BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF CONTROL AND INSTRUMENTATION
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

MONTE CARLO-BASED IDENTIFICATION STRATEGIES FOR STATE-SPACE MODELS

MONTE CARLO IDENTIFIKAČNÍ STRATEGIE PRO STAVOVÉ MODELY

DOCTORAL THESIS
DIZERTAČNÍ PRÁCE

AUTHOR AUTOR PRÁCE	Ing. MILAN PAPEŽ
ADVISOR VEDOUCÍ PRÁCE	prof. Ing. PETR PIVOŇKA, CSc.

BRNO 2018

ABSTRACT

State-space models are immensely useful in various areas of science and engineering. Their attractiveness results mainly from the fact that they provide a generic tool for describing a wide range of real-world dynamical systems. However, owing to their generality, the associated state and parameter inference objectives are analytically intractable in most practical cases. The present thesis considers two particularly important classes of nonlinear and non-Gaussian state-space models: conditionally conjugate state-space models and jump Markov nonlinear models. A key feature of these models lies in that—despite their intractability—they comprise a tractable substructure. The intractable part requires us to utilize approximate techniques. Monte Carlo computational methods constitute a theoretically and practically well-established tool to address this problem. The advantage of these models is that the tractable part can be exploited to increase the efficiency of Monte Carlo methods by resorting to the Rao-Blackwellization. Specifically, this thesis proposes two Rao-Blackwellized particle filters for identification of either static or time-varying parameters in conditionally conjugate state-space models. Furthermore, this work adopts recent particle Markov chain Monte Carlo methodology to design Rao-Blackwellized particle Gibbs kernels for state smoothing in jump Markov nonlinear models. The kernels are then used to facilitate maximum likelihood parameter inference in the considered models. The resulting experiments demonstrate that the proposed algorithms outperform related techniques in terms of the estimation precision and computational time.

KEYWORDS

Sequential Monte Carlo, particle Markov chain Monte Carlo, nonlinear and non-Gaussian state-space models, conditionally conjugate state-space models, jump Markov nonlinear models, state and parameter inference, identification of static and time-varying parameters

ABSTRAKT

Stavové modely jsou neobyčejně užitečné v mnoha inženýrských a vědeckých oblastech. Jejich atraktivita vychází především z toho faktu, že poskytují obecný nástroj pro popis široké škály dynamických systémů reálného světa. Nicméně, z důvodu jejich obecnosti, přidružené úlohy inference parametrů a stavů jsou ve většině praktických situacích nepoddajné. Tato dizertační práce uvažuje dvě zvláště důležité třídy nelineárních a ne-Gaussovských stavových modelů: podmíněně konjugované stavové modely a Markovsky přepínající nelineární modely. Hlavní rys těchto modelů spočívá v tom, že—navzdory jejich nepoddajnosti—obsahují poddajnou podstrukturu. Nepoddajná část požaduje abychom využily aproximační techniky. Monte Carlo výpočetní metody představují teoreticky a prakticky dobře etablovaný nástroj pro řešení tohoto problému. Výhoda těchto modelů spočívá v tom, že poddajná část může být využita pro zvýšení efektivity Monte Carlo metod tím, že se uchýlíme k Rao-Blackwellizaci. Konkrétně, tato doktorská práce navrhuje dva Rao-Blackwellizované částicové filtry pro identifikaci buďto statických anebo časově proměnných parametrů v podmíněně konjugovaných stavových modelech. Kromě toho, tato práce adoptuje nedávnou particle Markov chain Monte Carlo metodologii pro návrh Rao-Blackwellizovaných částicových Gibbsových jader pro vyhlazování stavů v Markovsky přepínajících nelineárních modelech. Tyto jádra jsou posléze použity pro inferenci parametrů metodou maximální věrohodnosti v uvažovaných modelech. Výsledné experimenty demonstrují, že navržené algoritmy překonávají příbuzné techniky ve smyslu přesnosti odhadu a výpočetního času.

KLÍČOVÁ SLOVA

Sekvenční Monte Carlo, particle Markov chain Monte Carlo, nelineární a ne-Gaussovské stavové modely, podmíněně konjugované stavové modely, Markovsky přepínající nelineární modely, inference stavů a parametrů, identifikace statických a časově proměnných parametrů

PAPEŽ, Milan. *Monte Carlo-Based Identification Strategies for State-Space Models*. Brno, 2018, 223 p. Doctoral thesis. Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation. Advised by prof. Ing. Petr Pivoňka, CSc.

DECLARATION

ACKNOWLEDGEMENT

I would like to thank my supervisor, Prof. Petr Pivoňka, for his encouragement. I want to express my gratitude to Prof. Pavel Václavek whose valuable support played an important role over the course of working on this thesis. Many thanks to Dr. John Leth and Dr. Torben Knudsen for being supportive and interested in my work throughout my stay at the Department of Automation and Control, Aalborg University. I am also grateful to Dr. Anthony Quinn for inspirational and fruitful collaboration over the last couple of months.

I am truly thankful to my family for their love and incredible support during all my life. Most importantly, I would like to thank my girlfriend for her love, patience, and encouragement over the years.

The research in chapters 3 and 4 was made possible by the grants FEKT-S-17-4234 and FEKT-S-14-2429, respectively. The work in chapters 4 (partially), 5, and 6 was carried out under the support from the project CEITEC 2020 (LQ1601). The research in chapter 7 was supported by the grant GAČR 18-15970S.

Brno . . . .

author’s signature

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Context

A state-space model is a generic tool to embody our intuition about time-space dependent and stochastic behaviour of a real-world dynamical system. The necessary step towards drawing conclusions about such a system is to observe data on it. The model and data are then used to carry out various statistical inference objectives, including the estimation of latent states and parameters. However, dynamical systems are mostly nonlinear and non-Gaussian, which makes the associated inference objectives analytically intractable and therefore poses a real challenge on the design of high-fidelity approximation techniques.

Sequential Monte Carlo (SMC) methodology [47] is particularly well suited for this aim. SMC methods provide approximate solutions based on generating a collection of random samples. A range of convergence results [217] for these approaches proves that as the number of samples increases, quantities of interest are approximated with increasingly high precision. This ability comes naturally with the question of high computational complexity. Fortunately, the computational power is still growing—albeit not as rapidly in the sense of the Moore's law as before, but rather in terms of parallel architectures [60]—which makes this question relative, but relevant mainly when the problem is high-dimensional or the computational resources are limited. However, there exist particularly useful and general classes of nonlinear and non-Gaussian state-space models that contain analytically tractable substructures. This feature is commonly utilized in the design of SMC methods in order to improve their computational efficiency through the Rao-Blackwellization [36]. In such cases, an algorithm relying on this principle can have the same estimation precision as an algorithm without this improvement but at a lower computational cost. The requirement of providing highly reliable approximate solutions to various inference objectives in state-space models has recently recorded a significant conceptual shift, namely the particle Markov chain Monte Carlo (MCMC) methodology [4]. Particle MCMC algorithms can be seen as exact approximations of the ideal MCMC procedures. These methods run an SMC method at each iteration in order to produce a single sample of a quantity of interest, making them highly computationally demanding. Therefore, even a slight improvement in the estimation accuracy of these methods can have a profound impact on the computational time.

This thesis is about algorithm design. The aim is to develop computationally efficient Monte Carlo techniques for two generic classes of nonlinear and non-Gaussian state-space models. The first class is formed by the conditionally conjugate state-space models. Their characteristic feature lies in that they contain an algebraically tractable substructure with respect to the parameters but an intractable substructure with respect to the unobserved states. These models have been applied to a broad range of diverse practical problems, including computer code performance tuning [86], flu epidemics tracking [59], vehicle navigation systems [167], target tracking [158], online recommendation services [226], estimation of the remaining useful life of batteries [139], learning of cellular dynamics in system biology [155], web activity modeling [146], optimization of portfolio returns [96], to mention a few. The second class is given by the jump Markov nonlinear models. Their key aspect is that they are formed by a finite number of nonlinear and non-Gaussian state-space configurations that switch according to a discrete-valued Markov chain. These configurations constitute the intractable part of the model, whereas the discrete chain forms the tractable part. These models have also become substantially popular in various practical applications, such as learning of consumption growth dynamics [97], traffic behavior analysis through video surveillance [13], virus-cell fusion identification [79], molecular bioimaging [197], detection of abrupt changes in financial markets [141], sensor networks [214], simultaneous localization and tracking [127], terrain-based navigation [23], estimation of drivers' behavior [128], etc. The design of precise and fast computational strategies can provide a substantial increase in efficiency in the above applications, potentially decreasing the cost of associated hardware tools.

Outline

The present thesis is divided into two parts. The first one is composed of the first two chapters and provides the reader with an introduction into the basic concepts and tools used throughout the thesis. The second one is formed by a number of *separate* chapters that summarize novel methods and solutions. These chapters are extended versions of the author's published papers.

Chapter 1

The purpose of Chapter 1 is to shortly describe the basic Monte Carlo principles for approximating general and intractable probability distributions, and to review more advanced methods such as sequential Monte Carlo and particle Markov chain Monte Carlo in their generic form. A minor contribution of this chapter consists in presenting a number of simple examples to better describe some of the characteristic features of the considered methods.

Chapter 2

Chapter 2 discusses applicability of the generic tools introduced in Chapter 1 to the state and parameter inference in state-space models. We provide a number of examples in this context and compare the performance of these methods with some alternative strategies. Although this chapter is mainly a literature review, it also proposes some minor methodological developments and adaptations. Specifically, in Section 2.5, we demonstrate that the two-filter smoothing can be seen as a special case of backward-filtering forward-smoothing algorithm, a time-reversed version of the forward-filtering backward-smoothing.

Chapter 3

Chapter 3 proposes a Rao-Blackwellized particle filter for estimation of static parameters in conditionally conjugate state-space models. The novelty lies in that we exploit the analytically tractable substructure of these models to design projection-based updates of the statistics representing posterior distribution of the parameters. The experiments demonstrate that the method achieves higher estimation accuracy in less computational time compared to a number of alternative approaches.

Chapter 4

In Chapter 4, we use methodology similar to Chapter 3 in order to propose a Rao-Blackwellized particle filter for estimation of time-varying parameters in conditionally conjugate state-space models. Their algebraically tractable substructure is here utilized to facilitate application of the alternative stabilized forgetting so that the method can compensate for the lack of knowledge of the parameter time-evolution model. We provide experiments revealing that the proposed method improves estimation accuracy at the cost of reduced computational time compared to procedures that use different forms of forgetting.

Chapter 5

The jump Markov nonlinear models constitute a particularly challenging class of state-space models that are usually encountered in application areas with abruptly changing data. Chapter 5 is concerned with exploiting the structural properties of these models to design—based on particle Markov chain Monte Carlo methodology—Rao-Blackwellized particle Gibbs kernels for the state smoothing. The experiments investigate the impact of Rao-Blackwellization in this context and prove that the proposed kernels provide improved efficiency compared to the related procedures.

Chapter 6

Chapter 6 takes advantage of the ideas of Chapter 5 and proposes an algorithm for maximum likelihood parameter inference in jump Markov nonlinear models. Specifically, we utilize the Rao-Blackwellized particle Gibbs kernel from Chapter 5 in order to design a particle stochastic approximation expectation maximization algorithm for the considered class of models. We show that the proposed method increases the convergence speed compared to algorithms without Rao-Blackwellization, thus being less demanding in terms of the computational resources.

Chapter 7

Chapter 7 develops a fully probabilistic design-based transfer learning strategy for a pair of Kalman filters. The key characteristic of the proposed approach is that it does not impose any dependence assumptions between the quantities of the involved filtering algorithms. The results demonstrate that the proposed method can outperform strategies that do specify such dependence assumptions. Although it may appear that this approach is not related to Monte Carlo methods, we argue that this method is generic—representing a framework for knowledge transfer between a pair of *Bayesian* filters—and the present application to the Kalman filtering context serves only as the first step towards realizing the knowledge transfer between more advanced state inference techniques, such as Gaussian or particle filters.

Notation

The notation introduced in this section is valid for the first two chapters. The last five chapters have an independent and separately introduced notation, which is—to a large extent—similar to the first two chapters.

Abbreviations

Abbreviation	Description
BS	Backward simulation
MC	Monte Carlo
SMC	Sequential Monte Carlo
CSMC	Conditional sequential Monte Carlo
MCMC	Markov chain Monte Carlo
PMCMC	Particle Markov chain Monte Carlo
ESS	Effective sample size
RNE	Relative numerical efficiency
IS	Importance sampling
SIS	Sequential importance sampling
SIR	Sequential importance resampling
SISR	Sequential importance sampling and resampling
ASIR	Auxiliary sequential importance resampling
ASISR	Auxiliary sequential importance sampling and resampling
SNIS	Self-normalized importance sampling
IID	Independent and identically distributed
SSM	State-space model
RMSE	Root-mean-square error
EM	Expectation maximization
MCEM	Monte Carlo expectation maximization
SAEM	Stochastic approximation expectation maximization
LL	Local linearization
SLLN	Strong law of large numbers
CLT	Central limit theorem

Symbols

Symbol	Description
$\mathbb{R}$	The set of real numbers
$\mathbb{N}$	The set of natural numbers
π	A distribution
$X \sim \pi$	A π-distributed random variable
X	A generic space
$X_{1:t} = (X_1, \dots, X_t)$	A sequence of random variables
$\mathbf{X}_t = X_{1:t}$	A sequence of random variables
$\pi(f)$	Expectation of a test function f under distribution π
$\mathsf{E}[X]$	Expectation of a random variable X
$\mathsf{V}[X]$	Variance of a random variable X
$\mathsf{C}[X, Y]$	Covariance of random variables X and Y
$\mathsf{E}_\pi[X]$	Expectation of a random variable X under distribution π
$\mathsf{V}_\pi[X]$	Variance of a random variable X under π
$\mathsf{C}_\pi[X, Y]$	Covariance of random variables X and Y under π
$\xrightarrow{a.s.}$	Almost sure convergence
$\xrightarrow{d}$	Convergence in distribution
δ_X	Dirac delta measure located at X
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean vector μ and covariance matrix Σ
$\mathcal{N}(x; \mu, \Sigma)$	Gaussian density function with mean vector μ and covariance matrix Σ
$\mathcal{G}a(\alpha, \beta)$	Gamma distribution with the shape α and β rate parameters
$U[a, b)^M$	Uniform distribution on half-open interval $[a, b)$
$\lfloor x \rfloor$	The largest integer smaller than or equal to x
$\pi \ll q$	π is absolutely continuous with respect to q

1 MONTE CARLO METHODS

1.1 Monte Carlo Sampling

Monte Carlo simulation refers to a process of drawing a set of N independent and identically distributed (IID) random samples, or *particles*, $(X^i)_{i=1}^N$, from a target distribution π defined on possibly high-dimensional space $\mathsf{X} \subseteq \mathbb{R}^{n_x}$. The samples carry information about the target distribution that can be used to capture some of its characteristic features and to perform statistical inference about the quantities of interest. The Monte Carlo approach is a very popular and highly universal numerical technique which is mostly employed in approximating analytically intractable probability distributions and associated expectations.

Consider we are able to draw a set of random samples $(X^i)_{i=1}^N$ from π, then the empirical point-mass approximation of π can be constructed as

$$\pi^{MC,N}(dx) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X^i}(dx), \tag{1.1}$$

where δ_X is the Dirac-delta measure located at X. The empirical measure (1.1) is suitable for approximating an intractable (or complicated) expectation of a test function $f : \mathsf{X} \to \mathbb{R}$ with respect to π,

$$\pi(f) = \int f(x)\pi(dx), \tag{1.2}$$

by the empirical expectation

$$\pi^{MC,N}(f) = \frac{1}{N}\sum_{i=1}^{N} f(X^i), \tag{1.3}$$

which is obtained by simply inserting (1.1) in (1.2) and utilizing the fact that δ_{X^i} is one at X^i and zero otherwise. The following theorem underlines the key principle of basic Monte Carlo integration methods.

Theorem 1.1. *Let us consider a sequence of IID random samples $(X^i)_{i=1}^N$ and a measurable function $f : \mathsf{X} \to \mathbb{R}$ such that $\mathsf{E}[|f(X)|] < \infty$, then the empirical expectation (1.3) converges almost surely to the exact one (1.2) as the number of samples N increases, that is,*

$$\pi^{MC,N}(f) \xrightarrow{a.s.} \pi(f),$$

for $N \longrightarrow \infty$, with $\xrightarrow{a.s.}$ denoting almost sure convergence.

Proof. See [20]. □

Theorem 1.1 is an application of the strong law of large numbers (SLLN, [20]), which states that—as the number of samples gets large—the empirical expectation approaches the true value with probability one. Although it reflects the nature of Monte Carlo methods, Theorem 1.1 does not say anything about the statistical properties of the estimator (1.3). We present them in the following definition.

