THE UNIVERSITY *of* EDINBURGH

Methodological contributions to state and parameter inference for state-space models

Kostas Tsampourakis

Advisor: Víctor Elvira

School of Mathematics

Doctor of Philosophy

June 2025

Dedicated to the struggle of the Palestinian people

Acknowledgements

I would like to thank my supervisor, Víctor Elvira, for the opportunity to pursue a PhD in a field outside of my academic background, and for his support throughout this process. I am grateful for his guidance, his patience, and his consistent feedback. I particularly appreciate his emphasis on the importance of writing and rewriting, which has been a key part of my development during the PhD.

I also wish to thank my colleagues at the School of Mathematics for their support and camaraderie. In particular, I am grateful to Nicola Branchini, Ben Cox, Mary Llewellyn, Sara P. Vieites, Torben Sell, Man Ho Suen, and Isabella Deutch for many enjoyable and thought-provoking conversations on research and broader topics.

I would like to acknowledge the staff at the University of Edinburgh, especially at the Bayes Centre and the School of Mathematics, for their support. I am especially thankful to Ezgi Kuyululu French for her responsiveness and helpfulness in administrative matters.

I would not be able to undergo this journey without the constant love and support of my family: my parents Vassilis and Kallia who always fostered my curiosity and sense of adventure, my aunt Rea and uncle Babis who are like second parents to me, my brother Stelios, and my cousins Aliki and Aris who are always there for me no matter what.

Finally, I would like to thank my friends in Greece and in Edinburgh for their love and support throughout this period. They are too many to name and I would not be able to complete this work without them.

Lay summary

In today's world, we collect data from nearly every aspect of life — from financial markets and environmental systems to health monitors and self-driving cars. Much of this data changes over time and reflects complex processes that we often cannot observe directly. For example, we may want to understand the true position of a moving vehicle using only GPS signals, or to track changes in an ecosystem using limited survey data. Making sense of these hidden, evolving systems is one of the central challenges in modern science and technology.

This thesis focuses on improving a widely used class of models known as state-space models. These models describe systems in terms of two parts: on one hand, a hidden state that changes over time, encompasses important information about the system; on the other, observed data coming from a measurement process, give us indirect clues about that hidden state. State-space models are powerful because they allow us to combine information from data with our knowledge about how the system works. They are used across many fields — including robotics, neuroscience, economics, and environmental science — wherever we need to understand processes that evolve over time but are only partly observable.

Working with state-space models involves two main tasks. The first is estimating the hidden state of the system based on the observed data. This is essential in tasks like tracking, forecasting, and decision-making. The second is learning the unknown parameters that govern how the system behaves and how the data is generated. This can help us better understand the system, improve predictions, or design better interventions. Both tasks are challenging because real-world systems are often complex, uncertain, and nonlinear, which makes the necessary calculations difficult or even impossible to do exactly.

This thesis presents two new methods that help solve these problems more effectively. The first method improves how we track hidden states in complex systems. Traditional approaches either rely on simple approximations that may be inaccurate or on random sampling methods that can be slow. The method developed in this work combines the strengths of both, adapting automatically to the nature of the system at each point in time. This results in a more reliable and efficient way to estimate what is happening inside the system, and works well even in situations where the system behavior is non-stationary.

The second method improves how we learn the parameters of a state-space model by using simulations. In this approach, we simulate many possible versions of the system and compare them to the actual observed data. This helps us identify which parameters best explain the system's behavior. A key insight behind the new method is that in many systems, the effects of early conditions

fade as time goes on. By using this idea, the method can extract more information from each simulation run, making the learning process faster, more stable, and more accurate — especially for long sequences of data.

Together, these contributions advance our ability to model, understand, and predict complex systems that evolve over time. The methods introduced in this thesis make state-space models more practical and effective, improving both the accuracy of hidden state estimation and the efficiency of learning model parameters. These advances are valuable across a wide range of applications, from guiding autonomous vehicles and managing financial risk to interpreting brain activity and monitoring natural systems. By making inference in state-space models more flexible and scalable, this work helps unlock new possibilities for data-driven decision-making in dynamic and uncertain environments.

Abstract

State-space models (SSMs) are a widespread framework for modeling dynamical systems in a wide variety of fields, including signal processing, robotics, neuroscience, and finance. These models represent systems in terms of latent variables that evolve over time according to Markovian dynamics, and are linked to observations through a probabilistic measurement process. Bayesian inference within SSMs comprises two core problems: state inference, which involves estimating the latent states given data and parameters, and parameter inference, which seeks to estimate unknown parameters of the model given observed data. Both problems are analytically tractable only in the special case of linear-Gaussian SSMs, where Kalman filtering and smoothing provide closed-form solutions. In general, however, practical inference in SSMs demands approximate methods due to nonlinearities and non-Gaussianity, as well as the intractability of the marginal likelihood over latent states.

This thesis is devoted to advancing the methodology for state and parameter inference in general SSMs. It begins by reviewing the classical literature, covering state inference methods such as Gaussian filters, Gaussian sum filters (GSFs), and particle filters (PFs), as well as parameter inference methods and discusses the limitations and trade-offs inherent to each approach. The thesis then proposes novel solutions for both state and parameter inference.

The first main contribution of this thesis is the development of the augmented Gaussian sum filter (AGSF), a novel class of Bayesian filters that addresses long-standing limitations of both GSFs and PFs. The AGSF is based on a Gaussian splitting scheme, that exploits a convolution identity to decompose a Gaussian distribution into a weighted mixture of lower-variance Gaussians. This representation enables a controlled trade-off between deterministic Gaussian approximations and stochastic particle-based methods. The AGSF generalizes standard GSFs and PFs, recovering both as limiting cases depending on the choice of augmentation covariances. Furthermore, an adaptive version of the AGSF is proposed that automatically sets these covariances via an optimization problem minimizing an upper bound on the mean squared error of moment estimates. The result is a flexible and robust filtering algorithm that dynamically adapts to the local nonlinearity of the model, blending the strengths of GSFs (efficiency) and PFs (accuracy and stability). Empirical results demonstrate that the AGSF achieves superior performance across various tasks, including maneuvering target tracking and systems with mixed linear and nonlinear dynamics.

The second major contribution is the introduction of truncated sequential neural likelihood (T-SNL), a new algorithm for parameter inference in SSMs

based on simulation-based inference (SBI). Traditional approaches to parameter inference suffer from high computational costs and instabilities due to the intractability of the SSM likelihood. Recent advances in SBI, particularly neural likelihood estimation via autoregressive normalizing flows have shown promise in high-dimensional, likelihood-free settings. Sequential neural likelihood (SNL) is a popular SBI method that learns a model of the likelihood, trained iteratively on simulated data near the posterior.

T-SNL builds on SNL by exploiting a key structural property of many SSMs: exponential forgetting, whereby the influence of initial conditions on the filtering distribution decays over time. T-SNL leverages this property by truncating the factors of the complete model likelihood. This truncation yields a significantly larger and more diverse training dataset from each simulation, vastly improving sample efficiency and training stability. Compared to SNL, T-SNL is able to make more efficient use of each simulation run, reducing the number of simulations required to reach high-quality posterior approximations. Moreover, T-SNL is easier to train than SNL, scales naturally to longer temporal sequences and is amortized. Experimental results show that T-SNL consistently outperforms existing SBI methods and classical inference techniques in both linear and nonlinear SSMs, including stochastic volatility models, and ecological population models.

Together, these contributions advance the state of the art in both state and parameter inference for SSMs. The AGSF provides a unified and adaptive framework for filtering that is capable of robustly interpolating between Gaussian and particle-based methods. T-SNL introduces a principled and scalable approach to simulator-based parameter inference by taking advantage of the temporal structure of SSMs. The thesis concludes with a discussion of future research directions, including the integration of AGSF into parameter inference frameworks, online learning settings, and applications in real-world domains such as control, neuroscience, and time-series forecasting.

Contents

Acknowledgements **v**

Lay summary **vii**

Abstract **x**

Contents **i**

List of acronyms **1**

1 Introduction **3**

2 Background **7**
2.1 State-space models 7
2.2 Inference of SSMs 9
2.3 Forgetting property of SSMs 12
2.4 Filters 14
2.4.1 Kalman filter 14
2.4.2 Extended Kalman filter 15
2.4.3 Unscented Kalman filter 15
2.4.4 Particle filters 16
2.5 Parameter inference algorithms 18
2.5.1 Markov chain Monte Carlo 20
2.5.2 Simulation-based inference and ABC 22

3 Augmented Gaussian sum filters **25**
3.1 Introduction 25
3.2 Background 29
3.2.1 Notation 29
3.2.2 General Gaussian filtering 29
3.2.3 Gaussian sum filters 33
3.3 Augmentation-based approximations 35
3.3.1 Augmentation of a Gaussian integral 36
3.3.2 Augmented Gaussian moment-matching 38
3.3.3 Automatic selection of $\boldsymbol{\Delta}$ 41
3.4 Augmented Gaussian Sum Filters 44
3.4.1 Framework description 44
3.4.2 The AGSF unifies GSF and BPF 50

3.4.3 Strategies for setting the parameters of AGSF 50
3.5 Numerical Experiments 51
3.6 Conclusions 58
3.7 Appendix for Chapter 3 64

4 Simulation-based inference of state-space models 73
4.1 Introduction 73
4.2 Background 75
4.2.1 Neural density estimation 75
4.2.2 Sequential neural likelihood 77
4.3 Neural likelihood estimation for inference in SSMs 79
4.3.1 Truncation of likelihood factors 80
4.3.2 The T-SNL algorithm 81
4.3.3 Choice of lag 83
4.4 Experiments 83
4.4.1 Setup 84
4.4.2 Results 86
4.4.3 Discussion 88
4.5 Conclusions 89
4.6 Appendix for Chapter 4 91

5 Conclusions 95

Bibliography 99

List of acronyms

ABC	Approximate bayesian computation
ACF	Auto-correlation function
AGSF	Augmented Gaussian sum filter
APF	Auxiliary particle filter
BPF	Bootstrap particle filter
EKF	Extended Kalman filter
EM	Expectation-maximization
ESS	Effective sample size
GMM	Gaussian moment matching
KDE	Kernel density estimate
L-AGSF	Linear-AGSF
L-GSF	Linear-GSF
LGSSM	Linear Gaussian state-space model
LV	Lotka-Volterra
MSE	Mean squared error
MH	Metropolis-Hastings
MCMC	Markov chain Monte Carlo
PDF	Probability density function
PF	Particle filter

PMCMC Particle Markov chain Monte Carlo

RWM Random walk Metropolis

SMC Sequential Monte Carlo

SMC-ABC Sequential Monte Carlo approximate bayesian computation

SBI Simulation-based inference

SSM State-space model

UT Unscented transform

UKF Unscented Kalman filter

U-AGSF Unscented-AGSF

U-GSF Unscented-GSF

Chapter 1

Introduction

The digital revolution of recent decades has enabled the collection, transmission, and analysis of information at an unprecedented scale, fundamentally changing nearly every aspect of human activity. Scientific research, engineering, policy-making, and even social interactions are being increasingly reshaped around the new information infrastructure that is being integrated into our modern societies [Emm21].

There are two central aspects of this revolution: data and algorithms. Data are the fundamental currency of information that encodes representations of the world, while algorithms provide the means to decode patterns from those representations and make decisions based on them. Data and algorithms are two faces of the same coin, and one cannot consider one without the other. Whether informing public health policy, guiding autonomous systems or financial decisions, the interaction between data and algorithms determines our ability to extract actionable insights, make reliable predictions, and respond adaptively to complex and uncertain environments.

Recently, advances in electronics and imaging have increased the scale, resolution, and precision of data-gathering across scientific and industrial domains. Moreover, the spread of smartphones, wearable devices, and high-speed wireless telecommunications has further contributed to an unprecedented richness of high-frequency data across all aspects of life. As a result, the size and complexity of our datasets has exploded in recent years, and so has the complexity of the patterns hidden within them. Modern datasets possess nontrivial structure: they may reflect spatial or temporal relations, be embedded in complex networks, or organized hierarchically across scales. Temporal structure in particular is extremely common in most applications. Most real-world phenomena evolve dynamically over time, and our ability to record them in real-time at an increasing rate has re-

cently exploded. Examples include records of neuronal dynamics, fluctuations in financial markets, changes in ecosystem populations, and the motion of physical systems.

Various approaches have been developed historically in different fields to model temporal data. In statistics and econometrics, classical time series models such as autoregressive integrated moving average and exponential smoothing have been used extensively for modeling and forecasting economic indicators and financial returns [Box+15]. In engineering and signal processing, techniques like Kalman filtering and spectral analysis have been central for real-time tracking and control of dynamical systems [AM12]. In recent years, machine learning and artificial intelligence have contributed highly flexible models such as recurrent neural networks (RNNs) [Sch19], temporal convolutional networks (TCNs)[Lea+17], and transformer-based architectures [Vas+17], which have found widespread success in fields like natural language processing [Bro+20], speech recognition [GMH13], robotics [Abd+20], and healthcare [Ner+23]. These methods differ in their interpretability, and ability to incorporate domain-specific knowledge.

One important class of models for temporal data is state-space models, widely used in robotics and signal processing. In a SSM, the system is described by an unobserved, noisy latent state that evolves over time, which generates the observed data through a separate observation model. The latent states typically encode the variables of interest, while the observation model captures the process by which measurements are generated conditionally on the states. The separation of the state and observation models provides two distinct advantages. First, by using an explicit observation model, it enables the incorporation of domain knowledge about the processes that generate the data. For example, in robotics, sensors such as LIDAR, cameras, and inertial measurement units (IMUs) provide indirect and noisy observations of the robot's true position and orientation. The observation model accounts for sensor characteristics, calibration, and environmental effects [Thr02b]. Second, the latent states can serve as a compact encoding of the data, summarizing the information in the observations. In neuroscience, for example, latent neural dynamics inferred from high-dimensional spike train recordings or magnetic resonance (fMRI) signals can act as low-dimensional representations that compress the activity into interpretable neural trajectories or cognitive states [Lin+19]. Moreover, SSMs are probabilistic methods treating all quantities as random variables described by probability distributions. Therefore, they are most useful in applications where we are interested in explicitly representing uncertainty in our beliefs about quantities of interest. They are widely used in fields such as control theory, econometrics and signal processing.

A SSM comprises of three sets of variables that are jointly modeled probabilistically: a set of latent states that capture the dynamics of the system, a set of observations that provide information about the system states and a set of parameters that characterize the dynamics and observation models. In most applications given the observations we aim to infer the hidden states and unknown parameters of the model. Therefore, two main tasks that arise when working with SSMs are state inference and parameter inference. State inference, also known as filtering and smoothing, refers to the task of computing posterior distributions of latent states given a sequence of observations and the model parameters. This task is central in applications such as navigation, object tracking, and time series denoising. Parameter inference, on the other hand, deals with computing the posterior of the static parameters that define the dynamics and observation models, given the data.

Both state and parameter inference in SSMs present computational challenges. For state inference, the primary difficulty comes from the intractability of probability density integrals that arise from nonlinear transformations of non-Gaussian random variables. Approximate methods such as Gaussian sum filters and particle filters have been developed to address this, which come with trade-offs between accuracy, computational cost, and ease of implementation. Parameter inference is further complicated by the fact that the likelihood is analytically intractable (also given by an intractable integral). Many parameter inference methods, such as expectation-maximization (EM) and Markov-chain Monte-Carlo (MCMC), use filters to obtain likelihood estimates recursively. Thus their accuracy is limited by the accuracy of the underlying filters. This is not the case for simulation-based methods which obtain likelihood surrogates via a number of simulations from the SSM by sampling the state and observation models.

This thesis presents contributions in both state and parameter inference by introducing two novel methods aimed at addressing these challenges. The first contribution focuses on filtering in complex, nonlinear SSMs and introduces a new algorithm that generalizes and unifies Gaussian sum filters and particle filters. The second contribution addresses the problem of parameter inference in SSMs through the lens of simulation-based inference. We compare SBI methods on this task, with a focus on SNL, a novel method based on neural density estimation. We show that SNL has limitations when applied to SSMs and propose an extension of SNL to address those limitations. The rest of this thesis is structured as follows.

Chapter 2 provides background on state-space models, including notation, key concepts, and a detailed overview of classical methods for state and parameter inference. It covers filtering and smoothing algorithms such as Kalman filters,

particle filters, and auxiliary particle filters, as well as inference methods like MCMC and simulation-based approaches.

Chapter 3 presents the first main contribution of this thesis: the augmented Gaussian sum filter. This chapter introduces the algorithm, provides theoretical analysis demonstrating how AGSF interpolates between Gaussian sum filters and particle filters, and proposes a version of the algorithm that is able to automatically adapt its behavior according to the local nature of the model. The chapter concludes with extensive empirical results illustrating the effectiveness of AGSF across a range of filtering tasks.

In **Chapter 4** we introduce the second main contribution: the truncated sequential neural likelihood method for parameter inference in SSMs. The T-SNL is an extension of SNL that is tailored to SSM inference. Here we motivate T-SNL based on the exponential forgetting property of SSMs. We compare T-SNL against other inference methods across multiple models and scenarios. We show that T-SNL is very sample efficient, and has several advantages over SNL.

Chapter 5 concludes the thesis by summarizing the contributions, discussing the strengths and limitations of the proposed methods, and outlining directions for future work, including their application to online learning, model selection, and integration into real-world systems.

Chapter 2

Background

In this chapter, we introduce the core concepts and methodologies that form the foundation of this thesis. We begin by defining state-space models and their probabilistic structure. Then we briefly present the main inference tasks associated with SSMs, including filtering, smoothing, prediction, and parameter inference. State and parameter inference are presented in more detail. Finally, we review approaches to these problems that are relevant to the contributions of this thesis. The goal of this chapter is to provide a general overview of these problems as well as introduce notation that is used later on. Further details and specific algorithmic developments are given in the respective chapters.

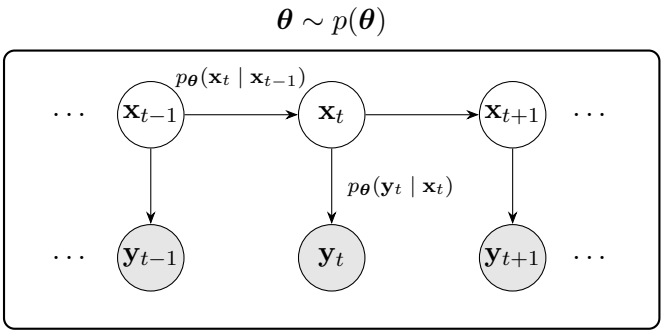


Figure 2.1: Graphical representation of a discrete-time SSM: the parameters come from a prior distribution. Conditional on the parameters, latent states follow Markovian dynamics (top layer) and observations (bottom layer) are generated conditional on the states.

2.1 State-space models

State-space models - also known as hidden Markov models - are a class of probabilistic models that describe the time-evolution of indirectly observed systems.

They are especially common in signal processing, control theory, econometrics and time series analysis.

A SSM describes the system at discrete time t by a latent *state vector* $\mathbf{x}_t$ in a *state space* $\mathcal{X}$. The state vector carries all information of interest about the system. A basic assumption of SSMs is that the time evolution of the state vector in the state space is Markovian[1] for all t. Consequently, the state dynamics is modeled as a Markov chain, the kernel of which is known as the *transition (or dynamics) model*.

Information about the system at each time t is obtained in the form of an *observation vector* $\mathbf{y}_t$ in a space $\mathcal{Y}$. The observation vector carries all information that we obtain about the system via measurement. It is assumed that the observation vector $\mathbf{y}_t$ only depends on $\mathbf{x}_t$. The measurement process is modeled by the conditional distribution of the observation vector given the state vector, known as the *observation (or measurement) model*. Formally a SSM is given by

$$\boldsymbol{\theta} \sim p(\boldsymbol{\theta}), \tag{2.1}$$

$$\mathbf{x}_t \sim p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t-1}),\ t = 1, \ldots, T, \tag{2.2}$$

$$\mathbf{y}_t \sim p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{x}_t),\ t = 1, \ldots, T, \tag{2.3}$$

where $p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the transition model and $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{x}_t)$ the observation model. The vector $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ with prior $p(\boldsymbol{\theta})$, encompasses the parameters of the transition and observation models of the SSM. A graphical representation of SSMs that is commonly used is shown in Fig. 2.1.

In many fields SSMs are formulated equivalently as

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{q}_t, \boldsymbol{\theta}), \tag{2.4}$$

$$\mathbf{y}_t = \mathbf{g}(\mathbf{y}_t, \mathbf{r}_t, \boldsymbol{\theta}), \tag{2.5}$$

for $t = 1, \ldots, T$, where $\mathbf{f}$ and $\mathbf{g}$ are transition and observation functions and $\mathbf{r}_t, \mathbf{q}_t$ are noise vectors. In this thesis we will use both formulations.

State-space models provide a flexible and principled framework for modeling dynamic systems in which the true state of the system is only partially observable. By capturing the temporal structure through Markovian dynamics and modeling uncertainty via probabilistic observation and transition models, SSMs can be tailored to a wide range of applications—from tracking physical systems in

[1]A dynamical system is said to be Markovian when each state depends on the past solely through its immediate predecessor: $P(X_t \mid X_{t-1}, \ldots, X_0) = P(X_t \mid X_{t-1})$

engineering to modeling latent economic indicators in econometrics. The separation of transition and observation models offers two key advantages: on one hand it allows domain knowledge to be encoded explicitly—for instance, by modeling sensor characteristics in robotics—and on the other it enables the latent states to serve as compact, interpretable summaries of complex data, such as neural activity in neuroscience. The dual formulation, either via conditional distributions or state-transition functions with noise inputs, allows SSMs to accommodate both classical parametric models and more complex nonlinear or simulation-based systems, as will be explored in the remainder of this thesis.

2.2 Inference of SSMs

When working with state-space models—as with probabilistic models more generally—the central task is inference: using observed data to update our beliefs about latent variables and parameters. In SSMs, this involves recovering information about the hidden states and model parameters from noisy, indirect observations. While the full joint posterior captures all such information, it is typically intractable and rarely needed in practice. Instead, inference focuses on marginal or conditional distributions of the full posterior, that are more relevant to downstream tasks and more accessible through approximate methods. This section outlines the core inference problems in SSMs and the strategies commonly used to solve them.

The inference problem of SSMs consists of computing the posterior distribution of the parameters $\boldsymbol{\theta}$ and states $\mathbf{x}_{1:T} \equiv \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ given the observations $\mathbf{y}_{1:T} \equiv \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$. The full posterior is given by

$$p(\boldsymbol{\theta}, \mathbf{x}_{1:T} | \mathbf{y}_{1:T}) = \frac{p(\boldsymbol{\theta}, \mathbf{x}_{1:T}, \mathbf{y}_{1:T})}{p(\mathbf{y}_{1:T})} = \frac{p(\boldsymbol{\theta}) p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T} | \boldsymbol{\theta})}{p(\mathbf{y}_{1:T})} \tag{2.6}$$

where

$$p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T} | \boldsymbol{\theta}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_t | \mathbf{x}_{t-1}) p_{\boldsymbol{\theta}}(\mathbf{y}_t | \mathbf{x}_t) \tag{2.7}$$

is the joint likelihood of the SSM and

$$p(\mathbf{y}_{1:T}) = \int p(\boldsymbol{\theta}) p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T} | \boldsymbol{\theta}) d\boldsymbol{\theta} d\mathbf{x}_{1:T} \tag{2.8}$$

is the model evidence. The full posterior in Eq. (2.6) represents the belief of a

Bayesian agent about the states and parameters of the model given the observations. In practice however this posterior, in addition to being highly intractable, is not the quantity that is typically needed for most downstream tasks. Instead, inference usually focuses on marginal distributions of the states or the parameters which are more useful and more computationally accessible through approximate methods. Below we give a brief overview of the main inference tasks that commonly arise in most applications of SSMs.

Filtering

The goal of *Bayesian filtering* is to compute the marginal posterior distribution, called *filtering distribution* $p(\mathbf{x}_t|\mathbf{y}_{1:t}, \boldsymbol{\theta})$ of the present state $\mathbf{x}_t$, given past observations, $\mathbf{y}_{1:t} \equiv \{\mathbf{y}_1, \ldots, \mathbf{y}_t\}$ and parameters $\boldsymbol{\theta}$. Given the filtering distribution at time $t-1$ and the new observation $\mathbf{y}_t$ the filtering distribution at time t can be computed via the following equations.

Box 2.1: Bayesian filtering equations

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}, \boldsymbol{\theta}) d\mathbf{x}_{t-1}, \tag{2.9}$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}, \boldsymbol{\theta}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta})}{\int p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta}) d\mathbf{x}_t}. \tag{2.10}$$

Equation (2.9) is known as the *prediction* and Eq. (2.10) as the *update step* and together they are known as the Bayesian filtering equations (for a derivation see [SS23]). Starting from an initial distribution $p(\mathbf{x}_0)$ the filtering equations can be solved recursively in order to obtain the sequence of filtering distributions $\{p(\mathbf{x}_t|\mathbf{y}_{1:t}, \boldsymbol{\theta})\}_{t=1}^T$. Algorithms which provide solutions to the filtering equations are known as *Bayesian filters*.

For general nonlinear SSMs the integrals that appear in Eqs. (2.9)-(2.10) are intractable and have to be approximated[2]. There are many methods for approximating the solutions of the filtering equations, based on numerical integration, enabling approximations or Monte-Carlo methods [SS23]. In the next section we review the most popular filtering algorithms.

[2]The filtering equations can be solved exactly only for linear-Gaussian SSMs described below.

Smoothing

The goal of *Bayesian smoothing* is to compute the marginal posterior distribution of a past state $\mathbf{x}_t$ given all available observations $\mathbf{y}_{1:T}$ and model parameters $\boldsymbol{\theta}$. This distribution, known as the *smoothing distribution*, is denoted by $p(\mathbf{x}_t|\mathbf{y}_{1:T}, \boldsymbol{\theta})$ for $t < T$. Given the full sequence of filtering distributions $\{p(\mathbf{x}_t|\mathbf{y}_{1:t}, \boldsymbol{\theta})\}_{t=1}^{T}$, the smoothing distribution can be computed recursively in a backward pass which proceeds from $t = T - 1$ down to $t = 1$ using the following relation:

$$p(\mathbf{x}_t|\mathbf{y}_{1:T}, \boldsymbol{\theta}) = p(\mathbf{x}_t|\mathbf{y}_{1:t}, \boldsymbol{\theta}) \int \frac{p_{\boldsymbol{\theta}}(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{x}_{t+1}|\mathbf{y}_{1:T}, \boldsymbol{\theta})}{p(\mathbf{x}_{t+1}|\mathbf{y}_{1:t}, \boldsymbol{\theta})} d\mathbf{x}_{t+1}. \tag{2.11}$$

This recursion combines the forward filtering distribution at time t with the smoothed distribution at time $t + 1$, allowing efficient computation of the full sequence of smoothed state marginals.

Prediction

Alternatively, we may be interested in predicting future states or observations given the current information. In this task, the goal is to compute the *predictive distribution* of the future state $\mathbf{x}_{t+k}$ or future observation $\mathbf{y}_{t+k}$ given observations up to time t and parameters $\boldsymbol{\theta}$, denoted by $p(\mathbf{x}_{t+k}|\mathbf{y}_{1:t}, \boldsymbol{\theta})$ or $p(\mathbf{y}_{t+k}|\mathbf{y}_{1:t}, \boldsymbol{\theta})$, respectively. These distributions can be computed recursively using the transition and observation models. For state prediction, the k-step ahead predictive distribution is obtained by repeated application of the transition model:

$$p(\mathbf{x}_{t+k}|\mathbf{y}_{1:t}, \boldsymbol{\theta}) = \int \cdots \int \prod_{i=1}^{k} p_{\boldsymbol{\theta}}(\mathbf{x}_{t+i}|\mathbf{x}_{t+i-1}) p(\mathbf{x}_t|\mathbf{y}_{1:t}, \boldsymbol{\theta}) d\mathbf{x}_t \cdots d\mathbf{x}_{t+k-1}. \tag{2.12}$$

The predictive distribution of a future observation $\mathbf{y}_{t+k}$ then follows by marginalizing over the predicted state:

$$p(\mathbf{y}_{t+k}|\mathbf{y}_{1:t}, \boldsymbol{\theta}) = \int p_{\boldsymbol{\theta}}(\mathbf{y}_{t+k}|\mathbf{x}_{t+k}), p(\mathbf{x}_{t+k}|\mathbf{y}_{1:t}, \boldsymbol{\theta}), d\mathbf{x}_{t+k}. \tag{2.13}$$

These predictive distributions are essential for forecasting and decision-making tasks under uncertainty.

Parameter inference

The tasks that we have described above assume that the parameters of the model $\boldsymbol{\theta}$ are given, and focus on inference of the latent states. In most practical ap-

plications however there are many parameters that are not known apriori and have to be inferred from the observations. The goal of *parameter inference* is to compute the posterior distribution over the model parameters $\boldsymbol{\theta}$ given the full set of observations $\mathbf{y}_{1:T}$. This distribution captures our belief about $\boldsymbol{\theta}$ after observing the data and is denoted by $p(\boldsymbol{\theta}|\mathbf{y}_{1:T})$. Using Bayes' rule the parameter posterior is given by

$$p(\boldsymbol{\theta}|\mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y}_{1:T})}, \tag{2.14}$$

where $p(\boldsymbol{\theta})$ is the prior distribution over parameters, and $p(\mathbf{y}_{1:T}|\boldsymbol{\theta})$ is the likelihood function, given by

$$p(\mathbf{y}_{1:T}|\boldsymbol{\theta}) = \int p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}|\boldsymbol{\theta}) d\mathbf{x}_{1:T}. \tag{2.15}$$

To perform inference on the parameter vector $\boldsymbol{\theta}$, we need to be able to evaluate the likelihood for given data $\mathbf{y}_{1:T}$ and parameters $\boldsymbol{\theta}$, which is an integral with respect to the latent states of the model. The integral can be computed exactly for linear-Gaussian SSMs by the Kalman filter given in Alg. 1. For nonlinear and non-Gaussian SSMs it is intractable, requiring numerical approximations. A common approach is particle filtering described in the next section.

The SSM likelihood can be written as a product of autoregressive conditionals,

$$p(\mathbf{y}_{1:T}|\boldsymbol{\theta}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t-1}), \tag{2.16}$$

where each conditional $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ is an intractable integral given by

$$p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \int p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{x}_t) p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t-1}) p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1:t}. \tag{2.17}$$

In filtering, the likelihood factors $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t-1})$ are computed as normalization constants of the filtering distribution and the factorization of Eq. (2.16) is used to approximate the likelihood.

2.3 Forgetting property of SSMs

Many state-space models possess the *exponential forgetting property* [DD04] which states that, for any $\mathbf{x}_0, \mathbf{x}_0' \in \mathcal{X}$, there exist constants $B \in (0, \infty)$ and $\lambda \in [0, 1)$ such that

$$||p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}_{1:t}, \mathbf{x}_0) - p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}_{1:t}, \mathbf{x}_0')||_{TV} \leq B\lambda^t, \tag{2.25}$$

Algorithm 1 Kalman filter

Input: Initial mean $\boldsymbol{\mu}_0$, initial covariance $\boldsymbol{\Sigma}_0$, observations $\mathbf{y}_{1:T}$
for $t = 1$ to T **do**
// Prediction step

$$\boldsymbol{\mu}_t^- = \mathbf{A}_{t-1}\boldsymbol{\mu}_{t-1} \tag{2.18}$$
$$\boldsymbol{\Sigma}_t^- = \mathbf{A}_{t-1}\boldsymbol{\Sigma}_{t-1}\mathbf{A}_{t-1}^\top + \mathbf{Q}_{t-1} \tag{2.19}$$

// Update step

$$\boldsymbol{\mu}_{\mathbf{y},t} = \mathbf{H}_t\boldsymbol{\mu}_t^- \tag{2.20}$$
$$\boldsymbol{\Sigma}_{\mathbf{y},t} = \mathbf{H}_t\boldsymbol{\Sigma}_t^-\mathbf{H}_t^\top + \mathbf{R}_t \tag{2.21}$$
$$\mathbf{K}_t = \boldsymbol{\Sigma}_t^-\mathbf{H}_t^\top\boldsymbol{\Sigma}_{\mathbf{y},t}^{-1} \tag{2.22}$$
$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_t^- + \mathbf{K}_t(\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y},t}) \tag{2.23}$$
$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_t^- - \mathbf{K}_t\boldsymbol{\Sigma}_{\mathbf{y},t}\mathbf{K}_t^\top \tag{2.24}$$

end for
Output: Filtered means $\boldsymbol{\mu}_{1:T}$ and covariances $\boldsymbol{\Sigma}_{1:T}$

where $p_{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{y}_{1:t}, \mathbf{x}_0)$ is the optimal filtering distribution, i.e. the exact solution to the filtering equations (Box 2.1), at time t when initialized at $\mathbf{x}_0$, with parameters $\boldsymbol{\theta}$ and observed data $\mathbf{y}_{1:t}$. $||\cdot||_{TV}$ is the total variation distance. This property occurs when the state process is uniformly ergodic and the observations satisfy certain conditions that can be found in [Kan+15; DD04].

It is straightforward to show that the likelihood factors $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t-1})$, which depend on the filtering distributions via Eq. (2.17), also satisfy this property: there exist constants $B \in (0, \infty)$ and $\lambda \in [0, 1)$ such that

$$||p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t}, \mathbf{x}_0) - p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t}, \mathbf{x}_0')||_{TV} \leq B\lambda^{t-1}. \tag{2.26}$$

Intuitively this property means that the optimal filter forgets its initial condition exponentially fast. Equivalently this means that past observations are forgotten fast.

In Chapter 4 we exploit this property to justify a truncation of the factors $p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t-1})$, whereby we approximate them by a density that conditions only on the L most recent observations

$$p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{1:t-1}) \approx p_{\boldsymbol{\theta}}(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}). \tag{2.27}$$

The logic is that if the forgetting property holds, and past observations are forgotten very fast, we may postulate that there exists only a window of past obser-

vations that influence the current one, and that observations outside the window are forgotten. We give more details on this approximation in Chapter 4.

2.4 Filters

Filtering is one of the core inference tasks in state-space models, concerned with computing the distribution of the current latent state given all observations up to the present. This section reviews two main computational approaches to filtering that are used throughout this thesis: the Kalman filter (and its variants EKF, UKF), which provides an exact solution for linear-Gaussian state-space models, and particle filters, a widely used class of methods for nonlinear and non-Gaussian systems. For the rest of the section we suppress explicit dependence on the parameters $\boldsymbol{\theta}$, which are held fixed. Additional details on Gaussian and Gaussian sum filters, which form the basis of the AGSF method introduced in the next chapter, are provided therein.

2.4.1 Kalman filter

A special case of the general state-space model that we have introduced is the linear-Gaussian state-space model (LGSSM), defined by

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{q}_t, \tag{2.28}$$

$$\mathbf{y}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{r}_t, \tag{2.29}$$

where $\mathbf{A}_t$ and $\mathbf{H}_t$ are sequences of real matrices, and the noise vectors follow $\mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ and $\mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$. Because both the dynamics and observation models are linear and all noise is Gaussian, the resulting joint, marginal, and conditional distributions over states and observations remain Gaussian at every time step [SS23]. LGSSMs are widely used in applications such as control systems, finance, signal processing, and navigation, due to their computational efficiency and analytical tractability [SS23].

The Kalman filter provides an exact, recursive solution to the Bayesian filtering problem in LGSSMs. Introduced in [Kal60], it updates the posterior distribution over the latent state $\mathbf{x}_t$ as new observations $\mathbf{y}_t$ arrive, using closed-form expressions for the predictive and updated mean and covariance. At each time t, the filtering distribution takes the form

$$p(\mathbf{x}_t | \mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \tag{2.30}$$

where $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ are the posterior mean and covariance. They are computed via the Kalman filter recursion, given the mean and covariance at the previous time step. The Kalman filter recurstion consists of two steps: the *prediction step* which computes the predictive distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ given the previous filtering distribution and the *update step* which does the Bayesian update from the predictive distribution to the new filtering distribution. The Kalman filter algorithm with its prediction and update steps is given in Alg. 1.

In addition to computing filtering distributions, the Kalman filter yields the exact value of the marginal likelihood of the observations $p(\mathbf{y}_{1:T}|\boldsymbol{\theta})$, which is useful for model comparison and parameter inference. Since the predictive distribution of $\mathbf{y}_t$ given past observations is Gaussian, the log-likelihood can be computed incrementally as

$$\log p(\mathbf{y}_{1:T}|\boldsymbol{\theta}) = \sum_{t=1}^{T} \log \mathcal{N}(\mathbf{y}_t|\boldsymbol{\mu}_{\mathbf{y},t}, \boldsymbol{\Sigma}_{\mathbf{y},t}), \tag{2.31}$$

where $\boldsymbol{\mu}_{\mathbf{y},t}$ and $\boldsymbol{\Sigma}_{\mathbf{y},t}$ are computed by the Kalman filter, Eqs. (2.20)-(2.21). For a derivation of the Kalman filter and further details, see [SS23].

