal meteorological society*, 137(656):553–597, 2011.

[28] P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, 2004.

[29] Feng Ding. State filtering and parameter estimation for state space systems with scarce measurements. *Signal Processing*, 104:369–380, 2014.

[30] P. M. Djurić, M. F. Bugallo, and J. Míguez. Density assisted particle filters for state and parameter estimation. In *Proceedings of the 29th IEEE ICASSP*, May 2004.

[31] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Míguez. Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38, September 2003.

[32] P. M. Djurić and J. Míguez. Sequential particle filtering in the presence of additive Gaussian noise with unknown parameters. In *Proceedings of ICASSP*, May 2002.

[33] A. Doucet, N. de Freitas, and N. Gordon. An introduction to sequential Monte Carlo methods. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 1, pages 4–14. Springer, 2001.

[34] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York (USA), 2001.

[35] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo Sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[36] G. Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[37] G. Evensen. The ensemble Kalman filter for combined state and parameter estimation. *IEEE Control Systems*, 29(3), 2009.

[38] Marco Frei and Hans R Künsch. Sequential state and observation noise covariance estimation using combined ensemble Kalman and particle filters. *Monthly Weather Review*, 140(5):1476–1495, 2012.

[39] Thomas C Gard. *Introduction to stochastic differential equations*. M. Dekker, 1988.

[40] M. Gerber and N. Chopin. Sequential quasi Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(3):509–579, 2015.

[41] A. Golightly and D. J. Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*, 1(6):807–820, 2011.

[42] N. Gordon, D. Salmond, and A. F. M. Smith. Novel approach to nonlinear and non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140(2):107–113, 1993.

[43] J. Hakkarainen, A. Ilin, A. Solonen, M. Laine, H. Haario, J. Tamminen, E. Oja, and H. Järvinen. On closure parameter estimation in chaotic systems. *Nonlinear Proc. Geoph.*, 19(1):127–143, 2012.

[44] John H Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964.

[45] Chris H Hamilton and Andrew Rau-Chaplin. Compact Hilbert indices: Space-filling curves for domains with unequal side lengths. *Information Processing Letters*, 105(5):155–163, 2008.

[46] Iraj Hassanzadeh and AF Mehdi. Design of augmented extended and unscented Kalman filters. *Journal of Applied Sciences*, 8(16):2901–2906, 2008.

[47] Hongwen He, Rui Xiong, Xiaowei Zhang, Fengchun Sun, and JinXin Fan. State-of-charge estimation of the lithium-ion battery using an adaptive extended Kalman filter based on an improved Thevenin model. *IEEE Transactions on vehicular technology*, 60(4):1461–1469, 2011.

[48] Yoshito Hirata. Fast time-series prediction using high-dimensional data: Evaluating confidence interval credibility. *Physical Review E*, 89(5):052916, 2014.

[49] Peter L Houtekamer and Herschel L Mitchell. A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129(1):123–137, 2001.

[50] Shin-ichi Ito, Hiromichi Nagao, Akinori Yamanaka, Yuhki Tsukada, Toshiyuki Koyama, Masayuki Kano, and Junya Inoue. Data assimilation for massive autonomous systems based on a second-order adjoint method. *Physical Review E*, 94(4):043307, 2016.

[51] Anders Chr Jensen, Susanne Ditlevsen, Mathieu Kessler, and Omiros Papaspiliopoulos. Markov chain Monte Carlo approach to parameter estimation in the Fitzhugh-Nagumo model. *Physical Review E*, 86(4):041114, 2012.

[52] S. J. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(2):401–422, March 2004.

[53] N. Kantas, A. Doucet, S. S. Singh, J. M. Maciejowski, and N. Chopin. On particle methods for parameter estimation in state-space models. *Statistical Science*, 30:328–351, August 2015.

[54] Sangjoon Kim, Neil Shephard, and Siddhartha Chib. Stochastic volatility: likelihood inference and comparison with ARCH models. *The review of economic studies*, 65(3):361–393, 1998.

[55] Taeklim Kim and Tae-Hyoung Park. Extended Kalman filter (EKF) design for vehicle position tracking using reliability function of radar and lidar. *Sensors*, 20(15):4126, 2020.

[56] G. Kitagawa. A self-organizing state-space model. *Journal of the American Statistical Association*, pages 1203–1215, 1998.