Definition 1.1. *Given a sequence of IID random samples $(X^i)_{i=1}^N$ and a measurable function $f : \mathsf{X} \to \mathbb{R}$ such that $\mathsf{E}[|f(X)|] < \infty$ and $\mathsf{E}[f(X)^2] < \infty$, we have*

$$\mathsf{E}[\pi^{MC,N}(f)] = \frac{1}{N}\sum_{i=1}^{N}\mathsf{E}[f(X^i)] = \pi(f), \tag{1.4}$$

$$\mathsf{V}[\pi^{MC,N}(f)] = \frac{1}{N^2}\sum_{i=1}^{N}\left(\mathsf{E}[f(X^i)^2] - \mathsf{E}[f(X^i)]^2\right) = \frac{1}{N}\mathsf{V}[f(X)], \tag{1.5}$$

where the expectation is taken with respect to π.

Definition 1.1 describes two properties of approximating integrals (1.2) based on Monte Carlo averages. The first one (1.4) demonstrates that the estimator (1.3) is unbiased, meaning that $\mathsf{E}[\pi^{MC,N}(f)] = \pi(f)$ for any N. The second one (1.5) states that the variance of (1.3) decreases with the increasing number of samples N, which allows us to make the variance arbitrarily small.

An important question is, what is the behaviour of the error when the estimator (1.3) approaches the true value of the integral (1.2) or, more precisely, what is the size and distribution of the error when the number of samples increases? The answer is provided in the following theorem.

Theorem 1.2. *Suppose $(X^i)_{i=1}^N$ is a sequence of IID random samples and $f : \mathsf{X} \to \mathbb{R}$ is a measurable function such that $\mathsf{E}[|f(X)|] < \infty$ and $\mathsf{E}[f(X)^2] < \infty$, then*

$$N^{\frac{1}{2}}[\pi^{MC,N}(f) - \pi(f)] \xrightarrow{d} \mathcal{N}(0, \mathsf{V}[f(X)]),$$

for $N \longrightarrow \infty$, where $\xrightarrow{d}$ labels the convergence in distribution.

Proof. See [117]. $\square$

Theorem 1.2 is the standard central limit theorem (CLT, [117]) and says that as the number of samples N increases, the asymptotic behaviour of the error follows approximately Gaussian distribution with the zero mean and variance $\mathsf{V}[f(X)]$. The theorem uncovers another key feature of the Monte Carlo approach, which consists in that the error convergences at $\mathcal{O}(N^{-\frac{1}{2}})$ rate.

On the one hand, this property is remarkable as the rate does not depend on the dimension n_x of the sample space X. Monte Carlo methods are then advertised as procedures that do not suffer from the curse of dimensionality—no matter the dimension they still converge at $\mathcal{O}(N^{-\frac{1}{2}})$ rate.[1] However, sampling from a high dimensional space X can often be computationally prohibitive. A question of repeatability of the random sequences also becomes important when the dimension is significantly large.

On the other hand, the convergence rate $\mathcal{O}(N^{-\frac{1}{2}})$ is rather slow, being a price for the robustness with respect to the high-dimensionality. Consider for instance that we want to reduce the error by an order of magnitude. Then, the convergence rate tells us that we need to produce 100 times more samples in order to accomplish this. Such a simple example demonstrates that the Monte Carlo techniques are computationally expensive. However, even a slight improvement in estimation accuracy of these methods can provide tremendous savings of the computational time, which motivates the development of computationally more efficient strategies. Although there are alternative families of methods for deterministic numerical integration, such as Newton-Cotes [176] and Gaussian [201] cubature rules[2], they typically outperform the Monte Carlo approach only when the dimension n_x is small. For example, the classical product rectangle rule provides $\mathcal{O}(N^{-\frac{1}{n_x}})$ convergence rate, which scales painfully in high-dimensional settings.

Theorem 1.2 generally demonstrates two possible ways of reducing the computational complexity of Monte Carlo methods. The first one is through the variance term $\mathsf{V}[f(X)]$ and the second one is through the exponent $\frac{1}{2}$. The variance can be influenced by changing the integrand, which is accomplished by the approaches such as antithetic variables, control variates, stratification, importance sampling, Rao-Blackwellization [183]. The exponent can be affected by the statistical properties of the samples. This is made possible by distributing them in a more clever way in the space X. A possible approach is (randomized) quasi-Monte Carlo method, which utilizes low-discrepancy point sets generated by means of suitably designed digital nets and sequences [161, 125]. The Halton sequence, for example, provides the $\mathcal{O}((\log N)^{n_x} N^{-1})$ (absolute) convergence rate (for sufficiently smooth functions). However, this rate can be dominated by the dimension-dependent $(\log N)^{n_x}$ term.

Theorem 1.2 describes the error with a certain probability, offering a possibility to compute confidence regions

$$(\pi^{MC,N}(f) \pm c_\alpha N^{-\frac{1}{2}} \sigma_N),$$

[1] Nevertheless, when devising more advanced Monte-Carlo constructions, such as those based on the importance sampling [135], the high-dimensional problems are usually difficult to address.

[2] The Newton-Cotes and Gaussian cubature rules space the points with an equal and unequal distance, respectively. The latter one does so based on the roots of certain polynomials [41].

where c_α is the $\alpha/2$ quantile of the standard Gaussian distribution and

$$\sigma_N^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\pi^{MC,N}(f) - f(X^i)\right)^2$$

is the empirical variance[3] [30]. Thus, we can use Theorem 1.2 to determine the number of samples N which is required for a particular experiment. Specifically, if we want a precision (error) level ϵ with a confidence level c_α, then we need $N = \epsilon^{-2}\sigma_N^2 c_\alpha$ samples. Here we see that, as the precision level gets tight, the computational burden grows fast. For example, the 0.95 confidence interval requires us to set $c_\alpha \approx 2$.

Another key characteristic of basic Monte-Carlo strategies lies in the $\mathcal{L}^p$-convergence, as shown by the following theorem.

Theorem 1.3. *Let $(X^i)_{i=1}^N$ be a sequence of IID random samples and $f : \mathsf{X} \to \mathbb{R}$ be a measurable function satisfying $\mathsf{E}[|f(X)|^p] < \infty$, for $p \geq 1$, then there exists $B_p^{MC} < \infty$ such that*

$$\mathsf{E}[|\pi^{MC,N}(f) - \pi(f)|^p]^{\frac{1}{p}} \leq N^{-\frac{1}{2}} B_p^{MC}.$$

Proof. See [40]. $\square$

Theorem 1.3 is a rephrased version of the Marcinkiewicz–Zygmund inequality [144] and describes asymptotic behaviour of the pth absolute central moment. It states that the bound on the moment scales with $N^{-\frac{1}{2}}$ and is proportional to an N-independent constant B_p^{MC} which usually grows exponentially with p [181].

An important property of Monte-Carlo methods is that the formula for computing the variance (1.5) reveals that the function f is required to be only square-integrable over X. This is a remarkably weak assumption which makes the Monte Carlo approaches suitable for a broad class of functions. Alternative integration methods, such as previously mentioned Newton-Cotes and Gaussian cubature rules, usually require more restrictive assumptions on the smoothness of the integrand.

A rather philosophical question with the Monte Carlo strategies lies in that—from the implementation point of view—it is usually not the case that standard computers are able to generate samples that are truly random but rather near random. This fact slightly undermines the validity of the theoretical results related to Monte Carlo methods as they embrace the true randomness [199]. However, empirical evidence demonstrates that the Monte Carlo techniques approximately behave according to the expectations delineated by the theoretical results, and they have proved to be immensely useful in a multitude of applications.

[3]Similarly to Definition 1.1, it can be shown that the estimator σ_N^2 is unbiased.

1.2 Importance Sampling

A fundamental issue with the basic Monte Carlo method is that a substantial amount of particles of the set $(X^i)_{i=1}^N$—drawn from a target distribution π defined on $\mathsf{X} \subseteq \mathbb{R}^{n_x}$—can be wasted in those parts of the space where the evaluations of a test function $f : \mathsf{X} \to \mathbb{R}$ contribute poorly to the integral approximation. A similar point of view is that one can be concerned with a certain measurable subset $A \subset \mathsf{X}$ where the function evaluations are relevant but the probability $\pi(A)$ is low, such as the rare-event analysis [27]. The time of computing such experiments can be unreasonably long due to waiting on a sufficient amount of particles being placed in A so that associated quantities converge to more precise estimates. Importance sampling is a generic approach which deals with these issues by focusing samples on the regions of X we deem important. This is accomplished by defining a user-selected importance or *proposal* distribution q which concentrates the samples towards suitable parts of X. Importance sampling is probably most widely used but also most difficult-to-tune technique for variance reduction of the basic Monte Carlo approach. The procedure is at the core of various more advanced, sampling-based, algorithms.

For a proposal distribution q satisfying $\pi \ll q$, one can suggest a simple extension of the target distribution π defined by

$$\pi^{IS}(dx) := w(x)q(dx), \tag{1.6}$$

where we introduce the *unnormalized* importance weight function $w(x) = \frac{\pi(dx)}{q(dx)}$. Given a set of random samples $(X^i)_{i=1}^n$ drawn from q, the importance sampling approximation of π can be formed by simply substituting the empirical form of the proposal distribution, $q^N(dx) = \frac{1}{N}\sum_{i=1}^N \delta_{X^i}(dx)$, in (1.6), that is,

$$\pi^{IS,N}(dx) = \frac{1}{N}\sum_{i=1}^{N} w(X^i)\delta_{X^i}(dx). \tag{1.7}$$

The approximation of the integral $\pi(f)$ of a test function $f : \mathsf{X} \to \mathbb{R}$ is then obtained by plugging (1.7) in (1.2), which results in

$$\pi^{IS,N}(f) = \frac{1}{N}\sum_{i=1}^{N} w(X^i)f(X^i). \tag{1.8}$$

One can notice that (1.7) and (1.8) are just reweighted versions of (1.1) and (1.3), respectively. The purpose of the importance weights is to compensate for the discrepancy between π and q. In other words, the individual Dirac-delta measures (or function evaluations) are weighted according to similarity between the distributions. The following theorem shows that the estimator (1.8) is strongly consistent.

Theorem 1.4. *For a sequence of IID random samples $(X^i)_{i=1}^N$ simulated from q such that $\pi \ll q$, and a measurable function $f : \mathsf{X} \to \mathbb{R}$ satisfying $\mathsf{E}[|f(X)|] < \infty$, the empirical expectation (1.8) converges almost surely to the exact one (1.2) as the number of samples N tends to infinity, thus,*

$$\pi^{IS,N}(f) \xrightarrow{a.s.} \pi(f),$$

for $N \longrightarrow \infty$.

Proof. See [20]. □

In the following definition, we present basic statistical properties of the importance sampling estimator (1.8).

Definition 1.2. *Assume a sequence of IID random samples $(X^i)_{i=1}^N$ drawn from q such that $\pi \ll q$ and a measurable function $f : \mathsf{X} \to \mathbb{R}$ for which $\mathsf{E}[|f(X)|] < \infty$ and $\mathsf{E}_q[f(X)^2 w(X)^2] < \infty$, then*

$$\mathsf{E}_q[\pi^{IS,N}(f)] = q(fw) = \pi(f), \tag{1.9}$$

$$\mathsf{V}_q[\pi^{IS,N}(f)] = \frac{1}{N}\Big(\mathsf{E}_q[f(X)^2 w(X)^2] - \pi(f)^2\Big) = \frac{1}{N}\mathsf{V}_q[f(X)w(X)]. \tag{1.10}$$

Definition 1.2 shows that the importance sampling preserves the statistical properties of the classic Monte Carlo approach. The first part (1.9) demonstrates that the estimator (1.8) is unbiased for any N, whereas the second part (1.10) says that we can decrease the variance arbitrarily low when increasing N. The requirement $\mathsf{E}[f(X)^2] < \infty$ is no longer enough to make the variance finite. In particular, an additional condition, $w(x) < \infty$ for all $x \in \mathsf{X}$, is required. An important observation in (1.10) is that only the first term in the middle part of (1.10) depends on q. Therefore, not only high values of the number of particles N but also a proper choice of the proposal distribution q are crucial for decreasing the variance. A simple application of Jensen's inequality reveals the lower bound of the first term in the middle part of (1.10) [3]

$$\mathsf{E}_q[f(X)^2 w(X)^2] \geq \mathsf{E}_q[|f(X)|w(X)]^2 = \mathsf{E}[|f(X)|]^2, \tag{1.11}$$

from which it follows that the lower bound is attained when the optimal proposal distribution satisfies

$$q^\star(dx) = \frac{|f(x)|\pi(dx)}{\int |f(x)|\pi(dx)}. \tag{1.12}$$

Let us now apply Jensen's inequality on the first term of $\mathsf{E}[f(X)]^2 - \pi(f)$ as

$$\mathsf{E}[f(X)]^2 - \pi(f) \geq \mathsf{E}[|f(X)|]^2 - \pi(f), \tag{1.13}$$

which corresponds to $N\mathsf{V}[\pi^{MC,N}(f)] \geq N\mathsf{V}_{q^\star}[\pi^{IS,N}(f)]$ and thus implies that the importance sampling is (almost surely) more efficient than the plain Monte Carlo approach. The problem with (1.12) is that we are not able to compute the denominator, which prevents us from achieving the optimal performance. On the one hand, this result tells us that we should choose the proposal distribution close to $|f(x)|\pi(dx)$. On the other hand, designing the proposal distribution with a specific function f makes a resulting estimation procedure less general. Therefore, the common requirement is to choose the proposal close to the situation where the variance of the importance weights is minimized, $q^\star = \pi$. In particular, the proposal distribution should be easy to sample from.

The next theorem states that the importance sampling estimator (1.8) is asymptotically Gaussian.

Theorem 1.5. *Let $(X^i)_{i=1}^N$ be a sequence of IID random samples drawn from q such that $\pi \ll q$, and let $f : \mathsf{X} \to \mathbb{R}$ be a measurable function satisfying $\mathsf{E}[|f(X)|] < \infty$ and $\mathsf{E}_q[f(X)^2 w(X)^2] < \infty$, then*

$$N^{\frac{1}{2}}[\pi^{IS,N}(f) - \pi(f)] \xrightarrow{d} \mathcal{N}(0, \mathsf{V}_q[f(X)w(X)]),$$

for $N \longrightarrow \infty$.

Proof. See [117]. □

Another important characteristic of the importance sampling is that there exists the $\mathcal{L}^p$ bound on pth absolute central moment of the estimator (1.8), as presented in the next theorem.

Theorem 1.6. *Consider a sequence $(X^i)_{i=1}^N$ of IID random samples drawn from q fulfilling $\pi \ll q$, and a measurable function $f : \mathsf{X} \to \mathbb{R}$ for which $\mathsf{E}[|f(X)w(X)|^p] < \infty$, where $p \geq 1$, then there exists $B_p^{IS} < \infty$ such that*

$$\mathsf{E}[|\pi^{IS,N}(f) - \pi(f)|^p]^{\frac{1}{p}} \leq N^{-\frac{1}{2}} B_p^{IS}.$$

Proof. See [40]. □

1.3 Self-Normalized Importance Sampling

In many applications, we need to draw a set of IID random samples $(X^i)_{i=1}^N$ from a target probability distribution

$$\pi(dx) = \frac{\gamma(dx)}{\gamma(1)} \tag{1.14}$$

defined on $\mathsf{X} \subseteq \mathbb{R}^{n_x}$. It is mostly the case that $\gamma(dx)$ can be evaluated point-wise but $\gamma(1)$ is intractable. Therefore, the standard importance sampling cannot be used directly to approximate (1.14). The self-normalized importance sampling is a technique which addresses this problem by applying the standard importance sampling to both the numerator and denominator of (1.14).

Let us consider we have the standard importance sampling approximation of the distribution γ given by

$$\gamma^{IS,N}(dx) = \frac{1}{N}\sum_{i=1}^{N} v(X^i)\delta_{X^i}(dx), \tag{1.15}$$

where $v(x) = \frac{\gamma(dx)}{q(dx)}$ is the unnormalized importance weight function, which implies $v(x) = w(x)\gamma(1)$ with w being defined in (1.6). Then, the self-normalized importance sampling approximation of the target distribution (1.14) is obtained by substituting (1.15) into the numerator and denominator of (1.14), providing us with

$$\pi^{SNIS,N}(dx) = \frac{\gamma^{IS,N}(dx)}{\gamma^{IS,N}(1)} = \sum_{i=1}^{N} W^i\delta_{X^i}(dx), \tag{1.16}$$

where

$$W^i := \frac{v(X^i)}{\sum_{j=1}^{N} v(X^j)} \tag{1.17}$$

is the *normalized* importance weight function. The representation $(X^i, W^i)_{i=1}^N$ of (1.16) is referred to as the *weighted particle system.* The integral $\pi(f)$ of a test function $f : \mathsf{X} \to \mathbb{R}$ is then approximated by inserting (1.16) in (1.2),

$$\pi^{SNIS,N}(f) = \sum_{i=1}^{N} W^i f(X^i). \tag{1.18}$$

It is noted that the form of (1.17) allows us to use proposal distributions which are known only up to a constant factor. As the unnormalized importance weights do not sum up to one, the approximation $\pi^{IS,N}$ is not an empirical probability measure, even if π is a probability measure. This issue is resolved here as the normalized importance weights (1.17) do sum up to one, and the approximation $\pi^{SNIS,N}$ is thus an empirical probability measure. The strong consistency of the estimator (1.18) is presented in the following theorem.