2.4.2 Extended Kalman filter

The extended Kalman filter (EKF) [SSM62] is an extension of the Kalman filter for nonlinear SSMs. The EKF at every iteration replaces the nonlinear functions $\mathbf{f}$ and $\mathbf{g}$ given in Eqs. (2.4)-(2.5) by their first-order Taylor expansion around the mean of the filtering or predictive distribution. In this way the nonlinear SSM is approximated by a LGSSM at every timestep on which we can apply the Kalman filter. The EKF algorithm is given in Alg. 2. More details on the linear approximation that underlies the EKF are given in the Chapter 3.

2.4.3 Unscented Kalman filter

The unscented Kalman filter (UKF) [WV01; JU04b], given in Alg. 3, was proposed to mitigate the drawbacks of the EKF, in particular its reliance on the computation of Jacobians, which can lead to significant numerical errors when dealing with highly nonlinear systems. The UKF uses an implicit linearization scheme known as the unscented transform (UT) [JU96]. The UT is a deterministic approximation and does not rely on differentiation, making it more robust in many applications where derivatives are not well defined or very noisy. Details on the unscented transform and its application to filtering are given in Chapter

Algorithm 2 Extended Kalman filter

Input: Initial mean $\boldsymbol{\mu}_0$, covariance $\boldsymbol{\Sigma}_0$, observations $\mathbf{y}_{1:T}$
for $t = 1$ to T **do**
// Prediction step

$$\boldsymbol{\mu}_t^- = \mathbf{f}(\boldsymbol{\mu}_{t-1}) \tag{2.32}$$
$$\boldsymbol{\Sigma}_t^- = \nabla \mathbf{f}(\boldsymbol{\mu}_{t-1}) \boldsymbol{\Sigma}_{t-1} \nabla \mathbf{f}(\boldsymbol{\mu}_{t-1})^T + \mathbf{Q} \tag{2.33}$$

// Update step

$$\boldsymbol{\mu}_{\mathbf{y}} = \mathbf{g}(\boldsymbol{\mu}_t^-) \tag{2.34}$$
$$\boldsymbol{\Sigma}_{\mathbf{y},t} = \nabla \mathbf{g}(\boldsymbol{\mu}_t^-) \boldsymbol{\Sigma}_t^- \nabla \mathbf{g}(\boldsymbol{\mu}_t^-)^T + \mathbf{R} \tag{2.35}$$
$$\mathbf{K}_t = \boldsymbol{\Sigma}_t^- \nabla \mathbf{g}(\boldsymbol{\mu}_t^-)^T \boldsymbol{\Sigma}_{\mathbf{y},t}^{-1} \tag{2.36}$$
$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_t^- + \mathbf{K}_t(\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y},t}) \tag{2.37}$$
$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_t^- - \mathbf{K}_t \boldsymbol{\Sigma}_{\mathbf{y},t} \mathbf{K}_t^T \tag{2.38}$$

end for
Output: Filtered means $\boldsymbol{\mu}_{1:T}$ and covariances $\boldsymbol{\Sigma}_{1:T}$

3.

2.4.4 Particle filters

Particle filters (PF) [GSS93; Kit96; DDG+01], also known as sequential Monte Carlo (SMC) methods [CP+20], are a class of Bayesian filtering algorithms that were developed to tackle the non-linear filtering problem when Gaussian approximations are not applicable. Unlike Gaussian filters which rely on distributional assumptions, particle filters approximate the filtering distribution using a set of weighted particles which are propagated through the nonlinearities and their weights updated via importance sampling [Elv+19b]. The general PF algorithm, also known as sequential importance resampling (SIR), is given in Table 2.5.

Particle filters also provide us with an estimate of the model likelihood, given by

$$\widehat{p}(\mathbf{y}_{1:T}|\boldsymbol{\theta}) = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_t^{(i)}, \tag{2.39}$$

where $\tilde{w}_t^{(i)}$ are the unnormalized weights defined in the SIR algorithm of Table 2.5 (see [DJ09] for a derivation).

Algorithm 3 Unscented Kalman filter

Input: Initial mean $\boldsymbol{\mu}_0$, covariance $\boldsymbol{\Sigma}_0$, observations $\mathbf{y}_{1:T}$
for $t = 1$ to T **do**
// Prediction step
1. Form the sigma points:

$$\begin{aligned}\boldsymbol{\sigma}_{t-1}^{(0)} &= \boldsymbol{\mu}_{t-1}, \\ \boldsymbol{\sigma}_{t-1}^{(\pm i)} &= \boldsymbol{\mu}_{t-1} + \sqrt{d_x + \lambda}\,[\sqrt{\boldsymbol{\Sigma}_{t-1}}]_i, \quad i = 1, \ldots, d_x,\end{aligned}$$

2. Propagate the sigma points through the dynamics:

$$\hat{\boldsymbol{\sigma}}_t^{(i)} = \mathbf{f}(\boldsymbol{\sigma}_{t-1}^{(i)}), \quad i = -d_x, \ldots, d_x$$

3. Compute predicted mean and covariance:

$$\begin{aligned}\boldsymbol{\mu}_t^- &= \sum_{i=-d_x}^{d_x} \omega_i \hat{\boldsymbol{\sigma}}_t^{(i)}, \\ \boldsymbol{\Sigma}_t^- &= \sum_{i=-d_x}^{d_x} \tilde{\omega}_i (\hat{\boldsymbol{\sigma}}_t^{(i)} - \boldsymbol{\mu}_t^-)(\hat{\boldsymbol{\sigma}}_t^{(i)} - \boldsymbol{\mu}_t^-)^\top + \mathbf{Q}\end{aligned}$$

// Update step
1. Form the sigma points from predicted mean and covariance:

$$\begin{aligned}\boldsymbol{\sigma}_t^{-(0)} &= \boldsymbol{\mu}_t^-, \\ \boldsymbol{\sigma}_t^{-(\pm i)} &= \boldsymbol{\mu}_t^- + \sqrt{d_x + \lambda}\,[\sqrt{\boldsymbol{\Sigma}_t^-}]_i, \quad i = 1, \ldots, d_x,\end{aligned}$$

2. Propagate the sigma points through the measurement model:

$$\hat{\mathbf{y}}_t^{(i)} = \mathbf{g}(\boldsymbol{\sigma}_t^{-(i)}), \quad i = -d_x, \ldots, d_x$$

3. Compute predicted observation, innovation covariance, and cross-covariance:

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{y},t} &= \sum_{i=-d_x}^{d_x} \omega_i \hat{\mathbf{y}}_t^{(i)}, \\ \boldsymbol{\Sigma}_{\mathbf{y},t} &= \sum_{i=-d_x}^{d_x} \tilde{\omega}_i (\hat{\mathbf{y}}_t^{(i)} - \boldsymbol{\mu}_{\mathbf{y},t})(\hat{\mathbf{y}}_t^{(i)} - \boldsymbol{\mu}_{\mathbf{y},t})^\top + \mathbf{R}, \\ \mathbf{C}_{\mathbf{xy},t} &= \sum_{i=-d_x}^{d_x} \tilde{\omega}_i (\boldsymbol{\sigma}_t^{(i)} - \boldsymbol{\mu}_t^-)(\hat{\mathbf{y}}_t^{(i)} - \boldsymbol{\mu}_{\mathbf{y},t})^\top\end{aligned}$$

4. Compute Kalman gain, updated mean and covariance:

$$\begin{aligned}\mathbf{K}_t &= \mathbf{C}_{\mathbf{xy},t} \boldsymbol{\Sigma}_{\mathbf{y},t}^{-1}, \\ \boldsymbol{\mu}_t &= \boldsymbol{\mu}_t^- + \mathbf{K}_t(\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y},t}), \\ \boldsymbol{\Sigma}_t &= \boldsymbol{\Sigma}_t^- - \mathbf{K}_t \boldsymbol{\Sigma}_{\mathbf{y},t} \mathbf{K}_t^\top\end{aligned}$$

end for
Output: Filtered means $\boldsymbol{\mu}_{1:T}$ and covariances $\boldsymbol{\Sigma}_{1:T}$

Bootstrap Particle Filter (BPF)

The Bootstrap Particle Filter (BPF) [GSS93] is the simplest particle filter. It is a special case of the SIR algorithm which uses the dynamics model as proposal distribution,

$$\pi(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{2.40}$$

This simplifies the weights update which becomes

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} \cdot p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}). \tag{2.41}$$

The BPF algorithm is given in Table 2.5.

Auxiliary Particle Filter (APF)

The Auxiliary Particle Filter [PS99] improves upon the BPF by incorporating a one-step look-ahead mechanism using the observation likelihood. It introduces an auxiliary variable to guide the resampling step with predictive weights. This reduces weight degeneracy and improves sample efficiency. The APF is given in Table 2.5. More details about the algorithm can be found in [Elv+19a].

Classical approaches to parameter inference in SSMs such as the EM algorithm and MCMC are likelihood-based. They rely on filters for likelihood evaluation, which can provide a noisy estimate in $O(T)$.

2.5 Parameter inference algorithms

In addition to inferring the hidden states of a state-space model, most applications require learning some or all of the model parameters that make up the vector $\boldsymbol{\theta}$ from observed data. This task, known as parameter inference, is generally more challenging than state inference. In the Bayesian framework, parameter inference corresponds to approximating the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y}_{1:T})$ or sampling from it. Thus, their accuracy is fundamentally limited by the quality of the underlying filtering algorithm. This is not the case for simulation-based methods such as approximate Bayesian computation (ABC) and neural-based methods, which bypass likelihood evaluation altogether by relying on model simulations.

In this section we review two broad classes of methods for parameter inference in SSMs that are used in this thesis: particle Markov chain Monte Carlo (MCMC) methods, and simulation-based inference (SBI) methods such as SMC-ABC. Discussion on neural-based SBI methods is postponed until Chapter 4.

SIR Particle Filter

Input: N, $\mathbf{y}_{1:T}$
for $i = 1$ to N **do**
  Sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$
  $w_0^{(i)} = \frac{1}{N}$
end for
for $t = 1$ to T **do**
  for $i = 1$ to N **do**
    Sample $\mathbf{x}_t^{(i)} \sim \pi(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t})$
    $\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \cdot \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)},\mathbf{y}_{1:t})}$
  end for
  $w_t^{(i)} = \tilde{w}_t^{(i)} / \sum_j \tilde{w}_t^{(j)}$
  if $N_{\text{eff}} <$ threshold **then**
    Resample $\mathbf{x}_t^{(i)}$, set $w_t^{(i)} = \frac{1}{N}$
  end if
end for
Output: $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$

Bootstrap Particle Filter

Input: N, $\mathbf{y}_{1:T}$
for $i = 1$ to N **do**
  Sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$
  $w_0^{(i)} = \frac{1}{N}$
end for
for $t = 1$ to T **do**
  for $i = 1$ to N **do**
    Sample $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$
    $\tilde{w}_t^{(i)} = p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$
  end for
  $w_t^{(i)} = \tilde{w}_t^{(i)} / \sum_j \tilde{w}_t^{(j)}$
  if $N_{\text{eff}} <$ threshold **then**
    Resample $\mathbf{x}_t^{(i)}$, set $w_t^{(i)} = \frac{1}{N}$
  end if
end for
Output: $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$

Auxiliary Particle Filter

Input: N, $\mathbf{y}_{1:T}$
for $i = 1$ to N **do**
  Sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$
  $w_0^{(i)} = \frac{1}{N}$
end for
for $t = 1$ to T **do**
  for $i = 1$ to N **do**
    $\tilde{w}_{t-1}^{(i)} = w_{t-1}^{(i)} \cdot p(\mathbf{y}_t \mid \hat{\mathbf{x}}_t^{(i)})$
  end for
  Normalize $\tilde{w}$ to get $\bar{w}$
  Resample $a_t^{(i)} \sim \bar{w}$
  for $i = 1$ to N **do**
    Sample $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(a_t^{(i)})})$
    $\tilde{w}_t^{(i)} = \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})}{p(\mathbf{y}_t|\hat{\mathbf{x}}_t^{(a_t^{(i)})})}$
  end for
  $w_t^{(i)} = \tilde{w}_t^{(i)} / \sum_j \tilde{w}_t^{(j)}$
end for
Output: $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}$

Table 2.1: SIR, BPF and APF algorithms

2.5.1 Markov chain Monte Carlo

Markov chain Monte Carlo (MCMC) methods are perhaps the most widely used algorithms for sampling from a target distribution [RC99; RC11]. The key idea behind MCMC is to construct a Markov chain whose stationary distribution is the target, so that, after a sufficient number of iterations, samples from the chain can be treated as approximate samples from the target. This can be done by finding a kernel for the Markov chain which satisfies a detailed balance condition with the target distribution.

An advantage of MCMC which makes it applicable to Bayesian inference is that we only need to know the target up to a normalization constant. The Bayesian posterior of a SSM is proportional to

$$p(\boldsymbol{\theta}|\mathbf{y}_{1:T}) \propto p(\boldsymbol{\theta}|\mathbf{y}_{1:T})p(\boldsymbol{\theta}). \tag{2.42}$$

This is convenient because we don't have to evaluate the marginal likelihood $p(\mathbf{y}_{1:T})$ which is an intractable integral over $\boldsymbol{\theta}$. This makes MCMC a general-purpose Bayesian inference tool.

The earliest and simplest way to ensure detailed balance is that of the Metropolis-Hastings (MH) algorithm [Met+53], which works as follows: given a current sample $\boldsymbol{\theta}^{(i)}$, the algorithm proposes a new candidate $\boldsymbol{\theta}'$ from a proposal distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i)})$, and accepts it with probability

$$\alpha = \min\left(1, \frac{p(\mathbf{y}_{1:T}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')q(\boldsymbol{\theta}^{(i)}|\boldsymbol{\theta}')}{p(\mathbf{y}_{1:T}|\boldsymbol{\theta}^{(i)})p(\boldsymbol{\theta}^{(i)})q(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i)})}\right). \tag{2.43}$$

If the proposal is accepted, the next state of the chain is set to $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}'$; otherwise, it remains at the current value, $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$. This simple acceptance-rejection rule ensures that the resulting Markov chain satisfies detailed balance and converges to the correct posterior distribution [RC99].

While the MH algorithm is conceptually simple and widely applicable, its performance depends critically on the choice of the proposal distribution. A poorly tuned proposal may result in low acceptance rates or slow exploration of the parameter space, leading to highly correlated samples and slow convergence. These limitations are especially pronounced in high-dimensional problems, where naive proposals fail to capture the geometry of the posterior.

To improve sampling efficiency, more advanced MCMC methods have been developed. One widely used class of such methods are gradient-based samplers, including Hamiltonian Monte Carlo (HMC) [Dua+87], which exploit gradient

information to propose moves informed by the local curvature of the posterior. HMC is particularly effective in high-dimensional and strongly correlated posteriors, as it can make distant proposals with high acceptance rates by simulating Hamiltonian dynamics in an augmented space.

In settings where the posterior is approximately Gaussian, or when the prior is Gaussian and the likelihood is weakly informative, the elliptical slice sampler (ESS) offers an attractive alternative [MAM10]. ESS is a parameter-free, automatically adaptive MCMC method that does not require gradients and often mixes well in problems where standard MH fails. It constructs proposals by slicing through an elliptical contour defined by the prior, and can be especially effective for models with Gaussian priors and non-conjugate likelihoods.

Overall, MCMC methods remain a foundational tool for Bayesian inference. In the context of state-space models, their practical use is hindered by the need to evaluate the marginal likelihood $p(\mathbf{y}_{1:T}|\boldsymbol{\theta})$, which is intractable in nonlinear and non-Gaussian models. The next subsection introduces particle-marginal methods which use particle filtering to approximate the marginal likelihood within MCMC.

Particle-marginal Metropolis-Hastings (PMMH)

For state-space models where the likelihood is intractable, we can approximate $p(\mathbf{y}_{1:T}|\boldsymbol{\theta})$ with a particle filter using Eq. (2.39), yielding the so-called particle-marginal MH algorithm [ADH10]. This results in Alg. 4 given below.

Algorithm 4 Particle-marginal Metropolis-Hastings (PMMH)

1: **Input:** Initial parameter $\boldsymbol{\theta}^{(0)}$, number of iterations N, proposal distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta})$
2: Evaluate $\hat{p}(\mathbf{y}_{1:T}|\boldsymbol{\theta}^{(0)})$ using a particle filter
3: **for** $i = 1$ to N **do**
4: Propose $\boldsymbol{\theta}' \sim q(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i-1)})$
5: Evaluate $\hat{p}(\mathbf{y}_{1:T}|\boldsymbol{\theta}')$ using a particle filter
6: Compute acceptance probability:

$$\alpha = \min\left(1, \frac{\hat{p}(\mathbf{y}_{1:T}|\boldsymbol{\theta}')\, p(\boldsymbol{\theta}')\, q(\boldsymbol{\theta}^{(i-1)}|\boldsymbol{\theta}')}{\hat{p}(\mathbf{y}_{1:T}|\boldsymbol{\theta}^{(i-1)})\, p(\boldsymbol{\theta}^{(i-1)})\, q(\boldsymbol{\theta}'|\boldsymbol{\theta}^{(i-1)})}\right)$$

7: With probability α, set $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}'$, else $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)}$
8: **end for**
9: **Output:** Samples $\{\boldsymbol{\theta}^{(i)}\}_{i=1}^{N}$

The performance of PMMH depends strongly on the variance of the likelihood estimator. Too few particles lead to noisy estimates and poor acceptance, while too many particles increase computational cost. Careful tuning of the number of particles is thus crucial in practice.

2.5.2 Simulation-based inference and ABC

More recently, a distinct approach on the Bayesian inference problem has become prominent which does not rely on explicit likelihood computations [Rub84]. In this approach, called simulation-based inference (SBI), instead of trying to evaluate the likelihood directly, e.g. by approximating an integral like the one in Eq. (2.15), we rely on multiple simulations from the model, which are used to obtain a surrogate of the likelihood.

The earliest SBI methods to be introduced are the so-called approximate Bayesian computation (ABC) methods [Tav+97; SFB18]. ABC methods propose parameter values repeatedly and simulate the model to generate synthetic data vectors. In the simplest version of ABC, known as *rejection ABC*, a proposed parameter is sampled from the model prior and is used to simulate a synthetic dataset. The distance between the synthetic dataset and the observations is quantified using an appropriate metric. If the distance is larger than a user-set tolerance the proposed parameter value is rejected. In this way we only accept parameters that produce datasets *close* to the observations. If $\boldsymbol{\theta}$ and $\mathbf{y}_o$ are the parameter and observation vectors, respectively, and the tolerance is set to ϵ, the rejection ABC sampler works by repeating the steps:

1. sample candidate: $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$,

2. simulate dataset: $\mathbf{y} \sim p(\mathbf{y}|\boldsymbol{\theta})$,

3. accept $\boldsymbol{\theta}$ if $d(\mathbf{y}, \mathbf{y}_o) \leq \epsilon$, else reject and repeat

until the required sample sized has been reached. In this way, the ABC rejection sampler produces samples from an approximation of the posterior, which depends on the tolerance ϵ. Decreasing the tolerance improves the approximation and in fact ABC methods have been shown to asymptotically converge to the true posterior as the tolerance goes to zero [Fra+18].

SMC-ABC

In practice a very small ϵ will lead to a vanishing acceptance rate, requiring an ever-increasing number of simulations to achieve a good posterior approximation. For rejection ABC, this problem is particularly pronounced since the proposal mechanism is fixed. Since the acceptance rate depends on the proposal mechanism as well as ϵ, improved algorithms adapt the proposal to achieve a non-vanishing acceptance rate while decreasing the tolerance. Notably, the ABC-MCMC [SFT07; Mar+03] algorithm uses the ABC sampler within an MCMC proposal kernel to perform likelihood-free MCMC, and SMC-ABC [Bea+09], uses SMC to target intermediate ABC-posteriors for a decreasing tolerance sequence while adapting the proposal to achieve a non-vanishing acceptance rate. In this work we focus on SMC-ABC, given in Alg. 5, which we take as a representative of ABC-type methods, since it has shown strong performance in the SSM inference problem [Ton+09].

Algorithm 5 SMC-ABC

Initialize tolerances $\epsilon_1, \dots, \epsilon_R$
while $r < R$ **do**
  for $i = 1$ to N **do**
    repeat
      if $r = 0$ **then**
        Sample $\boldsymbol{\theta}^{**} \sim p(\boldsymbol{\theta})$
      else
        Sample $\boldsymbol{\theta}^{*}$ from previous round with weights w_{r-1}
        Propose $\boldsymbol{\theta}^{**} \sim K_t(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{*})$
        if $p(\boldsymbol{\theta}^{**}) = 0$ **then**
          Continue
        end if
      end if
      Simulate $\mathbf{y}^{*}_{1:T} \sim p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta}^{**})$
    until $d(\mathbf{y}^{*}_{1:T}, \mathbf{y}^{(obs)}_{1:T}) < \epsilon_r$
    Set $\boldsymbol{\theta}^{(i)}_r \leftarrow \boldsymbol{\theta}^{**}$
    Compute weight:

$$w^{(i)}_r \leftarrow \begin{cases} 1, & \text{if } r = 0 \\ \dfrac{\pi(\boldsymbol{\theta}^{(i)}_r)}{\sum_{j=1}^{N} w^{(j)}_{r-1} K_t(\boldsymbol{\theta}^{(i)}_r \mid \boldsymbol{\theta}^{(j)}_{r-1})}, & \text{if} r > 0 \end{cases}$$

  end for
  Normalize weights $\{w^{(i)}_t\}_{i=1}^{N}$
  $r \leftarrow r + 1$
end while

Chapter 3

Augmented Gaussian sum filters

3.1 Introduction

State estimation, or filtering, is the problem of sequentially inferring the hidden states of a SSM given a record of observations. In the Bayesian setting, this problem is equivalent to obtaining the state posteriors, also known as the filtering distributions [SS23]. The filtering distributions satisfy a system of integral equations, the so-called filtering equations, given in Box 2.1. The filtering equations are not exactly solvable, except for the linear-Gaussian SSM in which dynamics and observation processes are linear transformations with additive Gaussian noise. In the LGSSM the filtering distributions are Gaussian distributions with means and covariance computed by the celebrated Kalman filter [Kal60; KB61].

When the models are nonlinear or the noises are non-Gaussian solutions have to be obtained by numerical approximations. There are various strategies for approximately solving the filtering equations, each relying on different assumptions about the models and the target distributions. One of the earliest approaches is the EKF [SSM62] which was introduced in the previous chapter. The EKF belongs to a larger class of algorithms known as *Gaussian filters* which also includes the UKF [WV00b], the *quadrature* and *cubature* Kalman filters [AHE07; IX00b; AH09]. Gaussian filters assume that the filtering distributions are Gaussian, and use moment-matching approximations to obtain the sequence of posteriors [SS23, Chapter 8]. Gaussian filters effectively linearize the problem resulting in algorithms that bear close resemblance to the Kalman filter (Alg.1). A set of Gaussian filters can be used in parallel to form a Gaussian mixture approximation of the filtering distributions in a family of algorithms known as *Gaussian sum filters* (GSF) [AS72a]. Gaussian sum filters are universal approximators, in the sense that in the limit of infinite components can approximate any probability

density function [SA71a].

Although Gaussian filters and GSFs have been applied with success to various fields for more than six decades[1], they also suffer from limitations which have hindered their applicability in many problems [Psi13]. One major limitation is the numerical instability of Gaussian filters (and GSFs) when applied to highly nonlinear systems. This instability often occurs for highly nonlinear systems as covariance matrices may become ill-conditioned, leading to numerically ill-posed matrix inversion problems which in turn lead to filter divergence. Moreover, since Gaussian filters effectively linearize the model, they benefit when the covariance matrices are small compared to the characteristic scales of the nonlinearities of the model. Practitioners have long sought of ways to limit the size of covariance matrices [SA71a].

For these reasons, a number of works have proposed methods which try to control the covariances of Gaussian components thus limiting the error due to ill-conditioning or covariance size. In [DBJ13], the authors use an entropy criterion to detect the nonlinearity and trigger a splitting of the component during the prediction step of continuous-time dynamical systems. In [FMK09] and [FK10], the authors propose a splitting scheme for models in which the dynamics and observation noise are expressed as Gaussian mixtures. The algorithm in [VR16] is focused on estimating the collision probability between tracked space object and uses the Gaussian splitting scheme of [DBJ13]. The algorithm in [HC13] develops a tracking algorithm based on a splitting scheme for application to autonomous robots. In [TZ18] the authors propose a Gaussian splitting scheme that can be used in the measurement update of a filter. Other recent work tries to control the mixture covariances by imposing *linear matrix inequalities* [Psi16; PSM15]. This approach requires the complicated machinery of matrix inequalities which is cumbersome to implement.

The limitations of the EKF and Gaussian filters more generally have led researchers to develop another approach that is based on particle approximations and Monte-Carlo and does not suffer from the limitations that come with Gaussian assumptions and linearization. This approach known as particle filtering, was reviewed in the previous chapter, and has been used widely for over two decades with a lot of success [GSS93; DDG+01]. Particle filters are sequential Monte-Carlo methods [CP+20] which approximate the filtering distribution us-

[1]It is historically interesting to note that the EKF was extensively used during the Apollo program. In fact EKF-enabled navigation was essential for guidance during Apollo 11's lunar landing `https://www.nasa.gov/wp-content/uploads/2015/04/techbul_20-03-nav_filter_042920.pdf`

ing an ensemble of particles that is propagated and weighted using importance sampling [Elv+19b; EM21]. They are a flexible class of algorithms which make few assumptions on the nature of the SSM and are relatively easy to implement [DJ09]. Moreover they enjoy strong theoretical convergence properties [CD02]. This has earned them success in multiple applications such as robotics [Thr02a], finance [LT11], and tracking [Gus+02] among others.

While particle filters offer flexibility for nonlinear filtering their main drawback is computational complexity since a large number of particles is needed to ensure good performance. This is especially pronounced in high dimensions due to the curse of dimensionality which often makes them impractical since the number of particles that are needed scales exponentially with dimension [DH03; PSM15]. This is in contrast to GSFs which can represent complex distributions with a small number of Gaussian components, albeit having their own drawbacks as we discussed.

Contributions: In this work, we introduce a novel way of controlling the covariance of Gaussian sum filters. We do so by exploiting variable augmentation and a convolution identity to split a Gaussian distribution into narrow components. In this way we introduce a novel Gaussian splitting scheme, which is very flexible, and gives the user control over filter covariances, which can be tuned to reduce the linearization errors in Gaussian filters. In this way we arrive at the *augmented Gaussian sum filter* (AGSF), a novel framework which generalizes GSFs and PFs into a unified filtering framework. This innovative approach blends the features of deterministic Gaussian filters and stochastic particle filters to exploit the trade-off between bias and variance in state estimation, resulting in an algorithm that effectively generalizes the two classes. The AGSF's adaptability is further enhanced by an optimization problem that automatically sets the augmentation covariances, allowing it to dynamically switch between Gaussian and particle filter behaviours based on the specific nonlinearities of each situation. This adaptability allows the AGSF to be very robust in a large variety of model behaviours. We reckon that for this reason the AGSF can be successful in many realistic scenarios such as those arising in robotics and target tracking applications.

Our work provides an elegant solution to a long-standing problem of Gaussian sum filtering, namely the control of covariances for the purpose of stabilizing the filter numerically and reducing the linearization errors. Although there are many works with a similar motivation [Psi16]-[PSM15], the contributions of this work are quite distinct. Firstly, the splitting scheme that is developed in this work is a very flexible and efficient way to split a Gaussian distribution. Its

advantages, compared to the schemes developed in the works [Psi16]-[PSM15] are its simplicity and flexibility, and we anticipate that it will find uses outside the context of nonlinear filtering. Furthermore, the splitting scheme allows us to unify the GSFs and the BPF in a common framework. The unification of the GSF and BPF algorithms allows us to propose a novel adaptive AGSF algorithm.

The main contributions are summarized as follows:

- We propose a novel splitting scheme by reinterpreting the well known Gaussian convolution identity. The scheme allows us to express a Gaussian distribution as a Gaussian mixture with narrow components, and do so in an efficient and flexible manner, compared to existing splitting schemes.

- We propose a novel class of filtering algorithms called augmented Gaussian sum filters which unifies the classes of Gaussian and particle filters. The AGSF uses a novel augmented Gaussian approximation method based on a Gaussian integral identity.

- We derive an adaptive version of the AGSF which can function like a particle filter, a Gaussian filter, or something in between, according to the local features of the nonlinearities. The adaptive version uses a novel optimization problem which is used to automatically select the augmentation covariance.

- We present simulation results showing the improved robustness of the AGSF compared to the bootstrap particle filter and GSFs. We also demonstrate the performance of the adaptive version of the AGSF in models possessing linear and strong nonlinear behaviour.

The rest of the chapter is structured as follows. Section 3.2 summarizes the general Gaussian, and Gaussian sum, filtering algorithm as well as EKF and UKF as special cases. In Section 3.3, we introduce the basic Gaussian augmentation and the augmented Gaussian approximations that it underpins. In Section 3.3.3, we introduce the optimization problem that is used to select the covariances automatically. In Section 3.4, we derive the novel AGSF algorithm and prove that it unifies the GSF and the BPF. Finally in Section 3.5 we present our experimental results comparing the AGSF to GSF and BPF.

3.2 Background

3.2.1 Notation

We denote by $\mathcal{N}(\cdot|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the multivariate Gaussian probability density function (PDF) with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. For any function $\boldsymbol{\phi}(\mathbf{x})$, the notation $\mathcal{I}[\cdot; \mathbf{z}, \boldsymbol{\Delta}]$ denotes the Gaussian integral

$$\mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathbf{z}, \boldsymbol{\Delta}] = \int \boldsymbol{\phi}(\mathbf{x}) \mathcal{N}(\mathbf{x}|\mathbf{z}, \boldsymbol{\Delta}) d\mathbf{x}.$$

We denote by $\mathbf{z}_{mn}$ and $\mathbf{s}_{mnl}$ the auxiliary variables introduced in the prediction and update steps of the AGSF algorithm, respectively. We also denote by $\boldsymbol{\Delta}_m^{(t)}$ and $\boldsymbol{\Lambda}_{mn}^{(t)}$ the augmentation covariances introduced at time t, for the prediction and update steps of the AGSF, respectively. Finally, we denote $a \wedge b = \min(a, b)$.

3.2.2 General Gaussian filtering

Gaussian filters are a class of filtering algorithms which make a Gaussian assumption on the filtering distributions and solve the filtering equations implicitly using moment matching [IX00a; SS23]. In what follows we describe the general Gaussian moment matching approximation and apply it to the filtering problem to obtain the general Gaussian filter. We also describe two versions of the Gaussian moment matching approximation that underly the EKF and the UKF that we have introduced in Chapter 2.

Gaussian moment matching

Gaussian moment matching (GMM) is a general technique for approximating the joint distribution of a pair of random variables by a Gaussian density which has first and second moments equal to those of the joint the we are trying to approximate. The *Gaussian moment matching of an additive transform* is given in Box 3.1.

Non-additive models. The moment matching approximation given in Box 3.1 can be applied to non-additive transformations of the form

$$\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{r}). \tag{3.1}$$

This generalization is straightforward and involves treating the variable $\tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{r})$ jointly, as the new state variable. The algorithm for non-additive models can be found in [SS23, Algorithm 8.2]. For notational clarity we only consider additive

models for the rest of the chapter and refer the reader to [SS23, Chapter 8] for details on Gaussian filters for non-additive models.

Box 3.1: Gaussian moment matching of an additive transform

Let $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^{d'}$ be random variables defined by

$$\mathbf{x} \sim \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}), \tag{3.2}$$
$$\mathbf{y} = \mathbf{f}(\mathbf{x}) + \mathbf{r}, \tag{3.3}$$

where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^{d'}$ and $\mathbf{r} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. The joint density of $(\mathbf{x}, \mathbf{y})$ is given by

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{f}(\mathbf{x}), \mathbf{R})\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}). \tag{3.4}$$

The GMM approximation of $p(\mathbf{x}, \mathbf{y})$ is the Gaussian density

$$p_{\text{GMM}}(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \middle| \begin{pmatrix} \boldsymbol{\mu}_\mathbf{x} \\ \boldsymbol{\mu}_\mathbf{y} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_\mathbf{x} & \mathbf{C}_{\mathbf{xy}} \\ \mathbf{C}_{\mathbf{xy}}^T & \boldsymbol{\Sigma}_\mathbf{y} \end{pmatrix} \right), \tag{3.5}$$

where,

$$\boldsymbol{\mu}_\mathbf{y} = \int \mathbf{f}(\mathbf{x})\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x})d\mathbf{x}, \tag{3.6}$$
$$\boldsymbol{\Sigma}_\mathbf{y} = \int (\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_\mathbf{y})(\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_\mathbf{y})^T\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x})d\mathbf{x} + \mathbf{R}, \tag{3.7}$$
$$\mathbf{C}_{\mathbf{xy}} = \int (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})(\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_\mathbf{y})^T\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x})d\mathbf{x}. \tag{3.8}$$

Linear and unscented approximations

In order to apply the Gaussian approximation to an additive transform (Box 3.1) to practical problems, we need to approximate the integrals of Eqs. (3.6)-(3.8). There are many ways to do this that would be valid in principle. In practice, there is a handful of methods that are usually applied to this problem, such as linearization, the unscented transform, and Gauss–Hermite quadrature, among others. We focus on linearization and the unscented transform, since our work builds on these methods.

Linearization involves approximating the nonlinear function by its first-order Taylor expansion around the mean. The *linear approximation of an additive*

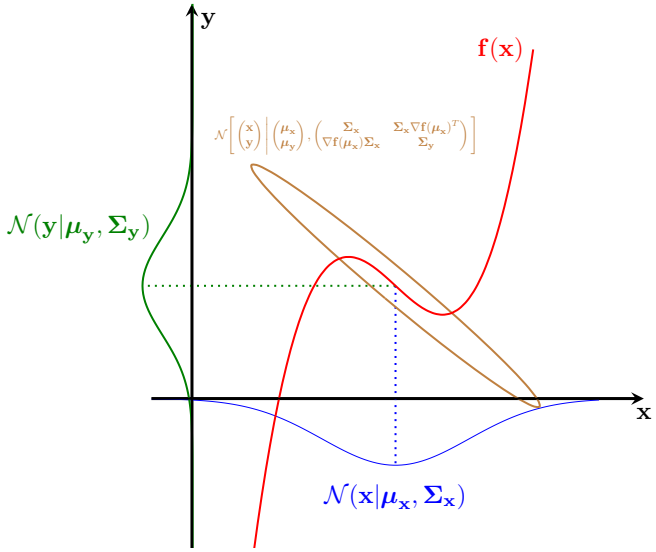


Figure 3.1: Visual illustration of the linear approximation of an additive transform. The prior Gaussian distribution for x (blue) is propagated through a linear approximation to yield a joint Gaussian approximation (brown) and the marginal of y (green)

Algorithm 6 General Gaussian filter

Input: Initial mean $\boldsymbol{\mu}_0$, covariance $\boldsymbol{\Sigma}_0$, observations $\mathbf{y}_{1:T}$
for $t = 1$ to T **do**
// Prediction step

$$\boldsymbol{\mu}_t^- = \int \mathbf{f}(\mathbf{x}_{t-1})\,\mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1})\,d\mathbf{x}_{t-1} \tag{3.9}$$

$$\boldsymbol{\Sigma}_t^- = \int \left(\mathbf{f}(\mathbf{x}_{t-1}) - \boldsymbol{\mu}_t^-\right)\left(\mathbf{f}(\mathbf{x}_{t-1}) - \boldsymbol{\mu}_t^-\right)^\top \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1})\,d\mathbf{x}_{t-1} + \mathbf{Q}_{t-1} \tag{3.10}$$

// Update step

$$\boldsymbol{\mu}_{\mathbf{y},t} = \int \mathbf{g}(\mathbf{x}_t)\,\mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\mu}_t^-, \boldsymbol{\Sigma}_t^-)\,d\mathbf{x}_t \tag{3.11}$$

$$\boldsymbol{\Sigma}_{\mathbf{y},t} = \int \left(\mathbf{g}(\mathbf{x}_t) - \boldsymbol{\mu}_{\mathbf{y},t}\right)\left(\mathbf{g}(\mathbf{x}_t) - \boldsymbol{\mu}_{\mathbf{y},t}\right)^\top \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\mu}_t^-, \boldsymbol{\Sigma}_t^-)\,d\mathbf{x}_t + \mathbf{R}_t \tag{3.12}$$

$$\mathbf{C}_{\mathbf{xy},t} = \int \left(\mathbf{x}_t - \boldsymbol{\mu}_t^-\right)\left(\mathbf{g}(\mathbf{x}_t) - \boldsymbol{\mu}_{\mathbf{y},t}\right)^\top \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\mu}_t^-, \boldsymbol{\Sigma}_t^-)\,d\mathbf{x}_t \tag{3.13}$$

$$\mathbf{K}_t = \mathbf{C}_{\mathbf{xy},t}\boldsymbol{\Sigma}_{\mathbf{y},t}^{-1} \tag{3.14}$$

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_t^- + \mathbf{K}_t(\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y},t}) \tag{3.15}$$

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_t^- - \mathbf{K}_t\boldsymbol{\Sigma}_{\mathbf{y},t}\mathbf{K}_t^\top \tag{3.16}$$

end for
Output: Filtered means $\boldsymbol{\mu}_{1:T}$ and covariances $\boldsymbol{\Sigma}_{1:T}$

transform [SS23, Alg. 7.1] is given in Box 3.2. A visual illustration of the linear approximation is given in Fig. 3.1. When this approximation is used repeatedly in each prediction and update step of the general Gaussian filter of Alg. 6 the resulting algorithm is the extended Kalman filter (Alg. 2).