[57] E. Koblents and J. Míguez. A population Monte Carlo scheme with transformed weights and its application to stochastic kinetic models. *Statistics and Computing*, 25(2):407–425, 2015.

[58] P. J. Van Leeuwen. A variance-minimizing filter for large-scale applications. *Monthly Weather Review*, 131(9):2071–2084, 2003.

[59] P. J. Van Leeuwen. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, 136(653):1991–1999, 2010.

[60] Hong Li, Eugenia Kalnay, and Takemasa Miyoshi. Simultaneous estimation of covariance inflation and observation errors within an ensemble Kalman filter. *Quarterly Journal of the Royal Meteorological Society*, 135(639):523–533, 2009.

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

[62] J. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 10, pages 197–223. Springer, 2001.

[63] J. S. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, September 1998.

[64] J. S. Liu, R. Chen, and T. Logvinenko. A theoretical framework for sequential importance sampling with resampling. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 11, pages 225–246. Springer, 2001.

[65] E Lourens, Edwin Reynders, Guido De Roeck, Geert Degrande, and Geert Lombaert. An augmented Kalman filter for force identification in structural dynamics. *Mechanical Systems and Signal Processing*, 27:446–460, 2012.

[66] Jan Mandel. *Efficient implementation of the ensemble Kalman filter*. University of Colorado at Denver and Health Sciences Center, Center for Computational Mathematics, 2006.

[67] I. P. Mariño and J. Míguez. On a recursive method for the estimation of unknown parameters of partially observed chaotic systems. *Physica D*, 220:175–182, September 2006.

[68] I. P. Mariño, A. Zaikin, and J. Miguez. A comparison of Monte Carlo-based Bayesian parameter estimation methods for stochastic models of genetic networks. *PLOS ONE*, 12(8):e0182015, 2017.

[69] Luca Martino, David Luengo, and Joaquín Míguez. *Independent Random Sampling Methods*. Springer, 2018.

[70] Engin Masazade, Makan Fardad, and Pramod K Varshney. Sparsity-promoting extended Kalman filtering for target tracking in wireless sensor networks. *IEEE Signal Processing Letters*, 19(12):845–848, 2012.

[71] Henrique MT Menegaz, João Y Ishihara, Geovany A Borges, and Alessandro N Vargas. A systematization of the unscented Kalman filter theory. *IEEE Transactions on automatic control*, 60(10):2583–2598, 2015.

[72] Renate Meyer, David A Fournier, and Andreas Berg. Stochastic volatility: Bayesian computation using automatic differentiation and the extended Kalman filter. *The Econometrics Journal*, 6(2):408–420, 2003.

[73] J. Míguez. Analysis of parallelizable resampling algorithms for particle filtering. *Signal Processing*, 87(12):3155–3174, 2007.

[74] J. Míguez, D. Crisan, and P. 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.

[75] J. Míguez, I. P. Mariño, and M. A. Vázquez. Analysis of a nonlinear importance sampling scheme for Bayesian parameter estimation in state-space models. *Signal Processing*, 142:281–291, January 2018.

[76] J. Miguez and M. A. Vázquez. A proof of uniform convergence over time for a distributed particle filter. *Signal Processing*, 122:152–163, 2016.

[77] Bongki Moon, Hosagrahar V Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the Hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):124–141, 2001.

[78] Christian A Naesseth, Fredrik Lindsten, and Thomas B Schön. High-dimensional filtering using nested sequential Monte Carlo. *IEEE Transactions on Signal Processing*, 67(16):4177–4188, 2019.

[79] Christopher Nemeth, Paul Fearnhead, and Lyudmila Mihaylova. Sequential Monte Carlo methods for state and parameter estimation in abruptly changing environments. *IEEE Transactions on Signal Processing*, 62(5):1245–1255, 2013.

[80] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*, volume 63. SIAM, 1992.

[81] J. Olsson, O. Cappé, R. Douc, and E. Moulines. Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli*, 14(1):155–179, 2008.

[82] Edward Ott, Brian R Hunt, Istvan Szunyogh, Aleksey V Zimin, Eric J Kostelich, Matteo Corazza, Eugenia Kalnay, DJ Patil, and James A Yorke. A local ensemble Kalman filter for atmospheric data assimilation. *Tellus A*, 56(5):415–428, 2004.