Theorem 1.7. *Let $(X^i)_{i=1}^N$ be a sequence of IID random samples drawn from q such that $\pi \ll q$, and let $f : \mathsf{X} \to \mathbb{R}$ be a measurable function fulfilling $\mathsf{E}[|f(X)|] < \infty$, then the empirical expectation (1.18) converges almost surely to the exact one (1.2) as the number of samples N increases, that is,*

$$\pi^{SNIS,N}(f) \xrightarrow{a.s.} \pi(f),$$

for $N \longrightarrow \infty$.

Proof. See [30]. $\square$

We note here that the proof of Theorem 1.7 relines on $\gamma^{IS,N}(1) \xrightarrow{a.s.} \gamma(1)$, demonstrating that $\gamma^{IS,N}(1)$ is a strongly consistent estimator of $\gamma(1)$, a feature which is immensely useful in Bayesian inference. An extension of this principle is important in sequential Monte Carlo methods.

The basic statistical properties of the self-normalized importance sampling estimator (1.18) are shown in the next definition.

Definition 1.3. *Let $(X^i)_{i=1}^N$ be a sequence of IID random samples drawn from q which satisfies $\pi \ll q$, let $f : \mathsf{X} \to \mathbb{R}$ be a measurable function such that $\mathsf{E}[|f(X)|] < \infty$ and $\mathsf{E}_q[f(X)^2 v(X)^2] < \infty$, and let $\mathsf{E}_q[v(X)^2] < \infty$, then*

$$\begin{aligned} \mathsf{E}_q[\pi^{SNIS,N}(f)] &= \pi(f) + \frac{1}{N}\Big(\pi(f)\mathsf{V}_q[w(X)] \\ &\qquad - \mathsf{C}_q[w(X)f(X), w(X)]\Big) + \mathcal{O}(N^{-2}), \qquad (1.19) \\ \mathsf{V}_q[\pi^{SNIS,N}(f)] &= \frac{1}{N}\mathsf{V}_q[w(X)f(X)] - \frac{1}{N}\Big(2\pi(f)\mathsf{C}_q[w(X)f(X), w(X)] \\ &\qquad - \pi(f)^2\mathsf{V}_q[w(X)]\Big) + \mathcal{O}(N^{-2}), \qquad (1.20) \end{aligned}$$

with C_q denoting the covariance with respect to q.

The estimator (1.16) is given by a ratio of quantities that are computed with the same set of random samples, making the numerator and denominator dependent. Therefore, to obtain Definition 1.3, we need to apply the delta method. Definition 1.3 shows that the self-normalized importance sampling estimator (1.16) is biased for a finite N. The dominating term of the bias vanishes linearly with increasing the number of samples N, thus allowing to control its size. The variance decreases with linear dependence, as in the case of the basic Monte Carlo and importance sampling approaches. However, the structure of (1.20) reveals an important difference. Specifically, when the correlation between $w(X)f(X)$ and $w(X)$ grows, the variance (1.20) can outperform the variance of the basic Monte Carlo (1.5) and plain importance sampling (1.10) estimators [135].

The asymptotic Gaussianity of the self-normalized importance sampling estimator (1.18) is shown below.

Theorem 1.8. *Consider a sequence of IID random samples $(X^i)_{i=1}^N$ drawn from q such that $\pi \ll q$, a measurable function $f : \mathsf{X} \to \mathbb{R}$ for which $\mathsf{E}[|f(X)|] < \infty$ and $\mathsf{E}_q[f(X)^2 v(X)^2] < \infty$, and $\mathsf{E}_q[v(X)^2] < \infty$, then*

$$N^{\frac{1}{2}}[\pi^{SNIS,N}(f) - \pi(f)] \xrightarrow{d} \mathcal{N}(0, \sigma^2(f)),$$

for $N \longrightarrow \infty$. Here,

$$\sigma^2(f) := \mathsf{E}\Big[[f(X) - \pi(f)]^2 w(X)\Big].$$

Proof. The proof follows from the multivariate central limit theorem [210] and the delta method [20]. $\square$

Similarly as before, the self-normalized importance sampling estimator (1.18) allows us to find the associated $\mathcal{L}^p$ error bound, as presented in the following theorem.

Theorem 1.9. *For a sequence $(X^i)_{i=1}^N$ of IID random samples drawn from q such that $\pi \ll q$ and a measurable function $f : \mathsf{X} \to \mathbb{R}$ with $\mathsf{E}[|f(X)v(X)|^p] < \infty$, where $p \geq 1$, we have $B_p^{SNIS} < \infty$ satisfying*

$$\mathsf{E}[|\pi^{SNIS,N}(f) - \pi(f)|^p]^{\frac{1}{p}} \leq N^{-\frac{1}{2}} B_p^{SNIS}.$$

Proof. See [30]. $\square$

1.3.1 Effective Sample Size

The performance of the self-normalized importance sampling is significantly affected by our choice of the proposal distribution q. We discussed previously that one should choose q as close as possible to the target distribution π. However, the question is how to assess the difference between π and q, especially when π can be evaluated only up to the constant factor. A possible approach how to deal with this issue is to apply the relative numerical efficiency (RNE, [73]). This quantity is defined as the ratio of the variance of the self-normalized importance sampling estimator when choosing the proposal distribution as the target distribution, $q = \pi$, to the variance of this estimator with an arbitrary distribution q, that is,

$$\text{RNE} := \frac{\mathsf{V}[\pi^{MC,M}(f)]}{\mathsf{V}_q[\pi^{SNIS,N}(f)]}. \tag{1.21}$$

It can be shown—see, e.g. [119]—that the variance of the self-normalized importance sampling estimator can be approximated by

$$\mathsf{V}_q[\pi^{SNIS,N}(f)] \approx \mathsf{V}[\pi^{MC,N}(f)](1 + \mathsf{V}_q[w(X)]). \tag{1.22}$$

Consequently, substituting (1.5) in (1.21) and (1.22) leads to

$$\text{RNE} \approx \frac{N}{M} \frac{1}{1 + \mathsf{V}_q[w(X)]}.$$

The situation RNE $= 1$ indicates that the performance of the standard Monte Carlo method and self-normalized importance sampling is equal. The value of M for which RNE $= 1$ is therefore of particular interest and defines the effective sample size

$$N_{\text{ess}} := \frac{N}{1 + \mathsf{V}_q[w(X)]}, \tag{1.23}$$

which is the number of particles required by the plain Monte Carlo sampling in order to have approximately the same performance as the self-normalized importance sampling.

The result (1.23) is quite universal in the sense that we do not need to use the function evaluations but only the importance weights. As discussed in [165], the effective sample size involving the function evaluations may be more accurate, as the performance of the importance sampling methods also depends on a concrete form of f. However, specific assumptions about f make algorithms based on this principle less generic. The effective sample size ranges in $(0, N]$. The bigger the difference between π and q, the higher the variance term $\mathsf{V}_q[w(X)]$ and the lower the effective sample size. The choice $q = \pi$ results in $N_{\text{ess}} = N$, thus matching the performance of the plain Monte Carlo method.

Although the form of (1.23) is suitable for explaining the principle of the effective sample size, it is not very convenient for practical computations. Let us therefore express (1.23) in the alternative form

$$N_{\text{ess}} = N\frac{\gamma(1)}{\pi(v)}, \tag{1.24}$$

which can simply be approximated as

$$\widehat{N}_{\text{ess}} := N\frac{\gamma^{IN,N}(1)}{\pi^{IS,N}(v)} = \frac{1}{\sum_{i=1}^{N}(W^i)^2}. \tag{1.25}$$

When using the effective sample size, we need to be aware that (1.22) is derived based on neglecting higher-order terms of the Taylor series. If the influence of these terms is substantial, then (1.23) becomes less reliable. The effective sample size based on discrepancy measures [145] offers an alternative approach.

1.4 Sequential Importance Sampling

A commonly encountered situation in various statistical inference objectives is the requirement of approximating a joint target probability distribution

$$\pi_t(dx_{1:t}) = \frac{\gamma_t(dx_{1:t})}{\gamma_t(1)} \tag{1.26}$$

defined on X^t with $\mathsf{X} \subseteq \mathbb{R}^{n_x}$. We assume that $\gamma_t(dx_{1:t})$ is point-wise known but $\gamma_t(1)$ is unknown. We can of course approximate this distribution by the self-normalized importance sampling. However, the problem is that the dimension of X^t grows with t, and so does the computational complexity of the importance sampling algorithm. Moreover, applying the importance sampling at a given iteration t would require us to throw away all computations associated with π_{t-1} and start those related to π_t

from the beginning. The sequential importance sampling [88] addresses these issues by reusing the approximations across the consecutive iterations, thus making the computational complexity fixed.

The empirical approximation of (1.26) is constructed in the same way as with the plain self-normalized importance sampling—except it is defined on the extended space X^t—and is given by

$$\pi_t^{SIS,N}(dx_{1:t}) = \sum_{i=1}^{N} W_t^i \delta_{X_{1:t}^i}(dx_{1:t}),$$

where

$$W_t^i := \frac{v_t(X_{1:t}^i)}{\sum_{j=1}^{N} v_t(X_{1:t}^j)}. \tag{1.27}$$

The associated approximation of the integral $\pi(f)$ of a test function $f : \mathsf{X}^t \to \mathbb{R}$ is

$$\pi_t^{SIS,N}(f) = \sum_{i=1}^{N} W_t^i f(X_{1:t}^i). \tag{1.28}$$

The basic idea of the sequential importance sampling is to find a recursive formulae for time-evolution of the proposal distribution and the unnormalized importance weights, which can simply be accomplished by

$$q_t(dx_{1:t}) = m_t(dx_t|x_{1:t-1})q_{t-1}(dx_{1:t-1}), \tag{1.29}$$

with $q_1(dx_1) = m_1(dx_1)$, and

$$v_t(x_{1:t}) = \frac{\gamma_t(dx_{1:t})}{q_t(dx_{1:t})} = \frac{\gamma_{t-1}(dx_{1:t-1})}{q_{t-1}(dx_{1:t-1})}\alpha_t(x_{1:t}) = v_{t-1}(x_{1:t-1})\alpha_t(x_{1:t}), \tag{1.30}$$

where

$$\alpha_t(x_{1:t}) := \frac{\gamma_t(dx_{1:t})}{\gamma_{t-1}(dx_{1:t-1})m_t(x_t|x_{1:t-1})} \tag{1.31}$$

defines the *incremental* importance weight function. Let us consider that we have the weighted particle system from the previous iteration, $(X_{1:t-1}^i, W_{t-1}^i)_{i=1}^N$. The *recursive step* of the sequential importance sampling can be split into two parts. In the first part, the recursion for evolving the proposal distribution (1.29) suggests to (i) keep the previous particle trajectory $X_{1:t-1}^i$ intact, (ii) sample the current particle according to

$$X_t^i \sim m_t(\cdot|X_{1:t-1}^i), \tag{1.32}$$

and, (iii) form an updated particle trajectory

$$X_{1:t}^i := (X_t^i, X_{1:t-1}^i).$$

In the second part, we (i) compute the incremental weights $\alpha_t(X_{1:t}^i)$, (ii) apply them together with W_{t-1}^i in the recursive formula for computing the unnormalized

Algorithm 1 Sequential importance sampling (SIS)

```
A. Initial step: ($t = 1$)
    1. Sample $X_1^i \sim m_1(\cdot)$.
    2. Compute $W_1^i \propto v_1(X_1^i)$.
B. Recursive step: ($t = 2, \dots, T$)
    1. Sample $X_t^i \sim m_t(\cdot|X_{1:t-1}^i)$ and set $X_{1:t}^i := (X_t^i, X_{1:t-1}^i)$.
    2. Compute $W_t^i \propto v_t(X_{1:t}^i)$ according to (1.30).
```

importance weights (1.30), and (iii) use (1.27) to compute the normalized importance weights W_t^i. Note that W_{t-1}^i can be used in place of $v_{t-1}(X_{1:t-1}^i)$ as the normalizing factor of W_{t-1}^i is canceled out in (1.27). These operations produce the current particle system $(X_{1:t}^i, W_t^i)_{i=1}^N$, thus concluding the recursive step.

The *initial step* generates the particle system $(X_1^i, W_1^i)_{i=1}^N$ based on the basic self-normalized importance sampling. The whole procedure can now be summarized in Algorithm 1, where we follow the convention that all i-dependent operations are performed for $i = 1, \dots, N$. Note this procedure generates a set of *IID* sample trajectories $(X_{1:t}^i)_{i=1}^N$.

The sequential importance sampling is equivalent to the self-normalized importance sampling on the extended space X^t, which makes all the theoretical results presented in Section 1.3 applicable here as well. However, the recursive nature of sequential importance sampling can reveal some interesting characteristics of this approach. The steps B1 and B2 are often referred to as the *propagation* (or mutation [37]) and *weighting* (or correction) steps, respectively. In the propagation step, we investigate the properties of the estimator $q_t^N(f)$, which can be used to estimate $\pi_t(f)$ before proceeding into the weighting step. In the weighting step, we are—similarly as with the plain self-normalized importance sampling—interested in the properties of the estimator $\pi_t^{SIS,N}(f)$. The following definition analyzes these two stages.

Definition 1.4. *Consider a collection of IID random samples $(X_{1:t}^i)_{i=1}^N$ drawn from q_t such that $\pi_t \ll q_t$, and a measurable function $f : \mathsf{X}^t \to \mathbb{R}$ satisfying $\mathsf{E}[|f(X_{1:t})|] < \infty$, $\mathsf{E}_{q_t}[f(X_{1:t})^2 v_t(X_{1:t})^2] < \infty$, and $\mathsf{E}_{q_t}[|f(X_{1:t})|] < \infty$, and let $\mathsf{E}_{q_t}[v_t(X_{1:t})^2] < \infty$, then, for the propagation step (B1), it holds that*

$$\mathsf{E}_{q_t}[q_t^N(f)] = q_t(f), \tag{1.33}$$

$$\mathsf{V}_{q_t}[q_t^N(f)] = \frac{1}{N}\Big(\mathsf{E}_{q_{t-1}}\Big[\mathsf{V}_{m_t}[f(X_{1:t})|X_{1:t-1}]\Big] + \mathsf{V}_{q_{t-1}}\Big[\mathsf{E}_{m_t}[f(X_{1:t})|X_{1:t-1}]\Big]\Big), \tag{1.34}$$

and, for the weighting step (B2), we have

$$\mathsf{E}_{q_t}[\pi^{SIS,N}(f)] := \pi_t(f) + \frac{1}{N}\mathsf{E}[(f(X_{1:t}) - \pi(f))w_t(X_{1:t})], \tag{1.35}$$

$$\mathsf{V}_{q_t}[\pi^{SIS,N}(f)] := \frac{1}{N}\mathsf{V}_{q_t}[(f(X_{1:t}) - \pi(f))w_t(X_{1:t})], \tag{1.36}$$

where the equality by definition follows from neglecting the higher order terms, as they vanish quickly for increasing N.

Definition 1.4 demonstrates that the estimator $q_t^N(f)$ is unbiased for any N (as expected). However, as presented in (1.34)—which follows from applying the law of total variance—its variance can only increase by executing the propagation steps. The formulae (1.35) and (1.36) are simply rearranged versions of (1.19) and (1.20), respectively. The appearance of w_t in these terms is important for further analysis. As mentioned in Section 1.3, the importance weight function w_t satisfies $w_t \propto v_t$. The unnormalized importance weight function is a density function by definition, $v_t : \mathsf{X}^t \to \mathbb{R}_+$. Similarly as before—based on the law of total variance—we can find the recursive formula for computing its variance

$$\mathsf{V}[v_t(X_{1:t})] = \mathsf{V}[v_{t-1}(X_{1:t-1})] + \mathsf{E}\Big[v_t^2(X_{1:t-1})\mathsf{V}[v_t(X_t|X_{1:t-1})|X_{1:t-1}]\Big]. \tag{1.37}$$

As both terms on the r.h.s. of (1.37) are always positive, the variance of the unnormalized importance weights v_t can only increase over the iterations. This phenomenon is commonly referred to as *weight degeneracy*. We can see from (1.35) and (1.36) that $w_t \propto v_t$ influences the bias and variance and thus the quality of the estimator (1.28). We should also consider these comments in the context of (1.19) and (1.20), which shows that the quality of the estimator can sometimes improve. However, experimental evidence demonstrates that this happens only rarely (during the initial iterations of the algorithm in most cases).