The unscented transform approximates Gaussian integrals by propagating a set of sigma points through the nonlinear function and matching the resulting moments. This method provides robust approximations for moderate nonlinearities without requiring Jacobians, which might not be available, or be ill-conditioned. The *unscented approximation of an additive transform* [SS23, Alg. 8.15] is given in Box 3.3. When this approximation is used repeatedly in each prediction and update step of the general Gaussian filter of Alg. 6 the resulting algorithm is the unscented Kalman filter (Alg. 3).

Box 3.2: Linear approximation of an additive transform

The linear approximation of an additive transform is a general way of approximating the joint density of a pair of random variables $\mathbf{x}$ and $\mathbf{y}$ as in Eqs. (3.2)-(3.3) by using a linear Taylor expansion of $\mathbf{f}$ around $\boldsymbol{\mu}_\mathbf{x}$,

$$\mathbf{f}(\mathbf{x}) \simeq \mathbf{f}(\boldsymbol{\mu}_\mathbf{x}) + \nabla\mathbf{f}(\boldsymbol{\mu}_\mathbf{x})(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}) + \mathbf{r}. \tag{3.17}$$

Substituting Eq. (3.17) into Eqs. (3.6)-(3.8) we obtain the moments of the linear Gaussian approximation:

$$\boldsymbol{\mu}_\mathbf{y} = \mathbf{f}(\boldsymbol{\mu}_\mathbf{x}), \tag{3.18}$$
$$\boldsymbol{\Sigma}_\mathbf{y} = \nabla\mathbf{f}(\boldsymbol{\mu}_\mathbf{x})\boldsymbol{\Sigma}_\mathbf{x}\nabla\mathbf{f}(\boldsymbol{\mu}_\mathbf{x})^T + \mathbf{R}, \tag{3.19}$$
$$\mathbf{C}_\mathbf{xy} = \boldsymbol{\Sigma}_\mathbf{x}\nabla\mathbf{f}(\boldsymbol{\mu}_\mathbf{x})^T. \tag{3.20}$$

A visual illustration of the linear approximation can be found in Fig. 3.1.

Gaussian filter

The Gaussian moment matching approximation defined by Eqs. (3.5)-(3.8) can be applied to filtering at time t as follows:

Prediction. Assuming that the filtering distribution is Gaussian at time $t-1$,

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}), \tag{3.21}$$

the prediction Eq. (2.9) is the computation of the $\mathbf{x}_t$–marginal of the joint distribution $p(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:t-1})$. Since the $\mathbf{x}_{t-1}$–marginal is assumed to be Gaussian, and

$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}) + \mathbf{q}_t$ we are in the scenario of Eqs. (3.2)-(3.3). Hence we can use Eq. (3.5) to approximate the joint $p(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:t-1})$ by a moment-matched Gaussian. The $\mathbf{x}_t$–marginal is also Gaussian and is given by $p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^-, \boldsymbol{\Sigma}_t^-)$ where $\boldsymbol{\mu}_t^-$ and $\boldsymbol{\Sigma}_t^-$ are given in Eqs. (3.9)-(3.10) of Alg. 6.

Update. Similarly, the update Eq. (2.10) is the computation of the $\mathbf{x}_t$–conditional of $p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1})$. Since the $\mathbf{x}_t$–marginal computed in the prediction step is Gaussian, and $\mathbf{y}_t = \mathbf{g}(\mathbf{x}_t) + \mathbf{r}_t$ we are again in the scenario of Eqs. (3.2)-(3.3). Hence we can again use Eq. (3.5) to approximate the joint $p(\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1})$ by the moment-matched Gaussian. Finally, the conditional is given by $p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ where $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ are given in Eqs. (3.15)-(3.16) of Alg. 6.

By recursively repeating the above steps we obtain the general Gaussian filter algorithm given in Alg. 6. The formulation of the Gaussian filter algorithm in this generic form enables the use of many numerical integration methods for the computation of Eqs. (3.9)-(3.13), leading to a variety of Gaussian filters, such as the EKF, UKF, quadrature and cubature Kalman filters among others.

3.2.3 Gaussian sum filters

One obvious limitation of Gaussian filters is their inability to express multimodal densities. This can be mitigated by running multiple Gaussian filters in parallel, and weigh them accordingly, to obtain a class of algorithms that is known as *Gaussian sum filters* [SA71b]. Any Gaussian filter can be used to construct the corresponding GSF as follows.

Let $\{p_m(\mathbf{x}_t|\mathbf{y}_{1:t-1})\}_{t=1}^T$ and $\{p_m(\mathbf{x}_t|\mathbf{y}_{1:t})\}_{t=1}^T$ denote the predicted and filtering distributions respectively of the m^{th} Gaussian filter for $m = 1, \ldots, M$. These are

$$p_m(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^{-(m)}, \boldsymbol{\Sigma}_t^{-(m)}), \tag{3.22}$$

$$p_m(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^{(m)}, \boldsymbol{\Sigma}_t^{(m)}), \tag{3.23}$$

where $(\boldsymbol{\mu}_t^{-(m)}, \boldsymbol{\Sigma}_t^{-(m)})$ and $(\boldsymbol{\mu}_t^{(m)}, \boldsymbol{\Sigma}_t^{(m)})$ are computed by the m^{th} Gaussian filter. Consider the filtering distribution of the GSF at time $t-1$,

$$p_{\text{GSF}}(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \sum_{m=1}^{M} w_{t-1}^{(m)} p_m(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \sum_{m=1}^{M} w_{t-1}^{(m)} \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)}). \tag{3.24}$$

Substituting Eq. (3.24) into the prediction Eq. (2.9) we obtain the predicted

Box 3.3: Unscented approximation of an additive transform

Another way to approximate $p(\mathbf{x}, \mathbf{y})$ with a Gaussian uses the so-called *unscented transform* (UT) [JU96]. The UT deterministically chooses a set of *sigma-points* that captures the first and second order moments of the distribution of $\mathbf{x}$. The sigma-points are then propagated through the nonlinearity $\mathbf{f}$, and the resulting points are used to reconstruct the joint Gaussian [SS23, Algorithm 8.15].
Formally, $2d_x + 1$ sigma-points are selected as follows,

$$\boldsymbol{\sigma}^{(0)} = \boldsymbol{\mu}, \tag{3.25}$$

$$\boldsymbol{\sigma}^{(\pm i)} = \boldsymbol{\mu} \pm \sqrt{d_x + \lambda} \cdot \boldsymbol{\Sigma}_{\bullet i}^{1/2},\ i = 1, \dots, d_x, \tag{3.26}$$

where $\{\boldsymbol{\sigma}^{(i)}\}_{i=-d_x}^{d_x}$ are the sigma points, $\boldsymbol{\Sigma}_{\bullet i}^{1/2}$ is the i^{th} column of the matrix square-root of the covariance $\boldsymbol{\Sigma}$ and $\lambda = \alpha^2(d_x + \kappa) - d_x$, where α and κ are parameters that determine the spread of the sigma-points around the mean [WV01]. The sigma-points are used to construct estimates of the moments of (3.4) as follows:

$$\boldsymbol{\mu}_{\mathbf{y}}^{\sigma} = \sum_{i=-d_x}^{d_x} \omega_i \mathbf{f}(\boldsymbol{\sigma}^{(i)}), \tag{3.27}$$

$$\boldsymbol{\Sigma}_{\mathbf{y}}^{\sigma} = \sum_{i=-d_x}^{d_x} \tilde{\omega}_i (\mathbf{f}(\boldsymbol{\sigma}^{(i)}) - \boldsymbol{\mu}_{\mathbf{y}}^{\sigma})(\mathbf{f}(\boldsymbol{\sigma}^{(i)}) - \boldsymbol{\mu}_{\mathbf{y}}^{\sigma})^T + \mathbf{R}, \tag{3.28}$$

$$\mathbf{C}_{\mathbf{xy}}^{\sigma} = \sum_{i=-d_x}^{d_x} \tilde{\omega}_i (\boldsymbol{\sigma}^{(i)} - \boldsymbol{\mu})(\mathbf{f}(\boldsymbol{\sigma}^{(i)}) - \boldsymbol{\mu}_{\mathbf{y}}^{\sigma})^T, \tag{3.29}$$

where

$$\omega_0 = \frac{\lambda}{d_x + \lambda},\ \tilde{\omega}_0 = \frac{\lambda}{d_x + \lambda} + (1 - \alpha^2 + \beta), \tag{3.30}$$

$$\omega_i = \tilde{\omega}_i = \frac{1}{2(d_x + \lambda)},\ i = \pm 1, \dots, \pm d_x, \tag{3.31}$$

and where β is an additional parameter that can be used to incorporate prior information on the distribution of $\mathbf{x}$ [WV01]. Finally, we use the estimates of Eqs. (3.27)-(3.29) to build the Gaussian approximation

$$p(\mathbf{x}, \mathbf{y}) \approx \mathcal{N}\left(\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \middle| \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_{\mathbf{y}}^{\sigma} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma} & \mathbf{C}_{\mathbf{xy}}^{\sigma} \\ \mathbf{C}_{\mathbf{xy}}^{\sigma\ T} & \mathbf{S}_{\mathbf{y}}^{\sigma} \end{pmatrix} \right). \tag{3.32}$$

distribution of the GSF

$$p_{\text{GSF}}(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \sum_{m=1}^{M} w_{t-1}^{(m)} \int p(\mathbf{x}_t|\mathbf{x}_{t-1})\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)})d\mathbf{x}_{t-1} \tag{3.33}$$

$$\approx \sum_{m=1}^{M} w_{t-1}^{(m)}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^{-(m)}, \boldsymbol{\Sigma}_t^{-(m)}), \tag{3.34}$$

where in the second line we used the fact that each of the Gaussian filters approximately satisfies the prediction Eq. (2.9). Substituting this into the update Eq. (2.10) we obtain

$$p_{\text{GSF}}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)\sum_{m=1}^{M} w_{t-1}^{(m)}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^{-(m)}, \boldsymbol{\Sigma}_t^{-(m)})}{\int p(\mathbf{y}_t|\mathbf{x}_t)\sum_{m=1}^{M} w_{t-1}^{(m)}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^{-(m)}, \boldsymbol{\Sigma}_t^{-(m)})d\mathbf{x}_t} \tag{3.35}$$

$$= \frac{\sum_{m=1}^{M} w_{t-1}^{(m)} p(\mathbf{y}_t|\mathbf{x}_t)\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^{-(m)}, \boldsymbol{\Sigma}_t^{-(m)})}{\sum_{m=1}^{M} w_{t-1}^{(m)} \int p(\mathbf{y}_t|\mathbf{x}_t)\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^{-(m)}, \boldsymbol{\Sigma}_t^{-(m)})d\mathbf{x}_t} \tag{3.36}$$

$$\approx \sum_{m=1}^{M} w_t^{(m)}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^{(m)}, \boldsymbol{\Sigma}_t^{(m)}) \tag{3.37}$$

where in the last line we used the fact that the Gaussian filters approximately satisfy the update Eq. (2.10),

$$w_t^{(m)} = \frac{w_{t-1}^{(m)}\mathcal{N}(\mathbf{y}_t|\boldsymbol{\mu}_{\mathbf{y},t}^{(m)}, \boldsymbol{\Sigma}_{\mathbf{y},t}^{(m)})}{\sum_{m=1}^{M} w_{t-1}^{(m)}\mathcal{N}(\mathbf{y}_t|\boldsymbol{\mu}_{\mathbf{y},t}^{(m)}, \boldsymbol{\Sigma}_{\mathbf{y},t}^{(m)})}, \tag{3.38}$$

and $\{(\boldsymbol{\mu}_{\mathbf{y},t}^{(m)}, \boldsymbol{\Sigma}_{\mathbf{y},t}^{(m)})\}_{t=1}^{T}$ are computed by the m^{th} Gaussian filter.

Gaussian sum filters provide an expressive way for representing filtering distributions since any density can be expressed as a Gaussian mixture with a large enough number of components [AS72b]. Their main drawback is that the computational cost increases significantly with the number of components. Particle filters described below offer an alternative way of representing multimodal densities at a much lower cost.

3.3 Augmentation-based approximations

In this section, we present a novel *augmented Gaussian approximation* which generalizes the Gaussian moment-matching approximations that were summarized in Section 3.2.2. The proposed procedure is based on a well known identity which is used to re-express a Gaussian density as a Gaussian mixture with multiple

components. Mirroring Sections 2.4.2-2.4.3, we derive the augmented linear and unscented approximations. In Section 3.4, we apply the *augmented Gaussian approximation* to filtering and derive the class of *augmented Gaussian sum filters*.

3.3.1 Augmentation of a Gaussian integral

The augmentation is based on the Gaussian integral identity,

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}) = \int \mathcal{N}(\mathbf{x}|\boldsymbol{\Gamma}\mathbf{z} + \mathbf{c}, \boldsymbol{\Delta})\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\Sigma}_\mathbf{z})d\mathbf{z}, \tag{3.39}$$

where $\mathbf{x} \in \mathbb{R}^{d_x}$ and $\mathbf{z} \in \mathbb{R}^{d_z}$. On the left hand side of Eq.(3.39) we have a Gaussian PDF with known mean $\boldsymbol{\mu}_\mathbf{x}$ and covariance $\boldsymbol{\Sigma}_\mathbf{x}$. On the right we have an integral over a latent variable $\mathbf{z}$. The integral is interpreted as a mixture of kernels $\mathcal{N}(\mathbf{x}|\boldsymbol{\Gamma}\mathbf{z}+\mathbf{c}, \boldsymbol{\Delta})$ with weights $\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\Sigma}_\mathbf{z})$. Therefore Eq.(3.39) represents the expression of a known Gaussian density as a dense Gaussian mixture. For Eq. (3.39) to hold, the following relations need to be satisfied by the parameters $\mathbf{c}, \boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\Gamma}, \boldsymbol{\Sigma}_\mathbf{z}, \boldsymbol{\Delta}$:

$$\boldsymbol{\mu}_\mathbf{x} = \boldsymbol{\Gamma}\boldsymbol{\mu}_\mathbf{z} + \mathbf{c}, \tag{3.40}$$
$$\boldsymbol{\Sigma}_\mathbf{x} = \boldsymbol{\Gamma}\boldsymbol{\Sigma}_\mathbf{z}\boldsymbol{\Gamma}^T + \boldsymbol{\Delta}, \tag{3.41}$$
$$\boldsymbol{\Delta} \succeq \mathbf{0}, \boldsymbol{\Sigma}_\mathbf{z} \succeq \mathbf{0}. \tag{3.42}$$

Note that Eqs. (3.40)-(3.42) are under-determined and therefore have multiple solutions. This can be easily seen by the decomposition of Eq. (3.41) which is not unique and the fact that for any choice of $\boldsymbol{\Gamma}$ and $\boldsymbol{\mu}_\mathbf{z}$, Eq. (3.40) can be satisfied by selecting $\mathbf{c} = \boldsymbol{\mu}_\mathbf{x} - \boldsymbol{\Gamma}\boldsymbol{\mu}_\mathbf{z}$. For simplicity we make the choices

$$d_z = d_x, \tag{3.43}$$
$$\mathbf{c} = \mathbf{0}, \tag{3.44}$$
$$\boldsymbol{\mu}_\mathbf{z} = \boldsymbol{\mu}_\mathbf{x}, \tag{3.45}$$
$$\boldsymbol{\Gamma} = \mathbf{I}, \tag{3.46}$$
$$\boldsymbol{\Sigma}_\mathbf{z} = \boldsymbol{\Sigma}_\mathbf{x} - \boldsymbol{\Delta}. \tag{3.47}$$

With these choices, the only free parameter left to choose is the covariance matrix $\boldsymbol{\Delta}$, which must satisfy $\boldsymbol{\Sigma}_\mathbf{x} \succeq \boldsymbol{\Delta} \succeq \mathbf{0}$. The augmentation procedure for the choice of Eqs. (3.43)-(3.47) is summarized in Box 3.4.

Box 3.4: Gaussian augmentation

Given a Gaussian density of a random variable $\mathbf{x} \in \mathbb{R}^{d_x}$ with known mean $\boldsymbol{\mu}_\mathbf{x}$ and covariance $\boldsymbol{\Sigma}_\mathbf{x}$, the *augmentation* of $\mathbf{x}$ is the introduction of a new random variable $\mathbf{z}$ via the equation

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}) = \int \mathcal{N}(\mathbf{x}|\mathbf{z}, \boldsymbol{\Delta})\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x} - \boldsymbol{\Delta})d\mathbf{z}, \tag{3.48}$$

where $\mathbf{z} \in \mathbb{R}^{d_x}$ and $\boldsymbol{\Delta}$ is a free parameter such that $\boldsymbol{\Sigma}_\mathbf{x} \succeq \boldsymbol{\Delta} \succeq \mathbf{0}$. The marginal of the augmentation variable is $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x} - \boldsymbol{\Delta})$ and the conditional, $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{z}, \boldsymbol{\Delta})$, referred to as the *kernel* of the augmentation. The covariance matrix $\boldsymbol{\Delta}$ is a parameter of the augmentation that is set by the user. By sampling the augmentation variable we obtain a re-approximation of the original Gaussian by a mixture of the kernels

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}) \simeq \frac{1}{N}\sum_{n=1}^{N} \mathcal{N}(\mathbf{x}|\mathbf{z}_n, \boldsymbol{\Delta}), \tag{3.49}$$

centered at $\mathbf{z}_n \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x} - \boldsymbol{\Delta})$ for $n = 1, \ldots, N$.

Interpolation property of the Gaussian augmentation. The augmentation that was introduced in Box 3.4 has an important interpolation property. To explain this property we consider the transformation in the space of probability density functions $\mathcal{P}(\mathbb{R}^{d_x})$, defined by

$$\mathcal{T}(\mathcal{N}(\cdot|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}); \boldsymbol{\Delta}) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{N}(\cdot|\mathbf{z}_n, \boldsymbol{\Delta}). \tag{3.50}$$

This transformation is equivalent to the approximation of Eq. (3.49) since the right hand side of this equation is equal to $\mathcal{T}(\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}); \boldsymbol{\Delta})$. Moreover, it is easy to verify that as a function of the augmentation covariance $\boldsymbol{\Delta}$, the transformation $\mathcal{T}$ has the following limiting behaviors:

$$\mathcal{T}(\mathcal{N}(\cdot|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}); \boldsymbol{\Delta}) \to \frac{1}{N}\sum_{n=1}^{N} \delta_{\mathbf{x}_n}(\cdot) \text{ as } \boldsymbol{\Delta} \to \mathbf{0}, \tag{3.51}$$

$$\mathcal{T}(\mathcal{N}(\cdot|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}); \boldsymbol{\Delta}) \to \mathcal{N}(\cdot|\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}) \text{ as } \boldsymbol{\Delta} \to \boldsymbol{\Sigma}_\mathbf{x}, \tag{3.52}$$

where $\mathbf{x}_n \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x})$. In words, as $\boldsymbol{\Delta}$ shrinks to $\mathbf{0}$, the transformation of the original Gaussian tends to a particle approximation whereas when $\boldsymbol{\Delta}$ tends to

$\boldsymbol{\Sigma}_{\mathbf{x}}$, the transformation tends to the identity. Thus the transformation we have defined interpolates between a particle approximation and the identity. This interpolation property also extends to approximations of Gaussian integrals.

Given a Gaussian integral

$$\mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{N}(\cdot|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})] = \int \boldsymbol{\phi}(\mathbf{x}) \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}) d\mathbf{x}, \tag{3.53}$$

we consider the following approximation

$$\begin{aligned} \mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{N}(\cdot|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})] &\approx \mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{T}(\mathcal{N}(\cdot|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}); \boldsymbol{\Delta})] && (3.54) \\ &= \frac{1}{N} \sum_{n=1}^{N} \int \boldsymbol{\phi}(\mathbf{x}) \mathcal{N}(\mathbf{x}|\mathbf{z}_n, \boldsymbol{\Delta}) d\mathbf{x} && (3.55) \\ &= \frac{1}{N} \sum_{n=1}^{N} \mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{N}(\cdot|\mathbf{z}_n, \boldsymbol{\Delta})]. && (3.56) \end{aligned}$$

This approximation can be expressed as a transformation

$$\mathcal{T}(\mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{N}(\cdot|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})]; \boldsymbol{\Delta})) = \mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{T}(\mathcal{N}(\cdot|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}); \boldsymbol{\Delta})], \tag{3.57}$$

which as a function of the covariance $\boldsymbol{\Delta}$ has the following limiting behaviors:

$$\mathcal{T}(\mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{N}(\cdot|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})]; \boldsymbol{\Delta})) \to \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) \text{ as } \boldsymbol{\Delta} \to \mathbf{0}, \tag{3.58}$$

$$\mathcal{T}(\mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{N}(\cdot|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})]; \boldsymbol{\Delta})) \to \mathcal{I}[\boldsymbol{\phi}(\mathbf{x}); \mathcal{N}(\cdot|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})] \text{ as } \boldsymbol{\Delta} \to \boldsymbol{\Sigma}_{\mathbf{x}}. \tag{3.59}$$

Therefore the integral transformation interpolates between a Monte Carlo approximation and the original integral.

3.3.2 Augmented Gaussian moment-matching

We combine the augmentation of the previous section with Gaussian moment-matching to obtain the *augmented Gaussian moment-matching approximation* of a pair $(\mathbf{x}, \mathbf{y})$ as defined in Eqs. (3.2)-(3.3). Substituting Eq. (3.49) into Eq. (3.4)

for the joint of $(\mathbf{x}, \mathbf{y})$ we obtain

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{f}(\mathbf{x}), \mathbf{R})\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}) \tag{3.60}$$

$$\simeq \mathcal{N}(\mathbf{y}|\mathbf{f}(\mathbf{x}), \mathbf{R})\frac{1}{N}\sum_{n=1}^{N}\mathcal{N}(\mathbf{x}|\mathbf{z}_n, \boldsymbol{\Delta}) \tag{3.61}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\mathcal{N}(\mathbf{y}|\mathbf{f}(\mathbf{x}), \mathbf{R})\mathcal{N}(\mathbf{x}|\mathbf{z}_n, \boldsymbol{\Delta}) \tag{3.62}$$

$$\simeq \frac{1}{N}\sum_{n=1}^{N}\mathcal{N}\left(\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}\middle|\begin{pmatrix}\mathbf{z}_n\\ \boldsymbol{\mu}_{\mathbf{y},n}\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Delta} & \mathbf{C}_{\mathbf{xy},n}\\ \mathbf{C}_{\mathbf{xy},n}^T & \boldsymbol{\Sigma}_{\mathbf{y},n}\end{pmatrix}\right), \tag{3.63}$$

where $\mathbf{z}_n \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta})$, and

$$\boldsymbol{\mu}_{\mathbf{y},n} = \int \mathbf{f}(\mathbf{x})\mathcal{N}(\mathbf{x}|\mathbf{z}_n, \boldsymbol{\Delta})d\mathbf{x}, \tag{3.64}$$

$$\boldsymbol{\Sigma}_{\mathbf{y},n} = \int (\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_{\mathbf{y}})(\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_{\mathbf{y}})^T\mathcal{N}(\mathbf{x}|\mathbf{z}_n, \boldsymbol{\Delta})d\mathbf{x} + \mathbf{R}, \tag{3.65}$$

$$\mathbf{C}_{\mathbf{xy},n} = \int (\mathbf{x} - \mathbf{z}_n)(\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_{\mathbf{y}})^T\mathcal{N}(\mathbf{x}|\mathbf{z}_n, \boldsymbol{\Delta})d\mathbf{x}. \tag{3.66}$$

In Eq. (3.61), we use the mixture approximation of Eq. (3.49), whereas in Eq. (3.63) we use for each mixture component of Eq. (3.62) the *Gaussian moment matching of an additive transform* given in Box 3.1. To generalize to a non-additive transform we use [SS23, Algorithm 8.2] instead. The augmented Gaussian moment-matching approximation is summarized in Box 3.5

Box 3.5: Augmented Gaussian moment-matching

The augmented Gaussian moment-matching approximation of the joint density of a pair of random variables $(\mathbf{x}, \mathbf{y})$ defined in Eqs. (3.2)-(3.3), is given by

$$p_{\text{AGMM}}(\mathbf{x}, \mathbf{y}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{N}\left(\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}\middle|\begin{pmatrix}\mathbf{z}_n\\ \boldsymbol{\mu}_{\mathbf{y},n}\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Delta} & \mathbf{C}_{\mathbf{xy},n}\\ \mathbf{C}_{\mathbf{xy},n}^T & \boldsymbol{\Sigma}_{\mathbf{y},n}\end{pmatrix}\right), \tag{3.67}$$

where $\mathbf{z}_n \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta})$ and $\boldsymbol{\mu}_{\mathbf{y},n}, \boldsymbol{\Sigma}_{\mathbf{y},n}, \mathbf{C}_{\mathbf{xy},n}$, given in Eqs. (3.64)-(3.66).

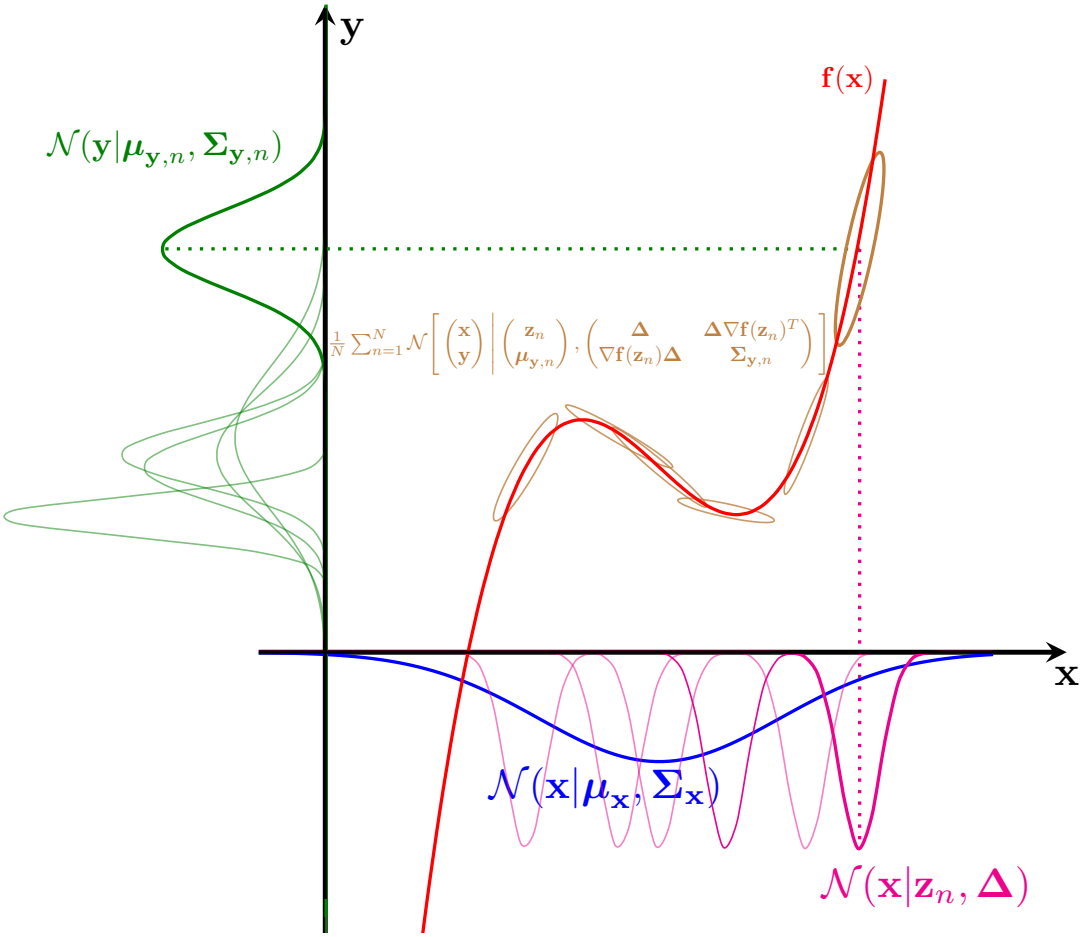


Figure 3.2: Visual illustration for the augmented linear approximation. The prior Gaussian (blue), is split into Gaussian components (magenta) which are used for the joint approximation (brown) and the approximation of the marginal of $\mathbf{y}$ (green).

Augmented linear approximation

By applying the *linear approximation of an additive transform* to approximate the moments in Eqs. (3.64)-(3.66), we obtain

$$\boldsymbol{\mu}_{\mathbf{y},n} = \mathbf{f}(\mathbf{z}_n), \tag{3.68}$$

$$\boldsymbol{\Sigma}_{\mathbf{y},n} = \nabla\mathbf{f}(\mathbf{z}_n)\boldsymbol{\Delta}\nabla\mathbf{f}(\mathbf{z}_n)^T + \mathbf{R}, \tag{3.69}$$

$$\mathbf{C}_{\mathbf{xy},n} = \boldsymbol{\Delta}\nabla\mathbf{f}(\mathbf{z}_n)^T, \tag{3.70}$$

where $\nabla\mathbf{f}$ denotes the Jacobian of $\mathbf{f}$. An illustration of the joint approximation coming from the augmented linear approximation is shown in Fig. 3.2. Using the non-additive version of the linear approximation [SS23, Algorithm 7.2], we can generalize the augmented linear approximation to non-additive models.

Augmented unscented approximation

Similarly, we can use the unscented approximation to approximate the Eqs. (3.64)-(3.66). In this way, we obtain

$$\boldsymbol{\mu}^{\sigma}_{\mathbf{y},n} = \sum_{i=-d_x}^{d_x} \omega_i \mathbf{f}(\boldsymbol{\sigma}^{(i)}_n), \tag{3.71}$$

$$\boldsymbol{\Sigma}^{\sigma}_{\mathbf{y},n} = \sum_{i=-d_x}^{d_x} \tilde{\omega}_i (\mathbf{f}(\boldsymbol{\sigma}^{(i)}_n) - \boldsymbol{\mu}^{\sigma}_{\mathbf{y},n})(\mathbf{f}(\boldsymbol{\sigma}^{(i)}_n) - \boldsymbol{\mu}^{\sigma}_{\mathbf{y},n})^T + \mathbf{Q}, \tag{3.72}$$

$$\mathbf{C}^{\sigma}_{\mathbf{xy},n} = \sum_{i=-d_x}^{d_x} \tilde{\omega}_i (\boldsymbol{\sigma}^{(i)}_n - \mathbf{z}_n)(\mathbf{f}(\boldsymbol{\sigma}^{(i)}_n) - \boldsymbol{\mu}^{\sigma}_{\mathbf{y},n})^T, \tag{3.73}$$

where $\{\boldsymbol{\sigma}^{(i)}_n\}_{i=-d_x}^{d_x}$ denote the sigma-points used to reconstruct $\mathcal{N}(\mathbf{z}_n, \boldsymbol{\Delta})$. Using the non-additive version of the unscented approximation [SS23, Algorithm 8.15], we can generalize the augmented linear approximation to non-additive models.

3.3.3 Automatic selection of $\boldsymbol{\Delta}$

In this section, we motivate a procedure for determining the augmentation covariance $\boldsymbol{\Delta}$ automatically via a semidefinite program. Our aim is to exploit the *interpolation property* that was described in Sec. 3.3.2. We derive the program in the context of the *augmented linear approximation* (Section 3.3.2), which allows tractable analytical computations.

The augmented linear estimator of the mean of variable $\mathbf{y}$ is given by

$$\widehat{\boldsymbol{\mu}}_{\mathbf{y}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{f}(\mathbf{z}_n), \tag{3.74}$$

where $\mathbf{z}_n \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta})$. The following theorem expresses the MSE of the estimator in Eq. (3.74) as a function of the covariance parameters $\boldsymbol{\Sigma}_{\mathbf{x}}$ and $\boldsymbol{\Delta}$, the number of Monte-Carlo samples N, the Jacobian, and the Hessian of the non-linearity $\mathbf{f}$. It will be exploited later to determine the augmentation covariances automatically.

Theorem 3.1. *Let $\mathbf{f} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ be twice continuously differentiable. Then the MSE of the estimator $\widehat{\boldsymbol{\mu}}_{\mathbf{y}}$ defined in Eq. (3.74) is given by*

$$\begin{aligned} MSE(\widehat{\boldsymbol{\mu}}_{\mathbf{y}}) \simeq \frac{1}{N}\operatorname{Tr}\left((\boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta})\nabla\mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}})^T\nabla\mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}})\right) \\ + \frac{1}{4}\sum_{i=1}^{d_y}\operatorname{Tr}\left(\boldsymbol{\Delta}\nabla^2\mathbf{f}_i(\boldsymbol{\mu}_{\mathbf{x}})^2\right), \end{aligned} \tag{3.75}$$

by ignoring terms of higher order in $\boldsymbol{\Delta}$.

Proof. Given in the Appendix for Chapter 3. □

The mean squared error (MSE) of the estimator is given by two terms, the first term of Eq. (3.75) corresponds to the error due to Monte Carlo sampling (variance), and the second term to the error due to the linearization (bias). Thus $\boldsymbol{\Delta}$ can be seen as a tuning parameter which controls the trade-off between the two sources of error. This is apparent in the two limiting cases: for $\boldsymbol{\Delta} \to \mathbf{0}$ there is pure Monte Carlo sampling (no linearization) and for $\boldsymbol{\Delta} \to \boldsymbol{\Sigma}_{\mathbf{x}}$ there is pure linearization (no sampling).

Theorem 3.1 suggests a way for choosing $\boldsymbol{\Delta}$ by minimizing the right-hand side of Eq. (3.75). This leads to the *semi-definite program*

$$\begin{aligned} &\min_{\boldsymbol{\Delta}} \Psi(\boldsymbol{\Delta}; \gamma, \boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}, \mathbf{f}), \\ &\text{s.t. } \boldsymbol{\Delta} \succeq \mathbf{0},\ \boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta} \succeq \mathbf{0}, \end{aligned} \tag{3.76}$$

where

$$\begin{aligned} \Psi(\boldsymbol{\Delta}; \gamma, \boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}, \mathbf{f}) = \frac{\gamma}{N}\operatorname{Tr}\left((\boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta})\nabla\mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}})^T\nabla\mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}})\right) \\ + \frac{1}{4}\sum_{i=1}^{d_y}\operatorname{Tr}\left(\boldsymbol{\Delta}\nabla^2\mathbf{f}_i(\boldsymbol{\mu}_{\mathbf{x}})\right)^2. \end{aligned} \tag{3.77}$$

The parameter γ is introduced to give the user control over the relative importance of the two terms. By controlling γ the user influences the behaviour of the estimator in Eq. (3.74) towards that of a Monte-Carlo or a linear estimator. We have included an ablation study of γ in the supplementary material of this paper.

For affine functions $\mathbf{f}$, the second term tends to zero and the minimizer of Eq. (3.76) will be $\boldsymbol{\Delta} = \boldsymbol{\Sigma}_{\mathbf{x}}$, leading to a purely linear approximation of the mean. On the other hand, when the first term of Eq. (3.77) vanishes, the program will select a zero covariance, corresponding to a pure Monte Carlo approximation. In general, when both terms of Eq. (3.77) are active, the optimal covariance will balance linearization and Monte Carlo error to minimize the objective. There are many optimization algorithms that can be used to solve Eq. (3.76), such as projected gradient descent or interior point methods [Ber97]. These algorithms however introduce additional computational complexity to the filtering algorithm, which solves Eq. (3.76) at every iteration.

In this work we consider cases where the degree of nonlinearity of the function $\mathbf{f}$ is comparable in all directions of the state space. In such cases, it is not necessary to solve the full optimization problem, and we use Eq. (3.76) to define a simpler problem that has an analytical solution, thus reducing the complexity of the algorithm. We arrive at the simpler problem by assuming that the augmentation covariance is proportional to the original covariance as shown in the following theorem.