[83] A. Papavasiliou. A uniformly convergent adaptive particle filter. *Journal of Applied Probability*, 42(4):1053–1068, 2005.

[84] A. Papavasiliou. Parameter estimation and asymptotic stability in stochastic filtering. *Stochastic Processes and Their Applications*, 116:1048–1065, 2006.

[85] Diego Pazó, A Carrassi, and Juan M López. Data assimilation by delay-coordinate nudging. *Quarterly Journal of the Royal Meteorological Society*, 142(696):1290–1299, 2016.

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

[87] Daniel Rey, Michael Eldridge, Mark Kostuk, Henry DI Abarbanel, Jan Schumann-Bischoff, and Ulrich Parlitz. Accurate state and parameter estimation in nonlinear systems with sparse observations. *Physics Letters A*, 378(11-12):869–873, 2014.

[88] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications.* Artech House, Boston, 2004.

[89] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer, 2004.

[90] J. Uhlmann S. J. Julier and H. F. Durrant-Whyte. A new method for the non linear transformation of means and covariances in filters and estimators. *IEEE Transactions Automatic Control*, 3:477–482, March 2000.

[91] J. Míguez S. Pérez-Vieites, I. P. Mariño. Probabilistic scheme for joint parameter estimation and state prediction in complex dynamical systems. *Physical Review E*, 98(6), 063305, 2017.

[92] Naratip Santitissadeekorn and Christopher Jones. Two-stage filtering for joint state-parameter estimation. *Monthly Weather Review*, 143(6):2028–2042, 2015.

[93] G. Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions Signal Processing*, 50(2):281–289, February 2002.

[94] Vladislav B Tadic. Analyticity, convergence, and convergence rate of recursive maximum-likelihood estimation in hidden markov models. *IEEE Transactions on Information Theory*, 56(12):6406–6432, 2010.

[95] Ruey S Tsay. *Analysis of financial time series*, volume 543. John wiley & sons, 2005.

[96] Eric Vanden-Eijnden et al. Fast communications: Numerical techniques for multi-scale dynamical systems with stochastic effects. *Communications in Mathematical Sciences*, 1(2):385–391, 2003.

[97] C. Vergé, C. Dubarry, P. Del Moral, and E. Moulines. On parallel implementation of sequential Monte Carlo methods: the island particle model. *Statistics and Computing*, pages 1–18, 2013.

[98] Audronė Virbickaitė, Hedibert F Lopes, M Concepción Ausín, and Pedro Galeano. Particle learning for Bayesian semi-parametric stochastic volatility model. *Econometric Reviews*, 2019.

[99] Femke C Vossepoel and Peter Jan van Leeuwen. Parameter estimation using a particle method: Inferring mixing coefficients from sea level observations. *Monthly weather review*, 135(3):1006–1020, 2007.

[100] Michail D Vrettas, Manfred Opper, and Dan Cornford. Variational mean-field algorithm for efficient inference in large systems of stochastic differential equations. *Physical Review E*, 91(1):012148, 2015.

[101] Xuedong Wang, Tiancheng Li, Shudong Sun, and Juan M Corchado. A survey of recent advances in particle filters and remaining challenges for multitarget tracking. *Sensors*, 17(12):2707, 2017.

[102] E Weinan, Di Liu, and Eric Vanden-Eijnden. Analysis of multiscale methods for stochastic differential equations. *Communications on Pure and Applied Mathematics*, 58(11):1544–1585, 2005.

[103] Greg Welch, Gary Bishop, et al. An introduction to the Kalman filter. 1995.

[104] Jingxin Ye, Daniel Rey, Nirag Kadakia, Michael Eldridge, Uriel I Morone, Paul Rozdeba, Henry DI Abarbanel, and John C Quinn. Systematic variational method for statistical nonlinear state and parameter estimation. *Physical Review E*, 92(5):052901, 2015.

[105] Hongjuan Zhang, H-JH Franssen, Xujun Han, Jasper A Vrugt, and Harry Vereecken. State and parameter estimation of two land surface models using the ensemble Kalman filter and the particle filter. *Hydrology and Earth System Sciences*, 21(9):4927–4958, 2017.