An example of the weight degeneracy is presented in the top row of Fig. 1.1 where we see that the weight W_t^2 converges to one and the remaining weights to zero. This effect can be measured by the effective sample size discussed in Section 1.3.1, which decreases with W_t^2 approaching one.

The recursive formula (1.37) renders the need for variance reduction techniques, although, in the present case, this can only postpone the inflation. As suggested by (1.36), the natural mechanism to decrease the variance is to have the number of particles multiple times greater compared to the current iteration, $N \gg t$, which is unrealistic for long sequences. Alternatively, the impact of (1.37) can be counteracted by the choice of the proposal distribution (1.32). The optimal proposal distribution which minimizes the variance of $v_t(x_{1:t})$ is defined by

$$m_t^\star(dx_t|x_{1:t-1}) = \gamma_t(dx_t|x_{1:t-1}). \tag{1.38}$$

Inserting this into (1.31) provides us with

$$\alpha_t^\star(x_{1:t}) = \frac{\gamma_t(dx_{1:t-1})}{\gamma_{t-1}(dx_{1:t-1})},$$

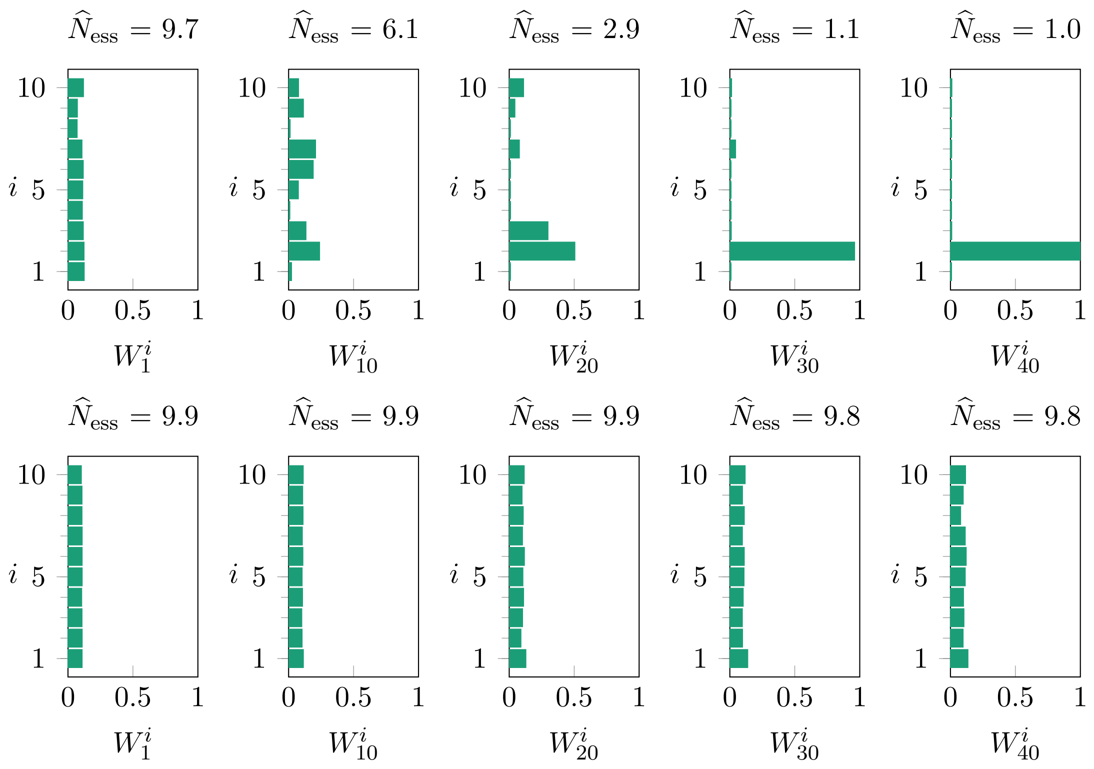

Fig. 1.1: An example of weight degeneracy. We consider $\gamma_t(x_{1:t}) := \gamma(x_1) \prod_{i=1}^{t} \gamma(x_i|x_{i-1})$ where $\gamma(x_1) = \mathcal{N}(x_1; 0, 1)$ and $\gamma(x_i|x_{i-1}) = \mathcal{N}(x_i; 0.8x_{i-1}, 1)$. The normalized weights $W_t^{1:10}$ for $t = (1, 10, 20, 30, 40)$ computed by Algorithm 1 with the proposal density $m(x_t|x_{t-1}) := \mathcal{N}(x_t; 0.8x_{t-1}, 2)$ (top) and $m(x_t|x_{t-1}) := \mathcal{N}(x_t; 0.8x_{t-1}, 1.05)$ (bottom).

whose variance is zero conditionally on $X_{1:t-1}$, making the variance of (1.30) zero as well [56]. However, (1.38) cannot be computed under a closed-form solution, expect simple cases (as it contains the integral with respect to X_t). We show one of such simple examples in the bottom row of Fig. 1.1. We can observe that there is no weight converging to one and the effective sample size decreases at a slower rate. The importance distribution is near-optimal in this example $m(\cdot|x_{t-1}) \approx \gamma(\cdot|x_{t-1})$.

1.5 Resampling

Resampling is a procedure which computes an approximation $\pi^{R,M}$ of a target distribution π defined on $\mathsf{X} \subseteq \mathbb{R}^{n_x}$ by using an already existing approximation π^N of the same target distribution π. The method is most often applied after the self-normalized importance sampling, $\pi^N := \pi^{SNIS,N}$, in order to duplicate the particles with high weights and discard the ones with low weights. In such a case, it is also referred to as sampling importance resampling [186]. The resampling operation is mostly applied in the sequential context to improve performance of an associated algorithm across multiple iterations.

Let us consider we have the weighted particle system $(X^i, W^i)_{i=1}^N$ representing the self-normalized importance sampling approximation $\pi^{SNIS,N}$ of π. The resampling procedure uses the basic Monte Carlo approach discussed in Section 1.1 to obtain the uniformly-weighted particle system $(\bar{X}^i, M^{-1})_{i=1}^M$ representing $\pi^{R,M}$. The resampling can be seen as a process where parent particles (X^i) can have multiple or no offspring particles $(\bar{X}^i)$ [4]. The method proceeds in two steps. First, we sample a set of ancestor indices $\mathbf{A} \coloneqq (A^1, \ldots, A^M)$ from a discrete probability distribution defined on $(1, \ldots, N)^M$,

$$\mathbf{A} \sim r(\cdot|\mathbf{W}) \coloneqq \prod_{i=1}^{M} \mathcal{F}(A^i|\mathbf{W}), \tag{1.39}$$

which is parameterized by the set of normalized importance weights $\mathbf{W} \coloneqq (W^1, \ldots, W^N) \in [0,1]^N$. Second, we use the ancestor indices to assign parent particles to offspring particles as $\bar{X}^i \coloneqq X^{A^i}$ for $i = 1, \ldots, M$. However, practical implementation of the first step often consists of obtaining a set of ancestor counts $\mathbf{O} \coloneqq (O^1, \ldots, O^N)$, where $O^i = \sum_{j=1}^M \mathbb{1}(A^j = i)$ is the number of offspring of ith particle, from a discrete probability distribution,

$$\mathbf{O} \sim s(\cdot|\mathbf{W}), \tag{1.40}$$

defined on $(1, \ldots, M)^N$, and applying a deterministic procedure to convert the ancestor counts O^i to the ancestor indices A^i. An important property of (1.40) lies in that the samples $\mathbf{O}$ should satisfy *the unbiasedness condition*

$$\mathsf{E}[O^i|\mathbf{W}] = MW^i, \tag{1.41}$$

for $i = 1, \ldots, N$. After finishing the procedure, we can formulate an associated estimator of a test function $f : \mathsf{X} \to \mathbb{R}$ as

$$\pi^{R,M}(f) = \frac{1}{M}\sum_{i=1}^{M} f(\bar{X}^i) = \frac{1}{M}\sum_{i=1}^{N} O^i f(X^i). \tag{1.42}$$

In the sequential context, the ancestor indices allow us to construct genealogy of the particle evolution, as will be discussed in Section 1.6. To discuss some of the basic properties of the resampling procedure, let us present the following definition.

Definition 1.5. *Let the assumptions of Definition 1.3 be met and let* $\mathbf{O}$ *satisfy (1.41), then*

$$\mathsf{E}[\pi^{R,M}(f)] = \mathsf{E}_q[\pi^{SNIS,N}(f)], \tag{1.43}$$

$$\mathsf{V}[\pi^{R,M}(f)] = \mathsf{V}_q[\pi^{SNIS,N}(f)] + \mathsf{E}[\{\pi^{R,M}(f) - \pi^{SNIS,N}(f)\}^2], \tag{1.44}$$

where the expectation and variance are taken w.r.t. the randomness in $\pi^{R,M}(f)$.

Multinomial [84]	Sample $\bar{U}^{1:N} \sim U[0,1)^N$ and set $U^{1:N} := \bar{U}^{1:N}$;
Stratified [114]	Sample $\bar{U}^{1:N} \sim U[0,1)^N$ and set $U^i := \frac{(i-1)+\bar{U}^i}{N}$ for $i = 1, \ldots, N$;
Systematic [32]	Sample $\bar{U} \sim U[0,1)$ and set $U^i := \frac{(i-1)+\bar{U}}{N}$ for $i = 1, \ldots, N$;
	then, find O^i as the number of times $U^i \in \big(\sum_{j=1}^{i-1} W^i, \sum_{j=1}^{i} W^i\big]$ for $i = 1, \ldots, N$.
Residual [219]	For $i = 1, \ldots, N$, set $O_a^i := \lfloor MW^i \rfloor$, use modified weights $\bar{W}^i \propto MW^i - O_a^i$ to obtain O_b^i for the remaining $M - \sum_{i=1}^N O_a^i$ particles with one of the above techniques, and set $O^i := O_a^i + O_b^i$.

Tab. 1.1: Popular resampling procedures. Here, $U[a,b)^M$ is uniform distribution on the M-fold half-open interval $[a,b)^M$ and $\lfloor x \rfloor$ is the largest integer smaller than or equal to x.

The first result (1.43) reveals that—when (1.41) holds—the resampling introduces no additional bias compared to the self-normalized importance sampling estimator (1.18). The second result (1.44) shows that the resampling procedure can only increase the variance after the self-normalized importance sampling step. Therefore, an estimate of $\pi(f)$ should be computed before the resampling. Moreover, the result (1.44) motivates us to perform sampling in (1.40) with variance reduction techniques and/or only when the effective sample size—discussed in Section 1.3.1—is not large enough. The variance can be affected by changing the mechanism of generating random numbers in the Monte Carlo sampling. We present the most popular resampling schemes in Tab. 1.1.

All these approaches satisfy (1.41) and can be implemented with $\mathcal{O}(N)$ computational complexity. For a detailed introduction to the principles underlying these methods and an experimental comparison, see [90]. The multinomial, stratified, and residual resampling techniques generate particles $(\bar{X}^i)_{i=1}^M$ that are conditionally IID given $(X^i)_{i=1}^N$ and provide asymptotic variance which tends to zero for $M \to \infty$. However, a theoretical comparison presented in [52] shows that this is not the case for the systematic resampling. Additionally, based on comparing the conditional variances of the considered resampling schemes, it is proved in [52] that the residual and stratified resampling outperform the multinomial one. A recent review of resampling strategies beyond those presented in Tab. 1.1 can be found in [126]. The resampling schemes are generally hard to parallelize as we are required to perform summation over the importance weights. Recently, two resampling schemes—referred to as Metropolis and rejection resampling—that do not involve such a requirement, were proposed in [157].

The variance of the normalized importance weights is non-zero before the resampling step. The application of the plain Monte Carlo approach to sample from $\pi^{SNIS,N}$ facilitates construction of the *uniformly-weighted* particle system. This is a key feature as it implies that the variance of the importance weights becomes zero.

Algorithm 2 The sequential Monte Carlo (SMC) algorithm

```
A. Initial step: $(t = 1)$
    1. Sample $X_1^i \sim m_1(\cdot)$.
    2. Compute $W_1^i \propto v_1(X_1^i)$.
B. Recursive step: $(t = 2, \dots, T)$
    1. Sample $A_{t-1}^i \sim \mathcal{F}(\cdot|\mathbf{W}_{t-1})$.
    2. Sample $X_t^i \sim m_t(\cdot|X_{1:t-1}^{A_{t-1}^i})$ and set $X_{1:t}^i := (X_t^i, X_{1:t-1}^{A_{t-1}^i})$.
    3. Compute $W_t^i \propto v_t(X_{1:t}^i)$ according to (1.49).
```

As will be presented in Section 1.6, resampling is the solution to weight degeneracy. However, it introduces a different problem, which lies in that resampling generally decreases the diversity of the original particle system, as we make multiple copies of the particles with high weights.

1.6 Sequential Monte Carlo

Sequential Monte Carlo (SMC) methodology [56, 47] is a general tool which unifies various algorithms for approximating a sequence of target distributions $(\pi_t)_{t=1}^T$ under a single framework, where π_t is defined on X^t and known only up to the normalizing factor as in (1.26). The sequential Monte Carlo methods are also referred to as *sequential importance (sampling and) resampling*. The resampling is of key importance here, as it allows us to deal with the weight degeneracy problem of the plain sequential importance sampling.

The SMC approximation of π_t and the integral $\pi_t(f)$ of a test function $f : \mathsf{X}^t \to \mathbb{R}$ are obtained in the same way as with the self-normalized importance sampling and are given by

$$\pi_t^N(dx_{1:t}) = \sum_{i=1}^N W_t^i \delta_{X_{1:t}^i}(dx_{1:t}), \tag{1.45}$$

and

$$\pi_t^N(f) = \sum_{i=1}^N W_t^i f(X_{1:t}^i). \tag{1.46}$$

respectively, where W_t^i is given by (1.27). Let us remind that (1.45) is fully determined by the weighted particle system $(X_{1:t}^i, W_t^i)_{i=1}^N$, where $(X_{1:t}^i)$ are termed particle trajectories that represent a hypothetical evolution of the true trajectory $x_{1:t}$, while the weights (W_t^i) provide the assessment of how the corresponding particle trajectories contribute to the resulting approximation.

The fundamental principles of the SMC methodology are rooted in the sequential importance sampling and resampling discussed in Section 1.4 and Section 1.5, respectively. Let us consider we have the weighted particle system computed at the preceding iteration, $(X_{1:t-1}^i, W_{t-1}^i)_{i=1}^N$. The *recursive step* of an SMC method can be divided into three parts. The first part performs the resampling, which simply generates a set of ancestor indices $(A_{t-1}^i)_{i=1}^N$ according to

$$A_{t-1}^i \sim \mathcal{F}(\cdot|\mathbf{W}_{t-1}), \qquad i = 1, \ldots, N.$$

The ancestor indices $(A_{t-1}^i)_{i=1}^N$ are then used in the sequential importance sampling, which is given by the remaining two parts. The second part consists of simulating the particles $(X_t^i)_{i=1}^N$ from the proposal distribution,

$$X_t^i \sim m_t(\cdot|X_{1:t-1}^{A_{t-1}^i}), \tag{1.47}$$

and extending the previous trajectories based on

$$X_{1:t}^i := (X_{1:t-1}^{A_{t-1}^i}, X_t^i). \tag{1.48}$$

Here, the index A_{t-1}^i is used to assign the parent trajectory to the offspring particle X_t^i in (1.47) and to extend the parent trajectory to the offspring trajectory $X_{1:t}^i$ in (1.48). The third part computes—based on (1.27)—the normalized importance weights W_t^i for $i = 1, \ldots, N$, where the unnormalized importance weight function now satisfies

$$v_t(x_{1:t}) := \frac{\gamma_t(dx_{1:t})}{\gamma_{t-1}(dx_{1:t-1})m_t(x_t|x_{1:t-1})}. \tag{1.49}$$

Note we do not use the recursive formula for computing the weights as in (1.30) due to the fact that we perform resampling at each iteration. Such an SMC setting is often referred to as *sequential importance resampling.* The above sequence of operations leads to the current particle system $(X_{1:t}^i, W_t^i)_{i=1}^N$, which closes the recursive step.

The *initial step* is simply formed by the standard self-normalized importance sampling where we first sample the particles $(X_1^i)_{i=1}^N$ from the initial proposal distribution $X_1^i \sim m_1(\cdot)$ and then compute the normalized importance weights $W_1^i \propto v_1(X_1^i)$ for $i = 1, \ldots, N$, where $v_1(x_1) = \gamma_1(x_1)/m_1(x_1)$. We summarize this SMC procedure in Algorithm 2 which can generally be seen as a procedure for propagating π_t^N in time. The computational complexity of Algorithm 2 scales with $\mathcal{O}(TN)$ operations, where T is the total amount of iterations. Note that we implicitly assume the multinomial resampling scheme in line B1. Naturally, we can replace this line by the alternative resampling strategies discussed in Section 1.5.