Theorem 3.2. *The program of Eq.* (3.76)*, when* $\boldsymbol{\Delta}$ *is constrained to be proportional to the covariance* $\boldsymbol{\Delta} = \rho\boldsymbol{\Sigma}_{\mathbf{x}}$*, is solved by,*

$$\rho^* = \frac{2\gamma}{N} \frac{\operatorname{Tr}\left(\boldsymbol{\Sigma}_{\mathbf{x}} \nabla \mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}})^T \nabla \mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}})\right)}{\sum_{i=1}^{d_y} \operatorname{Tr}\left(\boldsymbol{\Sigma}_{\mathbf{x}} \nabla^2 \mathbf{f}_i(\boldsymbol{\mu}_{\mathbf{x}})\right)^2} \wedge 1. \tag{3.78}$$

Proof. Substituting $\boldsymbol{\Delta} = \rho\boldsymbol{\Sigma}_{\mathbf{x}}$ into Eq. (3.77) we obtain the objective for ρ:

$$\begin{aligned} \Psi(\rho) = &\frac{\gamma}{N}(1-\rho) \operatorname{Tr}\left(\boldsymbol{\Sigma}_{\mathbf{x}} \nabla \mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}})^T \nabla \mathbf{f}(\boldsymbol{\mu}_{\mathbf{x}})\right) \\ &+ \rho^2 \frac{1}{4} \sum_{i=1}^{d_y} \operatorname{Tr}\left(\boldsymbol{\Sigma}_{\mathbf{x}} \nabla^2 \mathbf{f}_i(\boldsymbol{\mu}_{\mathbf{x}})\right)^2, \end{aligned} \tag{3.79}$$

which is a quadratic function of ρ. Setting $\Psi'(\rho) = 0$ and solving for ρ it is straightforward to obtain the desired expression. Noting that the constraints of the optimization problem (3.76) are translated to $0 \leq \rho \leq 1$, we obtain Eq. (3.78). □

In the context of our filtering framework presented in the next section we use Eq.

(3.78) to derive a novel adaptive filtering algorithm. Note that Theorems 3.1 and 3.2 have not been proved for the case of the unscented estimator because of the intractability of the bias. In our experiments with the unscented AGSF we use Eq. (3.78) and demonstrate its empirical success.

Finally, it is of interest to solve Eq. (3.76) for the complete augmentation covariances when the function $\mathbf{f}$ is highly nonlinear in some, but not all directions of the state space. In such scenarios, it becomes favorable to split the Gaussian components along the directions of high nonlinearity. This results in covariance matrices $\mathbf{\Delta}$ which are shortened in the directions of high nonlinearity and elongated in the directions of low nonlinearity.

3.4 Augmented Gaussian Sum Filters

We now derive a novel class of filtering algorithms, called *augmented Gaussian sum filters*, by leveraging the augmented Gaussian approximations presented in Section 3.3.2. Our framework uses the approximation at each time step in the prediction and update steps of the filtering procedure. We note that all algorithms presented in this section are derived for additive models for notational simplicity and can be generalized to non-additive models straightforwardly.

In Section 3.4.1, we start by presenting a generic version of the algorithm, which uses the generic *augmented Gaussian approximation* of Section 3.3.1. We then particularize using the augmented linear and unscented approximations, and derive the linear-AGSF (L-AGSF) and unscented-AGSF (U-AGSF) algorithms respectively. In Section 3.4.2, we prove that the novel AGSF algorithm interpolates between the well known *Gaussian sum filter* and *bootstrap particle filter*, which are special cases of the novel algorithm. Finally, in Section 3.4.3 we discuss strategies for selecting the parameters of the AGSF algorithm.

3.4.1 Framework description

Generic AGSF

The new filter is summarized in Alg. 7 in its basic form. The prior is initialized as a mixture of M Gaussians and iterates over prediction and update steps, whose derivations are given below. We assume that the augmentation covariance matrices $\{\mathbf{\Delta}_m^{(t)}\}_{m=1}^{M}$ and $\{\mathbf{\Lambda}_{mn}^{(t)}\}_{m,n=1}^{M,N_m}$ are parameters of the algorithm set by the user at run time.

Also note that at each step, the number of components of the original mixture

grows. To ensure that our algorithm will run on fixed memory and cost, we do a resampling step at the end of the update step (although other mechanisms are also possible).

Prediction: Here we derive the predictive density of our filter. We assume that the filtering distribution at time $t-1$ is given by the following Gaussian mixture of M components

$$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \simeq \sum_{m=1}^{M} w_{t-1}^{(m)} \mathcal{N}(\mathbf{x}_{t-1}|\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)}). \tag{3.80}$$

Substituting into Eq. (2.9), we obtain for the predictive distribution of $\mathbf{x}_t$,

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \simeq \sum_{m=1}^{M} w_{t-1}^{(m)} p_m(\mathbf{x}_t|\mathbf{y}_{1:t-1}), \tag{3.81}$$

where

$$p_m(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int \mathcal{N}(\mathbf{x}_t|\mathbf{f}(\mathbf{x}_{t-1}), \mathbf{Q}) \mathcal{N}(\mathbf{x}_{t-1}|\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)}) d\mathbf{x}_{t-1}. \tag{3.82}$$

This is the marginal PDF of $\mathbf{x}_t$ with the joint PDF being

$$p_m(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t|\mathbf{f}(\mathbf{x}_{t-1}), \mathbf{Q}) \mathcal{N}(\mathbf{x}_{t-1}|\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)}). \tag{3.83}$$

We will approximate the joint $p_m(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:t-1})$ by a Gaussian mixture using the *augmented Gaussian moment-matching approximation* of Section 3.3.2. We will use the resulting marginal approximation for $p_m(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ which will also be given as a Gaussian mixture. Using Eqs. (3.63)-(3.66), we obtain for the joint,

$$p_m(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{y}_{1:t-1}) \simeq \frac{1}{N_m} \sum_{n=1}^{N_m} \mathcal{N}_{mn}(\mathbf{x}_{t-1}, \mathbf{x}_t),$$

where,

$$\mathcal{N}_{mn}(\mathbf{x}_{t-1}, \mathbf{x}_t) = \mathcal{N}\left[\begin{pmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_t \end{pmatrix} \middle| \begin{pmatrix} \mathbf{z}_{mn} \\ \boldsymbol{\mu}_{t,mn}^{-} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Delta}_m^{(t)} & \mathbf{C}_{mn}^{-} \\ \mathbf{C}_{mn}^{-\,T} & \boldsymbol{\Sigma}_{t,mn}^{-} \end{pmatrix} \right], \tag{3.84}$$

and $\boldsymbol{\mu}_{t,mn}^{-}, \boldsymbol{\Sigma}_{t,mn}^{-}$ are given in Eqs. (3.91)-(3.92), and

$$\mathbf{C}_{mn}^{-} = \mathcal{I}[(\mathbf{x}_{t-1} - \mathbf{z}_{mn})(\mathbf{f}(\mathbf{x}_{t-1}) - \boldsymbol{\mu}_{t,mn}^{-})^T; \mathbf{z}_{mn}, \boldsymbol{\Delta}_m^{(t)}]. \tag{3.85}$$

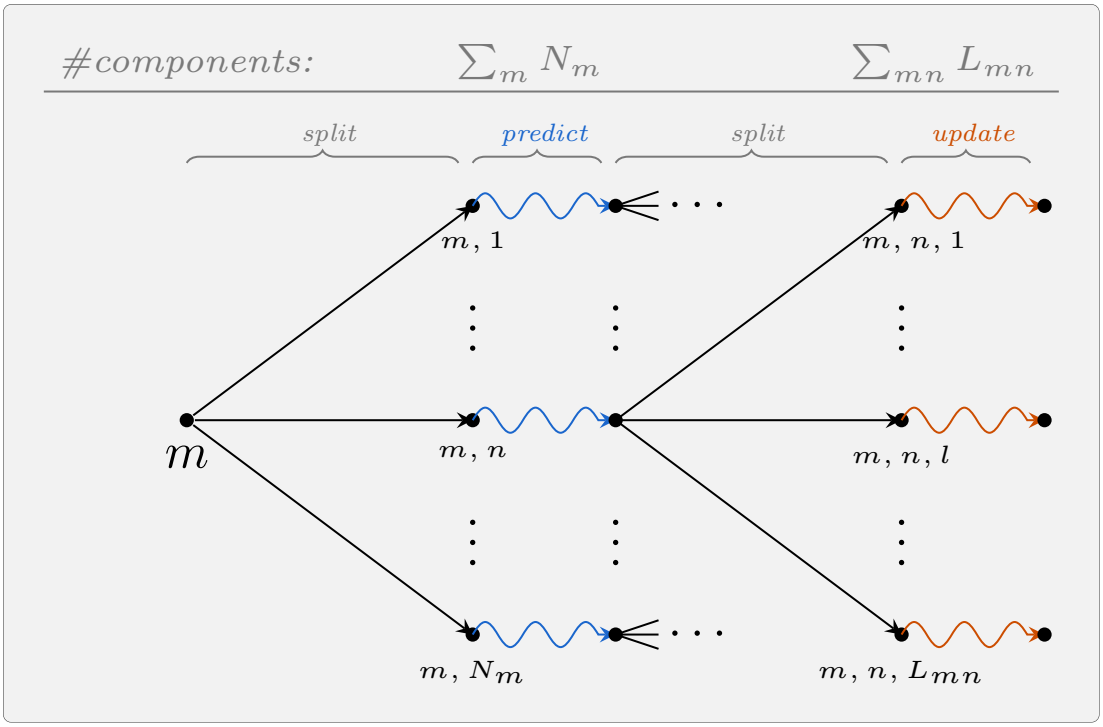


Figure 3.3: Splitting, prediction and update steps for a component of the AGSF.

The marginal approximation is given by

$$p_m(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \simeq \frac{1}{N_m}\sum_{n=1}^{N_m}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}^-_{t,mn}, \boldsymbol{\Sigma}^-_{t,mn}). \tag{3.86}$$

Hence from Eq. (3.81) we obtain the predictive distribution given in Eq. (3.90).

Update: Here we show how the filtering PDF in Eq. (3.93) is obtained. Using the update equation, Eq. (2.10), with the prior given by Eq. (3.90), we approximate the posterior of $\mathbf{x}_t$ as

$$\widehat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto \sum_{m=1}^{M}\sum_{n=1}^{N_m} w_{mn}\mathcal{N}(\mathbf{y}_t|\mathbf{g}(\mathbf{x}_t), \mathbf{R})\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}^-_{t,mn}, \boldsymbol{\Sigma}^-_{t,mn}). \tag{3.87}$$

As in the prediction step, we use the augmentation, to approximate the joint

$$p_{mn}(\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{y}_t|\mathbf{g}(\mathbf{x}_t), \mathbf{R})\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}^-_{t,mn}, \boldsymbol{\Sigma}^-_{t,mn})$$

by a Gaussian sum. Using Eq. (3.63) we obtain

$$p_{mn}(\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1}) \simeq \frac{1}{L_{mn}}\sum_{\ell=1}^{L_{mn}}\mathcal{N}_{nm\ell}(\mathbf{x}_t, \mathbf{y}_t), \tag{3.88}$$

where,

$$\mathcal{N}_{nm\ell}(\mathbf{x}_t, \mathbf{y}_t) = \mathcal{N}\left[\begin{pmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{pmatrix} \middle| \begin{pmatrix} \mathbf{s}_{mn\ell} \\ \boldsymbol{\mu}_{\mathbf{y},mn\ell} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Lambda}_{mn}^{(t)} & \mathbf{C}_{mn\ell} \\ \mathbf{C}_{mn\ell}^T & \mathbf{S}_{\mathbf{y},mn\ell} \end{pmatrix} \right], \tag{3.89}$$

and $\boldsymbol{\mu}_{\mathbf{y},mn\ell}, \mathbf{S}_{\mathbf{y},mn\ell}$ and $\mathbf{C}_{mn\ell}$ are given in Eqs. (3.97)-(3.99).

Combining Eqs. (3.87)-(3.88) and applying Eq. (3.151) from the Appendix for Chapter 3, we obtain Eq. (3.93) for the posterior of $\mathbf{x}_t$.

Resampling: During the prediction and update steps of our algorithm, the number of Gaussian components is increased. We start with M components before the prediction step to obtain $\sum_m N_m$ before and $\sum_{mn} L_{mn}$ after the update step, respectively. To ensure that our algorithm runs on a constant memory budget, we perform resampling at the end of the update step to reduce the number of components to M although other mechanisms could be performed, such as Gaussian mixture reduction [SH09; Cro+11a]. An iteration of the generic AGSF algorithm for one of the mixture components is graphically illustrated in Fig. 3.3

Linear AGSF

The generic algorithm presented in Sec. 3.4.1 can not be used in practice since the expectations of Eqs. (3.91)-(3.92) and (3.97)-(3.99) need to be approximated using some numerical scheme. We derive the linear-AGSF (L-AGSF) algorithm by using the linear Taylor approximation of the nonlinearities $\mathbf{f}$ and $\mathbf{g}$ for Eqs.(3.91)-(3.92) and (3.97)-(3.99). In this way, we obtain Eqs. (3.102)-(3.103) and (3.105)-(3.107). The L-AGSF algorithm is summarized in Alg. 8.

Unscented AGSF

We approximate the expectations in Eqs. (3.91)-(3.92) and (3.97)-(3.99) using the unscented approximation that we introduced in Sec. 3.3.2. For brevity, we omit the detailed version of the unscented AGSF, which however features in the numerical experiments in Section 3.5.

Algorithm 7 Augmented Gaussian sum filter

1: **Parameters:** N_m, $\mathbf{\Delta}_m^{(t)}$, and L_{mn}, $\mathbf{\Lambda}_{mn}^{(t)}$ for $m = 1, \ldots, M$ $n = 1, \ldots, N_m$ and $t = 1, \ldots, T$

2: **Initialization:** $\{w_0^{(m)}, \boldsymbol{\mu}_0^{(m)}, \mathbf{\Sigma}_0^{(m)}\}_{m=1}^M$

3: for $t = 1, \ldots, T$:

4: **Prediction:**

$$\widehat{p}(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \sum_{m=1}^{M}\sum_{n=1}^{N_m} w_{mn}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{t,mn}^-, \mathbf{\Sigma}_{t,mn}^-), \tag{3.90}$$

where

$$\boldsymbol{\mu}_{t,mn}^- = \mathcal{I}[\mathbf{f}(\mathbf{x}_{t-1}); \mathbf{z}_{mn}], \tag{3.91}$$
$$\mathbf{\Sigma}_{t,mn}^- = \mathcal{I}[(\mathbf{f}(\mathbf{x}_{t-1}) - \boldsymbol{\mu}_{t,mn}^-)(\mathbf{f}(\mathbf{x}_{t-1}) - \boldsymbol{\mu}_{t,mn}^-)^T; \mathbf{z}_{mn}] + \mathbf{Q}, \tag{3.92}$$

where $\mathbf{z}_{mn} \sim \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{t-1}^{(m)}, \mathbf{\Sigma}_{t-1}^{(m)} - \mathbf{\Delta}_m^{(t)})$ and $w_{mn} = w_{t-1}^{(m)}/N_m$, for $n = 1, \ldots, N_m$; $m = 1, \ldots, M$.

5: **Update:**

$$\widehat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \sum_{m=1}^{M}\sum_{n=1}^{N_m}\sum_{\ell=1}^{L_{mn}} w_{mnl}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{mn\ell}, \mathbf{\Sigma}_{mn\ell}), \tag{3.93}$$

where

$$w_{mnl} \propto (w_{mn}/L_{mn})\mathcal{N}(\mathbf{y}_t|\boldsymbol{\mu}_{\mathbf{y},mn\ell}, \mathbf{S}_{\mathbf{y},mn\ell}), \tag{3.94}$$
$$\boldsymbol{\mu}_{mn\ell} = \mathbf{s}_{mn\ell} + \mathbf{G}_{mn\ell}(\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y},mn\ell}), \tag{3.95}$$
$$\mathbf{\Sigma}_{mn\ell} = \mathbf{\Lambda}_{mn}^{(t)} - \mathbf{G}_{mn\ell}\mathbf{S}_{\mathbf{y},mn\ell}\mathbf{G}_{mn\ell}^T, \tag{3.96}$$

with,

$$\boldsymbol{\mu}_{\mathbf{y},mn\ell} = \mathcal{I}[\mathbf{g}(\mathbf{x}_t); \mathbf{s}_{mn\ell}, \mathbf{\Lambda}_{mn}^{(t)}], \tag{3.97}$$
$$\mathbf{S}_{\mathbf{y},mn\ell} = \mathcal{I}[(\mathbf{g}(\mathbf{x}_t) - \boldsymbol{\mu}_{\mathbf{y},mn\ell})(\mathbf{g}(\mathbf{x}_t) - \boldsymbol{\mu}_{\mathbf{y},mn\ell})^T; \mathbf{s}_{mn\ell}, \mathbf{\Lambda}_{mn}^{(t)}] + \mathbf{R}, \tag{3.98}$$

and,

$$\mathbf{C}_{mn\ell} = \mathcal{I}[(\mathbf{x}_t - \mathbf{s}_{mn\ell})(\mathbf{g}(\mathbf{x}_t) - \boldsymbol{\mu}_{\mathbf{y},mn\ell})^T; \mathbf{s}_{mn\ell}, \mathbf{\Lambda}_{mn}^{(t)}], \tag{3.99}$$

$$\mathbf{G}_{mn\ell} = \mathbf{C}_{mn\ell}\mathbf{S}_{\mathbf{y},mn\ell}^{-1}, \tag{3.100}$$

where $\mathbf{s}_{mn\ell} \sim \mathcal{N}(\mathbf{s}|\boldsymbol{\mu}_{t,mn}^-, \mathbf{\Sigma}_{t,mn}^- - \mathbf{\Lambda}_{mn}^{(t)})$ for $m = 1, \ldots, M, n = 1, \ldots, N_m$ and $\ell = 1, \ldots, L_{mn}$.

48: **Resampling:** For $m' = 1, \ldots, M$, sample a triplet $(mn\ell)$ with probability $w_{mn\ell}$ and set

$$\boldsymbol{\mu}_t^{(m')} = \boldsymbol{\mu}_{mn\ell} \;;\; \mathbf{\Sigma}_t^{(m')} = \mathbf{\Sigma}_{mn\ell}.$$

Set $w_t^{(m)} = 1/M$.

Algorithm 8 Linear AGSF

1: **Parameters:** N_m, $\boldsymbol{\Delta}_m^{(t)}$, and L_{mn}, $\boldsymbol{\Lambda}_{mn}^{(t)}$ for $m = 1, \ldots, M$ $n = 1, \ldots, N_m$ and $t = 1, \ldots, T$

2: **Initialization:** $\{w_0^{(m)}, \boldsymbol{\mu}_0^{(m)}, \boldsymbol{\Sigma}_0^{(m)}\}_{m=1}^{M}$

3: for $t = 1, \ldots, T$:

4: **Prediction:**

$$\widehat{p}(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \sum_{m=1}^{M}\sum_{n=1}^{N_m} w_{mn}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{t,mn}^{-}, \boldsymbol{\Sigma}_{t,mn}^{-}), \tag{3.101}$$

where

$$\boldsymbol{\mu}_{t,mn}^{-} = \mathbf{f}(\mathbf{z}_{mn}), \tag{3.102}$$
$$\boldsymbol{\Sigma}_{t,mn}^{-} = \nabla\mathbf{f}(\mathbf{z}_{mn})\boldsymbol{\Delta}_m^{(t)}\nabla\mathbf{f}(\mathbf{z}_{mn})^T + \mathbf{Q}, \tag{3.103}$$

for $\mathbf{z}_{mn} \sim \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)} - \boldsymbol{\Delta}_m^{(t)}), w_{mn} = w_{t-1}^{(m)}/N_m$, $n = 1, \ldots, N_m$; $m = 1, \ldots, M$.

5: **Update:**

$$\widehat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \sum_{m=1}^{M}\sum_{n=1}^{N_m}\sum_{\ell=1}^{L_{mn}} w_{mnl}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{mn\ell}, \boldsymbol{\Sigma}_{mn\ell}), \tag{3.104}$$

where

$$w_{mnl} \propto (w_{mn}/L_{mn})\mathcal{N}(\mathbf{y}_t|\boldsymbol{\mu}_{\mathbf{y},mn\ell}, \mathbf{S}_{\mathbf{y},mn\ell}),$$
$$\boldsymbol{\mu}_{mn\ell} = \mathbf{s}_{mn\ell} + \mathbf{G}_{mn\ell}(\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y},mn\ell}),$$
$$\boldsymbol{\Sigma}_{mn\ell} = \boldsymbol{\Lambda}_{mn}^{(t)} - \mathbf{G}_{mn\ell}\mathbf{S}_{\mathbf{y},mn\ell}\mathbf{G}_{mn\ell}^T,$$

with

$$\boldsymbol{\mu}_{\mathbf{y},mn\ell} = \mathbf{g}(\mathbf{s}_{mn\ell}), \tag{3.105}$$
$$\mathbf{S}_{\mathbf{y},mn\ell} = \nabla\mathbf{g}(\mathbf{s}_{mn\ell})\boldsymbol{\Lambda}_{mn}^{(t)}\nabla\mathbf{g}(\mathbf{s}_{mn\ell})^T + \mathbf{R}, \tag{3.106}$$
$$\mathbf{G}_{mn\ell} = \boldsymbol{\Lambda}_{mn}^{(t)}\nabla\mathbf{g}(\mathbf{s}_{mn\ell})^T\mathbf{S}_{\mathbf{y},mn\ell}^{-1}, \tag{3.107}$$

where $\mathbf{s}_{mn\ell} \sim \mathcal{N}(\mathbf{s}|\boldsymbol{\mu}_{t,mn}^{-}, \boldsymbol{\Sigma}_{t,mn}^{-} - \boldsymbol{\Lambda}_{mn}^{(t)})$ for $m = 1, \ldots, M, n = 1, \ldots, N_m$ and $\ell = 1, \ldots, L_{mn}$.

6: **Resampling:** For $m' = 1, \ldots, M$, sample a triplet $(mn\ell)$ with probability $w_{mn\ell}$ and set

$$\boldsymbol{\mu}_t^{(m')} = \boldsymbol{\mu}_{mn\ell}\ ;\ \boldsymbol{\Sigma}_t^{(m')} = \boldsymbol{\Sigma}_{mn\ell}. \tag{3.108}$$

Set $w_t^{(m)} = 1/M$.

3.4.2 The AGSF unifies GSF and BPF

The AGSF has the *bootstrap particle filter* and the *Gaussian sum fitler* as special cases. This is proved in the theorem that follows.

Theorem 3.3. *The AGSF unifies the GSF and BPF in the following way:*

(i) When at every timestep, the choices $N_m = 1, L_{mn} = 1$, *and* $\mathbf{\Delta}_m^{(t)} \to \mathbf{0}$, $\mathbf{\Lambda}_{mn}^{(t)} \to \mathbf{0}$ *are made for* $m = 1, \ldots, M$, *then Alg. 7 is identical to the BPF for an additive state-space model.*

(ii) When at every timestep, the choices $\mathbf{\Delta}_m^{(t)} = \mathbf{\Sigma}_{t-1}^{(m)}$, $\mathbf{\Lambda}_{mn}^{(t)} = \mathbf{\Sigma}_{t,mn}^{-}$ *are made for all* m, n, *then the prediction and update steps of Alg. 7 are identical to those of a GSF for an additive state-space model.*

Proof. Given in the Appendix for Chapter 3. □

Theorem 3.3 shows that the BPF and the GSF algorithms emerge as special cases in the AGSF framework for particular settings of the covariance parameters. The behaviour of the AGSF for intermediate settings of the parameters is an interpolation between the BPF and the GSF which is determined by the choice of the augmentation covariances, is discussed in the next section. Finally, we note that if in the definition of the augmented estimator of Eq. (3.49), instead of vanilla Monte-Carlo we use importance sampling [EM21] with some proposal of our choice, the resulting AGSF algorithm will be an interpolation between a GSF and a particle filter which uses this particular proposal, this however is not proved in this paper.

3.4.3 Strategies for setting the parameters of AGSF

The choice of the augmentation covariances plays an important role in the performance of the AGSF algorithms Algs. 7 and 8. These covariances have to be set at every time step of the AGSF algorithms and they have to satisfy the constraints

$$\mathbf{\Delta}_m^{(t)} \succeq \mathbf{0}, \ \mathbf{\Sigma}_{t-1}^{(m)} - \mathbf{\Delta}_m^{(t)} \succeq \mathbf{0}, \tag{3.109}$$

$$\mathbf{\Lambda}_{mn}^{(t)} \succeq \mathbf{0}, \ \mathbf{\Sigma}_{t,mn}^{-} - \mathbf{\Lambda}_{mn}^{(t)} \succeq \mathbf{0}, \tag{3.110}$$

for $m = 1, \ldots, M$, $n = 1, \ldots, N_m$ and $t = 1, \ldots, T$.

Moreover, the choices of $\mathbf{\Delta}_m^{(t)}$ and $\mathbf{\Lambda}_{mn}^{(t)}$, at time step t will determine whether the prediction and update steps, respectively, of the AGSF will resemble that of the BPF or the GSF, or if the AGSF operates in a novel, in between setting.

When $\boldsymbol{\Delta}_m^{(t)} \simeq \mathbf{0}$ and $\boldsymbol{\Lambda}_{mn}^{(t)} \simeq \mathbf{0}$ the behaviour will be more like that of a BPF, whereas when $\boldsymbol{\Delta}_m^{(t)} \simeq \boldsymbol{\Sigma}_{t-1}^{(m)}$ and $\boldsymbol{\Lambda}_{mn}^{(t)} \simeq \boldsymbol{\Sigma}_{t,mn}^{-}$ more like that of a GSF. When $\boldsymbol{\Delta}_m^{(t)}$ and $\boldsymbol{\Lambda}_{mn}^{(t)}$ are chosen in some intermediate values, i.e., not close to $\mathbf{0}$ or $\boldsymbol{\Sigma}_{t-1}^{(m)}$ and $\boldsymbol{\Sigma}_{t,mn}^{-}$ respectively, then the behaviour of the AGSF will be in between the two.

Hence it is reasonable to adopt the following heuristic. *At time step t:*

1. *if the function* $\mathbf{f}$ *is highly nonlinear we choose* $\boldsymbol{\Delta}_m^{(t)}$ *close to* $\mathbf{0}$*. If* $\mathbf{f}$ *is almost linear we choose* $\boldsymbol{\Delta}_m^{(t)}$ *close to* $\boldsymbol{\Sigma}_{t-1}^{(m)}$*, otherwise choose an intermediate* $\boldsymbol{\Delta}_m^{(t)}$*.*

2. *if the function* $\mathbf{g}$ *is highly nonlinear we choose* $\boldsymbol{\Lambda}_{mn}^{(t)}$ *close to* $\mathbf{0}$*. If* $\mathbf{g}$ *is almost linear we choose* $\boldsymbol{\Lambda}_{mn}^{(t)}$ *close to* $\boldsymbol{\Sigma}_{t,mn}^{-}$*, otherwise choose an intermediate* $\boldsymbol{\Lambda}_{mn}^{(t)}$*.*

Note that this intuition behind the heuristic is captured in the optimization problem of Eq. (3.76), where the quantification of the nonlinearity of $\mathbf{f}$ and $\mathbf{g}$ is done through the Hessian.

In our experiments, we use two different strategies to choose the covariances:

- *Proportional*: $\boldsymbol{\Delta}_m^{(t)} = \rho_1 \boldsymbol{\Sigma}_{t-1}^{(m)}$ for some $0 \leq \rho_1 \leq 1$ and $\boldsymbol{\Lambda}_{mn}^{(t)} = \rho_2 \boldsymbol{\Sigma}_{t,mn}^{-}$ for some $0 \leq \rho_2 \leq 1$.

- *Automatic:* We use the optimization problem in Eq. (3.76) or the approximate version of Eq. (3.78) to set the covariances automatically.

In both strategies the constraints in Eqs. (3.109)-(3.110) are satisfied. The two strategies can be used in any combination for choosing $\boldsymbol{\Delta}_m^{(t)}$ and $\boldsymbol{\Lambda}_{mn}^{(t)}$. For example we may at each time-step choose $\boldsymbol{\Delta}_m^{(t)}$ proportionally, i.e., $\boldsymbol{\Delta}_m^{(t)} = \rho \boldsymbol{\Sigma}_{t-1}^{(m)}$, while choosing $\boldsymbol{\Lambda}_{mn}^{(t)}$ automatically, using the optimization problem.

3.5 Numerical Experiments

In this section, we present experimental results involving the novel AGSF algorithms as well as the classical GSF, BPF, and APF algorithms. In Experiment A, we compare the performance of the various algorithms for a classical maneuvering target tracking application. In Experiment B, we present an ablation study for the proportionality parameters ρ_1 and ρ_2, highlighting how the performance of the AGSF algorithm varies, and how it interpolates between the GSF and BPF for particular choices of ρ_1 and ρ_2. Finally, in Experiment C, we evaluate the performance of an adaptive AGSF algorithm which selects the augmentation covariances automatically by exploiting Eq. (3.78). All algorithms are implemented in JAX [Bra+18] and the experiments are run on an Apple M2 Pro CPU.

Experiment A: Maneuvering target tracking

In this section, we consider a classical example from the signal processing literature, namely that of the tracking a maneuvering target. We consider the problem of tracking an object in 2D, given measurements of its bearing with respect to the sensor and range , i.e. its distance from the sensor. We describe this with the following state-space model,

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{G}\mathbf{q}_t, \tag{3.111}$$

$$\mathbf{y}_t = \begin{pmatrix} \sqrt{x_{1t}^2 + x_{2t}^2} \\ \tan^{-1}(x_{2t}/x_{1t}) \end{pmatrix} + \mathbf{r}_t, \tag{3.112}$$

where $\mathbf{x}_t = (x_{1t}, v_{1t}, x_{2t}, v_{2t})$ is the 4-dimensional state vector, (x_{1t}, x_{2t}) is the position vector of the object and (v_{1t}, v_{2t}) the velocity vector.

The motion of the maneuvering object is allowed to switch between the *constant-velocity* (CV) and *constant-turn* (CT) modes, described by the matrices,

$$\mathbf{F}_{CV} = \begin{pmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.113}$$

and

$$\mathbf{F}_{\pm}^{CT} = \begin{pmatrix} 1 & \frac{\sin(\Omega_t^{\pm} dt)}{\Omega_t^{\pm}} & 0 & -\frac{1-\cos(\Omega_t^{\pm} dt)}{\Omega_t^{\pm}} \\ 0 & \cos(\Omega_t^{\pm} dt) & 0 & -\sin(\Omega_t^{\pm} dt) \\ 0 & \frac{1-\cos(\Omega_t^{\pm} dt)}{\Omega_t^{\pm}} & 1 & \frac{\sin(\Omega_t^{\pm} dt)}{\Omega_t^{\pm}} \\ 0 & \sin(\Omega_t^{\pm} dt) & 0 & \cos(\Omega_t^{\pm} dt) \end{pmatrix}, \tag{3.114}$$

where,

$$\Omega_t^{\pm} = \pm \frac{a}{\sqrt{v_{1t}^2 + v_{2t}^2}}, \tag{3.115}$$

$dt = 1.0$,. In experiments A.1 and A.2 we choose $a = 0.5$ and $a = 0.05$ respectively. The dynamics of the target are defined by the matrix,

$$\mathbf{F}_t = \begin{cases} \mathbf{F}_{+}^{CT}, & t \leq 2T/5 \\ \mathbf{F}_{CV}, & 2T/5 < t \leq 3T/5 \\ \mathbf{F}_{-}^{CT}, & 3T/5 < t \leq T \end{cases}, \tag{3.116}$$

which describes an object doing a clockwise circular motion, followed by a con-

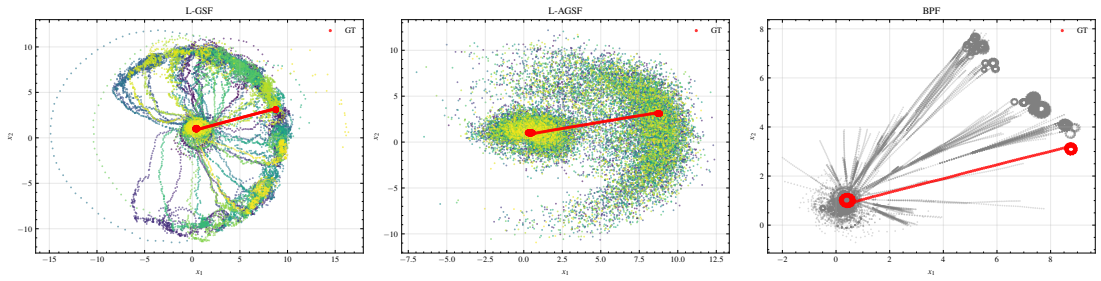


Figure 3.4: Comparison of the estimated trajectories between L-GSF, L-AGSF, and BPF for the same example run of Experiment A.2. The actual object trajectory is shown in red.

stant velocity motion, followed by an anti-clockwise circular motion.

The dynamical noise vector $\mathbf{q}_t$ is 2-dimensional with covariance matrix $\mathbf{Q} = 10^{-6}\mathbf{I}_2$. Finally,

$$\mathbf{G} = \begin{pmatrix} 0.5 & 1 \\ 1 & 0 \\ 0 & 0.5 \\ 0 & 1 \end{pmatrix}, \tag{3.117}$$

and $\mathbf{R} = \sigma^2\mathbf{I}_2$, where σ^2 takes values in $\{25 \times 10^{-1}, 25 \times 10^{-3}, 25 \times 10^{-6}\}$. All model parameters and switching behaviours are known by all filters.

We compare the estimation performance for the various algorithms. To measure estimation performance we use two metrics, the mean-squared error (MSE), and the log-probability error (LPE), both averaged across time that are defined below.

$$\text{MSE(alg)} = \frac{1}{T}\sum_{t=1}^{T} ||(\mathbf{x}_{t,i} - \widehat{\mathbf{x}}_{t,i}^{(\text{alg})})||^2, \tag{3.118}$$

$$\text{LPE(alg)} = \frac{1}{T}\sum_{t=1}^{T} -\log \widehat{p}_{\text{alg}}(\mathbf{x}_t|\mathbf{y}_{1:t-1}) \tag{3.119}$$

where alg $\in$ {L-GSF, U-GSF, L-AGSF, U-AGSF, BPF, APF}, $\widehat{p}_{\text{alg}}(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ is the filtered posterior estimate of the algorithm and $\widehat{\mathbf{x}}_t$ the mean estimate. The MSE measures the accuracy of the point estimate (here, the posterior mean), i.e. how close $\widehat{\mathbf{x}}_{t,i}^{(\text{alg})}$ is to the true latent state $\mathbf{x}_{t,i}$ on average across time. In contrast, the LPE allows us to assess the quality of the entire filtered distribution $\widehat{p}_{\text{alg}}(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})$, by penalizing algorithms that assign low probability density to the realised (true) state; this captures both miscalibration and over/under-confidence, not just errors in the mean.

In each experiment, we consider three different settings for each algorithm, varying the number of components or particles used. For each algorithm setting we do $N_{\text{sim}} = 100$ simulations of the state-space model in Eqs. (3.111)-(3.112) for $T = 500$ timesteps. In both experiments (A.1 and A.2), augmentation covariances for the prediction and update steps are chosen to be proportional to the filter covariance with $\rho_1 = \rho_2 = 0.9$. We report the error metrics for Experiments A.1 and A.2 in Tables 3.2 and 3.3 respectively. In several runs of the L-GSF algorithm, the filters diverged producing NAN outputs. When this happened we have included in parentheses the percentage of successful (non-divergent) runs.

Our results show that the AGSF is the most robust method across scenarios compared to PFs and GSFs. One the one hand, PFs often exhibit extreme particle degeneracy (even with resampling), which leads to an overconfident posterior estimate often localized far from the true state. On the other hand GSFs show an opposite tendency: large component covariances assign non-vanishing weights to small likelihood components, leading to very diffuse posteriors. The AGSF balances these two extreme behaviors in an efficient way, using the augmentation covariances to avoid degeneracy on the one hand, while limiting overall posterior spread on the other. These behaviors are illustrated in the example of Fig. 3.4.

These claims are supported by our experimental results shown in Tables 3.2 and 3.3. Across all scenarios the PFs exhibit degeneracy leading to overconfident estimates, but the frequency and severity depend strongly on both the observation noise and the particle number. In the higher- and moderate-noise settings ($\sigma^2 = 25 \cdot 10^{-1}$ and $\sigma^2 = 25 \cdot 10^{-3}$), this occurs for particle counts up to $M = 10^4$, and its probability decreases as M increases. In the small-noise regime ($\sigma^2 = 25 \cdot 10^{-6}$), however, degeneracy remains present even at $M = 10^5$ with non-vanishing probability. This is due to the form of the weights of the BPF and APF, which are proportional to the observation distribution $p(\mathbf{y}_t|\mathbf{x}_t)$. When the observation noise is very small, this is a very narrow Gaussian, and thus it assigns non-vanishing weights to particles that are closest to the observation-consistent manifold, while assigning essentially zero weight to the vast majority of particles. As a result, after normalization and resampling, only a tiny subset of particles survives, the effective sample size collapses, and the filter becomes prone to locking onto an incorrect, overconfident mode if the proposed particles happen to lie away from the true state. For the GSF family, there is an opposite tendency: due to large component covariances non-negligible weights are assigned to components with low likelihood; effectively, only a small subset of samples meaningfully contributes to the posterior estimate at each step. This produces very broad posterior estimates which often diverge.