To demonstrate how the resampling addresses the weight degeneracy problem, we continue with the example in Fig. 1.1 and present normalized importance weights computed by Algorithm 2 in the top row of Fig. 1.2. We see that there is no weight converging to one, demonstrating that the resampling counteracts the accumulation of approximation error over time—the error caused by the weight degeneracy.

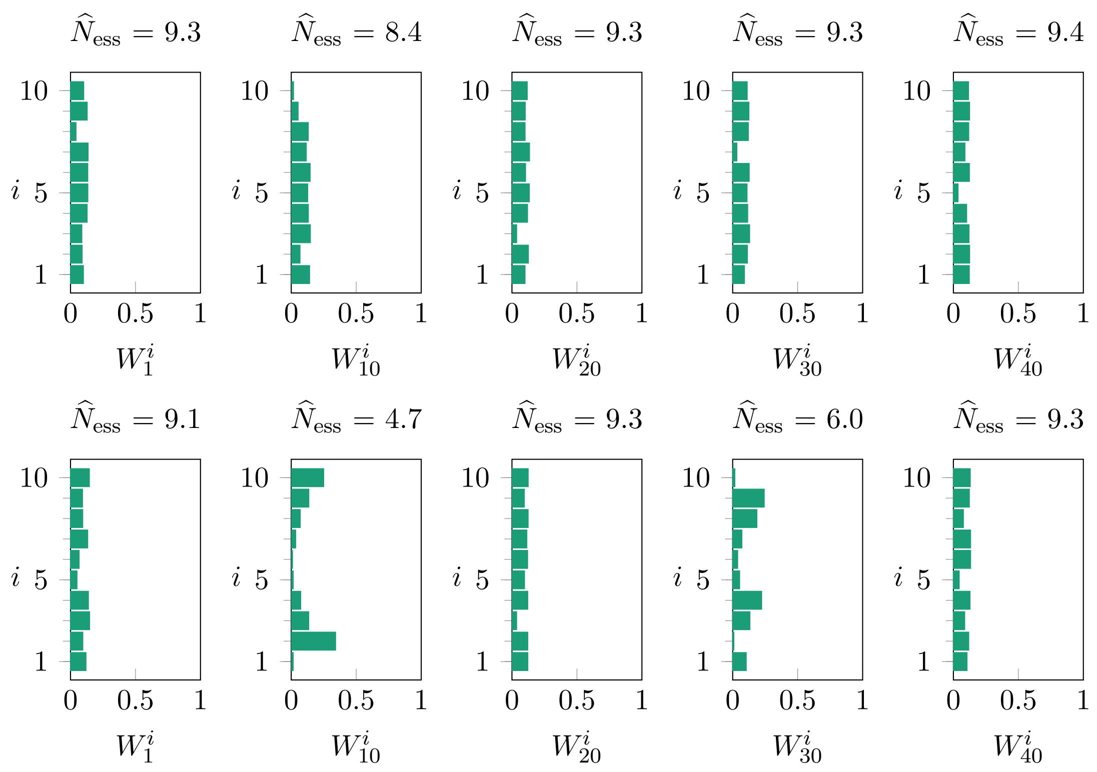

Fig. 1.2: An example of weight degeneracy. We consider $\gamma_t(x_{1:t}) \coloneqq \gamma(x_1)\prod_{i=1}^{t}\gamma(x_i|x_{i-1})$ where $\gamma(x_1) = \mathcal{N}(x_1;0,1)$ and $\gamma(x_i|x_{i-1}) = \mathcal{N}(x_i;0.8x_{i-1},1)$. The normalized weights $W_t^{1:10}$ for $t = (1,10,20,30,40)$ computed by the sequential importance resampling—Algorithm 2—(top) and sequential importance sampling and resampling (bottom), with the proposal density being $m(x_t|x_{t-1}) \coloneqq \mathcal{N}(x_t;0.8x_{t-1},2)$ in both these cases.

Nevertheless, the resampling procedure introduces a different problem, which we now describe. The main purpose of the ancestor indices $\mathbf{A}_{1:t-1} \coloneqq A_{1:t-1}^{1:N}$ is to allow us to trace the genealogy of the particle trajectories. By keeping the record of these indices, we can retrospectively determine the index B_n^i of the ith ancestor particle at some past time n in the history of the associated particle trajectory $X_{1:t}^i$ according to

$$B_n^i \coloneqq A_n^{B_{n+1}^i}, \tag{1.50}$$

for $n = t-1,\dots,1$, starting with $B_t^i = i$. Based on this backward recursion, one can assemble ith full particle trajectory as $X_{1:t}^i \coloneqq X_{1:t}^{B_{1:t}^i} = (X_1^{B_1^i},\dots,X_t^{B_t^i})$. We demonstrate particle trajectories that are generated in this way on an example presented in Fig. 1.3. We observe that, as the iterations increase, the trajectories are progressively more similar in the initial time range. Eventually, in the final frame, we can even notice that the trajectories are the same for the first 14 iterations. What we see in the present example is a direct consequence of the resampling operation, which inevitably causes the loss of the particle diversity by eliminating the trajectories with the low weights and duplicating those with the high weights. This phenomenon is

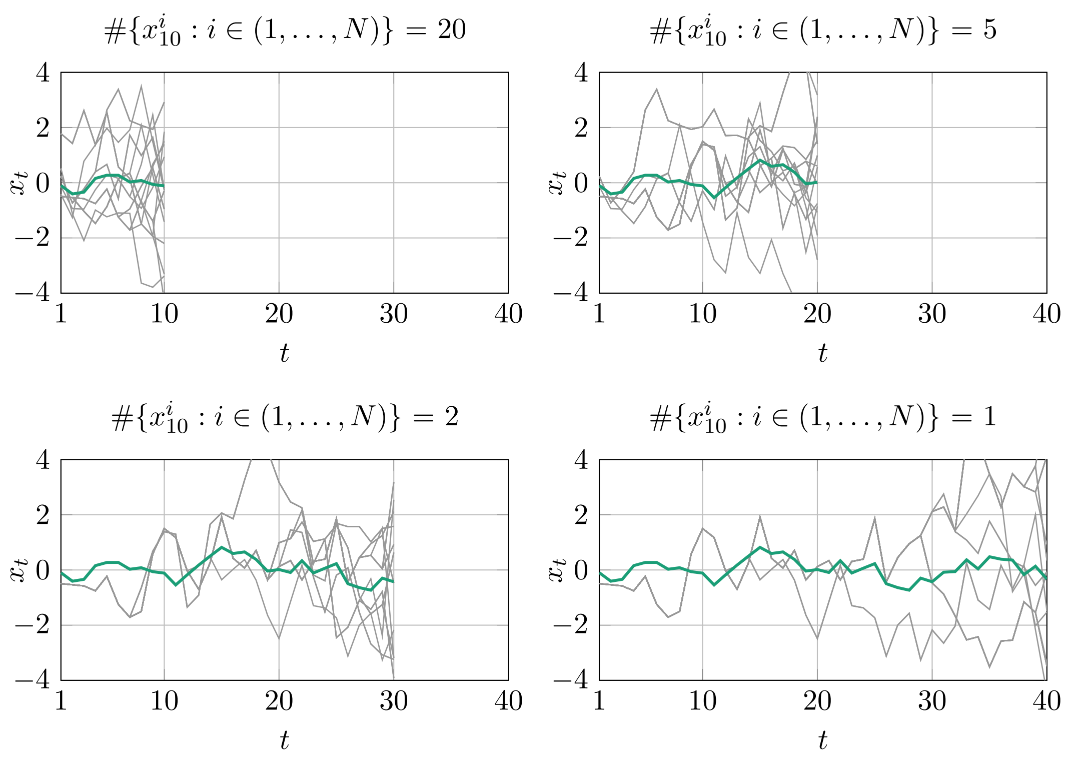

Fig. 1.3: An example of path degeneracy. We consider $\gamma_t(x_{1:t}) \coloneqq \gamma(x_1)\prod_{i=1}^{t}\gamma(x_i|x_{i-1})$ where $\gamma(x_1) = \mathcal{N}(x_1; 0, 1)$ and $\gamma(x_i|x_{i-1}) = \mathcal{N}(x_i; 0.8x_{i-1}, 1)$. The particle trajectories (—), $x_{1:t}^{1:N}$, and the associated trajectory estimate (—), for $t = (10, 20, 30, 40)$ and $N = 20$, computed by Algorithm 2 with the proposal density $m(x_t|x_{t-1}) \coloneqq \mathcal{N}(x_t; 0.8x_{t-1}, 4)$.

commonly referred to as *path degeneracy* [6] or sample impoverishment.

Alternative formulation of the SMC algorithm performs resampling only when the approximate value of the effective sample size (1.25) drops below a certain threshold N_{th}. Such an SMC setup is commonly known as *sequential importance sampling and resampling* and can be adopted as a first step towards reducing the impact of the particle path degeneracy problem. We can see in the bottom row of Fig. 1.2 that the effective sample size of the sequential importance sampling and resampling drops to lower values. This is caused by the fact that the threshold value is $N_{\text{th}} = N/2$, allowing the effective sample size to go to lower values. We can observe that the more uniform the weights the lower the variance and the higher the effective sample size, which is in agreement with (1.23).

Algorithm 2 requires us to select the sequence of importance distributions $(m_t)_{t=1}^{T}$. The choice of these distributions follows the same guideline as discussed in Section 1.4, thus we should select the sequence to be as close as possible to the optimal one $(m_t^\star)_{t=1}^{T}$. The main idea here is that the optimal proposal distribution is supposed to decrease the number of times the resampling step is triggered by decreasing the variance of the importance weights. If it is not possible to apply the optimal proposal distribution—which, indeed, is the most common case—one is advised to design an approximate proposal [173, 32]. Although adopting an approximate optimal proposal can lead to an increase in the diversity of the particle trajectories, the improvement brought by this strategy is rather weak, as will be demonstrated in Chapter 2. Another way of counteracting the path degeneracy is to utilize MCMC moves to diversify the particle trajectories [78]. There exists various types of the transition kernels to achieve this [42]. An alternative approach is to run an ensemble of SMC methods [98]. However, the most important role in counteracting the path degeneracy is played by backward simulation [80] and particle Markov chain Monte Carlo methods [4], which we discuss later on in this chapter.

For a latter reference, we present the density of all variables generated by the SMC algorithm

$$\psi(\mathbf{x}_{1:T}, \mathbf{a}_{1:T-1}) = \left\{\prod_{i=1}^{N} m_1(x_1^i)\right\} \prod_{t=2}^{T} \left\{r(\mathbf{a}_{t-1}|\mathbf{w}_{t-1}) \prod_{i=1}^{N} m_t(x_t^i | x_{1:t-1}^{a_{t-1}^i})\right\}, \tag{1.51}$$

where we use $\mathbf{x}_t \coloneqq x_t^{1:N} = (x_t^1, \dots, x_t^N)$ and $\mathbf{w}_t = (w_t^1, \dots, w_t^N)$. Here, we recall that r is the ancestor sampling distribution appearing in (1.39). The density (1.51) is of key importance when devising more advanced Monte Carlo-based strategies, such as particle Markov chain Monte Carlo [4] and SMC^2 [39].

The SMC methods—despite resetting the weights by resampling—perform a conditional self-normalized importance sampling step at each iteration by drawing the samples in (1.47) and computing (1.49). When the dimension of the marginal space X is high, poor performance should be expected as we face the problem of importance sampling in high-dimensions. The quality of approximation of the target distribution decreases with increasing the dimension of X, even when compensated for by an exponentially increasing number of particles [19]. The distinguishing feature of SMC methods—attractive mainly in the field of Bayesian statistics—lies in that they provide unbiased estimate of the normalizing factor $\gamma_t(1)$, [154].

1.7 Backward Simulation

Backward simulation [80] is a principled approach for sampling from a target probability distribution π_T defined on X^T, where $\mathsf{X} \subseteq \mathbb{R}^{n_x}$. We assume that π_T is known only up to the normalization factor, as in (1.26). The target distribution can be approximated by the SMC approach discussed in Section 1.6. However, as demonstrated in Fig. 1.3, the resulting approximation suffers from the path degeneracy problem. The particle trajectories of such an approximation are strongly dependent for $t \ll T$. A backward simulator first runs the forward SMC algorithm to store the particle systems $(\mathbf{X}_{1:t}, \mathbf{W}_t)$ for $t = 1, \dots, T$ and then removes the dependency by sampling from these particle systems in a backward sweep for $t = T, \ldots, 1$. Indeed, the backward simulator can be seen as an algorithm which applies an additional resampling sweep in the time-reverse direction. Thus, contrary to the forward SMC method, the backward simulator is an offline procedure.

The backward simulator is designed on the basis of factorizing the target distribution according to

$$\pi_T(dx_{1:T}) = k_T(dx_{1:t}|x_{t+1:T})\pi_T(dx_{t+1:T}), \tag{1.52}$$

where the backward transition kernel k_T can be written as [218, 133]

$$k_T(dx_{1:t}|x_{t+1:T}) \propto \frac{\gamma_T(x_{1:T})}{\gamma_t(x_{1:t})}\pi_t(dx_{1:t}). \tag{1.53}$$

We describe the recursive step of the backward simulator in two parts. Assume that we have the partial backward trajectories from the previous iteration (taking the time reverse perspective), $\tilde{X}^{1:M}_{t+1:T}$, where M denotes the number of backward trajectories. Furthermore, consider that we have the weighted particle systems $(\mathbf{X}_{1:t}, \mathbf{W}_t)_{t=1}^T$ computed by the forward SMC procedure in Algorithm 2. In the first part, we use the forward empirical approximation π_t^N—the current particle system $(\mathbf{X}_{1:t}, \mathbf{W}_t)$—to approximate the backward transition kernel (1.53) with

$$k_T^N(dx_{1:t}|\tilde{x}_{t+1:T}) = \sum_{i=1}^N W^i_{t|T}\delta_{X^i_{1:t}}(dx_{1:t}), \tag{1.54}$$

where

$$W^i_{t|T} \propto W^i_t \frac{\gamma_T(X^i_{1:t}, \tilde{x}_{t+1:T})}{\gamma_t(X^i_{1:t})}. \tag{1.55}$$

The computation of this kernel is performed by just evaluating the weights (1.55) conditionally on $\tilde{X}^j_{t+1:T} = \tilde{x}^j_{t+1:T}$ for $j = 1, \ldots, M$ and $i = 1, \ldots, N$. To simplify the subsequent assertion, we introduce $\mathbf{W}^j_{t|T} := W^{1:N,j}_{t|T}$ to denote a set of the backward weights. In the second part, we utilize the approximate kernel (1.54) to extend the backward particle trajectories, $\tilde{X}^{1:M}_{t+1:T}$. This is done by first sampling the index B^j_t based on the set of weights $\mathbf{W}^j_{t|T}$ according to

$$B^j_t \sim \mathcal{F}(\cdot|\mathbf{W}^j_{t|T}).$$

Consequently, we apply the index B^j_t to select from only the (fully diversified) current set of particles, $(X^i_t)_{i=1}^N$, while discarding the (poorly diversified) set of partial forward trajectories, $(X^i_{1:t-1})_{i=1}^N$. Finally, we keep the future backward trajectory $\tilde{X}^j_{t+1:T}$ intact—as the factorization (1.52) suggests—and concatenate it with the selected particle,

$$\tilde{X}^j_{t:T} := (X^{B^j_t}_t, \tilde{X}^j_{t+1:T}).$$

Algorithm 3 The backward simulator

```
A. Initial step: $(t = T)$
    1. Sample $B_t^j \sim \mathcal{F}(\cdot|\mathbf{W}_T)$ and set $\tilde{X}_T^j := X_t^{B_t^j}$.
B. Recursive step: $(t = T-1, \dots, 1)$
    1. Compute $\mathbf{W}_{t|T}^j$ according to (1.55).
    2. Sample $B_t^j \sim \mathcal{F}(\cdot|\mathbf{W}_{t|T}^j)$ and set $\tilde{X}_{t:T}^j := (X_t^{B_t^j}, \tilde{X}_{t+1:T}^j)$.
```

These operations are repeated for each $j = 1, \dots, M$. The recursive step of the backward simulator is now completed. Note that the use of the current particles is of key importance here as they have unreduced diversity. However, since we use the full trajectories $\mathbf{X}_{1:t}$ to compute the backward transition kernel before the sampling of the indices $\mathbf{B}_t$, the method is still affected by the path degeneracy problem into a certain degree [134].

The initial step simply selects $\tilde{X}_T^j := X_T^{B_T^j}$ with $B_t^j \sim \mathcal{F}(\cdot|\mathbf{W}_T)$ for $j = 1, \dots, M$; where $\mathbf{W}_T$ is taken from the forward SMC-based approximation π_T^N at the final time step $t = T$. We summarize the backward sampling procedure in Algorithm 3, where all j-dependent operations are performed for $j = 1, \dots, M$.