The AGSF variants sit between these two extremes in a way that bolster their interpretation as an interpolation between PFs and GSFs. In regimes where PFs require massive budgets to be accurate, AGSF attains much smaller LPE at modest mixture sizes while keeping MSE in a reasonable range; and in the low-noise settings where PFs collapse catastrophically, AGSF remains stable and continues to assign high probability to the realised state, without becoming as diffuse as the GSF. Overall, the results support the claim that AGSF interpolates between the PF's overly concentrated and the GSF's overly diffuse behavior, producing posterior estimates that are better balanced and more protected towards catastrophic failure.

Experiment B: Ablation Study

In this section, we perform a systematic evaluation of the proportional AGSF algorithms in terms of the parameters ρ_1 and ρ_2. We use the state-space model,

$$\mathbf{x}_t = \mathbf{\Phi}\mathbf{x}_{t-1} + \mathbf{q}_t, \tag{3.120}$$

$$\mathbf{y}_t = u_t \mathbf{V}_t \mathbf{r}_t + (1 - u_t)(\mathbf{H}\mathbf{x}_t + \mathbf{r}_t) \tag{3.121}$$

where,

$$\mathbf{V}_t = \beta \mathrm{diag}(e^{\mathbf{x}_{t,1}/\sigma}, \ldots, e^{\mathbf{x}_{t,d_x}/\sigma}), \tag{3.122}$$

and $u_t \in [0, 1]$ is a known input parameter that allows the model to switch from a linear-Gaussian (LG) model for $u_t = 0$, to a multivariate stochastic volatility (MSV) model for $u_t = 1$ [COA09]. We use the parameter values, $\mathbf{\Phi} = 0.8\mathbf{I}$, $\mathbf{H} = \mathbf{I}$, $\beta = 0.5$, $\sigma = 4.0$, $\mathbf{Q} = 10\mathbf{I}$ and $\mathbf{R} = 3 \times 10^{-2}\mathbf{I}$. The observation noise vector has mean $\mathbf{r}_0 = 10^{-4}\mathbf{1}$ proportional to the all-ones vector. We set the dimensionality of the system to $d_x = d_y = 4$ and use the inputs $u_t = \sin^2(0.1t)$. This defines a switching behaviour that periodically shifts from LG to MSV model. For each pair of parameters (ρ_1, ρ_2) in,

$$\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}^2, \tag{3.123}$$

we simulate the state space model for $T = 200$ timesteps and run the L-AGSF and U-AGSF algorithms with $M = 10, N = 5, L = 5$ for $N_{\mathrm{sim}} = 100$ repetitions. We report our results in the heatmaps of Fig. 3.5. To obtain each of the squares in the 11×11 grids we compute the MSE, defined in Eq. (3.118), for the L-AGSF (top) and U-AGSF (bottom) algorithms.

Moreover, we have included error values for the GSF and the BPF, as well as

Table 3.1: **Experiment B.** MSE and runtime for some settings of the GSF, BPF, and AGSF algorithms. The AGSF is run with $M = 10, N = 5, L = 5$ components.

	PARAMETERS	MSE	time(s)
L-GSF	$M = 1$	$89.001 \pm 35.888(100\%)$	0.25 ± 0.02
	$M = 10$	$66.884 \pm 15.829(100\%)$	0.32 ± 0.05
	$M = 1000$	$66.884 \pm 15.829(100\%)$	0.60 ± 0.05
U-GSF	$M = 1$	$89.074 \pm 35.886(100\%)$	0.70 ± 1.10
	$M = 10$	$69.173 \pm 17.106(100\%)$	1.75 ± 7.12
	$M = 1000$	$73.668 \pm 28.392(100\%)$	2.34 ± 0.10
L-AGSF	$\rho_1 = 0.9, \rho_2 = 0.4$	$\mathbf{6.582 \pm 0.145(100\%)}$	1.02 ± 0.07
	$\rho_1 = 0.95, \rho_2 = 0.95$	$19.632 \pm 6.624(100\%)$	1.03 ± 0.08
	$\rho_1 = 0.05, \rho_2 = 0.05$	$17.258 \pm 1.587(100\%)$	1.02 ± 0.09
U-AGSF	$\rho_1 = 0.9, \rho_2 = 0.4$	$6.582 \pm 0.141(100\%)$	1.30 ± 0.07
	$\rho_1 = 0.95, \rho_2 = 0.95$	$19.633 \pm 6.621(100\%)$	1.35 ± 0.11
	$\rho_1 = 0.05, \rho_2 = 0.05$	$17.321 \pm 2.587(100\%)$	1.28 ± 0.13
BPF	$M = 10$	$126.556 \pm 2.889(100\%)$	0.46 ± 0.04
	$M = 1000$	$58.781 \pm 5.428(100\%)$	0.64 ± 0.04
	$M = 100000$	$9.184 \pm 1.438(100\%)$	6.56 ± 0.12

some runs of the AGSF algorithm in Table 3.1. We report the best value run of the L-AGSF and U-AGSF algorithms which occurs for $(\rho_1 = 0.9, \rho_2 = 0.4)$, as well as the corner cases $(\rho_1 = 0.95, \rho_2 = 0.95)$ and $(\rho_1 = 0.05, \rho_2 = 0.05)$ which correspond to L-GSF/U-GSF and BPF, respectively. We have included tables with the MSE values that correspond to Fig. 3.5 in the supplementary material.

Theorem 3.3 suggests that the performance of the AGSF algorithms should coincide with the GSF (with $M = 10$) when $\rho_1 \to 1$ and $\rho_2 \to 1$, and with the BPF (with $M = 10$) when $\rho_1 \to 0$ and $\rho_2 \to 0$. This is confirmed nicely in top right corner of the plot, which corresponds to the case $\rho_1 \simeq 1$ and $\rho_2 \simeq 1$. It is also confirmed in the bottom left corner corresponding to $\rho_1 \simeq 0$ and $\rho_2 \simeq 0$[2]. Moreover, we see that for $\rho_2 = 0.05, 0.1$ the high error persists for all values of ρ_1. Finally, we note that for a large range of parameter values, the AGSF algorithms have a smaller MSE than both GSF and BPF algorithms.

[2]In the implementation of our algorithm, we set ρ_1 and ρ_2 close to 0 and 1 instead of using the exact values to avoid value errors due to covariance matrices.

Experiment C: Adaptive AGSF

In this section, we present the adaptive version of the AGSF and evaluate its performance against other algorithms. We show that by using Eq. (3.78) the adaptive AGSF algorithm is able to change its behaviour in real time to locally match the performance of a GSF or a BPF. This allows the adaptive AGSF to achieve better performance than the GSF and BPF in linear and nonlinear modes, respectively.

We evaluate the algorithms in the state-space model defined by Eqs. (3.120)-(3.122), which uses an input parameter $u_t \in [0, 1]$ to switch from a linear-Gaussian model to a multivariate stochastic volatility model. We evaluate the algorithms for two different switching behaviours, which we call Experiment C.1 and Experiment C.2, defined by

$$u_t^{(C.1)} = (1 - \sin(0.2t))/2, \tag{3.124}$$

$$u_t^{(C.2)} = \mathbb{I}[t > T/2], \tag{3.125}$$

respectively. In both experiments we use $d_x = d_y = 10$ for the state and observation dimensions, and use $\mathbf{R} = 0.1\mathbf{I}$ for the observation noise covariance. In Experiment C.1, we gradually move from a LG to a MSV model in a periodic fashion and run the model for $T_1 = 140$ timesteps. In Experiment C.2, we run the model for $T_2 = 200$ timesteps and have a discontinuous change from $u_t = 0$ to $u_t = 1$, with the change point at $t = 100$. The AGSF algorithms are run with proportional augmentation covariances in the the prediction step with $\rho_1 = 0.9$, i.e., $\mathbf{\Delta}_m^{(t)} = 0.9\mathbf{\Sigma}_{t-1}^{(m)}$ for all t. In the update step the augmentation covariances are chosen proportional, with ρ_2 determined via Eq. (3.78) with $\gamma = 10^{-6}$

We report our results for Experiment C.1 and C.2 in Table 3.6. As can be seen in the tables the AGSF algorithms outperform the GSF and BPF algorithm in all cases. We also plot the error in time for the various algorithms in the top left and right panels of Fig. 3.6. As seen from the figures, the AGSF algorithm switches its behavior to match the best algorithm for the linear and nonlinear models. In the bottom right and left panels of Fig. 3.6 we plot the coefficient ρ_2 for Experiment C.1 and C.2 respectively. As we can see the coefficients are close to one when the model is linear and drop to zero when the model becomes nonlinear.

To the best of our knowledge this type of adaptive behaviour is novel and does not exist elsewhere in the literature. This experiment is a proof-of-concept for combining two filtering algorithms in this way, and the results are encouraging.

We are excited about the potential applications that this approach can have in multiple fields where Bayesian filtering is routinely used.

3.6 Conclusions

We have proposed the augmented Gaussian sum filter , a novel class of Bayesian filters which unifies Gaussian sum filters and particle filters. The algorithm utilizes a novel augmented Gaussian approximation which is based on introducing a set of latent variables and associated covariance parameters. By tuning the covariances we are able to interpolate between GSFs and PFs which are limiting cases in the class. The AGSF is able to have the best of both worlds, optimally exploiting the trade-off between bias and variance. This allows for the design of a novel adaptive algorithm which behaves more like a GSF or a PF according to the local nature of the nonlinearities. We note that for notational simplicity all algorithms have been derived for an additive SSM with the generalization to non-additive being straightforward. We have demonstrated experimentally the performance advantages of the AGSF in a target-tracking application and for state-space models possessing linear and strong nonlinear modes. In the future we are very interested in applying the AGSF for model selection and change-detection problems, as well as on datasets coming from realistic applications. Moreover, there are multiple possible extensions to the AGSF framework that we are considering. Among these are combination with more sophisticated importance sampling strategies, as well as adaptation of the number of particles in real time, resulting in a more efficient version of the AGSF.

Table 3.2: **Experiment A.1.** Comparison between the MSE and LPE for various settings of the L-GSF, U-GSF, and BPF with the proposed L-AGSF and U-AGSF algorithms. We compare the error metrics for $a = 0.5$ and three observation noise values. Lower values are better.

		$\sigma^2 = 25 \times 10^{-1}$		$\sigma^2 = 25 \times 10^{-3}$		$\sigma^2 = 25 \times 10^{-6}$	
	PARAMETERS	MSE	LPE	MSE	LPE	MSE	LPE
L-GSF	$M = 1 (EKF)$	201.47 ± 62.68	$3.07 \times 10^4 \pm 4.16 \times 10^3$	19.42 ± 8.65	$9.13 \times 10^3 \pm 2.18 \times 10^3$	0.21 ± 0.15	$9.42 \times 10^3 \pm 6.17 \times 10^3$
	$M = 100$	62.48 ± 13.46	972.78 ± 84.06 (50.0%)	69.19 ± 35.37	$3.07 \times 10^3 \pm 2.56 \times 10^3$ (95.0%)	0.44 ± 0.43	$1.77 \times 10^4 \pm 1.33 \times 10^4$
	$M = 1000$	101.22 ± 21.27	N/A (0%)	12.38 ± 11.04	92.67 ± 8.42 (90.0%)	$0.02 \pm 6.56 \times 10^{-3}$	$1.08 \times 10^3 \pm 138.07$
U-GSF	$M = 1 (UKF)$	15.37 ± 8.60	$6.18 \times 10^5 \pm 4.68 \times 10^5$	4.13 ± 1.97	$9.55 \times 10^3 \pm 8.92 \times 10^3$	$0.02 \pm 5.39 \times 10^{-3}$	713.35 ± 293.59
	$M = 100$	16.40 ± 8.57	0.73 ± 1.46	0.25 ± 0.13	21.29 ± 6.04	0.06 ± 0.04	144.36 ± 79.03
	$M = 1000$	22.07 ± 9.20	0.59 ± 0.99	0.40 ± 0.15	22.51 ± 6.99	$7.10 \times 10^{-3} \pm 1.42 \times 10^{-3}$	73.93 ± 39.61
L-AGSF	$M = 2, N = 5, L = 5$	75.62 ± 19.18	97.68 ± 30.62	1.29 ± 0.67	8.32 ± 8.57	$0.06 \pm 9.83 \times 10^{-3}$	-3.34 ± 0.06
	$M = 10, N = 5, L = 5$	21.92 ± 5.92	17.36 ± 3.44	0.33 ± 0.05	-0.48 ± 0.09	$0.02 \pm 1.40 \times 10^{-3}$	-3.44 ± 0.03
	$M = 100, N = 5, L = 5$	7.51 ± 2.23	5.25 ± 0.45	0.33 ± 0.05	-0.46 ± 0.07	$0.01 \pm 1.67 \times 10^{-3}$	-3.44 ± 0.02
U-AGSF	$M = 2, N = 5, L = 5$	38.42 ± 13.29	108.67 ± 51.99	0.50 ± 0.14	1.11 ± 1.93	$0.02 \pm 2.26 \times 10^{-3}$	-6.87 ± 0.24
	$M = 10, N = 5, L = 5$	34.53 ± 10.29	72.97 ± 34.87	0.21 ± 0.04	-1.86 ± 0.13	$0.01 \pm 1.27 \times 10^{-3}$	-7.21 ± 0.16
	$M = 100, N = 5, L = 5$	7.14 ± 1.92	4.52 ± 0.95	0.16 ± 0.02	-2.09 ± 0.07	$0.01 \pm 1.08 \times 10^{-3}$	-7.00 ± 0.17
BPF	$M = 100$	46.03 ± 29.72	$4.55 \times 10^6 \pm 2.49 \times 10^6$	15.41 ± 5.97	$6.69 \times 10^6 \pm 2.85 \times 10^6$	40.64 ± 21.23	$1.81 \times 10^7 \pm 1.09 \times 10^7$
	$M = 1000$	5.29 ± 2.94	$3.99 \times 10^5 \pm 3.14 \times 10^5$	3.71 ± 1.98	$1.42 \times 10^6 \pm 9.45 \times 10^5$	74.06 ± 37.09	$3.82 \times 10^7 \pm 2.15 \times 10^7$
	$M = 10000$	1.46 ± 0.65	$3.98 \times 10^3 \pm 3.92 \times 10^3$	$0.03 \pm 7.26 \times 10^{-3}$	66.97 ± 25.61	74.21 ± 33.05	$3.71 \times 10^7 \pm 1.48 \times 10^7$
	$M = 100000$	0.76 ± 0.20	98.03 ± 13.06	$0.03 \pm 5.06 \times 10^{-3}$	21.76 ± 10.65	21.13 ± 12.32	$1.10 \times 10^7 \pm 6.84 \times 10^6$
APF	$M = 100$	102.30 ± 31.25	$2.77 \times 10^7 \pm 8.34 \times 10^6$	15.49 ± 4.21	$6.38 \times 10^6 \pm 1.94 \times 10^6$	116.62 ± 34.29	$5.94 \times 10^7 \pm 1.59 \times 10^7$
	$M = 1000$	43.49 ± 23.42	$1.12 \times 10^7 \pm 6.06 \times 10^6$	20.31 ± 9.88	$8.62 \times 10^6 \pm 3.96 \times 10^6$	158.41 ± 101.61	$6.97 \times 10^7 \pm 4.64 \times 10^7$
	$M = 10000$	4.29 ± 2.95	$9.81 \times 10^5 \pm 1.05 \times 10^6$	1.68 ± 1.37	$6.48 \times 10^5 \pm 4.86 \times 10^5$	62.74 ± 38.49	$3.45 \times 10^7 \pm 2.10 \times 10^7$
	$M = 100000$	0.94 ± 0.23	349.70 ± 90.77	$0.03 \pm 9.04 \times 10^{-3}$	826.71 ± 712.70	0.94 ± 1.03	$4.66 \times 10^5 \pm 3.94 \times 10^5$

Table 3.3: **Experiment A.2.** Comparison between the MSE and LPE for various settings of the L-GSF, U-GSF, and BPF with the proposed L-AGSF and U-AGSF algorithms. We compare the error metrics for $a = 0.05$ and three observation noise values. Lower values are better.

		$\sigma^2 = 25 \times 10^{-1}$		$\sigma^2 = 25 \times 10^{-3}$		$\sigma^2 = 25 \times 10^{-6}$	
	PARAMETERS	MSE	LPE	MSE	LPE	MSE	LPE
L-GSF	$M = 1 (EKF)$	178.20 ± 46.16	$3.38 \times 10^3 \pm 896.60$	2.20 ± 1.90	669.81 ± 610.31	0.03 ± 0.02	465.31 ± 215.72
	$M = 100$	52.11 ± 9.41	25.12 ± 3.75	0.35 ± 0.17	15.34 ± 8.24	9.27 ± 9.12	$4.42 \times 10^5 \pm 4.69 \times 10^5$
	$M = 1000$	66.80 ± 20.83	4.73 ± 1.10 (95.0%)	0.30 ± 0.18	14.20 ± 15.84	$1.21 \times 10^{-3} \pm 7.16 \times 10^{-4}$	89.39 ± 96.67
U-GSF	$M = 1 (UKF)$	87.79 ± 48.81	$1.80 \times 10^4 \pm 1.60 \times 10^4$	2.95 ± 2.45	$1.00 \times 10^5 \pm 9.38 \times 10^4$	0.07 ± 0.06	567.01 ± 409.18
	$M = 100$	6.21 ± 4.63	7.26 ± 7.20	0.52 ± 0.33	-6.95 ± 0.62	$1.28 \times 10^{-3} \pm 4.92 \times 10^{-4}$	-15.07 ± 0.19
	$M = 1000$	44.40 ± 20.59	-1.59 ± 0.57	0.16 ± 0.05	-7.36 ± 0.45	$1.24 \times 10^{-3} \pm 4.46 \times 10^{-4}$	-15.37 ± 0.14
L-AGSF	$M = 2, N = 5, L = 5$	94.86 ± 28.93	86.91 ± 36.54	6.01 ± 5.06	58.43 ± 65.61	$7.97 \times 10^{-3} \pm 4.17 \times 10^{-4}$	-4.17 ± 0.02
	$M = 10, N = 5, L = 5$	21.45 ± 6.45	13.16 ± 6.14	0.56 ± 0.08	-1.28 ± 0.08	$3.60 \times 10^{-3} \pm 4.32 \times 10^{-4}$	-4.19 ± 0.02
	$M = 100, N = 5, L = 5$	9.88 ± 1.57	3.07 ± 0.07	0.50 ± 0.07	-1.37 ± 0.10	$2.93 \times 10^{-3} \pm 3.77 \times 10^{-4}$	-4.16 ± 0.02
U-AGSF	$M = 2, N = 5, L = 5$	54.71 ± 15.30	86.04 ± 30.01	16.88 ± 16.00	404.43 ± 349.29	$9.21 \times 10^{-3} \pm 1.34 \times 10^{-3}$	-7.49 ± 0.17
	$M = 10, N = 5, L = 5$	33.79 ± 8.33	36.29 ± 11.35	0.38 ± 0.07	-2.12 ± 0.09	$5.03 \times 10^{-3} \pm 4.49 \times 10^{-4}$	-7.56 ± 0.08
	$M = 100, N = 5, L = 5$	8.49 ± 1.72	2.52 ± 0.21	0.33 ± 0.04	-2.16 ± 0.07	$2.89 \times 10^{-3} \pm 2.98 \times 10^{-4}$	-7.62 ± 0.10
BPF	$M = 100$	42.78 ± 21.52	$1.61 \times 10^6 \pm 8.85 \times 10^5$	57.56 ± 35.13	$2.02 \times 10^7 \pm 1.36 \times 10^7$	124.04 ± 47.64	$5.98 \times 10^7 \pm 2.70 \times 10^7$
	$M = 1000$	2.21 ± 0.76	$1.99 \times 10^3 \pm 1.29 \times 10^3$	97.77 ± 59.90	$4.20 \times 10^7 \pm 2.61 \times 10^7$	151.37 ± 45.90	$8.16 \times 10^7 \pm 2.66 \times 10^7$
	$M = 10000$	1.34 ± 0.36	$1.29 \times 10^3 \pm 1.24 \times 10^3$	1.74 ± 1.08	$4.53 \times 10^5 \pm 3.82 \times 10^5$	159.01 ± 47.90	$7.47 \times 10^7 \pm 2.10 \times 10^7$
	$M = 100000$	1.92 ± 0.80	90.31 ± 27.90	$0.02 \pm 3.01 \times 10^{-3}$	7.95 ± 8.45	98.94 ± 32.24	$4.71 \times 10^7 \pm 1.36 \times 10^7$
APF	$M = 100$	45.40 ± 13.02	$2.21 \times 10^6 \pm 5.56 \times 10^5$	70.76 ± 32.19	$2.96 \times 10^7 \pm 1.44 \times 10^7$	164.74 ± 30.83	$8.37 \times 10^7 \pm 1.52 \times 10^7$
	$M = 1000$	115.32 ± 92.06	$9.59 \times 10^6 \pm 8.30 \times 10^6$	18.87 ± 17.53	$8.94 \times 10^6 \pm 8.26 \times 10^6$	159.32 ± 34.55	$7.65 \times 10^7 \pm 1.70 \times 10^7$
	$M = 10000$	1.13 ± 0.50	$5.80 \times 10^3 \pm 2.93 \times 10^3$	7.49 ± 6.74	$3.34 \times 10^6 \pm 3.18 \times 10^6$	24.77 ± 12.20	$1.30 \times 10^7 \pm 5.86 \times 10^6$
	$M = 100000$	0.78 ± 0.24	183.63 ± 44.74	$0.02 \pm 3.31 \times 10^{-3}$	11.56 ± 11.28	68.97 ± 43.13	$3.61 \times 10^7 \pm 2.35 \times 10^7$

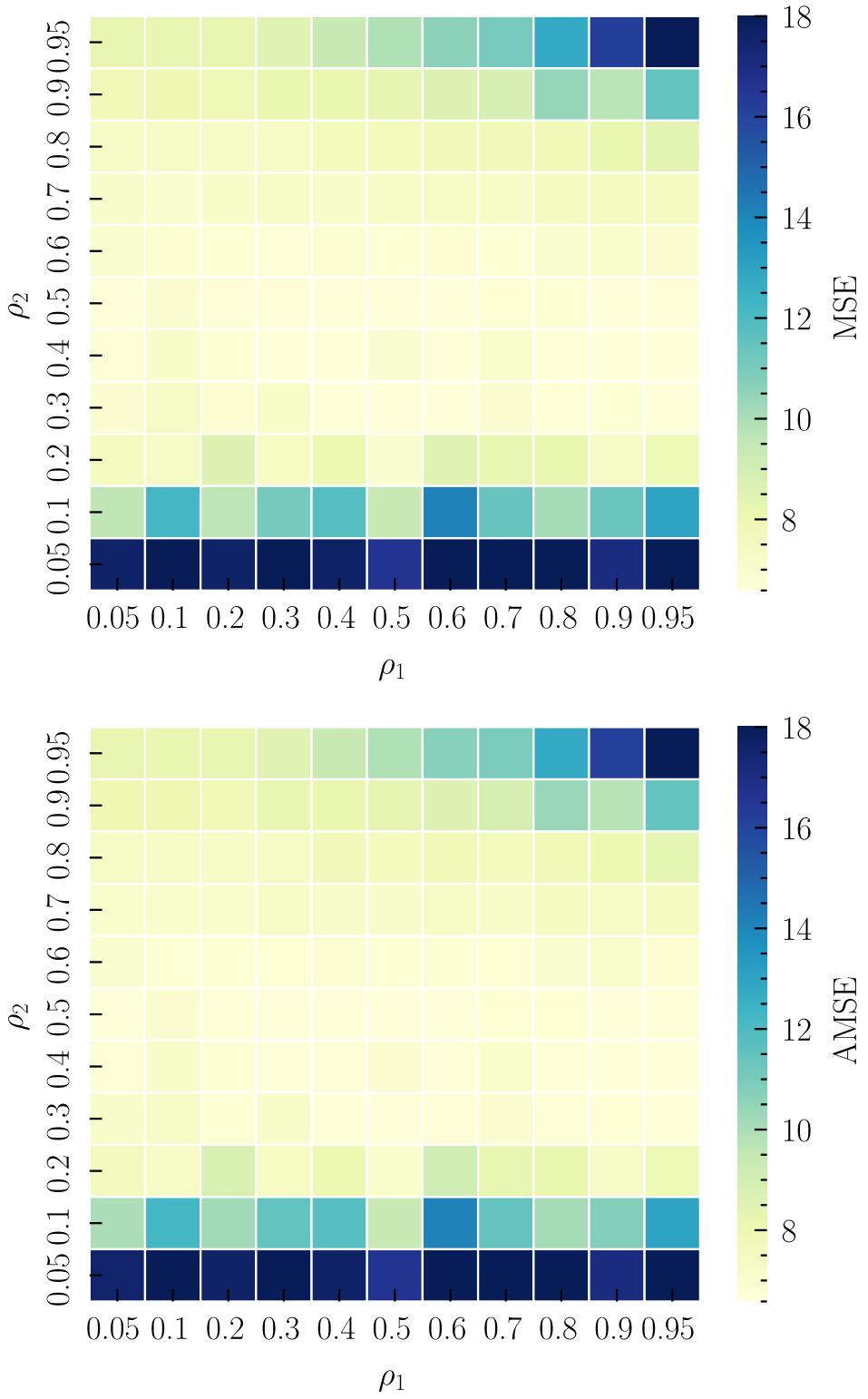


Figure 3.5: **Experiment B.** Ablation study for the parameters ρ_1 and ρ_2. Each square depicts the value of the MSE (Eq.(3.118)) for a given pair (ρ_1, ρ_2) *(Top)* L-AGSF *(Bottom)* U-AGSF.

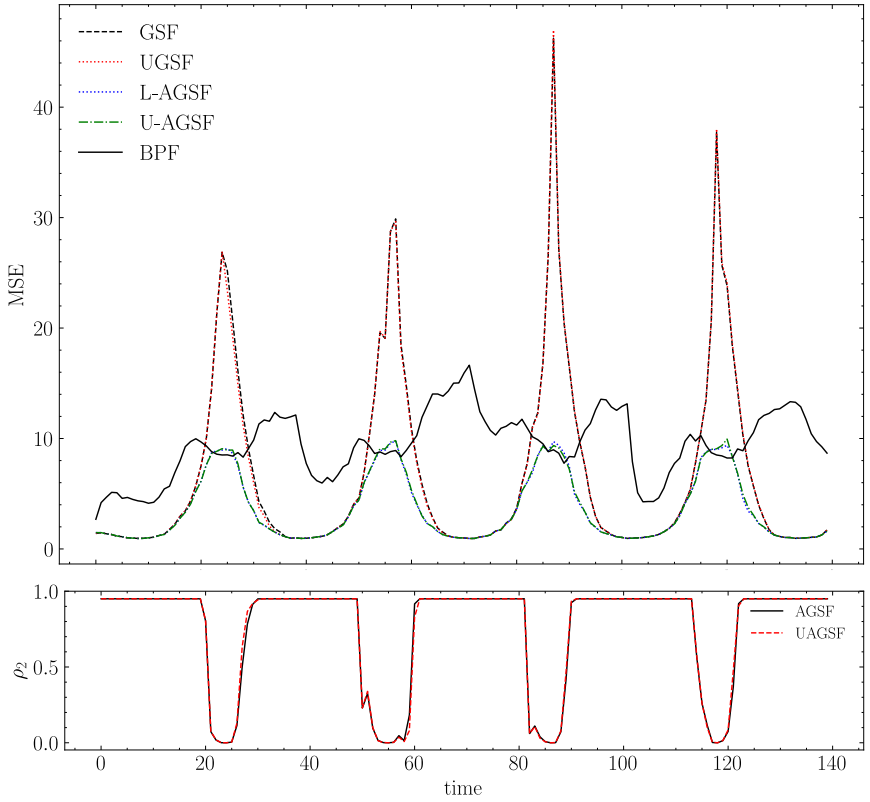
Table 3.4: Comparison between the MSE of various settings of the L-GSF, U-GSF, and BPF with the proposed L-AGSF and U-AGSF algorithms. Lower values are better. Best MSE in bold, second best underlined.

Experiment C.1

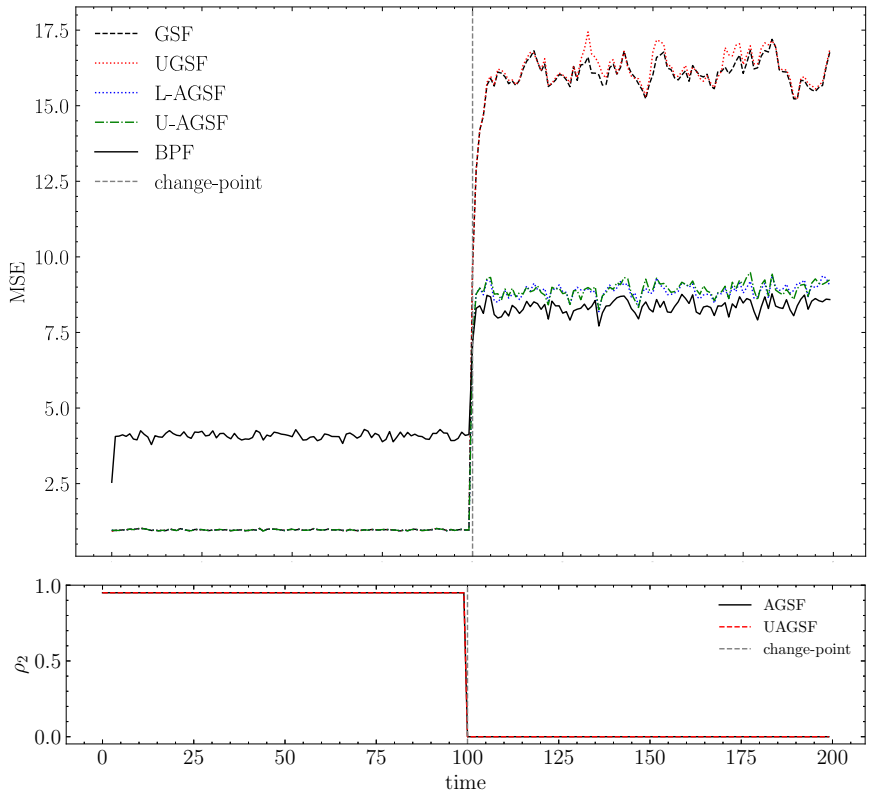
	PARAMETERS	MSE	time(s)
L-GSF	$M = 1$ (EKF)	$6.598 \pm 0.408(100\%)$	0.27 ± 0.03
	$M = 100$	$6.767 \pm 0.271(98\%)$	0.61 ± 0.05
	$M = 1000$	$6.584 \pm 0.394(100\%)$	1.20 ± 0.05
U-GSF	$M = 1$ (UKF)	$6.670 \pm 0.352(100\%)$	0.51 ± 0.03
	$M = 100$	$6.697 \pm 0.286(97\%)$	1.61 ± 0.09
	$M = 1000$	$6.670 \pm 0.353(100\%)$	8.11 ± 0.06
L-AGSF	$M = 2, N = 2, L = 2$	$5.733 \pm 0.150(100\%)$	1.16 ± 0.07
	$M = 10, N = 5, L = 5$	$3.972 \pm 0.053(100\%)$	1.78 ± 0.07
	$M = 100, N = 5, L = 5$	$\mathbf{3.550 \pm 0.023(100\%)}$	2.96 ± 0.10
U-AGSF	$M = 2, N = 2, L = 2$	$5.724 \pm 0.185(100\%)$	1.50 ± 0.05
	$M = 10, N = 5, L = 5$	$3.974 \pm 0.055(100\%)$	2.45 ± 0.06
	$M = 100, N = 5, L = 5$	$\underline{3.566 \pm 0.023(100\%)}$	5.71 ± 0.08
BPF	$M = 100$	$14.436 \pm 0.191(100\%)$	0.52 ± 0.05
	$M = 1000$	$12.531 \pm 0.246(100\%)$	0.79 ± 0.05
	$M = 50000$	$9.416 \pm 0.237(100\%)$	6.27 ± 0.10

Experiment C.2

	PARAMETERS	MSE	time(s)
L-GSF	$M = 1$ (EKF)	$8.462 \pm 0.031(100\%)$	0.28 ± 0.03
	$M = 100$	$8.462 \pm 0.031(100\%)$	0.66 ± 0.06
	$M = 1000$	$8.462 \pm 0.031(100\%)$	1.52 ± 0.09
U-GSF	$M = 1$ (UKF)	$8.536 \pm 0.047(99\%)$	0.52 ± 0.05
	$M = 100$	$8.536 \pm 0.047(99\%)$	1.69 ± 0.07
	$M = 1000$	$8.536 \pm 0.047(99\%)$	9.03 ± 0.08
L-AGSF	$M = 2, N = 2, L = 2$	$11.137 \pm 0.156(100\%)$	1.17 ± 0.08
	$M = 10, N = 5, L = 5$	$6.019 \pm 0.025(100\%)$	1.88 ± 0.09
	$M = 100, N = 5, L = 5$	$\mathbf{4.921 \pm 0.015(100\%)}$	3.65 ± 0.13
U-AGSF	$M = 2, N = 2, L = 2$	$11.134 \pm 0.147(100\%)$	1.51 ± 0.08
	$M = 10, N = 5, L = 5$	$6.022 \pm 0.023(100\%)$	2.58 ± 0.05
	$M = 100, N = 5, L = 5$	$\underline{4.941 \pm 0.016(100\%)}$	6.73 ± 0.20
BPF	$M = 100$	$10.046 \pm 0.034(100\%)$	0.54 ± 0.05
	$M = 1000$	$7.807 \pm 0.019(100\%)$	0.87 ± 0.04
	$M = 50000$	$6.208 \pm 0.014(100\%)$	8.87 ± 0.29



Experiment C.1. *(Top)* MSE plotted in time for L-GSF, U-GSF($M = 1000$), L-AGSF, U-AGSF($M = 100, N = 5, L = 5$), and BPF ($M = 50000$). The AGSF algorithms behave as well as the best algorithm in each mode. *(Bottom)* Proportionality parameter ρ_2. When the model is linear, AGSF behaves like GSF; when nonlinear ρ_2 drops to zero, and AGSF behaves like a particle filter. The plots of ρ_2 seem to coincide for L-AGSF and U-AGSF.



Experiment C.2. *(Top)* MSE of three algorithms over time, with a change at $t = 100$. AGSF nearly matches the best method in each segment. *(Bottom)* ρ_2 drops to zero at the change point, switching AGSF from GSF-like to BPF-like behavior. The plots of ρ_2 seem to coincide for L-AGSF and U-AGSF.

Figure 3.6: Comparison of AGSF behavior in Experiments C.1 and C.2.

3.7 Appendix for Chapter 3

Proof of Theorem 1

The theorem is an application of Taylor's expansion to the MSE of the estimator of Eq. (3.74). We write for the MSE,

$$\begin{aligned} MSE(\widehat{\boldsymbol{\mu}}_{\mathbf{y}}) &= \mathbb{E}\big[||\widehat{\boldsymbol{\mu}}_{\mathbf{y}} - \boldsymbol{\mu}_{\mathbf{y}}||_2^2\big] \\ &= \mathbb{E}\big[||\widehat{\boldsymbol{\mu}}_{\mathbf{y}} - \mathbb{E}[\widehat{\boldsymbol{\mu}}_{\mathbf{y}}] + \mathbb{E}[\widehat{\boldsymbol{\mu}}_{\mathbf{y}}] - \boldsymbol{\mu}_{\mathbf{y}}||_2^2\big] \\ &= \mathbb{E}\big[||\widehat{\boldsymbol{\mu}}_{\mathbf{y}} - \mathbb{E}[\widehat{\boldsymbol{\mu}}_{\mathbf{y}}]||_2^2\big] + ||\mathbb{E}[\widehat{\boldsymbol{\mu}}_{\mathbf{y}}] - \boldsymbol{\mu}_{\mathbf{y}}||_2^2, \end{aligned} \tag{3.126}$$

where the first term is the error due to Monte Carlo sampling, associated to the variance of the estimator, and the second term is the error due to linearization which corresponds to the bias. We now make a Taylor approximation in each term.

Variance term: We have

$$\mathbb{E}\big[||\widehat{\boldsymbol{\mu}}_{\mathbf{y}} - \mathbb{E}\widehat{\boldsymbol{\mu}}_{\mathbf{y}}||_2^2\big] = \mathbb{E}\big[\sum_{i=1}^{d_y} \big(\widehat{\boldsymbol{\mu}}_{\mathbf{y},i} - \mathbb{E}\widehat{\boldsymbol{\mu}}_{\mathbf{y},i}\big)^2\big] \tag{3.127}$$

$$= \sum_{i=1}^{d_y} \mathbb{V}ar\big[\widehat{\boldsymbol{\mu}}_{\mathbf{y},i}\big]. \tag{3.128}$$

For the variance of the scalar components of $\widehat{\boldsymbol{\mu}}_{\mathbf{y}}$ we have,

$$\mathbb{V}ar\big[\widehat{\boldsymbol{\mu}}_{\mathbf{y},i}\big] = \mathbb{V}ar\big[\frac{1}{N}\sum_{n=1}^{N} \mathbf{f}_i(\mathbf{z}_n)\big] = \frac{\mathbb{V}ar\big[\mathbf{f}_i(\mathbf{z})\big]}{N}. \tag{3.129}$$

The first order Taylor expansion of $\mathbf{f}_i(\mathbf{z})$ around $\boldsymbol{\mu}$ is

$$\mathbf{f}_i(\mathbf{z}) \simeq \mathbf{f}_i(\boldsymbol{\mu}) + \nabla \mathbf{f}_i(\boldsymbol{\mu})^T(\mathbf{z} - \boldsymbol{\mu}).$$

Substituting this into Eq. (3.129) we obtain

$$\mathbb{V}ar\big[\widehat{\boldsymbol{\mu}}_{\mathbf{y},i}\big] \simeq \frac{1}{N}\nabla \mathbf{f}_i(\boldsymbol{\mu})^T(\boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta})\nabla \mathbf{f}_i(\boldsymbol{\mu}). \tag{3.130}$$

Therefore we have for the variance term

$$\begin{aligned}\mathbb{E}\big[||\widehat{\boldsymbol{\mu}}_{\mathbf{y}} - \mathbb{E}\widehat{\boldsymbol{\mu}}_{\mathbf{y}}||_2^2\big] &\simeq \frac{1}{N}\sum_{i=1}^{d_y} \nabla \mathbf{f}_i(\boldsymbol{\mu})^T(\boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta})\nabla \mathbf{f}_i(\boldsymbol{\mu}) \\ &= \frac{1}{N}\operatorname{Tr}\big(\nabla \mathbf{f}(\boldsymbol{\mu})^T \nabla \mathbf{f}(\boldsymbol{\mu})(\boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Delta})\big).\end{aligned} \tag{3.131}$$

Bias term: Similarly, the bias term is decomposed as

$$||\mathbb{E}[\widehat{\boldsymbol{\mu}}_{\mathbf{y}}] - \boldsymbol{\mu}_{\mathbf{y}}||_2^2 = \sum_{i=1}^{d_y} (\mathbb{E}\widehat{\boldsymbol{\mu}}_{\mathbf{y},i} - \boldsymbol{\mu}_{\mathbf{y},i})^2. \tag{3.132}$$

We can approximate the terms in the sum by

$$\mathbb{E}[\widehat{\boldsymbol{\mu}}_{\mathbf{y},i}] - \boldsymbol{\mu}_{\mathbf{y},i} = \mathbb{E}[\mathbf{f}_i(\mathbf{z})] - \mathbb{E}\big[\mathbb{E}[\mathbf{f}_i(\mathbf{x})|\mathbf{z}]\big] \tag{3.133}$$

$$= \mathbb{E}\big[\mathbf{f}_i(\mathbf{z}) - \mathbb{E}[\mathbf{f}_i(\mathbf{x})|\mathbf{z}]\big] \tag{3.134}$$

$$\simeq -\frac{1}{2}\operatorname{Tr}\big(\boldsymbol{\Delta}\mathbb{E}[\nabla^2 \mathbf{f}_i(\mathbf{z})]\big), \tag{3.135}$$

where we have used the second order Taylor expansion of $\mathbf{f}_i(\mathbf{x})$ around $\mathbf{z}$

$$\mathbf{f}_i(\mathbf{x}) \simeq \mathbf{f}_i(\mathbf{z}) + \nabla \mathbf{f}_i(\mathbf{z})^T(\mathbf{x} - \mathbf{z}) + \frac{1}{2}(\mathbf{x} - \mathbf{z})^T \nabla^2 \mathbf{f}_i(\mathbf{z})(\mathbf{x} - \mathbf{z}),$$

and taken the expectation with respect to $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{z}, \boldsymbol{\Delta})$. Finally, by using the approximation $\mathbb{E}[\nabla^2 \mathbf{f}_i(\mathbf{z})] \simeq \nabla^2 \mathbf{f}_i(\boldsymbol{\mu})$ and substituting into Eq. (3.132), we obtain for the bias term

$$||\mathbb{E}[\widehat{\boldsymbol{\mu}}_{\mathbf{y}}] - \boldsymbol{\mu}_{\mathbf{y}}||_2^2 \simeq \frac{1}{4}\sum_{i=1}^{d_y} \operatorname{Tr}\big(\boldsymbol{\Delta}\nabla^2 \mathbf{f}_i(\boldsymbol{\mu})\big)^2. \tag{3.136}$$

Proof of Theorem 3

(i) For an overview of the BPF algorithm we refer the reader to [SS23, Algorithm 11.9]. We initialize the AGSF recursion with a particle approximation (i.e. take $\boldsymbol{\Sigma}_{t-1}^{(m)} \to \mathbf{0},\ m = 1, \ldots, M$) and show that it is identical to that of the BPF.

Prediction: When $\boldsymbol{\Delta}_m^{(t)} = \boldsymbol{\Sigma}_{t-1}^{(m)} \to \mathbf{0}$ we have $\mathbf{z}_m = \boldsymbol{\mu}_{t-1}^{(m)},\ m = 1, \ldots, M$, where we have dropped the n index since we take $N_m = 1$ for all m. Moreover, the expectations of Eqs. (3.91), (3.92), are with respect to the delta measure, since

$$\mathcal{N}(\mathbf{x}_{t-1}|\mathbf{z}_m, \boldsymbol{\Delta}_m^{(t)}) \to \delta(\mathbf{x}_{t-1} - \mathbf{z}_m), \tag{3.137}$$

when $\boldsymbol{\Delta}_m^{(t)} \to \mathbf{0}$. Hence, we may substitute in these expressions $\mathbf{x}_{t-1} = \mathbf{z}_m = \boldsymbol{\mu}_{t-1}^{(m)}$ to obtain $\boldsymbol{\mu}_m^- = \mathbf{f}(\boldsymbol{\mu}_{t-1}^{(m)})$ and $\boldsymbol{\Sigma}_m^- = \mathbf{Q}$ for all m.

Update: In the update step of the AGSF, we have (dropping indices n, ℓ), $\mathbf{s}_m \sim \mathcal{N}(\mathbf{f}(\boldsymbol{\mu}_{t-1}^{(m)}), \mathbf{Q})$, which is the PDF of the dynamical model $p(\cdot|\mathbf{x}_{t-1} = \boldsymbol{\mu}_{t-1}^{(m)})$.

Moreover, in Eq. (3.99) the expectation is taken with respect to $\delta(\mathbf{x}_t - \mathbf{s}_m)$, since $\boldsymbol{\Lambda}_m = \mathbf{0}$, hence $\mathbf{C}_m = \mathbf{0}$ and the gain $\mathbf{G}_m$ is also zero. Hence, the update Eqs. (3.95), (3.96) become, $\boldsymbol{\mu}_t^{(m)} = \mathbf{s}_m \sim \mathcal{N}(\mathbf{f}(\boldsymbol{\mu}_{t-1}^{(m)}), \mathbf{Q})$ and $\boldsymbol{\Sigma}_t^{(m)} = \mathbf{0}$. By identifying the means $\boldsymbol{\mu}_t^{(m)}$ with the particles of the BPF, we have $\boldsymbol{\mu}_t^{(m)} \sim p(\cdot|\boldsymbol{\mu}_{t-1}^{(m)})$ which is the sampling step of the BPF. Moreover, the weights of the particles are given by the likelihood $w_m \propto p(\mathbf{y}_t|\boldsymbol{\mu}_t^{(m)})$, just as in the BPF, as can be verified by Eqs. (3.94), (3.97), (3.98). Finally, we have a resampling step with probabilities being proportional to the weights. Hence, we have reproduced a complete BPF recursion. Note that the use of resampling is instrumental for the AGSF to capture the BPF as a special case.

(ii) For the m-th component of the AGSF, the choice $\boldsymbol{\Delta}_m^{(t)} = \boldsymbol{\Sigma}_{t-1}^{(m)}$ implies that all sampled particles

$$\mathbf{z}_{mn} \sim \mathcal{N}(\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)} - \boldsymbol{\Delta}_m^{(t)}) \tag{3.138}$$

are identical to the mean $\boldsymbol{\mu}_{t-1}^{(m)}$. Hence, we can set $N_m = 1$ for each m without losing generality. Moreover from Eqs. (3.91), (3.92) we can see that the expectations are taken with respect to the distribution $\mathcal{N}(\mathbf{x}_{t-1}|\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)})$, which are the original component distributions of the GSF.

Similarly for the update step, the sampled particles all coincide with the means of the predicted distributions $\boldsymbol{\mu}_{t,mn}^-$. Moreover once again, it is easy to see that the updated means and covariances are identical to those of a GSF.

Conditional of a Gaussian mixture

Let the joint distribution of a pair of random vectors $\mathbf{x}$ and $\mathbf{y}$ be given by the Gaussian mixture,

$$p(\mathbf{x}, \mathbf{y}) = \sum_{m=1}^{M} w_m \mathcal{N}_m(\mathbf{x}, \mathbf{y}), \tag{3.139}$$

where,

$$\mathcal{N}_m(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left[\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \middle| \begin{pmatrix} \boldsymbol{\mu}_{\mathbf{x},m} \\ \boldsymbol{\mu}_{\mathbf{y},m} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{x},m} & \mathbf{C}_m \\ \mathbf{C}_m^T & \boldsymbol{\Sigma}_{\mathbf{y},m} \end{pmatrix} \right], \tag{3.140}$$

are the components of the mixture. Let us denote by $\mathcal{N}_m(\mathbf{x}), \mathcal{N}_m(\mathbf{y})$ and $\mathcal{N}_m(\mathbf{x}|\mathbf{y})$, the marginals of each component and the conditional of $\mathbf{x}$ given $\mathbf{y}$, given by

respectively,

$$\mathcal{N}_m(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\mathbf{x},m}, \boldsymbol{\Sigma}_{\mathbf{x},m}), \tag{3.141}$$

$$\mathcal{N}_m(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_{\mathbf{y},m}, \boldsymbol{\Sigma}_{\mathbf{y},m}), \tag{3.142}$$

$$\mathcal{N}_m(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y},m}, \boldsymbol{\Sigma}_{\mathbf{x}|\mathbf{y},m}), \tag{3.143}$$

where,

$$\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y},m} = \boldsymbol{\mu}_{\mathbf{x},m} + \mathbf{C}_m \boldsymbol{\Sigma}_{\mathbf{y},m}^{-1}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y},m}), \tag{3.144}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}|\mathbf{y},m} = \boldsymbol{\Sigma}_{\mathbf{x},m} - \mathbf{C}_m \boldsymbol{\Sigma}_{\mathbf{y},m}^{-1} \mathbf{C}_m^T, \tag{3.145}$$

see [SS23, Lemma A.3]. It is straightforward to see that the marginals of the mixture are given by,

$$p(\mathbf{x}) = \sum_{m=1}^{M} w_m \mathcal{N}_m(\mathbf{x}), \tag{3.146}$$

$$p(\mathbf{y}) = \sum_{m=1}^{M} w_m \mathcal{N}_m(\mathbf{x}). \tag{3.147}$$

For the conditional of $\mathbf{x}$ given $\mathbf{y}$ we have,

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \tag{3.148}$$

$$= \frac{\sum_{m=1}^{M} w_m \mathcal{N}_m(\mathbf{x}, \mathbf{y})}{\sum_{m=1}^{M} w_m \mathcal{N}_m(\mathbf{y})} \tag{3.149}$$

$$= \frac{\sum_{m=1}^{M} w_m \mathcal{N}_m(\mathbf{y}) \mathcal{N}_m(\mathbf{x}|\mathbf{y})}{\sum_{m=1}^{M} w_m \mathcal{N}_m(\mathbf{y})} \tag{3.150}$$

$$= \sum_{m=1}^{M} \tilde{w}_m \mathcal{N}_m(\mathbf{x}|\mathbf{y}) \tag{3.151}$$

where,

$$\tilde{w}_m = \frac{w_m \mathcal{N}_m(\mathbf{y})}{\sum_{m=1}^{M} w_m \mathcal{N}_m(\mathbf{y})}.$$

Unscented AGSF algorithm

In Table 3.5 we present the algorithm for the unscented AGSF. The values of the weights ω_i are given in Box. (3.3).

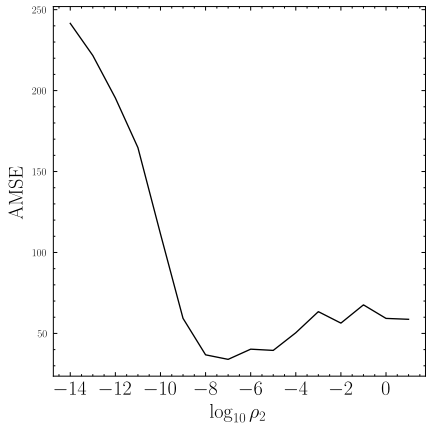


Figure 3.7: MSE plotted for each value of $\log_{10}\gamma$.

Ablation Study for γ

Here we include a small study for the effect of the parameter γ in Eq. (47) of the thesis. We use the setting of Experiment C.1 and we take the parameter $\gamma = 10^i$ for $i = -14, \ldots, -1, 0, 1, 2$. We run the L-AGSF algorithm with $M = 10, N = 5, L = 5$ for each of those values $N_{\text{sim}} = 100$ times and plot the MSE as a function of γ. We also plot the averaged value of ρ_2 for the sequence of the experiment for comparison with the bottom panels of Fig. 3.6 of the thesis.

U-AGSF: Prediction step

Parameters: N_m, $\boldsymbol{\Delta}_m^{(t)}$ for $m = 1, \ldots, M$; N_m sigma-points per component
Initialization: $\{w_0^{(m)}, \boldsymbol{\mu}_0^{(m)}, \boldsymbol{\Sigma}_0^{(m)}\}_{m=1}^{M}$
for $t = 1$ to T **do**
  for $m = 1$ to M **do**
    for $n = 1$ to N_m **do**
      Sample $\mathbf{z}_{mn} \sim \mathcal{N}(\boldsymbol{\mu}_{t-1}^{(m)}, \boldsymbol{\Sigma}_{t-1}^{(m)} - \boldsymbol{\Delta}_m^{(t)})$
      $\boldsymbol{\sigma}_{mn}^{(0)} = \mathbf{z}_{mn}$
      $\boldsymbol{\sigma}_{mn}^{(\pm i)} = \mathbf{z}_{mn} \pm \sqrt{d_x + \lambda} \cdot [\boldsymbol{\Delta}_m^{(t)}]_{\bullet i}^{1/2}$
      $\boldsymbol{\mu}_{t,mn}^{-} = \sum_i \omega_i \mathbf{f}(\boldsymbol{\sigma}_{mn}^{(i)})$
      $\boldsymbol{\Sigma}_{t,mn}^{-} = \sum_i \omega_i (\mathbf{f}(\boldsymbol{\sigma}_{mn}^{(i)}) - \boldsymbol{\mu}_{t,mn}^{-})(\cdots)^T + \mathbf{Q}$
      Set $w_{mn} = w_{t-1}^{(m)} / N_m$
    end for
  end for
end for

U-AGSF: Update and resampling steps

for $t = 1$ to T **do**
  for $m = 1$ to M **do**
    for $n = 1$ to N_m **do**
      for $\ell = 1$ to L_{mn} **do**
        Sample $\mathbf{s}_{mn\ell} \sim \mathcal{N}(\boldsymbol{\mu}_{t,mn}^{-}, \boldsymbol{\Sigma}_{t,mn}^{-} - \boldsymbol{\Lambda}_{mn}^{(t)})$
        $\boldsymbol{\sigma}_{mn\ell}^{(0)} = \mathbf{s}_{mn\ell}$
        $\boldsymbol{\sigma}_{mn\ell}^{(\pm i)} = \mathbf{s}_{mn\ell} \pm \sqrt{d_x + \lambda} \cdot [\boldsymbol{\Lambda}_{mn}^{(t)}]_{\bullet i}^{1/2}$
        $\boldsymbol{\mu}_{\mathbf{y},mn\ell} = \sum_i \omega_i \mathbf{g}(\boldsymbol{\sigma}_{mn\ell}^{(i)})$
        $\mathbf{S}_{\mathbf{y},mn\ell} = \sum_i \omega_i (\mathbf{g}(\boldsymbol{\sigma}_{mn\ell}^{(i)}) - \boldsymbol{\mu}_{\mathbf{y},mn\ell})(\cdots)^T + \mathbf{R}$
        $\mathbf{C}_{mn\ell} = \sum_i \tilde{\omega}_i (\boldsymbol{\sigma}_{mn\ell}^{(i)} - \mathbf{s}_{mn\ell})(\mathbf{g}(\boldsymbol{\sigma}_{mn\ell}^{(i)}) - \boldsymbol{\mu}_{\mathbf{y},mn\ell})^T$
        $\mathbf{G}_{mn\ell} = \mathbf{C}_{mn\ell} \mathbf{S}_{\mathbf{y},mn\ell}^{-1}$
        $\boldsymbol{\mu}_{mn\ell} = \mathbf{s}_{mn\ell} + \mathbf{G}_{mn\ell}(\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y},mn\ell})$
        $\boldsymbol{\Sigma}_{mn\ell} = \boldsymbol{\Lambda}_{mn}^{(t)} - \mathbf{G}_{mn\ell} \mathbf{S}_{\mathbf{y},mn\ell} \mathbf{G}_{mn\ell}^T$
        $w_{mn\ell} \propto (w_{mn}/L_{mn}) \mathcal{N}(\mathbf{y}_t | \boldsymbol{\mu}_{\mathbf{y},mn\ell}, \mathbf{S}_{\mathbf{y},mn\ell})$
      end for
    end for
  end for
  Normalize weights $w_{mn\ell}$
  Resample M triplets (m, n, ℓ) proportional to $w_{mn\ell}$
  Set $\boldsymbol{\mu}_t^{(m')} = \boldsymbol{\mu}_{mn\ell}, \boldsymbol{\Sigma}_t^{(m')} = \boldsymbol{\Sigma}_{mn\ell}, w_t^{(m')} = 1/M$
end for

Table 3.5: Prediction and update/resampling steps of the Unscented Augmented Gaussian Sum Filter (U-AGSF).

Tables for Experiment B

$\rho_2\rho_1$	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95
0.95	8.23	8.25	8.23	8.50	9.37	9.91	10.63	11.02	12.79	16.20	19.75
0.9	7.82	7.93	7.84	8.19	8.14	8.29	8.64	8.82	10.42	9.74	11.51
0.8	7.30	7.38	7.36	7.38	7.62	7.63	7.73	7.68	7.77	8.13	8.40
0.7	7.15	7.00	7.18	7.21	7.15	7.22	7.23	7.23	7.52	7.48	7.46
0.6	7.00	6.88	6.85	6.73	6.88	6.83	6.86	6.81	7.00	7.06	6.93
0.5	6.67	6.93	6.72	6.71	6.66	6.67	6.61	6.79	6.82	6.64	6.72
0.4	6.73	7.20	6.83	6.59	6.71	6.94	6.64	7.06	6.65	6.59	6.59
0.3	6.92	7.38	6.89	7.17	6.66	6.65	6.67	6.91	6.72	6.81	6.64
0.2	7.51	7.35	8.57	7.41	7.99	7.02	8.48	8.27	8.16	7.33	7.87
0.1	9.57	12.21	9.65	11.09	11.81	9.43	14.14	11.44	10.15	11.39	12.98
0.05	17.56	18.31	17.63	18.02	17.68	16.70	18.32	19.79	18.15	17.05	18.50

Table 3.6: Ablation study MSE for L-AGSF

$\rho_2\rho_1$	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95
0.95	8.19	8.17	8.27	8.54	9.38	9.92	10.69	10.98	12.80	16.15	19.81
0.9	7.92	7.93	7.82	8.23	8.13	8.30	8.63	8.93	10.41	9.76	11.53
0.8	7.30	7.36	7.36	7.38	7.66	7.62	7.80	7.62	7.78	8.05	8.36
0.7	7.12	7.07	7.08	7.22	7.13	7.14	7.28	7.18	7.49	7.38	7.48
0.6	7.01	6.82	6.85	6.77	6.89	6.81	6.84	6.80	6.98	7.07	6.88
0.5	6.66	6.93	6.70	6.66	6.63	6.63	6.63	6.77	6.79	6.63	6.70
0.4	6.73	7.20	6.82	6.67	6.73	6.93	6.71	7.06	6.64	6.58	6.62
0.3	7.14	7.37	6.83	7.18	6.67	6.65	6.68	6.91	6.76	6.80	6.62
0.2	7.60	7.34	8.79	7.42	7.97	7.08	9.15	8.26	8.14	7.31	7.87
0.1	9.99	12.21	10.20	11.50	11.79	9.43	14.13	11.45	10.15	10.79	12.99
0.05	17.55	18.31	17.63	18.01	17.68	16.70	18.32	19.79	18.15	17.04	18.50

Table 3.7: Ablation study MSE for U-AGSF

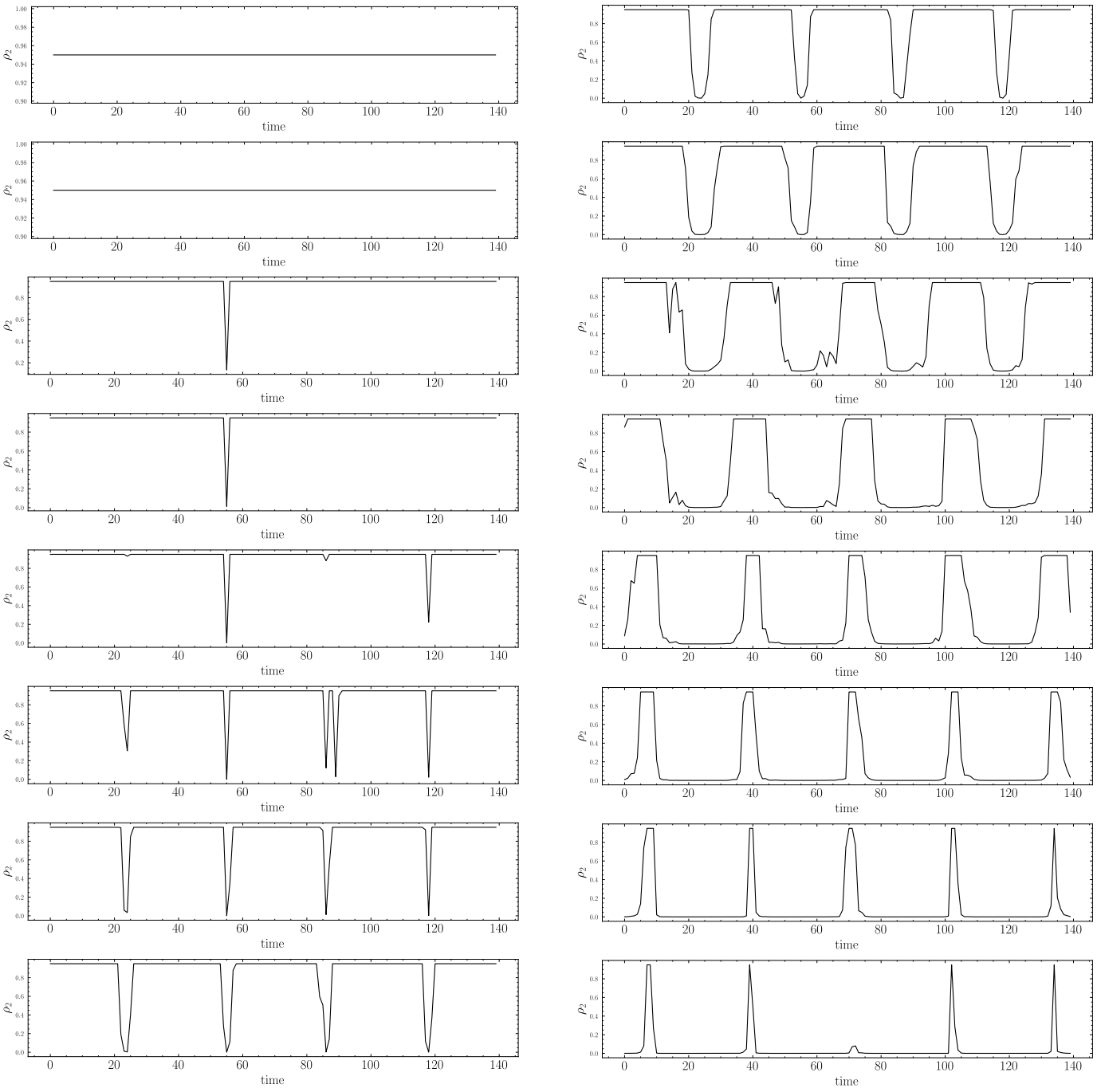


Figure 3.8: Values of ρ_2 plotted in time for each value of γ. *(Left column)* From $\gamma = 10$ at the top to $\gamma = 10^{-6}$ at the bottom. *(Right column)* From $\gamma = 10^{-7}$ at the top to $\gamma = 10^{-14}$ at the bottom.

Chapter 4

Simulation-based inference of state-space models

4.1 Introduction

Parameter inference is a basic problem that arises when working with SSMs since all algorithms, like filters and smoothers, require parameter values as inputs in order to operate. In many applications we may have estimates of such values, available through direct measurement. For example, in tracking applications sensor characteristics might be available from specifications or calibration procedures. However, in many practical scenarios parameters are unknown or partially known. This is especially true for example in black-box methods like neural networks, for which parameters (e.g. weights) do not have a direct association with quantities in the real world that they can be estimated from. In such cases parameters have to be treated as latent variables and estimated from data.

The main difficulty of SSM parameter inference is the intractability of the likelihood function, which is a very high dimensional integral over the complete sequence of latent states, given in Eq. (2.15). Due to the Markovian structure of the SSM, likelihood estimates are computed recursively in linear time by filters, as described in the previous chapter. Commonly used algorithms, such as EM and MCMC, rely on filters to obtain noisy likelihood estimates. Therefore the accuracy of these algorithms is limited by the accuracy of the underlying filters. This is not the case for simulation-based, or likelihood-free, methods which avoid likelihood approximations and instead rely on repeated simulations from the generative model.

In contrast to standard methods, which rely on noisy likelihood approximations, simulation-based methods work by simulating parameters and observations

from the model, which are used to create a surrogate of the likelihood. This approach is particularly useful when the likelihood function is intractable or expensive to evaluate, but simulations from the model are readily available. Independence from the whims of noisy likelihood estimates has made simulation-based inference a popular alternative to standard Bayesian inference tools [Col+24].

The earliest SBI method to be introduced is approximate Bayesian computation (ABC) [Tav+97], which proposes parameter samples from the prior and accepts them if they generate simulated data sufficiently close to the observed data, as explained in Chapter 2. Many extensions of ABC have been proposed since [SFB18; Pes+23], most notably MCMC-ABC [Mar+03] and SMC-ABC [Bea+09; Sim+21]. More recently methods based on neural density estimation [Ger+15], such as sequential neural posterior (SNP) [PM16; Lue+17; GNM19], sequential neural likelihood (SNL) [PSM19], and sequential neural ratio (SNR) [Tho+22; HBL20] have become the state-of-art in a variety of problems [CBL20; Lue+21; Col+24].

With the abundance of simulation tools recently available to practitioners, such as GPUs and powerful software suites, SBI has emerged as an important subfield of Bayesian inference that can be applied to many real-world scenarios, with applications ranging from earthquake prediction [SLW24], economic agent-based models [GR15; GRT17], astrophysics [Hah+17] and biology [Ver+23], to name a few. SBI methods have also been used successfully for parameter inference in SSMs [Ton+09; Mar+19; Dea+14; Aus+24a]. More recently there have been applications of neural SBI methods to SSM inference, in the context of VI and Gaussian processes [War+20; Ryd+21; Doe+18]. Other works which use neural SBI for SSMs have mainly focused on inference on the hidden states of the SSM [KS22; Aus+24b]. Moreover, up to date and to the best of our knowledge, there have not been any works evaluating the performance of NDE-based methods such as SNL or SNPE, which are the best-performing methods, to SSM parameter inference.

In this chapter we evaluate the performance of neural SBI methods for the parameter inference problem of SSMs. We focus on the evaluation of SNL, which is the best-performing neural SBI method [Lue+21; PSM19]. We show that while SNL in many cases outperforms widely used algorithms such as SMC-ABC and MCMC, it is not sample efficient. To address the limitations of SNL we introduce a novel, sample efficient variant, called truncated-SNL which is specifically tailored for SSMs. It is based on a well known forgetting property of SSMs, which allows us to truncate the observation sequence and instead of learning the full likelihood function, learning a conditional likelihood function in a much smaller

space. T-SNL improves on SNL in several ways: it is more sample efficient, mote stable and robust during training, it scales better to longer temporal sequences and it is amortized.

The main contributions of this chapter are summarized as follows:

- We evaluate SNL on for parameter inference in SSMs. We compare to current methods to assess performance and understand tradeoffs of various approaches.
- We propose a novel variant called T-SNL, which exploits a well known forgetting property of SSMs to simplify the learning task.
- We show that T-SNL is very sample efficient and outperforms traditional methods. We show that T-SNL improves on SNL in several ways.

The rest of the chapter is organized as follows. In Section 4.2 we review the basic concepts used throughout the paper and introduce notation. In Section 4.3 we motivate T-SNL and detail the methodology. In Section 4.4 we present the main experimental results of this chapter and discuss our findings.

4.2 Background

4.2.1 Neural density estimation

Neural density estimation (NDE) methods use neural networks to model an unknown data distribution [Ger+15], given samples from the distribution. The model is trained on a given dataset typically by maximizing the average log-likelihood using stochastic-gradient optimization. Recently, NDE methods have been applied to SBI and have achieved state-of-the-art results [Lue+21]. They have been shown to achieve a trade-off between accuracy and simulation cost that is optimal among current methods, being able to learn complex likelihoods with a comparatively small number of simulations of the model [PM16; PSM19; CBL20]. We briefly review the *masked autoencoder for distribution estimation* (MADE) [Ger+15] and the *masked autoregressive flow* (MAF) [PPM17] models that form the basis of sequential neural likelihood [PSM19], reviewed in the next section.

MADE

Given data $\{\mathbf{y}_m\}_{m=1}^M$ in $\mathbb{R}^D$, we are interested in learning the data density $p(\mathbf{y})$. The masked autoencoder for distribution estimation model [Ger+15] is an au-

toregressive density estimator. Autoregressive density estimators use the identity

$$p(\mathbf{y}) = \prod_{d=1}^{D} p(y_d|y_{1:d-1}) \tag{4.1}$$

and use neural networks to parametrize the conditionals. One common choice is to use Gaussian conditionals,

$$p(y_d|y_{1:d-1}) = \mathcal{N}(y_d|\mu_d, \sigma_d^2), \tag{4.2}$$

and feedforward neural networks to model the mean and variance

$$\mu_d = g_{\mu_d}(y_{1:d-1}; \boldsymbol{\phi}_{\mu_d}), \tag{4.3}$$

$$\sigma_d = g_{\sigma_d}(y_{1:d-1}; \boldsymbol{\phi}_{\sigma_d}), \tag{4.4}$$

where $\boldsymbol{\phi}_{\mu_d}$ and $\boldsymbol{\phi}_{\sigma_d}$ are trainable parameters of the neural networks g_{μ_d} and g_{σ_d}, respectively, for $d = 1, \ldots, D$.

Instead of using D separate neural networks to model the conditionals, MADE uses masking to compute all means and covariances using a single neural network:

$$(\mu_{1:D}, \sigma_{1:D}) = \mathbf{g}_{\text{MADE}}(\mathbf{y}; \boldsymbol{\phi}), \tag{4.5}$$

where $\boldsymbol{\phi}$ are all trainable parameters. The function $\mathbf{g}_{\text{MADE}}$ is a MLP, where the weight matrix of each layer is multiplied by a binary mask which drops some connections, as introduced in [Ger+15]. This ensures that μ_d and σ_d, only depend on $y_{1:d-1}$. MADE layers are stacked in series in what is known as a masked autoregressive flow model, described bellow.

MAF

Autoregressive models such as MADE are normalizing flows[1]. We can see this by writing Eq. (4.2) as

$$y_d = u_d \sigma_d(y_{1:d-1}) + \mu_d(y_{1:d-1}), \tag{4.6}$$

with $u_d \sim \mathcal{N}(0, 1)$, for $d = 1, \ldots, D$. This defines an invertible transformation $\mathbf{y} = \mathbf{f}(\mathbf{u})$, where $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The inverse of this transformation $\mathbf{u} = \mathbf{f}^{-1}(\mathbf{y})$ is

[1]Normalizing flows are a class of generative models that transform a simple base distribution into a complex target distribution through a sequence of invertible and differentiable mappings, allowing both efficient sampling and exact likelihood evaluation via the change-of-variables formula.

given elementwise by

$$u_d = (y_d - \mu_d(y_{1:d-1}))\sigma_d(y_{1:d-1})^{-1}. \tag{4.7}$$

The Jacobian of this transformation is triangular due to the autoregressive property, and its determinant is given by

$$\Big| \det \frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{y}} \Big| = \prod_{d=1}^{D} \frac{1}{\sigma_d}. \tag{4.8}$$

The MAF is constructed by stacking multiple MADE flows

$$\mathbf{y} = \mathbf{f}_K \circ \cdots \mathbf{f}_1(\mathbf{u}), \tag{4.9}$$

where each $\mathbf{f}_k$ is a MADE flow transformation as the one in Eq. (4.6), with inverse as the one in Eq. (4.7). This results in a very flexible model that can be trained efficiently by standard gradient methods. Moreover, $p(\mathbf{y})$ can be calculated by

$$p(\mathbf{y}) = p(\mathbf{u}) \Big| \det \frac{\partial \mathbf{f}_{MAF}^{-1}}{\partial \mathbf{y}} \Big|, \tag{4.10}$$

where $\mathbf{f}_{MAF} = \mathbf{f}_K \circ \cdots \mathbf{f}_1$, and

$$\Big| \det \frac{\partial \mathbf{f}_{MAF}^{-1}}{\partial \mathbf{y}} \Big| = \prod_{k=1}^{K} \prod_{d=1}^{D} \frac{1}{\sigma_d^{(k)}}, \tag{4.11}$$

where $\sigma_{1:D}^{(k)}$ are scale parameters computed by the k^{th} MADE layer. More importantly, $p(\mathbf{y})$ can be evaluated by doing a single pass through the model, making it well-suited for applications where fast likelihood evaluations are needed.

Conditional NDE. Autoregressive models such as MADE and MAF can be easily extended to conditional density estimation: the problem of estimating the conditional density $p(\mathbf{y}|\boldsymbol{\theta})$ given examples $\{(\boldsymbol{\theta}_m, \mathbf{y}_m)\}_{m=1}^{M}$. This can be done using NDE, by treating $\boldsymbol{\theta}$ as the leading dimensions of an augmented data vector $(\boldsymbol{\theta}, \mathbf{y})$ and only modeling the conditionals that correspond to $\mathbf{y}$.

4.2.2 Sequential neural likelihood

Neural density estimation methods can be naturally applied to the problem of SBI, since they can be employed to model unknown probability densities. There are three classes of neural-SBI methods targeting the posterior, likelihood ratio or likelihood of the model. *Neural posterior estimation* methods learn a neural

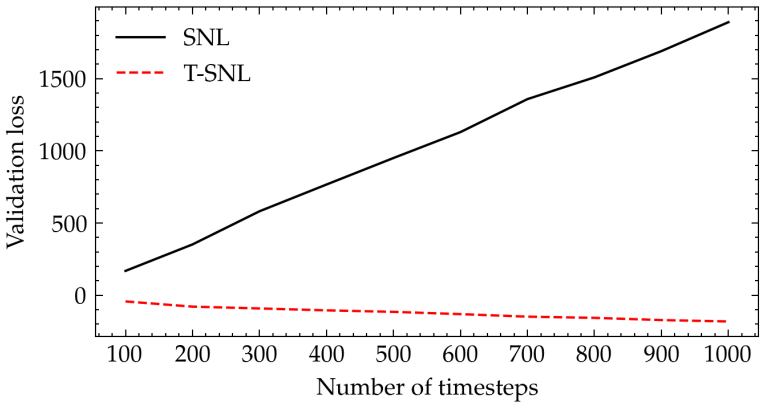


Figure 4.1: Validation losses of SNL and T-SNL during training for an increasing number of simulation timesteps. While for SNL the loss increases with the number of timesteps, for T-SNL it remains constant.

estimate of the posterior by training on proposed parameter-data pairs [PM16; Lue+17]. *Neural ratio estimation* methods estimate the density ratios that are used withing MCMC or other schemes [Tho+22; HBL20]. Finally, *neural likelihood estimation* methods use the simulator to learn a surrogate of the model likelihood [PSM19]. Sequential neural likelihood, which is a neural likelihood estimation method, has been shown to be a very sample efficient way of performing SBI, outperforming other alternatives in various tasks [PSM19; Lue+21].