Algorithm 3 generates a uniformly-weighted particle system $(\tilde{X}_{1:T}^j, M^{-1})_{j=1}^M$ containing a set of conditionally IID samples from the target distribution π_T. This particle system can be used to form the associated empirical approximation

$$\pi_T^{BS,M}(dx_{1:T}) = \frac{1}{M} \sum_{j=1}^{M} \delta_{\tilde{X}_{1:T}^j}(dx_{1:T}). \tag{1.56}$$

The computational complexity of Algorithm 3 for producing M trajectories of the length T scales with $\mathcal{O}(TNM)$ operations.

1.8 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) simulation [3] refers to a mechanism suitable for generating a set of R random samples $(X[k])_{k=1}^R$ from a target distribution π defined on $\mathsf{X} \subseteq \mathbb{R}^{n_x}$, with π being known only up to the normalizing factor, as in (1.14). The samples are not independent—as in the case of basic Monte Carlo simulation—but constitute a Markov chain. If the target distribution π coincides with the stationary distribution of the chain, then after reaching the stationary regime, the samples are approximately distributed according to π and can be used to address associated inference objectives. The primary aim in designing MCMC methods is to make the time before entering the stationary regime—commonly referred to as the transient phase—as short as possible.

Algorithm 4 The Markov chain Monte Carlo sampler

```
A. Initial step: ($k = 1$)
    * Sample $X[1] \sim \mu(\cdot)$.
B. Recursive step: ($k = 2, \dots, R$)
    * Sample $X[k] \sim \mathcal{K}(X[k-1], \cdot)$.
```

The stochastic behaviour of a time-homogeneous Markov chain $(X[k])_{k=1}^{R}$ is fully determined by the pair of distributions $(\mu, \mathcal{K})$ where μ is the initial probability distribution on X, and $\mathcal{K}$ constitutes the Markov transition kernel which is a probability distribution $\mathcal{K}(x, \cdot)$ for $x \in \mathsf{X}$ and a measurable function $\mathcal{K}(\cdot, A)$ for $A \subseteq \mathsf{X}$. A simple procedure for simulating the Markov chain according to $(\mu, \mathcal{K})$ is presented in Algorithm 4, which can be seen as a generic Markov chain Monte Carlo sampler. We start by simulating the first sample $X[1]$ in the initial step, and then continue by using the sample from previous iteration $X[k-1]$ to generate a new one $X[k]$ in the recursive step. The samples can then be used to compute the empirical expectation of a test function $f : \mathsf{X} \to \mathbb{R}$ under π as

$$\pi^{MCMC,R}(f) = \frac{1}{R} \sum_{k=1}^{R} f(X[k]). \tag{1.57}$$

Whenever we devise a particular MCMC algorithm, we practically design a specific transition kernel $\mathcal{K}$. To develop an algorithm which produces Markov chains that are suitable for estimating $\pi(f)$ by the empirical average (1.57), we have to fulfill certain conditions. The following theorem defines such assumptions and underlines the basic principle of the MCMC methodology.

Theorem 1.10. *If $(X[k])_{k=1}^{R}$ is a π-irreducible, aperiodic, Markov chain with invariant distribution π, and $f : \mathsf{X} \to \mathbb{R}$ is a measurable function such that $\mathsf{E}[|f(X)|] < \infty$; then, for π-almost every initial state $X[1]$, the empirical expectation (1.57) converges almost surely to the exact one (1.2) as the number of samples R increases,*

$$\pi^{MCMC,R}(f) \xrightarrow{a.s.} \pi(f),$$

for $R \longrightarrow \infty$.

Proof. See [30] Section 14.2.6. □

The transition kernel should generally be chosen to capture important features of π and to be easy to sample from. More precise requirements are stated by Theorem 1.10 which constitutes the strong law of large numbers for the Markov chains and states that—as long as we fulfill its requirements—we can design a successful MCMC algorithm. The first requirement is that $\mathcal{K}$ should admit π as its stationary distribution. The stationary distribution characterizes the stable behaviour of the chain and, indeed, it is the first step towards constructing standard MCMC schemes. In the stable regime, for $k \geq n$, the consecutive samples $X[k-1]$ and $X[k]$ are approximately distributed according to π. However, even when $\mathcal{K}$ admits π as its stationary distribution, there is no guarantee that the chain will converge to the stationary regime. To ensure this, we also need the chain to be π-irreducible, that is, for any initial state $X[1] \in \mathsf{X}$, there exists a positive probability of entering any set for which π has positive probability [205]. Moreover, we require the chain to be aperiodic, which states that there are no measurable partitions of the space that can be entered at certain regularly spaced intervals. A stronger version of Theorem 1.10 can be formulated if we additionally assume that the chain is Harris recurrent [152]. The almost sure convergence then holds irrespective of the initial state.

Under the assumptions detailed in [101], it can be demonstrated that the error of MCMC methods follows the central limit theorem

$$R^{\frac{1}{2}}[\pi^{MCMC,R}(f) - \pi(f)] \xrightarrow{d} \mathcal{N}(0, \sigma^2_{MCMC}(f)),$$

for $R \longrightarrow \infty$. Here,

$$\sigma^2_{MCMC}(f) = \mathsf{V}_\pi[f(X_1)] + 2\sum_{k=1}^{\infty} \mathsf{C}_\pi[f(X_1), f(X_k)] < \infty.$$

Thus, the MCMC methods converge under the standard $\mathcal{O}(R^{-\frac{1}{2}})$ rate.

The MCMC methods can generally be separated into two main classes known as Metropolis-Hastings and Gibbs samplers. Since we do not use the Metropolis-Hastings algorithm in this thesis, we leave it from the discussion in this section.

1.8.1 The Gibbs Sampler

To simplify the subsequent description, let us consider that the target distribution $\pi(dx)$ admits a probability density function $\pi : \mathsf{X} \rightarrow \mathbb{R}^+$ with respect to a dominating measure which we (abusively) denote by dx. To facilitate construction of the Gibbs sampler [72], we need that the quantity of interest can be separated into at least two components $X \coloneqq (Z, \Theta)$. The algorithm is then referred to as the two-stage Gibbs sampler [183]. More generally, we can also have a target density that contains more than two variables, $\pi(x_{1:t})$; however, for simplicity, we do not consider this case here and rather refer the reader to [183, 30] for more details. The main motivation for resorting to Gibbs sampler lies in that, in some situations, it may be more convenient and more tractable to sample from the conditional densities $\pi(z|\theta)$ and $\pi(\theta|z)$ rather than directly from the joint density $\pi(z, \theta)$.

The basic idea of the Gibbs sampler is that, given the previous state $X[k-1]$, we obtain the current state $X[k]$ by first drawing $\Theta[k] \sim \pi(\cdot|Z[k-1])$ and then

Algorithm 5 The Gibbs sampler

```
A. Initial step: $(k = 1)$
    1. Sample $\Theta[1], Z[1] \sim \mu(\cdot)$.
B. Recursive step: $(k = 2, \ldots, R)$
    1. Sample $\Theta[k] \sim \pi(\cdot|Z[k-1])$.
    2. Sample $Z[k] \sim \pi(\cdot|\Theta[k])$.
```

$Z[k] \sim \pi(\cdot|\Theta[k])$. More specifically, conditionally on $Z[k-1]$, we sample $\Theta[k]$—without utilizing the previous state $\Theta[k-1]$—and then use this current state to condition sampling of $Z[k]$. These two steps define a complete *sweep* of the Gibbs sampler and can individually be described by their respective transition kernels,

$$\mathcal{K}_\theta(x, dx^*) = \delta_z(dz^*)\pi(\theta^*|z)d\theta^*,$$
$$\mathcal{K}_z(x, dx^*) = \delta_\theta(d\theta^*)\pi(z^*|\theta)dz^*.$$

It can be shown, see, e.g., [30] Section 6.2, that each of these kernels admits π as its stationary density. A complete Gibbs transition kernel is given by $\mathcal{K} = \mathcal{K}_\theta \mathcal{K}_z$ and can be summarized by the recursive step of Algorithm 5.

In the case of using the Gibbs kernel to produce samples from a high-dimensional target density $\pi(x_{1:t})$ with strongly correlated quantities $X_{1:t}$, the performance depends on a particular setting of the updating scheme [184]. If we decide to design the sweep so that the quantities are updated individually, then the resulting kernel will suffer from poor mixing and the updates will be negligible. There are generally two ways how to address this problem known as *blocking* and *collapsing* [137]. Blocking is a strategy which separates the sequence $X_{1:t}$ into a number of smaller blocks and the sweep is designed to sample these blocks separately. Collapsing is a technique where certain quantities in $X_{1:t}$ are marginalized out and the sweep is created to sample only the remaining quantities in the standard way.

1.9 Particle Markov Chain Monte Carlo

Various statistical inference objectives often lead to the requirement of approximating a joint target density

$$\pi(\theta, x_{1:T}) = \frac{\gamma(\theta, x_{1:T})}{\gamma(1)}, \tag{1.58}$$

where $\Theta \in \mathsf{\Theta} \subseteq \mathbb{R}^{n_\theta}$ and $X_t \in \mathsf{X} \subseteq \mathbb{R}^{n_x}$, for $t = 1, \ldots, T$, are some unknown static and dynamic quantities of interest, respectively. We consider that $\gamma : \mathsf{\Theta} \times \mathsf{X}^T \to \mathbb{R}^+$ is point-wise known whereas $\gamma(1) := \int_{\mathsf{\Theta} \times \mathsf{X}^T} \gamma(\theta, x_{1:T}) d\theta dx_{1:T}$ is unknown. The approximation can be performed by applying the MCMC techniques. However, although these methods converge under rather weak assumptions, their success strongly depends on our ability to design suitable proposal densities. Specifically, as mentioned in Section 1.8, their performance can degrade when the joint quantities are sampled individually and there exists a complex dependence structure among them. This is even more pronounced if the target density is high-dimensional. The particle MCMC methodology [4] addresses these issues by utilizing sequential Monte Carlo methods [56] to construct high-dimensional proposal distributions for MCMC techniques. The particle MCMC approach has recently shown to be the key enabler for more sophisticated Bayesian inference problems than ever before.

The underlying idea of the particle MCMC methodology is to use exact MCMC techniques to sample from an extended target density $\tilde{\pi}^N$ which is defined on the space of the static quantity and all the quantities generated by the SMC algorithm, $\Theta \times \mathbf{X} \coloneqq \Theta \times \mathsf{X}^{NT} \times (1, \ldots, N)^{N(T-1)+1}$, that is,

$$\tilde{\pi}^N(\theta, k, \mathbf{x}_{1:T}, \mathbf{a}_{1:T-1}) \coloneqq \frac{\pi(\theta, x^k_{1:T})}{N^T} \frac{\psi^\theta(\mathbf{x}_{1:T}, \mathbf{a}_{1:T-1})}{m^\theta_1(x_1^{b^k_1}) \prod_{t=2}^T r(b^k_{t-1} | \mathbf{w}_{t-1}) m^\theta_t(x_t^{b^k_t} | x_{1:t-1}^{b^k_{t-1}})}, \tag{1.59}$$

where ψ^θ is defined in (1.51). The key feature of (1.59) lies in that it admits (1.58) as the marginal density. A specific particle MCMC method is designed as a Markov transition kernel $\mathcal{K}$ on the extended space $\Theta \times \mathbf{X}$ with (1.59) being the invariant density. The generic sampler follows exactly the same steps as in Algorithm 4. However, the produced chain contains only the samples targeting the marginal density, $(\Theta[k], X_{1:T}[k])_{k=1}^R$, whereas the auxiliary variables are discarded at each iteration.

A remarkable feature of the particle MCMC methods is that the associated transition kernel leaves π invariant for a finite number of particles N. Thus, the number of particles N does not need to tend to infinity for these methods to converge. They preserve the convergence properties of the basic MCMC procedures that converge with the number of iterations R going to infinity. The particle MCMC methods overcome the difficulties related to the particle path degeneracy problem discussed in Section 1.6. However, they still suffer from this issue in the sense that it affects their convergence speed. The particle MCMC methods are generally highly computationally demanding, as they require us to run a full forward sweep of an SMC method to generate only a single particle trajectory at each iteration. If we run such techniques for R iterations, then their computational complexity scales at least with $\mathcal{O}(RNT)$ operations. A conceptual simplification provided by the particle MCMC methods is that the problem of designing proposal distributions for MCMC reduces to the problem of designing proposal distributions for SMC.

Similarly as in Section 1.8, as we do not use the Metropolis-Hastings algorithm in this thesis, we leave its particle MCMC version from the discussion.

Algorithm 6 The conditional sequential Monte Carlo (CSMC) update

```
A. Initial step: $(t = 1)$
    1. Sample $X_1^i \sim m_1(\cdot)$ for $i \neq B_1^K$.
    2. Compute $W_1^i \propto v_1(X_1^i)$.
B. Recursive step: $(t = 2, \ldots, T)$
    1. Sample $\mathbf{A}_{t-1}^{-B_t^K} \sim r(\cdot|\mathbf{W}_{t-1}, A_{t-1}^{B_t^K} = B_{t-1}^K)$.
    2. Sample $X_t^i \sim m_t(\cdot|X_{1:t-1}^{A_{t-1}^i})$ for $i \neq B_t^K$ and set $X_{1:t}^i := (X_t^i, X_{1:t-1}^{A_{t-1}^i})$.
    3. Compute $W_t^i \propto v_t(X_{1:t}^i)$ according to (1.49).
```

1.9.1 The Particle Gibbs Sampler

A typical way of designing a Gibbs sampler for the target density (1.58) is to sample from $\pi(\theta|x_{1:T})$ and $\pi_\theta(x_{1:T})$. We assume that the first factor $\pi(\theta|x_{1:T})$ is tractable, which is commonly the case if conjugate models are chosen. However, the second factor $\pi_\theta(x_{1:T})$ is notoriously intractable, except the simplest situations, which prevents us from completing the Gibbs sweep. To address this problem, we can construct a Gibbs sampler with the target density (1.59) on the extended space $\Theta \times \mathbf{X}$. The resulting algorithm is referred to as the particle Gibbs sampler [4] and its sweep is given by the three parts

$$\Theta^* \sim \tilde{\pi}^N(\cdot|k, x_{1:T}^k, b_{1:T-1}^k) = \pi(\cdot|x_{1:T}^k), \tag{1.60a}$$

$$\mathbf{X}_{1:T}^{*,-b_{1:T}^k}, \mathbf{A}_{1:T-1}^{*,-b_{2:T}^k} \sim \tilde{\pi}^N(\cdot|\theta^*, k, x_{1:T}^k, b_{1:T-1}^k), \tag{1.60b}$$

$$K^* \sim \tilde{\pi}^N(\cdot|\theta^*, \mathbf{x}_{1:T}^{*,-b_{1:T}^k}, \mathbf{a}_{1:T-1}^{*,-b_{2:T}^k}, x_{1:T}^k, b_{1:T-1}^k), \tag{1.60c}$$

where $\mathbf{X}_{1:T}^{-b_{1:T}^k} := (\mathbf{X}_1^{-b_1^k}, \ldots, \mathbf{X}_T^{-b_T^k})$ and $\mathbf{X}_t^{-n} = (X_t^1, \ldots, X_t^{n-1}, X_t^{n+1}, \ldots, X_t^N)$. The same type of notation holds also for the ancestor indices. The first part samples the parameters based on some previous trajectory. Even when the implementation of this part is not possible due to intractability of the density in (1.60a), we can use the Metropolis-Hastings algorithm to obtain the samples. To implement the second part (1.60b), where the density coincides with the second fraction on the r.h.s. of (1.59), we need a specific type of the SMC algorithm refereed to as the *conditional SMC (CSMC) update* [4]. The key idea behind this approach is to guarantee that a single particle trajectory $X_{1:t}$ with the ancestral lineage $B_{1:T}$ survives all the resampling steps during the full run of the SMC algorithm. The procedure only samples $N-1$ particle trajectories according to Algorithm 2 but keeps one prespecified trajectory intact. Thus, conditionally on $X_{1:T}^K = x_{1:T}^k$ and $B_{1:T}^K = b_{1:T}^k$, the method proceeds as delineated in Algorithm 6. The third part samples an index from a probability mass function which is conditioned on all random quantities generated by the CSMC algorithm and the prespecified trajectory. This is equivalent to sampling the index based on the final set of the normalized importance weights $\mathbf{W}_T$ or, in other words, to sampling from $\pi_\theta^N(x_{1:T})$. The index is then used to prespecify the trajectory

Algorithm 7 The particle Gibbs sampler

```
A. Initial step: $(i = 1)$
    1. Sample $\Theta[1], X_{1:T}[1] \sim \mu(\cdot)$.
B. Recursive step: $(i = 2, \ldots, R)$
    1. Sample $\Theta[i] \sim \pi(\cdot|X_{1:T}[i-1])$.
    2. Conditionally on $X_{1:T}[i-1]$, $B_{1:T}[i-1]$, and $\Theta[i]$, run Algorithm 6.
    3. Sample $K$ with $\mathbb{P}(K = l) \coloneqq W_T^l$ and set $X_{1:T}[i] = X_{1:T}^K$ and $B_{1:T}[i] = B_{1:T}^K$.
```

for the next iteration. A simplified listing of the particle Gibbs sampler (1.60) is presented in Algorithm 7.