In Alg. 9 we outline the SNL algorithm for the parameter inference problem of state-space models. The method proceeds in rounds. At each round r, a proposal distribution over parameters $\pi_r(\boldsymbol{\theta})$ is used to sample parameters $\boldsymbol{\theta}_n$, via MCMC. For each sample, synthetic observations $\mathbf{y}_{1:T}^{(n)}$ are drawn from the state-space model. These parameter-observation pairs form a dataset $\mathcal{D}$ used to train a MAF model $q_{\boldsymbol{\phi}}(\mathbf{y}_{1:T}|\boldsymbol{\theta})$.

The training dataset $\mathcal{D}_{r+1}$ can be constructed in different ways depending on the strategy: using all past data (ALL), only the most recent batch (LAST), or selecting the simulations that most closely match the observations (BEST). Once trained, the surrogate likelihood is evaluated at the observed data $\mathbf{y}_{1:T}^{(obs)}$, and combined with the prior to form an updated posterior estimate. This posterior then becomes the proposal π_{r+1} for the next round, focusing future simulations on more relevant regions of parameter space. Through this iterative refinement, SNL improves the quality of the learned likelihood and concentrates simulations in regions supported by the posterior, resulting in more efficient and accurate inference.

Algorithm 9 Sequential Neural Likelihood (SNL)

1: **Initialization** Set $\pi_0(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$ and $\mathcal{D}_0 = \emptyset$

2: **For** $r = 0, \ldots, R-1$**:**

3: **Simulation:** Sample parameters from the proposal and observations from the SSM:

$$\boldsymbol{\theta}_n \sim \pi_r(\boldsymbol{\theta}), \tag{4.12}$$

$$\mathbf{y}_{1:T}^{(n)} \sim p(\mathbf{y}_{1:T}|\boldsymbol{\theta}_n), \tag{4.13}$$

for $n = 1, \ldots, N$. Set up the dataset

$$\mathcal{D} = \{(\boldsymbol{\theta}_n, \mathbf{y}_{1:T}^{(n)})\}_{n=1}^{N}. \tag{4.14}$$

4: **Set up dataset:** One of three options is used to construct the training dataset at round r:

- ALL: $\mathcal{D}_{r+1} = \mathcal{D}_r \cup \mathcal{D}$
- LAST: $\mathcal{D}_{r+1} = \mathcal{D}$
- BEST: $\mathcal{D}_{r+1}$ is the set of the N datapoints from all rounds closest to the observations.

5: **Training:** Train $q_{\boldsymbol{\phi}}(\mathbf{y}_{1:T}|\boldsymbol{\theta})$ on $\mathcal{D}_{r+1}$.

6: **Posterior approximation:** Set

$$\widehat{p}(\boldsymbol{\theta}|\mathbf{y}_{1:T}^{(obs)}) \propto q_{\boldsymbol{\phi}}(\mathbf{y}_{1:T}^{(obs)}|\boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{4.15}$$

$$\pi_{r+1}(\boldsymbol{\theta}) = \widehat{p}(\boldsymbol{\theta}|\mathbf{y}_{1:T}^{(obs)}). \tag{4.16}$$

4.3 Neural likelihood estimation for inference in SSMs

In this section, we introduce a novel, sample-efficient variant of SNL (Alg. 9) which is tailored to the SSM inference problem. Our algorithm, denoted truncated-SNL, like SNL, learns a neural surrogate of the model likelihood sequentially. Unlike SNL, which uses simulations to train a MAF surrogate of the full SSM likelihood it works by replacing the likelihood factors $p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta})$ of Eq. (2.16),

by a truncated version $p(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}, \boldsymbol{\theta})$ which only conditions on the L most recent observations. This is equivalent to the assumption that the observation process is Markovian of order L. Intuitively, this is convenient because it reduces the problem of learning T different conditionals $\{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta})\}_{t=1}^{T}$ to that of learning a single conditional, namely $p(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}, \boldsymbol{\theta})$. This trick allows us to obtain a dataset for training T-SNL that is T times larger than the SNL dataset, for the same number of simulations. This makes the T-SNL algorithm much more sample efficient than SNL, as we show numerically in the next section. This section proceeds by motivating the truncation of the likelihood factors, before introducing the T-SNL algorithm in detail.

4.3.1 Truncation of likelihood factors

A key limitation of SNL when applied to state space models is that it approximates the full data likelihood $p(\mathbf{y}_{1:T}|\boldsymbol{\theta})$ directly via simulations. Due to the autoregressive structure of the SSM likelihood, this effectively amounts to learning T distinct conditional distributions of the form $p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta})$, one for each time step. While this strategy is general, it is inefficient: the conditionals share structure, yet SNL does not reuse information across time steps. As a result, each simulation yields only one training point for the surrogate model, limiting sample efficiency.

This redundancy becomes especially apparent in light of the exponential forgetting property of SSMs, introduced in Chapter 2. As discussed in Sec. 2.3, the likelihood factors $p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta})$ quickly forget past observations, which motivates a truncation of the observation sequence so that each observation only depends on L most recent observations

$$p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \boldsymbol{\theta}) \approx p(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}, \boldsymbol{\theta}). \tag{4.17}$$

Moreover we assume that the conditional $p(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}, \boldsymbol{\theta})$ is shared for all time-steps. This is equivalent to approximating the observation process as an L-order Markov chain. Under this approximation, all the conditional factors in the likelihood share the same structure and can be modeled using a single neural density estimator. This is formalized in the following approximation:

Approximation 1 (Observation process is approximately Markovian)**.** *We approximate the observation process by a Markov chain of order L. The kernel of the chain has conditional density $p(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}, \boldsymbol{\theta})$, such that the likelihood can be*

approximated as

$$p_L(\mathbf{y}_{1:T}|\boldsymbol{\theta}) \approx \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}, \boldsymbol{\theta}). \tag{4.18}$$

This truncated factorization greatly improves sample efficiency. Given a dataset $\mathcal{D} = \{(\boldsymbol{\theta}_n, \mathbf{y}_{1:T}^{(n)})\}_{n=1}^N$ consisting of N simulations, the kernel is trained by minimizing the log-likelihood on the lagged set $\mathcal{D}_L = \{\{(\boldsymbol{\theta}_n, \mathbf{y}_{t-L:t}^{(n)})\}_{t=1}^T\}_{n=1}^N$. By comparison SNL, which targets the full likelihood, is trained on $\mathcal{D}$. Thus SNL is trained on a dataset of size $|\mathcal{D}| = N$, while T-SNL is trained on a set of size $|\mathcal{D}_L| = N \times T$. This multiplication of the dataset size is a main strength of T-SNL which has the remarkable property that for some scenarios, it can tackle the inference problem with a single simulator run, $N = 1$. This makes T-SNL a very sample-efficient algorithm, as we show in the experiments section. We continue with a detailed description of the proposed methodology.

4.3.2 The T-SNL algorithm

We now present the full T-SNL algorithm for parameter inference in state space models. T-SNL replaces the full likelihood with a truncated factorization that is easier to estimate and more sample-efficient. The learned likelihood is used within an MCMC sampler to approximate the posterior. As in SNL, T-SNL uses sequential proposal adaptation, which allows the algorithm to focus simulations on high-probability regions of the parameter space.

Description of the algorithm

The T-SNL algorithm is detailed in Alg. 10. Fistly the parameter proposal is initialized as the model prior. Then for each round $r = 0, \ldots, R-1$, parameter samples are drawn from the current proposal π_r via MCMC in Eq. (4.19). Observation sequences are drawn from the simulator as shown in Eq. (4.20).

From the simulations a dataset $\mathcal{D}_L$ is constructed in Eq. (4.21). The training dataset $\mathcal{D}_{r+1}$ for training in round $r+1$ is then built according to one of three strategies ALL, LAST or BEST. ALL means that all past simulations are used, LAST than only the simulations from the current round are used and BEST means that the N datapoints from all rounds whose trajectories are closest to the observed data (in Euclidean distance)

Then the likelihood model is set up and trained. A MAF model is used to model the Markov kernel of the observation process.

Algorithm 10 Truncated SNL (T-SNL)

1: **Initialization** Set $\pi_0(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$ and $\mathcal{D}_0 = \{\}$

2: **For** $r = 0, \dots, R-1$**:**

3: **Simulation:** Sample parameters from the proposal and observations from the SSM:

$$\boldsymbol{\theta}_n \sim \pi_r(\boldsymbol{\theta}), \tag{4.19}$$

$$\mathbf{y}_{1:T}^{(n)} \sim p(\mathbf{y}_{1:T}|\boldsymbol{\theta}_n), \tag{4.20}$$

for $n = 1, \dots, N$. Set up the lagged dataset

$$\mathcal{D}_L = \{\{(\boldsymbol{\theta}_n, \mathbf{y}_{t-L:t}^{(n)})\}_{t=1}^{T}\}_{n=1}^{N}. \tag{4.21}$$

4: **Set up dataset:** One of three options is used to construct the training dataset at round r:

- ALL: $\mathcal{D}_{r+1} = \mathcal{D}_r \cup \mathcal{D}_L$
- LAST: $\mathcal{D}_{r+1} = \mathcal{D}_L$
- BEST: $\mathcal{D}_{r+1}$ is the set of the N datapoints from all rounds closest to the observations.

5: **Training:** Train $q_{\boldsymbol{\phi}}^{(L)}(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}, \boldsymbol{\theta})$ on $\mathcal{D}_{r+1}$ and compute likelihood approximation

$$q_{\boldsymbol{\phi}}^{(L)}(\mathbf{y}_{1:T}^{(obs)}|\boldsymbol{\theta}) = \prod_{t=1}^{T} q_{\boldsymbol{\phi}}^{(L)}(\mathbf{y}_t^{(obs)}|\mathbf{y}_{t-L:t-1}^{(obs)}, \boldsymbol{\theta}). \tag{4.22}$$

6: **Posterior approximation:** Set

$$\widehat{p}(\boldsymbol{\theta}|\mathbf{y}_{1:T}^{(obs)}) \propto q_{\boldsymbol{\phi}}^{(L)}(\mathbf{y}_{1:T}^{(obs)}|\boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{4.23}$$

$$\pi_{r+1}(\boldsymbol{\theta}) = \widehat{p}(\boldsymbol{\theta}|\mathbf{y}_{1:T}^{(obs)}). \tag{4.24}$$

The model, denoted $q_{\boldsymbol{\phi}}^{(L)}(\mathbf{y}_t|\mathbf{y}_{t-L:t-1}, \boldsymbol{\theta})$, is trained by maximizing the log-likelihood over the dataset $\mathcal{D}_{r+1}$. More specifically, we obtain $\boldsymbol{\phi}$ by minimizing the loss func-

tion,

$$\text{loss}(\boldsymbol{\phi}) = -\sum_{n} \log q_{\boldsymbol{\phi}}^{(L)}(\mathbf{y}_{1:T}^{(n)}|\boldsymbol{\theta}_n) \tag{4.25}$$

$$= -\sum_{n=1}^{N}\sum_{t=1}^{T} \log q_{\boldsymbol{\phi}}^{(L)}(\mathbf{y}_{t}^{(n)}|\mathbf{y}_{t-L:t-1}^{(n)}, \boldsymbol{\theta}_n). \tag{4.26}$$

The full likelihood estimate is given in Eq. (4.22). The learned likelihood is used to form the approximate posterior $\widehat{p}(\boldsymbol{\theta}|\mathbf{y}_{1:T}^{(obs)})$, given in Eq. (4.23). This approximation is used as the proposal $\pi_{r+1}(\boldsymbol{\theta})$ in the next round.

4.3.3 Choice of lag

To choose a value for the lag L of the T-SNL algorithm we estimate the autocorrelation function (ACF) of the observation sequence and choose L that achieves a small ACF. The forgetting property implies that $\mathbf{y}_{t-k}, \mathbf{y}_t$ should be independent when $t >> t-k$, hence the ACF should be low. In particular, for a sequence of observations $(\mathbf{y}_t)_{t=1}^{T}$ and lag L we calculate the autocorrelation matrix

$$\mathbf{C}_L = \frac{1}{T-L}\sum_{t=1}^{T-L}(\mathbf{y}_{t-L} - \bar{\mathbf{y}})(\mathbf{y}_t - \bar{\mathbf{y}})^T, \tag{4.27}$$

which is the sample autocorrelation matrix at lag L. The frobenius norm of this matrix is

$$||\mathbf{C}_L||_F = \text{trace}(\mathbf{C}_L^T\mathbf{C}_L)^{1/2} \tag{4.28}$$

and it is an estimate for the total autocorrelation among all dimensions of $\mathbf{y}$. We found the ACF to be a useful rule of thumb for the determination of L. In Fig. 4.2 we plot the ACFs of the models considered in the experiments.

4.4 Experiments

To evaluate the performance of T-SNL, we conduct a series of experiments comparing it against established inference methods on a range of state-space models of increasing complexity. Our goal is to assess the sample-efficiency and robustness of the proposed algorithm and highlight its advantages over SNL, in a variety of inference settings. We consider dynamical systems including the LGSSM and stochastic volatility models with varying dimension, and nonlinear population dynamics. For each setting, we measure the quality of posterior inference under

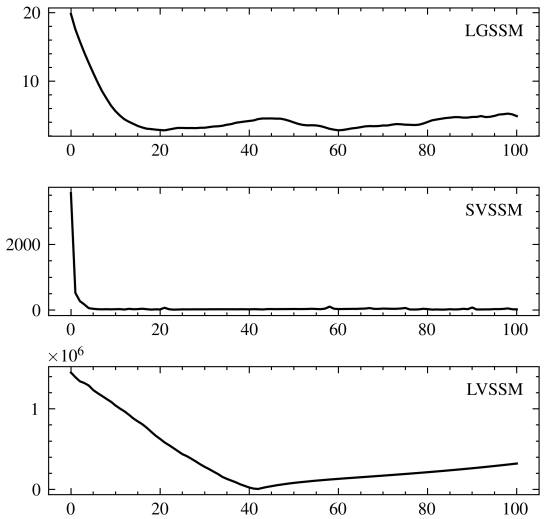


Figure 4.2: **ACF.** Plots of the autocorrelation function for each of the state-space models used in this work. Each plot is obtained by averaging multiple ACF estimates.

a fixed simulation budget and compare the trade-off between accuracy and simulation cost across methods. Below we describe the experimental setup, including the algorithms and evaluation metrics. The results are presented in the following subsection. Additional details and results are provided in the supplementary material.

4.4.1 Setup

Algorithms.

In our experiments we evaluate and compare the following algorithms.

SMC-ABC. We use SMC-ABC with a Gaussian kernel and resampling when the effective sample size is less than 50%. We use the adaptive method of [Sim+21] to select the sequence of tolerances ϵ_n and stopping rule. Details of our implementation of this method can be found in the supplementary material.

Particle MCMC. A bootstrap particle filter estimator of the model likelihood is used within a random walk Metropolis (RWM) kernel. We run the MCMC chain for 1000 steps.

SNL & T-SNL. For both SNL and T-SNL we use a MAF model composed of 5 MADE layers, each composed of 5 hidden layers, each of them with 32 hidden units and relu or tanh activations. After conducting several tests we found those architectures to achieve the best model size to performance tradeoff. The models are trained using the adam optimizer [KB14]. After the models are trained, we use

RWM or elliptical slice sampling [MAM10] to obtain samples from the posterior.

Metrics.

We use the following commonly used metrics to measure the performance of the algorithms [Lue+21].

Probability of true parameters One metric that we used to assess the accuracy is the negative log probability of the true parameters under a kernel density estimate on the posterior samples. More precisely, if $\widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_K$ are samples from an approximate posterior, and $\boldsymbol{\theta}_0$ is the true parameter value we define the error as

$$\mathcal{E}_{\text{KDE}} = -\log p_{\text{KDE}}(\boldsymbol{\theta}_0 | \widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_K), \tag{4.29}$$

where, $p_{\text{KDE}}(\boldsymbol{\theta} | \widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_K)$ is the pdf of a kernel density estimator with kernels placed at the sample points $\widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_K$. We use standard normal kernels for the evaluation of $\mathcal{E}_{\text{KDE}}$.

Minimum distance. Another informative measure of discrepancy is the minimum distance between posterior samples and the true parameter

$$\mathcal{E}_{\text{min}} = \min_k ||\boldsymbol{\theta}_0 - \widehat{\boldsymbol{\theta}}_k||. \tag{4.30}$$

In the supplementary material we show that $\mathcal{E}_{\text{KDE}}$ is upper and lower bounded by functions of $\mathcal{E}_{\text{min}}$. Advantages of $\mathcal{E}_{\text{min}}$ over $\mathcal{E}_{\text{KDE}}$ are that it is independent of the choice of kernel density and that it has an intuitive meaning.

Posterior bias and standard deviation. We also report the bias and standard deviation of the posterior samples

$$\text{bias} = ||\boldsymbol{\theta}_0 - \widehat{\boldsymbol{\theta}}||, \tag{4.31}$$

$$\text{st.dev.} = \sqrt{\frac{1}{K} \sum_{k=1}^{K} ||\widehat{\boldsymbol{\theta}}_k - \widehat{\boldsymbol{\theta}}||^2}, \tag{4.32}$$

where $\widehat{\boldsymbol{\theta}} = \sum_{k=1}^{K} \widehat{\boldsymbol{\theta}}_k / K$ is the sample mean. These quantities give us insight on the distribution of the samples. The bias informs us about the distance between the sample mean and the true parameter, while the standard deviation about the spread of the samples around their mean. Together they give us the MSE of the estimator which is equal to the sum of squares of bias and standard deviation.

Simulation cost. To measure the amount of data that each algorithm uses to perform inference, i.e., the simulation cost for each algorithm, we count the num-

ber of calls to the dynamics simulator $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. This is equal to the number of timesteps times the number of simulations of the full model that each algorithm uses. We choose this measure because it can be applied to all considered algorithms and is a constant multiple of the number of samples, which is the usual metric.

4.4.2 Results

We have conducted experiments on the state-space models described below. For each SSM we considered scenarios in which we target different parameters during inference. In each experiment we set the ground truth of the target parameters to a fixed value, then simulate observations and run inference for each algorithm, repeating for multiple trials. We plot the errors vs simulation cost averaged over trials.

Below we include a part of our experimental results and discuss our findings.
Linear Gaussian model. The linear-Gaussian SSM is used to describe systems that evolve over time with linear dynamics and observation models and Gaussian noise. It is given by the equations,

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{q}_t, \tag{4.33}$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \mathbf{r}_t, \tag{4.34}$$

where $\mathbf{x}_t \in \mathbb{R}^{d_x}$ and $\mathbf{y}_t \in \mathbb{R}^{d_y}$ are the state and observation vectors respectively, $\mathbf{A} \in \mathbb{R}^{d_x \times d_x}$ and $\mathbf{H} \in \mathbb{R}^{d_y \times d_x}$ are real matrices and $\mathbf{q}_t \sim \mathcal{N}(\mathbf{q}_0, \mathbf{Q}) \in \mathbb{R}^{d_x}, \mathbf{r}_t \sim \mathcal{N}(\mathbf{r}_0, \mathbf{R}) \in \mathbb{R}^{d_y}$ are noise vectors. The initial state has distribution $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$.

We consider inference of the dynamics covariance $\mathbf{Q}$ of a model with $d_x = d_y = 1$, setting the ground truth to $\mathbf{Q}_{gt} = 0.1\mathbf{I}$. The plots of metrics versus simulation cost are shown in Fig. 4.3. We see that T-SNL achieves good performance using significantly fewer simulations than other methods. From the plot of $\mathcal{E}_{\text{KDE}}$ we see that BPF-MCMC is able to achieve the best accuracy, albeit at a much higher cost.
Stochastic volatility model. This model, which is prominent in econometrics, describes the time-evolution of coupled financial assets. The model is defined by the equations

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + +\mathbf{q}_t, \tag{4.35}$$

$$\mathbf{y}_t = \mathbf{d} + \boldsymbol{\Sigma}_t^{1/2}\mathbf{r}_t, \tag{4.36}$$

where $\mathbf{y}_t$ is the vector of observed asset prices at time t with covariance $\mathbf{\Sigma}_t = \mathbf{D}_t\mathbf{C}\mathbf{D}_t$, where $\mathbf{D}_t$ is the matrix of volatilities which depends on the hidden state $\mathbf{x}_t$ by $\mathbf{D}_t = \text{diag}(e^{\mathbf{x}_t/2})$. Finally, $\mathbf{C}$ is the matrix of correlations of $\mathbf{y}_t$, the vectors and $\mathbf{d}$ are biases, and $\mathbf{q}_t, \mathbf{r}_t$, are Gaussian noise vectors with covariances $\mathbf{Q}$ and $\mathbf{R}$ respectively.

In Fig. 4.4 we present the results from the inference of matrix $\mathbf{C}$ for a 2D model. We take the ground truth to be

$$\mathbf{C}_{gt} = \begin{pmatrix} 1.00 & 0.52 \\ 0.52 & 1.00 \end{pmatrix}. \tag{4.37}$$

T-SNL exhibits the best error-to-cost tradeoff among the evaluated methods. It achieves similar or better accuracy than SNL while requiring significantly fewer simulations. In contrast, SNL performs well only after a larger number of simulations and tends to be noisy when data is limited. SMC-ABC shows high KDE error at low simulation budgets, which gradually improves as more simulations are added. The bias and standard deviation plots reveal distinct behaviors: BPF-MCMC tends to produce samples with low variance but persistent bias, suggesting that its estimates are tightly clustered but systematically miss the true parameter. In contrast, SMC-ABC produces samples with higher variance and smaller bias, which increases the chance of covering the true parameter. However, its variance remains considerably higher than that of both SNL and T-SNL, whose posteriors are more tightly concentrated around the ground truth.

Lotka-Volterra model. The Lotka-Volterra (LV) model [Lot20] is a population model for two interacting populations of predator and prey. The stochastic version of the model can be expressed as a chemical reaction network with four reactions

$$\begin{aligned} A + B &\xrightarrow{\rho_1} 2A, \\ A + B &\xrightarrow{\rho_2} A, \\ B &\xrightarrow{\rho_3} 2B, \\ A &\xrightarrow{\rho_4} \emptyset, \end{aligned}$$

where A and B are the predator and prey species respectively and $\rho_r > 0$ is the rate of reaction $r = 1, \ldots, 4$. The population dynamics of stochastic reaction networks can be simulated with the Gillespie algorithm [Gil77]. Details of the model and the algorithm can be found in [Wil18].

The LV simulator begins at initial populations $(n_{A,0}, n_{B,0}) = (50, 100)$ and

simulates a trajectory $\mathbf{x}_t = (n_{A,t}, n_{B,t})$, where $n_{s,t}$ is the population of species $s \in \{A, B\}$ at $t = 1, \ldots, T$. We take the observations to be noisy measurements of the prey population

$$y_t = n_{B,t} + e_{B,t}, \tag{4.38}$$

where $e_{B,t} \sim \mathcal{N}(0, \sigma_e^2)$.

Results are shown in Fig. 4.5. The overall picture is similar to that of the other two models. T-SNL is again able to achieve the best tradeoff between error and simulation cost. SMC-ABC is able to achieve similar performance as the simulation cost increases. In contrast SNL does not seem able to improve the estimate for larger samples. BPF-MCMC produces again highly confident biased estimates.

4.4.3 Discussion

Our experiments show that T-SNL is a very sample-efficient and robust algorithm for parameter inference in state-space models. The comparison between T-SNL and SMC-ABC shows that while SMC-ABC can perform well on simple, low-dimensional models such as the LGSSM, as the complexity of the simulator increases it becomes very inefficient. In contrast, T-SNL is consistently very sample-efficient, requiring a minimal number of samples to achieve near-optimal performance.

This is most evident in models with nonlinear dynamics, such as the stochastic volatility and Lotka-Volterra systems. In these cases, the rejection-based nature of SMC-ABC becomes increasingly inefficient, and tuning the tolerance schedule becomes critical. T-SNL, by contrast, scales naturally: its neural surrogate can handle complex likelihoods with little manual tuning, and truncation makes more efficient use of each simulation.

Compared to standard SNL, T-SNL offers several advantages. First, it is significantly more sample-efficient. By leveraging the Markov property of the observation process, T-SNL transforms each simulation into multiple training examples, allowing it to achieve lower errors with fewer simulator calls. Second, it is more stable and robust during training. The lower dimensionality of the truncated likelihood factors simplifies the learning problem, leading to smaller validation losses and fewer optimization issues. Related to that is the fact that the input size of the T-SNL model does not grow with the sequence length T: the conditional $q_{\boldsymbol{\phi}}(\mathbf{y}_t | \mathbf{y}_{t-L:t-1}, \boldsymbol{\theta})$ always sees a fixed-size input window. In contrast, SNL models the full likelihood and must process inputs of size proportional to T, which becomes increasingly difficult to scale. T-SNL not only avoids this

issue, but also benefits from it: as T increases, the size of the training dataset grows proportionally, further improving performance. This can be seen in Fig. 4.1 where the validation loss of SNL increases linearly with T while for T-SNL it decreases. Finally, T-SNL naturally supports amortized inference: once the conditional likelihood model is trained, it can be reused for new observations without retraining. For example, if a new observation $\mathbf{y}_{T+1}$ becomes available, the learned conditional $q_{\boldsymbol{\phi}}(\mathbf{y}_{T+1}|\mathbf{y}_{T+1-L:T}, \boldsymbol{\theta})$ can be used within an SMC or MCMC scheme to obtain samples from the updated posterior $p(\boldsymbol{\theta}|\mathbf{y}_{1:T+1})$. This is not the case for SNL, which upon receiving the new observation has no way of extending the support of the learned likelihood from $\mathbf{y}_{1:T}$ to $\mathbf{y}_{1:T+1}$.

Overall, our results highlight the importance of exploiting temporal structure in simulator-based inference for state-space models. By combining the flexibility of neural likelihood estimation with the Markovian structure imposed by the truncation, T-SNL makes better use of simulations and scales easily to longer sequences. This leads to an inference method that is not only very sample-efficient, but also practical to apply in complex, real-world models. Moreover, the advantage of being amortized means that T-SNL can incorporate streaming information as it is obtained, updating the posterior estimate along the way. This means that it can be used as a reliable component of real-time inference algorithms, such as nested algorithms for joint state and parameter estimation [PMM18].

4.5 Conclusions

In this work, we have explored parameter inference methods for state-space models, including SMC-ABC, particle MCMC, and SNL. We introduced T-SNL, a sample-efficient variant of SNL that leverages the temporal structure of SSMs by assuming that the observation sequence is approximately Markovian. Our experiments demonstrated that T-SNL outperforms other methods in both efficiency and robustness, particularly in complex and nonlinear models. This improvement arises mainly from its ability to convert each simulation into multiple training points. T-SNL improves upon SNL by being more sample-efficient, more stable and robust during training, and more scalable to longer temporal sequences. Finally, T-SNL is amortized, allowing the trained model to be reused as new data arrives. Overall, our findings show that T-SNL is a flexible and effective tool for inference in SSMs, and suggest several promising directions for future work, including applications to real-time systems and more structured neural models.

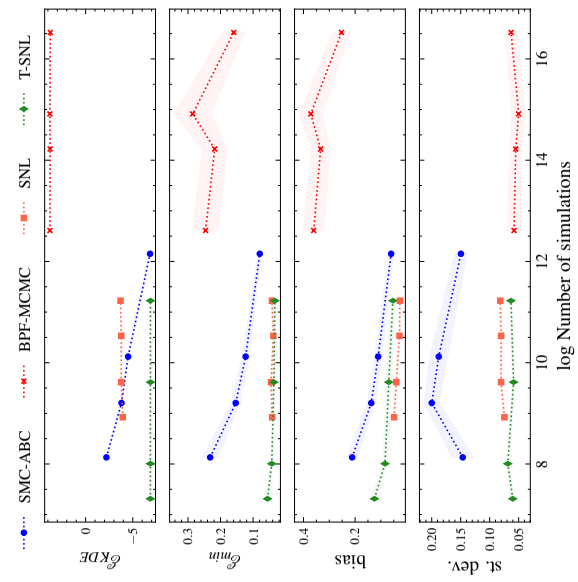


Figure 4.4: **Stochastic volatility model.** Results for the inference problem for the correlation matrix of a 2D stochastic volatility model. We plot errors vs number of simulations, bottom left is best.

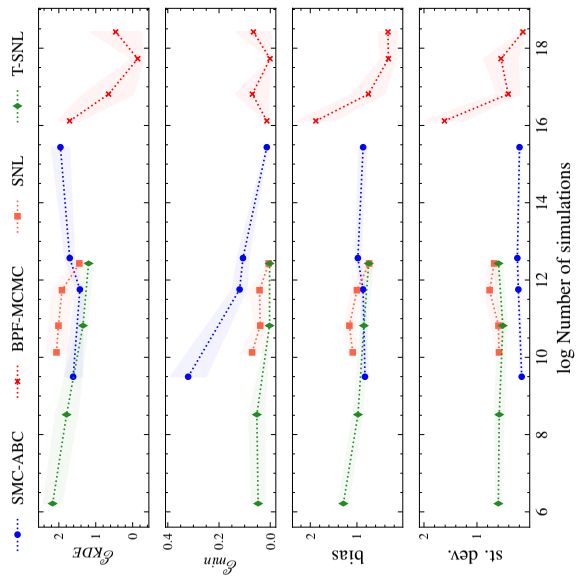


Figure 4.3: **LGSSM.** Results for the inference problem for the dynamics covariance of a linear-Gaussian SSM. We plot errors vs number of simulations, bottom left is best.

Figure 4.5: **Lotka-Volterra.** Results for the inference problem for the reaction rates of a Lotka-Volterra model. We plot errors vs number of simulations, bottom left is best.

4.6 Appendix for Chapter 4

SMC-ABC implementation details

In our implementation of SMC-ABC we have used the tolerance selection method of [Sim+21], which introduces a rule that compares consecutive approximations of the posterior, chooses how much to lower the tolerance at each step, and stops the simulation once the posterior stops changing significantly.

Tolerance selection subroutine

The method works by estimating the ratio between the approximate posterior distributions at consecutive iterations, $\widehat{p}_{\epsilon_{t-1}}(\boldsymbol{\theta})$ and $\widehat{p}_{\epsilon_t}(\boldsymbol{\theta})$, using kernel-based density ratio estimation. From this, a bound $q_t = 1/\sup_{\boldsymbol{\theta}} r(\boldsymbol{\theta})$ is computed, which reflects how much the posterior has changed. The next tolerance ϵ_{t+1} is then set as the q_t-quantile of the current distances between simulated and observed summaries. This ensures that tolerances decrease adaptively in proportion to how much the posterior contracts, without requiring manual scheduling. Moreover, a stopping rule is built in: when q_t exceeds a fixed threshold (e.g. 0.99), the procedure halts automatically, indicating that the posterior has stabilized and further reductions in tolerance would yield diminishing returns. Numerically, the method involves solving two optimization problems: one to fit the ratio function and one to find its maximum. Both problems can be formulated as fixed-point iterations, allowing for efficient, gradient-free solutions.

Fitting the ratio function

The ratio function is defined by:

$$r(\boldsymbol{\theta}) = \frac{\widehat{p}_{\epsilon_t}(\boldsymbol{\theta})}{\widehat{p}_{\epsilon_{t-1}}(\boldsymbol{\theta})}, \tag{4.39}$$

$$\widehat{p}_{\epsilon_t}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_t^{(n)}), \tag{4.40}$$

$$\widehat{p}_{\epsilon_{t-1}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1}^{(n)}). \tag{4.41}$$

We approximate the ratio by a Gaussian kernel density estimate

$$r_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \alpha_n e^{-||\boldsymbol{\theta}-\boldsymbol{\theta}_{t-1}^{(n)}||^2/2\sigma^2}. \tag{4.42}$$

In order for $r_{\boldsymbol{\alpha}}$ to be a proper density ratio, it must satisfy

$$
\begin{aligned}
1 &= \int r_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \widehat{p}_{\epsilon_{t-1}}(\boldsymbol{\theta}) d\boldsymbol{\theta} && (4.43)\\
&= \sum_{n=1}^{N} r_{\boldsymbol{\alpha}}(\boldsymbol{\theta}_{t-1}^{(n)}) && (4.44)\\
&= \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n e^{-||\boldsymbol{\theta}_{t-1}^{(n)} - \boldsymbol{\theta}_{t-1}^{(m)}||^2/2\sigma^2} && (4.45)\\
&= \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n E_{nm}^0 && (4.46)\\
&= \sum_{n=1}^{N} \alpha_n \sum_{m=1}^{N} E_{nm}^0 && (4.47)\\
&= \boldsymbol{\alpha}^T \mathbf{e}^0, && (4.48)
\end{aligned}
$$

where

$$
\begin{aligned}
E_{nm}^0 &= e^{-||\boldsymbol{\theta}_{t-1}^{(n)} - \boldsymbol{\theta}_{t-1}^{(m)}||^2/2\sigma^2} && (4.49)\\
\mathbf{e}_n^0 &= \sum_{m=1}^{N} E_{nm}^0. && (4.50)
\end{aligned}
$$

We set $\boldsymbol{\alpha}$ by maximizing the following function

$$
\begin{aligned}
\ell(\boldsymbol{\alpha}) &= \sum_{n=1}^{N} \log r_{\boldsymbol{\alpha}}(\boldsymbol{\theta}_t^{(n)}) - \lambda(1 - \boldsymbol{\alpha}^T \mathbf{e}^0)), && (4.51)\\
&= \sum_{n} \log \sum_{m} \alpha_m E_{nm} - \lambda(1 - \boldsymbol{\alpha}^T \mathbf{e}^0)), && (4.52)
\end{aligned}
$$

where $E_{nm} = e^{-||\boldsymbol{\theta}_t^{(n)} - \boldsymbol{\theta}_{t-1}^{(m)}||^2/2\sigma^2}$. We compute the gradient of this function

$$
\frac{\partial \ell(\boldsymbol{\alpha})}{\partial \alpha_k} = \sum_{n} \frac{E_{nk}}{\sum_m \alpha_m E_{nm}} - \lambda \mathbf{e}_k^0, \tag{4.53}
$$

If we multiply and divide the first term with α_k and set the gradient equal to zero we obtain

$$
\alpha_k = \frac{1}{\lambda \mathbf{e}_k^0} \sum_{n} \frac{\alpha_k E_{nk}}{\sum_m \alpha_m E_{nm}} \tag{4.54}
$$

Vectorizing we obtain

$$
\boldsymbol{\alpha} = \boldsymbol{\alpha} \odot \frac{1}{N\mathbf{e}^0} \odot \mathbf{E}^T \boldsymbol{\beta} = \mathbf{f}(\boldsymbol{\alpha}) \tag{4.55}
$$

where $\odot$ denotes element-wise multiplication, $\boldsymbol{\beta} = \frac{1}{\mathbf{E}\boldsymbol{\alpha}}$, $\mathbf{E}$ is the matrix with elements E_{nm} and $\lambda = N$ to satisfy the constraint. We solve the optimization problem by repeating the iteration

$$\boldsymbol{\alpha}^{(t)} = \mathbf{f}(\boldsymbol{\alpha}^{(t-1)}) \tag{4.56}$$

Finding the supremum

Our approximation of the ratio is $\widehat{r}_{\boldsymbol{\alpha}^\star}(\boldsymbol{\theta})$ and the corresponding value of $\widehat{c}_t$ is

$$\widehat{c}_t = \sup_{\boldsymbol{\theta}} \widehat{r}_{\boldsymbol{\alpha}^\star}(\boldsymbol{\theta}) \tag{4.57}$$

To find it we follow a similar procedure to the previous subsection. Our objective is to maximize

$$\widehat{r}_{\boldsymbol{\alpha}^\star}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \alpha_n^\star e^{-||\boldsymbol{\theta}-\boldsymbol{\theta}_{t-1}^{(n)}||^2/2\sigma^2} \tag{4.58}$$

with respect to $\boldsymbol{\theta}$. We take the gradient to obtain

$$\frac{\partial \widehat{r}_{\boldsymbol{\alpha}^\star}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{2\sigma^2} \sum_n \alpha_n^\star \frac{\partial}{\partial \boldsymbol{\theta}} ||\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1}^{(n)}||^2 e^{-||\boldsymbol{\theta}-\boldsymbol{\theta}_{t-1}^{(n)}||^2/2\sigma^2} \tag{4.59}$$

$$= -\frac{1}{\sigma^2} \sum_n \alpha_n^\star \frac{\partial}{\partial \boldsymbol{\theta}} (\boldsymbol{\theta} - \boldsymbol{\theta}_{t-1}^{(n)}) e^{-||\boldsymbol{\theta}-\boldsymbol{\theta}_{t-1}^{(n)}||^2/2\sigma^2}. \tag{4.60}$$

Setting equal to zero we obtain the fixed point iteration

$$\boldsymbol{\theta} = \frac{1}{\widehat{r}_{\boldsymbol{\alpha}^\star}(\boldsymbol{\theta})} \sum_n \alpha_n^\star e^{-||\boldsymbol{\theta}-\boldsymbol{\theta}_{t-1}^{(n)}||^2/2\sigma^2} \boldsymbol{\theta}_{t-1}^{(n)}. \tag{4.61}$$

To find the supremum we iterate Eq. (4.61) until convergence.

Since fixed-point iterations are algebraic updates they are very efficient to compute. This results in a very efficient implementation of SMC-ABC which tunes the tolerance schedule and stops automatically.

Relationship between $\mathcal{E}_{\text{KDE}}$ and $\mathcal{E}_{\text{min}}$

Here we derive an bounds of $\mathcal{E}_{\text{KDE}}$ in terms of $\mathcal{E}_{\text{min}}$. Given samples $\{\widehat{\boldsymbol{\theta}}_k\}_{k=1}^K$ from the approximate posterior we use the Gaussian KDE

$$p_{\text{KDE}}(\boldsymbol{\theta}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\boldsymbol{\theta}|\widehat{\boldsymbol{\theta}}_k, \sigma^2\mathbf{I}). \tag{4.62}$$

The log-pdf evaluated at the true parameters $\boldsymbol{\theta}_0$ can be approximated as follows

$$\log p_{\text{KDE}}(\boldsymbol{\theta}_0) = \log \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\boldsymbol{\theta} | \widehat{\boldsymbol{\theta}}_k, \sigma^2 \mathbf{I}) \tag{4.63}$$

$$= \log \frac{1}{K} \sum_{k=1}^{K} \frac{e^{-||\boldsymbol{\theta}_0 - \boldsymbol{\theta}_k||^2/2\sigma^2}}{(2\pi\sigma^2)^{d_\theta/2}} \tag{4.64}$$

$$= \log \sum_{k=1}^{K} e^{-||\boldsymbol{\theta}_0 - \boldsymbol{\theta}_k||^2/2\sigma^2} - \log K - \frac{d_\theta}{2} \log(2\pi\sigma^2) \tag{4.65}$$

$$= \text{LSE}(\{-\frac{1}{2\sigma^2} ||\boldsymbol{\theta}_0 - \boldsymbol{\theta}_k||^2\}_k) + C \tag{4.66}$$

where LSE is the log-sum-exp function and $C = -\log K - \frac{d_\theta}{2} \log(2\pi\sigma^2)$. To make the connection to $\mathcal{E}_{\text{min}}$ we use the following inequality which holds identically,

$$\max\{e_k\}_{k=1}^K \leq \text{LSE}(\{e_k\}_{k=1}^K) \leq \max\{e_k\}_{k=1}^K + \log K. \tag{4.67}$$

Combining Eqs. (4.66) and (4.67) we obtain for the error $\mathcal{E}_{\text{KDE}}$:

$$\frac{1}{2\sigma^2} \mathcal{E}_{\text{min}}^2 + \frac{d_\theta}{2} \log(2\pi\sigma^2) \leq \mathcal{E}_{\text{KDE}} \leq \frac{1}{2\sigma^2} \mathcal{E}_{\text{min}}^2 + \log K + \frac{d_\theta}{2} \log(2\pi\sigma^2) \tag{4.68}$$

Chapter 5

Conclusions

This thesis has addressed two of the central challenges in Bayesian inference for state-space models: state inference and parameter inference. These are foundational problems in the modeling and understanding of dynamical systems, with applications across a wide range of scientific and engineering disciplines. While state-space models offer a powerful probabilistic framework for capturing systems that evolve over time, their practical use is often limited by the computational difficulty of performing exact inference, especially in the presence of nonlinearities or non-Gaussian noise. To address these challenges, the thesis has introduced two novel methodological contributions, each advancing the state of the art in its respective domain.

The first contribution is the augmented Gaussian sum filter, a new class of Bayesian filters that unifies and extends the capabilities of Gaussian sum filters and particle filters. The AGSF framework introduces latent variables and associated augmentation covariances to construct a Gaussian mixture approximation to the filtering distribution. By tuning these covariances, AGSF interpolates smoothly between the deterministic behavior of GSFs and the stochastic nature of PFs. Moreover, the proposed adaptive AGSF variant automatically sets the augmentation parameters by minimizing an upper bound on the mean squared error of moment estimates, leading to a filter that dynamically adjusts to the local structure of the problem. Empirical results demonstrated the strong performance of AGSF in scenarios that challenge traditional filters, including systems with both linear and nonlinear dynamics. The algorithm's ability to combine the efficiency of Gaussian approximations with the robustness of particle methods offers a practical and theoretically grounded tool for state inference in complex models.

The second contribution is truncated sequential neural likelihood, a novel pa-

rameter inference algorithm tailored to SSMs. Building on the simulation-based inference paradigm, T-SNL extends the sequential neural likelihood approach by exploiting the exponential forgetting property inherent to many SSMs. This property suggests that the influence of the past on future observations decays over time, motivating a truncation of the model likelihood into shorter, approximately independent temporal windows. This truncation allows each simulation to yield multiple training samples, significantly improving the sample efficiency and stability of the likelihood learning process. The resulting algorithm is more robust to noise and overfitting, scales better with sequence length, and it is amortized for future observations. Experiments across a diverse set of models—including linear-gaussian, stochastic volatility, ecological highlighted the advantages of T-SNL over both traditional and recent SBI methods.

Together, these contributions represent complementary advances in the two core inference tasks for SSMs. AGSF provides a flexible and adaptive framework for online state estimation, while T-SNL offers an efficient and scalable solution for offline parameter estimation in simulation-based settings. Although developed independently, these methods open several avenues for integration. For example, the adaptive filtering capabilities of AGSF could be incorporated into likelihood-based parameter estimation pipelines, or used as a proposal mechanism in particle MCMC schemes. Likewise, the amortized inference nature of T-SNL suggests potential for use in online parameter learning, where models must adapt to streaming data in real time.

Looking ahead, several directions offer exciting opportunities for future work:

- **Extensions of the AGSF framework:** In this work, we have chosen the word framework to address the AGSF because of its extensibility. Future work could explore the use of more sophisticated proposal distributions beyond simple Monte Carlo sampling, such as variational or adaptive importance sampling schemes. Additionally, more advanced numerical integration methods could be incorporated to replace linearization or unscented transformations, enabling higher accuracy in capturing nonlinear dynamics. We view this plug-and-play flexibility as a strength of the framework, giving it a very wide range of possible behaviours. This was highlighted in this work by showing that the AGSF interpolates between Gaussian and particle filters. Therefore, the wide range of behaviours of the AGSF makes it likely that a practitioner will be able to find a solution to their problem within the framework.

- **Online learning and real-time inference:** Extending T-SNL to stream-

ing data settings, where model parameters are updated incrementally as new observations arrive, could enable adaptive decision-making in environments that change over time. This would be particularly valuable in applications such as autonomous systems, financial forecasting, and real-time diagnostics, where models must continually adapt to nonstationary dynamics without the need for full retraining.

- **Model selection and change detection:** The AGSF framework's ability to adjust its approximation based on local nonlinearities makes it well suited to detecting abrupt or gradual changes in system behavior. This opens the door to its use in tasks such as structural break detection, regime switching, and automatic model selection, where adaptive inference mechanisms are essential for identifying shifts in the underlying dynamics.

- **Temporal neural architectures:** The current T-SNL implementation employs a simple multilayer perceptron to model the conditional likelihood within each truncated segment. This architecture could be replaced with temporal neural networks such as convolutional models, which can better capture local temporal patterns, or transformer-based models, which can effectively handle long-range dependencies. These architectural improvements could enhance the expressiveness, sample efficiency, and robustness of the learned surrogate likelihood.

- **Hierarchical modeling of multi-scale dynamics:** Many real-world systems exhibit behavior across multiple timescales, such as fast fluctuations overlaying slow trends. A promising direction is to construct hierarchical T-SNL models, where each layer is responsible for capturing dynamics at a different resolution. This could improve interpretability, generalization, and predictive performance in complex systems such as climate models, neural recordings, or macroeconomic indicators.

- **Applications to real-world systems:** The proposed methods are especially applicable to domains where uncertainty, limited data, and nonlinear behavior intersect. Fields such as robotics (e.g., sensor fusion and motion tracking), neuroscience (e.g., latent neural dynamics), and environmental science (e.g., ecosystem monitoring) are particularly promising targets. Collaborations with domain experts could help tailor the algorithms for practical deployment and drive new methodological improvements.

In summary, this thesis has made methodological contributions to the field of probabilistic modeling for dynamical systems by developing adaptive, efficient,

and scalable tools for inference in state-space models. The proposed AGSF and T-SNL algorithms, in addition to being effective and practical solutions that can be readily used by practitioners, are also grounded in general principles that have broader applicability. The idea of introducing latent augmentation in AGSF provides a flexible mechanism to interpolate between different inference regimes, and this principle can be integrated into other filtering or smoothing algorithms beyond the specific setting considered here. Similarly, the truncation strategy used in T-SNL, which exploits the natural decay of temporal dependence in many state-space models, is a general approach that can benefit a wide range of temporal models. These underlying ideas — augmentation and truncation — are not limited to the algorithms developed in this thesis, but rather offer conceptual tools that can inspire new methods, improve existing ones, or be adapted to other modeling frameworks where efficiency, flexibility, and structure are required. As such, the contributions of this thesis go beyond their immediate implementation, and open up possibilities for further methodological development in the broader field of sequential and probabilistic modeling.

Bibliography

[Abd+20] Hazem Abdelkawy et al. "Spatio-temporal convolutional networks and N-ary ontologies for human activity-aware robotic system". In: *IEEE Robotics and Automation Letters* 6.2 (2020), pp. 620–627.

[ADH10] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. "Particle markov chain monte carlo methods". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 72.3 (2010), pp. 269–342.

[ADT24] Rafael Anderka, Marc Peter Deisenroth, and So Takao. "Iterated INLA for state and parameter estimation in nonlinear dynamical systems". In: *arXiv preprint arXiv:2402.17036* (2024).

[AH09] Ienkaran Arasaratnam and Simon Haykin. "Cubature kalman filters". In: *IEEE Transactions on automatic control* 54.6 (2009), pp. 1254–1269.

[AHE07] Ienkaran Arasaratnam, Simon Haykin, and Robert J. Elliott. "Discrete-Time Nonlinear Filtering Algorithms Using Gauss–Hermite Quadrature". In: *Proceedings of the IEEE* 95.5 (2007), pp. 953–977. DOI: 10.1109/JPROC.2007.894705.

[AM12] Brian DO Anderson and John B Moore. *Optimal filtering*. Courier Corporation, 2012.

[Aok13] Masanao Aoki. *State space modeling of time series*. Springer Science & Business Media, 2013.

[Arc+15] Evan Archer et al. "Black box variational inference for state space models". In: *arXiv preprint arXiv:1511.07367* (2015).

[AS72a] D. Alspach and H. Sorenson. "Nonlinear Bayesian estimation using Gaussian sum approximations". In: *IEEE Transactions on Automatic Control* 17.4 (1972), pp. 439–448. DOI: 10.1109/TAC.1972.1100034.

[AS72b] D. Alspach and H. Sorenson. "Nonlinear Bayesian estimation using Gaussian sum approximations". In: *IEEE Transactions on Automatic Control* 17.4 (1972), pp. 439–448. DOI: 10.1109/TAC.1972.1100034.

[AT14] Mehdi Aghagolzadeh and Wilson Truccolo. "Latent state-space models for neural decoding". In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2014, pp. 3033–3036.

[Aus+24a] Alexander Aushev et al. "Likelihood-free inference in state-space models with unknown dynamics". In: *Statistics and Computing* 34.1 (2024), p. 27.

[Aus+24b] Alexander Aushev et al. "Likelihood-free inference in state-space models with unknown dynamics". In: *Statistics and Computing* 34.1 (2024), p. 27.

[Bea+09] Mark A. Beaumont et al. "Adaptive approximate Bayesian computation". In: *Biometrika* 96.4 (2009), pp. 983–990. ISSN: 00063444, 14643510. (Visited on 10/16/2024).

[Ber97] Dimitri P Bertsekas. "Nonlinear programming". In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334.

[Bis94] Christopher M Bishop. "Mixture density networks". In: (1994).

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[Box+15] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[Bra+18] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/google/jax.

[BRA13] Enis Bayramoglu, Ole Ravn, and Nils Axel Andersen Electrical. "A novel hypothesis splitting method implementation for multi-hypothesis filters". In: *2013 10th IEEE International Conference on Control and Automation (ICCA)*. 2013, pp. 574–579. DOI: 10.1109/ICCA.2013.6564951.

[Bro+20] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[Buc+04] ST Buckland et al. “State-space models for the dynamics of wild animal populations”. In: *Ecological modelling* 171.1-2 (2004), pp. 157–175.

[Caf98] Russel E Caflisch. “Monte carlo and quasi-monte carlo methods”. In: *Acta numerica* 7 (1998), pp. 1–49.

[CBL20] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. “The frontier of simulation-based inference”. In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30055–30062. DOI: 10.1073/pnas.1912789117.

[CD02] Dan Crisan and Arnaud Doucet. “A survey of convergence results on particle filtering methods for practitioners”. In: *IEEE Transactions on signal processing* 50.3 (2002), pp. 736–746.

[CJP13] Nicolas Chopin, Pierre E Jacob, and Omiros Papaspiliopoulos. “SMC2: an efficient algorithm for sequential analysis of state space models”. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 75.3 (2013), pp. 397–426.

[CLN22] Karthik Comandur, Yunpeng Li, and Santosh Nannuru. “Particle Flow Gaussian Particle Filter”. In: *2022 25th International Conference on Information Fusion (FUSION)*. IEEE. 2022, pp. 1–6.

[COA09] Siddhartha Chib, Yasuhiro Omori, and Manabu Asai. “Multivariate Stochastic Volatility”. In: *Handbook of Financial Time Series*. Ed. by Thomas Mikosch et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 365–400. ISBN: 978-3-540-71297-8. DOI: 10.1007/978-3-540-71297-8_16. URL: https://doi.org/10.1007/978-3-540-71297-8_16.

[Col+24] ATLAS Collaboration et al. “An implementation of neural simulation-based inference for parameter estimation in ATLAS”. In: *arXiv preprint arXiv:2412.01600* (2024).

[CP+20] Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*. Vol. 4. Springer, 2020.

[Cro+11a] David F Crouse et al. “A look at Gaussian mixture reduction algorithms”. In: *14th International Conference on Information Fusion*. IEEE. 2011, pp. 1–8.

[Cro+11b] David F. Crouse et al. “A look at Gaussian mixture reduction algorithms”. In: *14th International Conference on Information Fusion*. 2011, pp. 1–8.

[Csi+10] Katalin Csilléry et al. “Approximate Bayesian computation (ABC) in practice”. In: *Trends in ecology & evolution* 25.7 (2010), pp. 410–418.

[DBJ13] Kyle J DeMars, Robert H Bishop, and Moriba K Jah. “Entropy-based approach for uncertainty propagation of nonlinear dynamical systems”. In: *Journal of Guidance, Control, and Dynamics* 36.4 (2013), pp. 1047–1057.

[DD04] Pierre Del Moral and Pierre Del Moral. *Feynman-kac formulae:Genealogical and Interacting Particle Systems with Applications*. Springer, 2004.

[DDG+01] Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*. Vol. 1. 2. Springer, 2001.

[Dea+14] Thomas A Dean et al. “Parameter estimation for hidden Markov models with intractable likelihoods”. In: *Scandinavian Journal of Statistics* 41.4 (2014), pp. 970–987.

[DH03] Fred Daum and Jim Huang. “Curse of dimensionality and particle filters”. In: *2003 IEEE aerospace conference proceedings (Cat. No. 03TH8652)*. Vol. 4. IEEE. 2003, 4_1979–4_1993.

[DJ09] Arnaud Doucet and Adam M Johansen. “A tutorial on particle filtering and smoothing: Fifteen years later”. In: *Handbook of nonlinear filtering* 12.656-704 (2009).

[Dju+03] Petar M Djuric et al. “Particle filtering”. In: *IEEE signal processing magazine* 20.5 (2003), pp. 19–38.

[Doe+18] Andreas Doerr et al. “Probabilistic recurrent state-space models”. In: *International conference on machine learning*. PMLR. 2018, pp. 1280–1289.

[DSB16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using real nvp”. In: *arXiv preprint arXiv:1605.08803* (2016).

[Dua+87] Simon Duane et al. “Hybrid Monte Carlo”. In: *Physics Letters B* 195.2 (1987), pp. 216–222. ISSN: 0370-2693. DOI: https://doi.org/10.1016/0370-2693(87)91197-X. URL: https://www.sciencedirect.com/science/article/pii/037026938791197X.

[Elv+19a] Victor Elvira et al. “Elucidating the Auxiliary Particle Filter via Multiple Importance Sampling [Lecture Notes]”. In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 145–152. DOI: 10.1109/MSP.2019.2938026.

[Elv+19b] Víctor Elvira et al. “Generalized Multiple Importance Sampling”. In: *Statistical Science* 34.1 (2019), pp. 129–155. DOI: 10.1214/18-STS668. URL: https://doi.org/10.1214/18-STS668.

[EM21] Víctor Elvira and Luca Martino. “Advances in Importance Sampling”. In: Feb. 2021, pp. 1–14. ISBN: 9781118445112. DOI: 10.1002/9781118445112.stat08284.

[EMC21] Víctor Elvira, Luca Martino, and Pau Closas. “Importance Gaussian Quadrature”. In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 474–488. DOI: 10.1109/TSP.2020.3045526.

[EMD21] Víctor Elvira, Joaquín Miguez, and Petar M Djurić. “On the performance of particle filters with adaptive number of particles”. In: *Statistics and Computing* 31 (2021), pp. 1–18.

[Emm21] Frank Emmert-Streib. “From the digital data revolution toward a digital society: Pervasiveness of artificial intelligence”. In: *Machine Learning and Knowledge Extraction* 3.1 (2021), pp. 284–298.

[EMR22] Víctor Elvira, Luca Martino, and Christian P Robert. “Rethinking the effective sample size”. In: *International Statistical Review* 90.3 (2022), pp. 525–550.

[FK10] Friedrich Faubel and Dietrich Klakow. “Further improvement of the adaptive level of detail transform: Splitting in direction of the nonlinearity”. In: *2010 18th European Signal Processing Conference*. 2010, pp. 850–854.

[FMK09] Friedrich Faubel, John McDonough, and Dietrich Klakow. “The split and merge unscented Gaussian mixture filter”. In: *IEEE Signal Processing Letters* 16.9 (2009), pp. 786–789.

[Fra+18] David T Frazier et al. “Asymptotic properties of approximate Bayesian computation”. In: *Biometrika* 105.3 (2018), pp. 593–607.

[Gar+15] Ángel F García-Fernández et al. “Posterior linearization filter: Principles and implementation using sigma points”. In: *IEEE transactions on signal processing* 63.20 (2015), pp. 5561–5573.

[GB99] Zoubin Ghahramani and Matthew Beal. "Variational inference for Bayesian mixtures of factor analysers". In: *Advances in neural information processing systems* 12 (1999).

[Ger+15] Mathieu Germain et al. "MADE: Masked Autoencoder for Distribution Estimation". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 881–889.

[GH12] Fredrik Gustafsson and Gustaf Hendeby. "Some Relations Between Extended and Unscented Kalman Filters". In: *IEEE Transactions on Signal Processing* 60.2 (2012), pp. 545–555. DOI: 10.1109/TSP.2011.2172431.

[Gil77] Daniel T Gillespie. "Exact stochastic simulation of coupled chemical reactions". In: *The journal of physical chemistry* 81.25 (1977), pp. 2340–2361.

[Gla+20] Joshua Glaser et al. "Recurrent switching dynamical systems models for multiple interacting neural populations". In: *Advances in neural information processing systems* 33 (2020), pp. 14867–14878.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee. 2013, pp. 6645–6649.

[GNM19] David Greenberg, Marcel Nonnenmacher, and Jakob Macke. "Automatic posterior transformation for likelihood-free inference". In: *International conference on machine learning*. PMLR. 2019, pp. 2404–2414.

[GR15] Jakob Grazzini and Matteo Richiardi. "Estimation of ergodic agent-based models by simulated minimum distance". In: *Journal of Economic Dynamics and Control* 51 (2015), pp. 148–165. ISSN: 0165-1889. DOI: https://doi.org/10.1016/j.jedc.2014.10.006. URL: https://www.sciencedirect.com/science/article/pii/S0165188914002814.

[Gre+12] Arthur Gretton et al. "A Kernel Two-Sample Test". In: *Journal of Machine Learning Research* 13.25 (2012), pp. 723–773. URL: http://jmlr.org/papers/v13/gretton12a.html.

[GRT17] Jakob Grazzini, Matteo G. Richiardi, and Mike Tsionas. "Bayesian estimation of agent-based models". In: *Journal of Economic Dynamics and Control* 77 (2017), pp. 26–47. ISSN: 0165-1889. DOI: https://doi.org/10.1016/j.jedc.2017.01.014. URL: https://www.sciencedirect.com/science/article/pii/S0165188917300222.

[GSS93] Neil J Gordon, David J Salmond, and Adrian FM Smith. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". In: *IEE proceedings F (radar and signal processing)*. Vol. 140. 2. IET. 1993, pp. 107–113.

[Gus+02] Fredrik Gustafsson et al. "Particle filters for positioning, navigation, and tracking". In: *IEEE Transactions on signal processing* 50.2 (2002), pp. 425–437.

[Hah+17] ChangHoon Hahn et al. "Approximate Bayesian computation in large-scale structure: constraining the galaxy–halo connection". In: *Monthly Notices of the Royal Astronomical Society* 469.3 (Apr. 2017), pp. 2791–2805. ISSN: 0035-8711. DOI: 10.1093/mnras/stx894. eprint: https://academic.oup.com/mnras/article-pdf/469/3/2791/17638868/stx894.pdf. URL: https://doi.org/10.1093/mnras/stx894.

[Ham94] James D Hamilton. "State-space models". In: *Handbook of econometrics* 4 (1994), pp. 3039–3080.

[HBL20] Joeri Hermans, Volodimir Begy, and Gilles Louppe. "Likelihood-free mcmc with amortized approximate ratio estimators". In: *International conference on machine learning*. PMLR. 2020, pp. 4239–4248.

[HC13] Frank Havlak and Mark Campbell. "Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments". In: *IEEE Transactions on Robotics* 30.2 (2013), pp. 461–474.

[Hil87] Francis Begnaud Hildebrand. *Introduction to numerical analysis*. Courier Corporation, 1987.

[IX00a] K. Ito and K. Xiong. "Gaussian filters for nonlinear filtering problems". In: *IEEE Transactions on Automatic Control* 45.5 (2000), pp. 910–927. DOI: 10.1109/9.855552.

[IX00b] K. Ito and K. Xiong. "Gaussian filters for nonlinear filtering problems". In: *IEEE Transactions on Automatic Control* 45.5 (2000), pp. 910–927. DOI: 10.1109/9.855552.

[Jaz07] Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[Joh+00] Rolf Johansson et al. "State-space system identification of robot manipulator dynamics". In: *Mechatronics* 10.3 (2000), pp. 403–418.

[JU04a] S.J. Julier and J.K. Uhlmann. "Unscented filtering and nonlinear estimation". In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422. DOI: 10.1109/JPROC.2003.823141.

[JU04b] S.J. Julier and J.K. Uhlmann. "Unscented filtering and nonlinear estimation". In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422. DOI: 10.1109/JPROC.2003.823141.

[JU96] Simon Julier and Jeffrey K. Uhlmann. *A General Method for Approximating Nonlinear Transformations of Probability Distributions*. Tech. rep. 1996.

[Kal60] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems". In: (1960).

[Kan+15] Nikolas Kantas et al. "On Particle Methods for Parameter Estimation in State-Space Models". In: *Statistical Science* 30.3 (Aug. 2015), pp. 328–351. DOI: 10.1214/14-STS511.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[KB61] Rudolph E Kalman and Richard S Bucy. "New results in linear filtering and prediction theory". In: (1961).

[KD03a] J.H. Kotecha and P.M. Djuric. "Gaussian particle filtering". In: *IEEE Transactions on Signal Processing* 51.10 (2003), pp. 2592–2601. DOI: 10.1109/TSP.2003.816758.

[KD03b] Jayesh H Kotecha and Petar M Djuric. "Gaussian sum particle filtering". In: *IEEE Transactions on signal processing* 51.10 (2003), pp. 2602–2612.

[Kit96] Genshiro Kitagawa. "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models". In: *Journal of computational and graphical statistics* 5.1 (1996), pp. 1–25.

[KM27] W. O. Kermack and A. G. McKendrick. "A contribution to the mathematical theory of epidemics". In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 115 (772 1927), pp. 700–721. DOI: 10.1098/rspa.1927.0118.

[KS22] Ramis Khabibullin and Sergei Seleznev. "Fast Estimation of Bayesian State Space Models Using Amortized Simulation-Based Inference". In: *arXiv preprint arXiv:2210.07154* (2022).

[LBD15] Tiancheng Li, Miodrag Bolic, and Petar M Djuric. "Resampling methods for particle filtering: classification, implementation, and strategies". In: *IEEE Signal processing magazine* 32.3 (2015), pp. 70–86.

[Lea+17] Colin Lea et al. "Temporal convolutional networks for action segmentation and detection". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 156–165.

[Lin+19] Scott Linderman et al. "Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in C. elegans". In: *BioRxiv* (2019), p. 621540.

[LKJ09] Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. "Generating random correlation matrices based on vines and extended onion method". In: *Journal of multivariate analysis* 100.9 (2009), pp. 1989–2001.

[LO09] Pierre L'Ecuyer and Art B Owen. *Monte Carlo and Quasi-Monte Carlo Methods 2008*. Springer, 2009.

[Lot20] Alfred J. Lotka. "Analytical Note on Certain Rhythmic Relations in Organic Systems". In: *Proceedings of the National Academy of Sciences* 6.7 (1920), pp. 410–415. DOI: 10.1073/pnas.6.7.410. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.6.7.410. URL: https://www.pnas.org/doi/abs/10.1073/pnas.6.7.410.

[LT11] Hedibert F Lopes and Ruey S Tsay. "Particle filters and Bayesian inference in financial econometrics". In: *Journal of Forecasting* 30.1 (2011), pp. 168–209.

[Lue+17] Jan-Matthis Lueckmann et al. "Flexible statistical inference for mechanistic models of neural dynamics". In: *Advances in neural information processing systems* 30 (2017).

[Lue+21] Jan-Matthis Lueckmann et al. “Benchmarking simulation-based inference”. In: *International conference on artificial intelligence and statistics*. PMLR. 2021, pp. 343–351.

[MAM10] Iain Murray, Ryan Adams, and David MacKay. “Elliptical slice sampling”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 541–548.

[Mar+03] Paul Marjoram et al. “Markov chain Monte Carlo without likelihoods”. In: *Proceedings of the National Academy of Sciences* 100.26 (2003), pp. 15324–15328. DOI: 10.1073/pnas.0306899100.

[Mar+19] Gael M Martin et al. “Auxiliary likelihood-based approximate Bayesian computation in state space models”. In: *Journal of Computational and Graphical Statistics* 28.3 (2019), pp. 508–522.

[May82] Peter S Maybeck. *Stochastic models, estimation, and control*. Academic press, 1982.

[Met+53] Nicholas Metropolis et al. “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[MS85] Leonard A McGee and Stanley F Schmidt. *Discovery of the Kalman filter as a practical tool for aerospace and industry*. Tech. rep. 1985.

[MW03] R. van der Merwe and E. Wan. “Gaussian mixture sigma-point particle filters for sequential probabilistic inference in dynamic state-space models”. In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP ’03)*. Vol. 6. 2003, pp. VI–701. DOI: 10.1109/ICASSP.2003.1201778.

[Ner+23] Subhash Nerella et al. “Transformers in healthcare: A survey”. In: *arXiv preprint arXiv:2307.00067* (2023).

[New+23] Ken Newman et al. “State-space models for ecological time-series data: Practical model-fitting”. In: *Methods in Ecology and Evolution* 14.1 (2023), pp. 26–42.

[Pan+10] Liam Paninski et al. “A new look at state-space models for neural data”. In: *Journal of computational neuroscience* 29 (2010), pp. 107–126.

[Pes+23] Henri Pesonen et al. “ABC of the future”. In: *International Statistical Review* 91.2 (2023), pp. 243–268. DOI: https://doi.org/10.1111/insr.12522.

[PM16] George Papamakarios and Iain Murray. “Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016.

[PMM18] Sara Pérez-Vieites, Inés P Mariño, and Joaquín Míguez. “Probabilistic scheme for joint parameter estimation and state prediction in complex dynamical systems”. In: *Physical Review E* 98.6 (2018), p. 063305.

[PPM17] George Papamakarios, Theo Pavlakou, and Iain Murray. “Masked Autoregressive Flow for Density Estimation”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[PS99] Michael K Pitt and Neil Shephard. “Filtering via simulation: Auxiliary particle filters”. In: *Journal of the American statistical association* 94.446 (1999), pp. 590–599.

[Psi13] Mark L Psiaki. “The blind tricyclist problem and a comparative study of nonlinear filters: A challenging benchmark for evaluating nonlinear estimation methods”. In: *IEEE Control Systems Magazine* 33.3 (2013), pp. 40–54.

[Psi14] Mark L. Psiaki. “The "Blob" Filter: Gaussian mixture nonlinear filtering with re-sampling for mixand narrowing”. In: *2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014*. 2014, pp. 393–406. DOI: 10.1109/PLANS.2014.6851397.

[Psi16] Mark L. Psiaki. “Gaussian Mixture Nonlinear Filtering With Resampling for Mixand Narrowing”. In: *IEEE Transactions on Signal Processing* 64.21 (2016), pp. 5499–5512. DOI: 10.1109/TSP.2016.2595503.

[PSM15] Mark L Psiaki, Jonathan R Schoenberg, and Isaac T Miller. “Gaussian sum reapproximation for use in a nonlinear filter”. In: *Journal of Guidance, Control, and Dynamics* 38.2 (2015), pp. 292–303.

[PSM19] George Papamakarios, David Sterratt, and Iain Murray. "Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, 16–18 Apr 2019, pp. 837–848.

[RAG03] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech house, 2003.

[Rao73] Calyampudi Radhakrishna Rao. *Linear statistical inference and its applications*. Wiley New York, 1973.

[RC11] Christian Robert and George Casella. "A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data". In: (2011).

[RC99] Christian P Robert and George Casella. *Monte Carlo statistical methods*. Vol. 2. Springer, 1999.

[RM15] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[RMC09] Håvard Rue, Sara Martino, and Nicolas Chopin. "Approximate Bayesian Inference for Latent Gaussian models by using Integrated Nested Laplace Approximations". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 71.2 (Apr. 2009), pp. 319–392. ISSN: 1369-7412. DOI: 10.1111/j.1467-9868.2008.00700.x. eprint: https://academic.oup.com/jrsssb/article-pdf/71/2/319/49686253/jrsssb_71_2_319.pdf. URL: https://doi.org/10.1111/j.1467-9868.2008.00700.x.

[RRM03] Christopher V Rao, James B Rawlings, and David Q Mayne. "Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations". In: *IEEE transactions on automatic control* 48.2 (2003), pp. 246–258.

[Rub84] Donald B Rubin. "Bayesianly justifiable and relevant frequency calculations for the applied statistician". In: *The Annals of Statistics* (1984), pp. 1151–1172.

[Ryd+21] Thomas Ryder et al. “The neural moving average model for scalable variational inference of state space models”. In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 12–22.

[SA71a] H.W. Sorenson and D.L. Alspach. “Recursive bayesian estimation using gaussian sums”. In: *Automatica* 7.4 (1971), pp. 465–479. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(71)90097-5. URL: https://www.sciencedirect.com/science/article/pii/0005109871900975.

[SA71b] H.W. Sorenson and D.L. Alspach. “Recursive bayesian estimation using gaussian sums”. In: *Automatica* 7.4 (1971), pp. 465–479. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(71)90097-5. URL: https://www.sciencedirect.com/science/article/pii/0005109871900975.

[Sär13] Simo Särkkä. *Bayesian filtering and smoothing*. Cambridge university press, 2013.

[Sch19] Robin M Schmidt. “Recurrent neural networks (rnns): A gentle introduction and overview”. In: *arXiv preprint arXiv:1912.05911* (2019).

[Sch78] Gideon Schwarz. “Estimating the dimension of a model”. In: *The annals of statistics* (1978), pp. 461–464.

[SDŠ11] Ondřej Straka, Jindřich Duník, and Miroslav Šimandl. “Gaussian sum unscented Kalman filter with adaptive scaling parameters”. In: *14th International Conference on Information Fusion*. IEEE. 2011, pp. 1–8.

[SFB18] Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. CRC press, 2018.

[SFT07] Scott A Sisson, Yanan Fan, and Mark M Tanaka. “Sequential monte carlo without likelihoods”. In: *Proceedings of the National Academy of Sciences* 104.6 (2007), pp. 1760–1765.

[SH09] Dennis Schieferdecker and Marco F Huber. “Gaussian mixture reduction via clustering”. In: *2009 12th international conference on information fusion*. IEEE. 2009, pp. 1536–1543.

[Sil18] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[Sim+21] Umberto Simola et al. “Adaptive approximate Bayesian computation tolerance selection”. In: *Bayesian analysis* 16.2 (2021), pp. 397–423.

[SLW24] Samuel Stockman, Daniel J Lawson, and Maximilian J Werner. "SB-ETAS: using simulation based inference for scalable, likelihood-free inference for the ETAS model of earthquake occurrences". In: *Statistics and Computing* 34.5 (2024), p. 174.

[SS23] Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*. Vol. 17. Cambridge university press, 2023.

[SSM62] Gerald L Smith, Stanley F Schmidt, and Leonard A McGee. *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration, 1962.

[Sun+13] Mikael Sunnåker et al. "Approximate bayesian computation". In: *PLoS computational biology* 9.1 (2013), e1002803.

[Tav+97] Simon Tavaré et al. "Inferring coalescence times from DNA sequence data". In: *Genetics* 145.2 (1997), pp. 505–518.

[TE23] Kostas Tsampourakis and Víctor Elvira. "An Augmented Gaussian Sum Filter through a mixture Decomposition". In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095899.

[Tho+22] Owen Thomas et al. "Likelihood-free inference by ratio estimation". In: *Bayesian Analysis* 17.1 (2022), pp. 1–31.

[Thr02a] Sebastian Thrun. "Particle Filters in Robotics." In: *UAI*. Vol. 2. Citeseer. 2002, pp. 511–518.

[Thr02b] Sebastian Thrun. "Probabilistic robotics". In: *Communications of the ACM* 45.3 (2002), pp. 52–57.

[TMN98] Petr Tichavsky, Carlos H Muravchik, and Arye Nehorai. "Posterior Cramér-Rao bounds for discrete-time nonlinear filtering". In: *IEEE Transactions on signal processing* 46.5 (1998), pp. 1386–1396.

[Ton+09] Tina Toni et al. "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". In: *Journal of the Royal Society Interface* 6.31 (2009), pp. 187–202.

[TZ18] Kirsten Tuggle and Renato Zanetti. "Automated splitting Gaussian mixture nonlinear measurement update". In: *Journal of Guidance, Control, and Dynamics* 41.3 (2018), pp. 725–734.

[Van+00] Rudolph Van Der Merwe et al. “The unscented particle filter”. In: *Advances in neural information processing systems* 13 (2000).

[Van04] Rudolph Van Der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. Oregon Health & Science University, 2004.

[Vas+17] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[Ver+23] Hippolyte Verdier et al. “Simulation-based inference for non-parametric statistical comparison of biomolecule dynamics”. In: *PLOS Computational Biology* 19.2 (Feb. 2023), pp. 1–24. DOI: 10.1371/journal.pcbi.1010088. URL: https://doi.org/10.1371/journal.pcbi.1010088.

[VR16] Vivek Vittaldev and Ryan P Russell. “Space object collision probability using multidirectional gaussian mixture models”. In: *Journal of Guidance, Control, and Dynamics* 39.9 (2016), pp. 2163–2169.

[War+20] Wil Ward et al. “Black-box inference for non-linear latent force models”. In: *International conference on artificial intelligence and statistics*. PMLR. 2020, pp. 3088–3098.

[Wil18] Darren J Wilkinson. *Stochastic modelling for systems biology*. Chapman and Hall/CRC, 2018.

[WV00a] E.A. Wan and R. Van Der Merwe. “The unscented Kalman filter for nonlinear estimation”. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.

[WV00b] Eric A Wan and Rudolph Van Der Merwe. “The unscented Kalman filter for nonlinear estimation”. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee. 2000, pp. 153–158.

[WV01] Eric A Wan and Rudolph Van Der Merwe. “The unscented Kalman filter”. In: *Kalman filtering and neural networks* (2001), pp. 221–280.

[ZT18] Renato Zanetti and Kirsten Tuggle. “A novel Gaussian mixture approximation for nonlinear estimation”. In: *2018 21st International Conference on Information Fusion (FUSION)*. IEEE. 2018, pp. 1100–1106.