Although the basic form of the particle Gibbs sampler presented in Algorithm 7 converges in the sense of Theorem 1.10, it suffers from the previously discussed path degeneracy problem. The trajectories generated by the CSMC update are in some sense close to the prespecified trajectory and are highly dependent for $t \ll T$, as presented in Fig. 1.3. Therefore, we can only sample a new prespecified trajectory which is to a large extent similar to the previous one. Given the fact the forgetting properties of the proposal kernel m_t are satisfactory, the generated trajectories are independent of the prespecified trajectory with a high probability. This is the key idea which allows the sampler to work; otherwise, without the forgetting properties, every new trajectory would be just the same as the previous one and the space would not be explored. However, the fact the trajectories are similar makes the particle Gibbs sampler to mix poorly.

A very straightforward remedy to this problem is to simply have the number of particles N significantly greater than T. This may increase the number of independent trajectories for $t \ll T$ and thus improve the mixing properties. Another way to achieve this is to choose a resampling strategy which decreases the number of times the particle trajectories are resampled [63]. A more efficient approach was suggested in [216] and lies in introducing a backward simulation sweep into the CSMC update in order to sample the ancestral lineage $B_{1:T}^K$ rather than tracing it back in the deterministic sense (1.50). This proposal was later improved in [134] so that the ancestor sampling can be performed in the single forward run of the CSMC update.

1.10 Rao-Blackwellization

A wide range of practical problems often require us to approximate a target density $\pi(x)$—defined on $\mathsf{X} \subseteq \mathbb{R}^{n_x}$—such that the quantity of interest is composite, $X = (U, V)$, and the factorization

$$\pi(u, v) = \pi^c(u|v)\pi^m(v), \tag{1.61}$$

where the conditional factor π^c is conditionally tractable given V but the marginal factor π^m is intractable, exists. We can proceed by approximating the target density directly with any method presented in the previous sections. However, in this case, such an approach may prove to be inefficient in terms of the estimation accuracy and computational complexity. Rao-Blackwellization is a principle which suggests to apply a Monte Carlo method to approximate only the marginal factor π^m and perform the computations associated with the conditional factor π^c under a closed-form solution. The accuracy of an estimator under (1.61) can then be lower than or at least as same as of an estimator based on the joint samples. A simple intuition behind Rao-Blackwellization is that focusing samples on the subspace $\mathsf{V} \subset \mathsf{X}$ is more efficient than on the compete space X.

Let us consider we have used one of the previously discussed methods to approximate the marginal factor π^m by the empirical approximation $\pi^{m,N}$ associated with the weighted particle system $(V^i, W^i)_{i=1}^N$. Then, the approximation of the target density is simply obtained by inserting $\pi^{m,N}$ in (1.61),

$$\pi^{RB,N}(u, dv) = \pi^c(u|v)\pi^{m,N}(dv). \tag{1.62}$$

The weighted particle system of $\pi^{RB,N}(u, dv)$ is then given by $(V^i, \pi^{i,c}, W^i)_{i=1}^N$, where $\pi^{i,c} := \pi^c(\cdot|V^i)$. The conditional factors $(\pi^{i,c})$ are often represented by a set of finite-dimensional statistics (S^i) that can be computed under a closed-form analytical expression for each V^i. Therefore, it is more common to use the weighted particle system defined by $(V^i, S^i, W^i)_{i=1}^N$ [55, 192], although the appearance of the full conditional factors $(\pi^{i,c})$ in the particle system can be seen, e.g., when dealing with the discrete-valued substructures [167]. The expected value of a test function $f : \mathsf{U} \times \mathsf{V} \to \mathbb{R}$ under π^c is tractable conditionally on (V^i). This allows us to express the estimator with respect to (1.61) as

$$\pi^{RB,N}(f) = \sum_{i=1}^N W^i \mathsf{E}[f(U, V^i)|V^i].$$

To assess the efficiency gain brought by a Rao-Blackwellized estimator $\pi^{RB,N}(f)$ compared to a non-Rao-Blackwellized estimator $\pi^N(f)$, we consider the variance decomposition

$$\mathsf{V}[\pi^N(f)] = \mathsf{V}[\pi^{RB,N}(f)] + \mathsf{E}[\mathsf{V}[\pi^N(f)|V]]. \tag{1.63}$$

This formula can only serve for a basic qualitative comparison between $\pi^{RB,N}(f)$ and $\pi^N(f)$ as the involved expectations cannot be evaluated except the trivial cases. Moreover, the comparison would depend on specific Monte Carlo methods adopted. Since the second term on the r.h.s. of (1.63) is always non-negative, we can state that $\pi^{RB,N}(f)$ can only have lower—or at worst the same—variance as $\pi^N(f)$. However, the size of the second term affects the trade-off between estimation accuracy and computational time. The computational cost at a given number of particles N is usually higher for $\pi^{RB,N}(f)$ than for $\pi^{N}(f)$, which is caused by the fact we need to evaluate the statistic S^i associated with $\pi^{c,i}$ for each $i = 1, \ldots, N$. On the one hand, if the second term is low, the variance of $\pi^{N}(f)$ and $\pi^{RB,N}(f)$ will be approximately the same, but the cost of computing $\pi^{RB,N}(f)$ will be higher at a given N. The Rao-Blackwellization is rather inefficient in such situations. On the other hand, if the second term is high, the variance of $\pi^{N}(f)$ and $\pi^{RB,N}(f)$ will be substantially different, and it could take a high N—and more computational resources—to match the variance of $\pi^{N}(f)$ with $\pi^{RB,N}(f)$. The Rao-Blackwellization is most efficient in this case. A concrete trade-off commonly depends on a given model and the uncertainty of associated quantities.

Practically all the methods discussed previously can have their Rao-Blackwellized variants. For example, applying Rao-Blackwellization in the context of importance sampling-based methods leads to canceling the conditional factors out in the importance weights, which in turn provides us with their lower variance. For the backward simulation, full Rao-Blackwellization requires us to design a Rao-Blackwellized forward and backward samplers separately.

Rao-Blackwellization can be applied even if the conditional factor π^c is analytically intractable. In such situations, we use a nested Monte Carlo method to approximate π^c conditionally on V^i by an empirical approximation $\pi^{i,c,M}$ for all $i = 1, \ldots, N$. The particle system $(V^i, \pi^{i,c,M}, W^{i,v})_{i=1}^{N}$ is then assembled in a way that each $\pi^{i,c,M}$ is represented by its own, local, particle system $(U^{i,j}, W^{i,j,u})_{j=1}^{M}$. This approach is referred to as exact approximate Rao-Blackwellization [100]. Here, the exactness means that the method converges for $N \to \infty$ and any number of particles $M \geq 1$ of the local particle filters. For a fixed N and $M \to \infty$, the procedure approaches the performance of a Rao-Blackwellized method designed for a target distribution with a tractable substructure. However, a question is whether such an approach computationally outperforms a plain Monte Carlo procedure with MN particles.

2 STATE AND PARAMETER INFERENCE IN STATE-SPACE MODELS

2.1 State-Space Models

A state-space model or a hidden Markov model [152] is the most widely used tool for modeling of dynamical systems. This fact follows mainly from its versatility and ability to cover a broad class of non-linear and non-Gaussian systems. The model is commonly applied in various areas of science and engineering, including signal processing, econometrics, bioinformatics, sociology, etc. However, the generality of state-space models comes at the price of not providing analytically tractable solutions to the associated inference problems in most practical cases of interest, which requires us to resort to approximate techniques.

2.1.1 Definition

A state-space model generally characterizes (or interprets) a bivariate, discrete-time, stochastic process $(X_t, Y_t)_{t=1}^{T}$, where the random variables[1] X_t and Y_t take values in some (perhaps discrete-valued) spaces $\mathsf{X} \subseteq \mathbb{R}^{n_x}$ and $\mathsf{Y} \subseteq \mathbb{R}^{n_y}$, respectively, with $n \in \mathbb{N}_+$ denoting their dimension. We refer to the variables $(Y_t)_{t=1}^{T}$ as observations and assume that they can be measured on a system under study. The variables $(X_t)_{t=1}^{T}$ are usually called as latent (unobserved) states, as they cannot directly be observed. The state-space model evolves according to the probability distributions given by

$$\mathbb{P}(X_t \in A | X_{t-1} = x_{t-1}, \Theta = \theta) := F_\theta(x_{t-1}, A), \tag{2.1a}$$

$$\mathbb{P}(Y_t \in B | X_t = x_t, \Theta = \theta) := G_\theta(x_t, B), \tag{2.1b}$$

for measurable subsets $A \subseteq \mathsf{X}$ and $B \subseteq \mathsf{Y}$. The states $(X_t)_{t=1}^{T}$ constitute a homogeneous Markov chain of an initial distribution $\mathbb{P}(X_t \in A | \Theta = \theta) := \mu_\theta(A)$ and transition kernel (2.1a). Thus, given the previous state, x_{t-1}, (2.1a) determines the probability that the current state moves into the set $A \subseteq \mathsf{X}$. The observations $(Y_t)_{t=1}^{T}$ are conditionally independent given $(X_t)_{t=1}^{T}$, and they have a marginal distribution defined by (2.1b). Similarly as before, given the current state, x_t, (2.1b) describes the probability that the current observation belongs into the set $B \subseteq \mathsf{Y}$. All the above-introduced distributions depend on a fixed parameterization Θ which takes values in a (mostly real-valued) space $\mathsf{\Theta} \subseteq \mathbb{R}^{n_\theta}$ and has a prior distribution $\mathbb{P}(\Theta \in C) := \nu(C)$, for all measurable $C \subseteq \mathsf{\Theta}$. A time-homogeneous state-space

[1]All random variables are defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

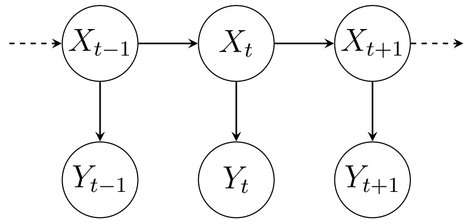

Fig. 2.1: Graphical model of a state-space model.

model is thus fully determined by the distributions $(\mu_\theta, F_\theta, G_\theta, \nu)$. The dependence structure of the variables in a state-space model is depicted in the graphical model in Fig. 2.1, where we leave Θ out as it is connected to all the nodes.

To simplify subsequent presentation, we assume that μ_θ and F_θ have probability density functions $\mu_\theta : \mathsf{X} \to \mathbb{R}_+$ and $f_\theta : \mathsf{X} \times \mathsf{X} \to \mathbb{R}_+$, respectively, both defined with respect to some dominating measure dx. Additionally, G_θ also admits a probability density function $g_\theta : \mathsf{X} \times \mathsf{Y} \to \mathbb{R}_+$ with respect to some dominating measure dy. The state-space model is then sometimes referred to as being *fully dominated* [30]. Moreover, we consider that there exists a dominating measure $d\theta$ such that ν has a probability density function $\nu : \Theta \to \mathbb{R}_+$. Note we abusively use the same symbol for the initial probability distributions and densities.

Given the set of densities $(\mu_\theta, f_\theta, g_\theta, \nu)$, the stochastic behavior of a state-space model over the full time horizon $(1, \dots, T)$, with $T \in \mathbb{N}_+$ being the final time index, is described by the joint density of the states, observations, and parameters,

$$p(\theta, x_{1:T}, y_{1:T}) = \nu(\theta) p_\theta(x_{1:T}, y_{1:T}) = \nu(\theta)\mu_\theta(x_1) \prod_{t=2}^{T} f_\theta(x_t|x_{t-1}) \prod_{t=1}^{T} g_\theta(y_t|x_t). \quad (2.2)$$

We continue by discussing basic concepts for inferring the quantities $(\Theta, X_{1:T})$.

2.1.2 Inference Objectives

The utility of the complete description (2.2) lies in that it facilitates formulation of practically all inference tasks related to state space models. A particular inference objective is merely a matter of being interested in certain conditional and marginal distributions of (2.2). The most complete statistical inference objective is to learn the model $(\mu_\theta, f_\theta, g_\theta, \nu)$ by computing the joint posterior density of the unknown states $X_{1:T}$ and parameters Θ based on a realization of the observation sequence $Y_{1:T} = y_{1:T}$, that is,

$$p(\theta, x_{1:T}|y_{1:T}) = \frac{p(\theta, x_{1:T}, y_{1:T})}{p(y_{1:T})}, \quad (2.3)$$

where $p(y_{1:T}) = \int p(\theta, x_{1:T}, y_{1:T}) d\theta dx_{1:T}$. The density (2.3) is then used to provide some of the characteristic features of the unknown quantities, such as their mean and variance. We divide the most common inference tasks, leading from the factorization $p(\theta, x_{1:T}|y_{1:T}) = p_\theta(x_{1:T}|y_{1:T})p(\theta|y_{1:T})$, into the following two major categories: (i) state and (ii) parameter inference.

2.1.3 State Inference

Various state inference objectives can generally be formulated by the joint density of the states given the observations,

$$p_\theta(x_{i:j}|y_{1:k}) = \frac{p_\theta(x_{i:j}, y_{1:k})}{p_\theta(y_{1:k})}, \tag{2.4}$$

where $1 \leq i \leq j \leq k$. A particular inference task is then only a matter of a proper choice of the indices i, j, and k. The most basic—but most important—state inference objective is the state filtering, which can be formulated by choosing $i = j = k = t$ in (2.4), where $t = 1, \ldots, T$ denotes the current time instance. The filtering task relies on all the observations gathered to the current time instance and uses them to compute the posterior density of the current state, $p_\theta(x_t|y_{1:t})$. The observation sequence here grows as the time index t increases. A related state inference objective is the state prediction which utilizes observations up to the current time step to obtain the l-step-ahead state posterior density—the density of a future state—$p_\theta(x_l|y_{1:t})$, where $l > t$. The one-step-ahead state prediction, $l = t + 1$, is the inherent part of the filtering procedure. Another state inference task is the state smoothing which considers the observation sequence of a certain size and computes the posterior density of a state sequence which is shorter than—or at most as long as—the observation sequence, $p_\theta(x_{i:j}|y_{1:t})$, where $1 \leq i \leq j \leq t$. We distinguish a number of special cases of the state smoothing that result from a specific choice of indices in (2.4). For $i = 1$ and $j = k = t$, we have the forward smoothing which computes the posterior density of the state sequence up to the current time instance given the corresponding observation sequence, $p_\theta(x_{1:t}|y_{1:t})$. As a special case of this approach, for $i = t - l + 1$, $j = t$, and $k = t$, we obtain the posterior density of a fixed-lag state sequence, $p_\theta(x_{t-l+1:t}|y_{1:t})$, where $l \geq 2$. This task is often referred to as the fixed-lag smoothing. So far, we have considered only the instances that can be performed in a sequential or online manner. The following two inference objectives are particularly suited for off-line scenarios. For $i = j < k = T$, we compute the posterior density of a past state given all available observations, $p_\theta(x_i|y_{1:T})$, which is known as the marginal smoothing. For $i = 1, j = k = T$, we compute the posterior density of all states given all observations, $p_\theta(x_{1:T}|y_{1:T})$, which is often known as the joint smoothing [30], constituting the most complete state-inference task we can perform. Finally, for $1 \leq i < j \leq k = T$, the objective is to compute the posterior density of a certain state sequence given all available observations,

Task	Indices
Filtering	$i = j = k = t$
Prediction	$i = j > k = t$
Joint smoothing	$i = 1, j = k = T$
Forward smoothing	$i = 1, j = k = t$
Fixed-lag smoothing	$i = t - l + 1, j = k = t$
Marginal smoothing	$i = j < k = T$
Fixed-interval smoothing	$1 \leq i < j \leq k = T$

Tab. 2.1: Common state inference tasks formulated by particular choices of the indices r and s in the joint state posterior distribution (2.4).

$p_\theta(x_{i:j}|y_{1:T})$, which is referred to as the fixed-interval smoothing. We summarize all these instances in Tab. 2.1. We point out that there is a difference between the interpretation of $p_\theta(x_{1:t}|y_{1:t})$ and $p_\theta(x_{1:T}|y_{1:T})$. While the former can only be computed in the forward direction, the latter one is also based on the backward computations.

The common problem of all described state inference objectives lies in that (2.4) is often known only up to the normalizing factor $p_\theta(y_{1:k})$. To compute $p_\theta(y_{1:k})$, one needs to deal with a complex high dimensional integral with respect to the latent state variables, which is notoriously intractable except a restricted amount of cases, including discrete-valued state-space models, linear Gaussian state-space models, and solutions based on the Fokker-Plank equation [44].

The state inference is an important intermediate step in addressing parameter inference objectives. For example, the joint state posterior distribution $p_\theta(x_{1:T}|y_{1:T})$ is often required for constructing expectation maximization algorithms and Gibbs samplers. The density (2.4) is a generic formulation of the Bayesian state inference objectives. We will adopt this approach throughout the present thesis and will not consider alternative, non-Bayesian, strategies.

2.1.4 Parameter Inference

The parameter inference in state-space models can be divided into two main methodological directions known as frequentist and Bayesian inference. The frequentist and Bayesian techniques share a mutual problem which consists in that we first need to deal with the unknown state sequence in order to facilitate parameter inference. There are generally two strategies how to address this problem [193]. The first one is *data augmentation* which treats the states as auxiliary variables that are estimated along with the parameters. The second one is *marginalization* where the states are marginalized out and only the parameters are of interest.

Frequentist Inference

The main objective of the frequentist or maximum likelihood inference [140] is to locate the point parameter estimate which maximizes the likelihood of the observed data sequence,

$$\widehat{\theta}_{\mathrm{ML}} = \underset{\theta \in \Theta}{\arg\max}\, p_\theta(y_{1:T}). \tag{2.5}$$

The unknown parameters are seen as a fixed, non-random, quantity θ. Under appropriate regularity conditions, the maximum likelihood approach provides strongly consistent and asymptotically normal estimators [83]. There are generally two problems we face with the application of the maximum likelihood estimation in the context of state-space models. First, $p_\theta(y_{1:T})$ cannot be computed under a close-form solution unless in a restricted number of models. The reason for this is that the complex high-dimensional integral needs to be computed in order to obtain $p_\theta(y_{1:T})$. Second, it is often the case that the maximization does not admit an explicit solution, although there are, again, certain model classes where a tractable solution can be found. Moreover, $p_\theta(y_{1:T})$ can contain several local maxima or the maximum likelihood estimate does not have to be unique. Indeed, we can encounter situations where $p_\theta(y_{1:T})$ has, for instance, two maxima of the same value but different location. The first problem is usually addressed with approximate state inference techniques. The second one commonly leads to the application of specific gradient-based search methods, if $p_\theta(y_{1:T})$ is sufficiently smooth.

Bayesian Inference

The central aim of Bayesian inference [17] is to compute the posterior distribution of the unknown parameters based on the observed data sequence, according to Bayes' rule,

$$p(\theta|y_{1:T}) = \frac{p(\theta, y_{1:T})}{p(y_{1:T})}, \tag{2.6}$$

where $p(\theta, y_{1:T}) = p_\theta(y_{1:T})\nu(\theta)$. The unknown parameters are treated as an unobserved random variable Θ with a prior distribution ν which represents our beliefs about the modeled quantity before processing any observations. Thus, Bayesian inference is a tool for updating prior (subjective) beliefs to posterior beliefs based on the observed (objective) data. Under appropriate assumptions, the Bayesian approach offers asymptotically consistent posterior distributions (the posterior concentrates around some θ_0) [76]. Similarly as before, there are again two main problems associated with the application of the Bayesian estimation in the context of state-space models. First, the observed data likelihood, $p_\theta(y_{1:T})$, is—in the same manner as with the frequentist inference—intractable due to the necessity of computing the high-dimensional integral. In simple scenarios, it is possible to compute

$p_\theta(y_{1:T})$ under a closed-form solution. However, even then, when the prior, $\nu(\theta)$, is not conjugate to the observed data likelihood, $p_\theta(y_{1:T})$, the computations associated with the posterior density (2.6) are intractable as there is the need to deal with the integration involved in the *marginal* likelihood, $p(y_{1:T})$. Second, even when $p_\theta(y_{1:T})$ and $\nu(\theta)$ are at least component-wise conjugate as, e.g., in mixture models, the posterior density (2.6) can have a multimodal shape, raising the question of how to appropriately pick the point parameter estimate. The first problem can be approached by resorting to approximate state inference procedures. The second problem is approached based on the answer to the above question. We can be interested in either the maximum a posteriori estimate or an estimate defined as the minimizer of a suitably chosen loss function [15]. In both these choices, we first need to resort to approximations in order to overcome the intractable integration in the denominator of (2.6).

2.1.5 Examples

In this section, we present a number of simple examples of state-space models that will be used throughout this chapter. The selection of these models is made so that we can demonstrate differences between various inference methods for state-space models. The attention is paid to simplicity so that distinctions among compared approaches can easily be seen.

Example 2.1 (Linear Gaussian model)**.** *The model is defined by*

$$\begin{aligned} X_t &= 0.95 X_{t-1} + W_t, \\ Y_t &= X_t + V_t, \end{aligned}$$

where $W_t \sim \mathcal{N}(\mu_w, \sigma_w^2)$ and $V_t \sim \mathcal{N}(\mu_v, \sigma_v^2)$ are IID Gaussian random variables with the mean μ and variance σ^2.

Example 2.2 (Nonlinear Gaussian model)**.** *Consider equations in the form [112]*

$$\begin{aligned} X_t &= \frac{X_{t-1}}{2} + \frac{25 X_{t-1}}{1 + X_{t-1}^2} + 8\cos(1.2t) + W_t, \\ Y_t &= \frac{X_t^2}{20} + V_t, \end{aligned}$$

where $W_t \sim \mathcal{N}(\mu_w, \sigma_w^2)$ and $V_t \sim \mathcal{N}(\mu_v, \sigma_v^2)$ are IID Gaussian random variables with the mean μ and variance σ^2.

The filtering and smoothing distributions for the model in Example 2.2 are generally bi-modal due to the square of the state variable in the observation equation.

Example 2.3 (Nonlinear and non-Gaussian model). *The model is characterized by the equations [209]*

$$X_t = 1 + \sin(0.04\pi t) + \frac{X_{t-1}}{2} + W_t,$$
$$Y_t = \begin{cases} \frac{X_t^2}{10} + V_t, & t \leq t_1 \\ \frac{X_t}{2} - 2 + V_t, & t > t_1 \end{cases},$$

where $W_t \sim \mathcal{G}a(\alpha, \beta)$ is IID Gamma random variable with the shape α and β rate parameters, and $V_t \sim \mathcal{N}(\mu, \sigma^2)$ is IID Gaussian random variable with the mean μ and variance σ^2.

2.2 Forward Filtering

The central aim of the forward filtering is to compute the posterior density of the current state given the observed data sequence, $p(x_t|y_{1:t})$. The observations arrive sequentially in time and are processed in the online manner. Note that, here and throughout the subsequent sections, we leave the dependence on Θ whenever dealing with the pure state inference objectives. The filtering density is generally computed as the marginal of the joint state posterior density, $p(x_t|y_{1:t}) = \int p(x_{1:t}|y_{1:t})dx_{1:t-1}$. Given the fact that the joint state posterior density is proportional to the joint density of the states and observations, $p(x_{1:t}|y_{1:t}) \propto p(x_{1:t}, y_{1:t})$, c.f. (2.2), the filtering density becomes, considering the conditional independence assumptions discussed in Section 2.1.1,

$$p(x_t|y_{1:t}) = \frac{g(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \tag{2.7a}$$

where

$$p(x_t|y_{1:t-1}) = \int_{\mathsf{X}} f(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}, \tag{2.7b}$$
$$p(y_t|y_{1:t-1}) = \int_{\mathsf{X}} g(y_t|x_t)p(x_t|y_{1:t-1})dx_t. \tag{2.7c}$$

At the initial time step $t = 1$, we have $p(x_1) = \mu(x_1)$. Here, (2.7a) and (2.7b) are often referred to as the data and time step, respectively. They together form a general recursive approach for computing the filtering density $p(x_t|y_{1:t})$, and—as a byproduct—also the one-step-ahead state (2.7b) and observation (2.7c) prediction densities. The forward filtering plays an important role in almost all more advanced state inference techniques presented in the sequel.

The only requirement to perform filtering according to (2.7) is to know the elements of a state-space model, (μ, f, g). Nevertheless, in nonlinear and non-Gaussian state-space models, $p(x_t, y_t|y_{1:t-1})$ and $p(x_t, x_{t-1}|y_{1:t-1})$ usually become highly complicated after a low number of iterations. This makes the posterior density (2.7a)—and the involved integrals (2.7b) and (2.7c)—analytically intractable. The exact solution of the filtering recursion (2.7) can be found in a restricted number of instances, such as discrete-valued state-space models and linear Gaussian state-space models, as noted before. Therefore, in most situations, we need to embrace approximate state inference techniques.

2.2.1 Gaussian Filters

Gaussian filters form a practically important and very often used class of algorithms for approximate state filtering in nonlinear state-space models. The underlying idea of these filters is based on the assumed density filtering [147], where we consider that the filtering density can be approximated by the Gaussian density. The utility of these filters is maximized when the model behaves—at least approximately—in the Gaussian way, and the model nonlinearities are not too severe.

Assumed Density Approximation

Let us consider we have an exact density $p : \mathsf{Z} \to \mathbb{R}_+$ which we intend to approximate by a simpler one $\widehat{p} : \mathsf{Z} \to \mathbb{R}_+$. Assumed density framework seeks an optimal approximate density $p^o : \mathsf{Z} \to \mathbb{R}_+$ as the minimizer of the Kullback-Leibler divergence from the exact density p to a user-defined feasible density $\bar{p} : \mathsf{Z} \to \mathbb{R}_+$,

$$p^o := \operatorname*{argmin}_{\bar{p}\in\mathsf{P}} \mathcal{D}(p, \bar{p}) = \operatorname*{argmin}_{\bar{p}(z)\in\mathsf{P}} \int_{\mathsf{X}} p(z) \log\left(\frac{p(z)}{\bar{p}(z)}\right) dz, \tag{2.8}$$

where P is a set of feasible densities. The feasible density is commonly chosen to be a member of an appropriate parametric family, such as the exponential family [12],

$$\bar{p}(z) := \exp\{\langle \eta(z), \bar{V}\rangle - \log \mathcal{I}(\bar{V})\}, \tag{2.9}$$

which is composed of the inner product $\langle \cdot, \cdot \rangle$ between a function η defined on Z and a statistic $\bar{V}$—both being of appropriate dimensions—and the normalizing factor $\mathcal{I}$. Then, after we set the gradient of $\mathcal{D}(p, \bar{p})$ with respect to the feasible statistic $\bar{V}$ to zero, we obtain

$$\mathsf{E}[\eta(Z)|V^o] = \mathsf{E}[\eta(Z)]. \tag{2.10}$$

The minimum of the optimization problem (2.8) is attained if the expected value w.r.t. the optimal density p^o is equal to the expected value w.r.t. the exact density p. In other words, (2.10) implies that we should compute V^o so that the above expected values are the same. However, this cannot be achieved as the intractability of $\mathsf{E}[\eta(Z)]$ is the reason why we resort to the approximate solution in the first place. Therefore, we should find an approximate statistic $\widehat{V}$ as the nearest one to optimal statistic V^o. This statistic is computed after approximating the expected value on the r.h.s. of (2.10) by an appropriate numerical technique. Then, the statistic $\widehat{V}$ characterizes the sought approximate density, $\widehat{p} \in \mathsf{P}$, which is more crude than the optimal one, p^o, and is the near-optimal solution to the optimization problem (2.8) under the restriction $\bar{p} \in \mathsf{P}$. From (2.10), we see that the assumed density approach allows us to find an optimal solution for matching the moments of the exact density, but we cannot expect to fit the entire shape of this density. Hence, this optimal solution is not ideal but still the best one we can obtain under the parametric family chosen. This technique is also referred to as the moment matching [21].

Principle

The Gaussian filters are suitable for general state-space models in the functional form given by

$$X_t = a(X_{t-1}, W_t), \tag{2.11a}$$

$$Y_t = b(X_t, V_t), \tag{2.11b}$$

where $W_t \in \mathsf{W} \subseteq \mathbb{R}^{n_w}$ and $V_t \in \mathsf{V} \subseteq \mathbb{R}^{n_v}$ are IID mutually independent noise variables—often referred to as the state and observation noise variables, respectively—and $a : \mathsf{X} \times \mathsf{W} \to \mathsf{X}$ and $b : \mathsf{X} \times \mathsf{V} \to \mathsf{Y}$ are measurable functions.

In the state filtering—not necessarily Gaussian filtering—the central objects of interest are the joint densities $p(x_t, x_{t-1}|y_{1:t-1})$ and $p(x_t, y_t|y_{1:t-1})$. The motivation here is that they admit all objects appearing in the filtering recursion (2.7) as the appropriate conditional or marginal densities. As mentioned previously, these terms usually become highly complicated after a number of iterations. The Gaussian filters seek the approximation by means of the assumed density framework. The result of this approach (2.10) states that, as long as we work with densities within an appropriate parametric family, the optimal solution can be obtained by equating the expectations of the optimal and exact densities. The parametric family in the context of the Gaussian filters is—as the name suggests—Gaussian, and we thus need to match the first and second order exact moments with respect to $p(x_t, x_{t-1}|y_{1:t-1})$ and $p(x_t, y_t|y_{1:t-1})$.

Let us consider that $p^o(x_{t-1}|y_{1:t-1}) = \mathcal{N}(x_{t-1}; x^o_{t-1|t-1}, P^o_{t-1|t-1})$ is the optimal filtering density from the previous iteration. The optimization problem (2.8) suggests that $p(x_t, x_{t-1}|y_{1:t-1})$ and $p(x_t, y_t|y_{1:t-1})$ should optimally be approximated by

$$p^o(x_t, x_{t-1}|y_{1:t-1}) = \mathcal{N}\left(\begin{bmatrix} x_{t-1} \\ x_t \end{bmatrix}; \begin{bmatrix} x^o_{t-1|t-1} \\ x^o_{t|t-1} \end{bmatrix}, \begin{bmatrix} P^o_{t-1|t-1} & N^o_{t|t-1} \\ (N^o_{t|t-1})^\top & P^o_{t|t-1} \end{bmatrix}\right), \tag{2.12}$$

where

$$\begin{aligned}
x^o_{t|t-1} &= \int a(x_{t-1}, w_t) p(x_{t-1}|y_{1:t-1}) p(w_t) dx_{t-1} dw_t,\\
P^o_{t|t-1} &= \int [a(x_{t-1}, w_t) - x^o_{t|t-1}][a(x_{t-1}, w_t) - x^o_{t|t-1}]^\top\\
&\qquad\qquad\qquad\qquad\qquad p(x_{t-1}|y_{1:t-1}) p(w_t) dx_{t-1} dw_t,\\
N^o_{t|t-1} &= \int [x_{t-1} - x^o_{t-1|t-1}][a(x_{t-1}, w_t) - x^o_{t|t-1}]^\top p(x_{t-1}|y_{1:t-1}) p(w_t) dx_{t-1} dw_t,
\end{aligned}$$

and

$$p^o(x_t, y_t|y_{1:t-1}) = \mathcal{N}\left(\begin{bmatrix} x_t \\ y_t \end{bmatrix}; \begin{bmatrix} x^o_{t|t-1} \\ y^o_{t|t-1} \end{bmatrix}, \begin{bmatrix} P^o_{t|t-1} & M^o_{t|t-1} \\ (M^o_{t|t-1})^\top & R^o_{t|t-1} \end{bmatrix}\right),$$

where

$$\begin{aligned}
y^o_{t|t-1} &= \int b(x_t, v_t) p(x_t|y_{1:t-1}) p(v_t) dx_t dv_t,\\
R^o_{t|t-1} &= \int [b(x_t, v_t) - y^o_{t|t-1}][b(x_t, v_t) - y^o_{t|t-1}]^\top p(x_t|y_{1:t-1}) p(v_t) dx_t dv_t,\\
M^o_{t|t-1} &= \int [x_t - x^o_{t|t-1}][b(x_t, v_t) - y^o_{t|t-1}]^\top p(x_t|y_{1:t-1}) p(v_t) dx_t dv_t,
\end{aligned}$$

respectively, from which we see that—according to (2.10)—the optimal moments are equal to the exact ones. As discussed before, the computation of these moments is prevented by the problem definition. In this context, the assumed density framework addresses this issue by first adopting the approximate densities

$$\hat{p}(x_t, x_{t-1}|y_{1:t-1}) = \mathcal{N}\left(\begin{bmatrix} x_{t-1} \\ x_t \end{bmatrix}; \begin{bmatrix} \hat{x}_{t-1|t-1} \\ \hat{x}_{t|t-1} \end{bmatrix}, \begin{bmatrix} \hat{P}_{t-1|t-1} & \hat{N}_{t|t-1} \\ \hat{N}^\top_{t|t-1} & \hat{R}_{t|t-1} \end{bmatrix}\right), \tag{2.15}$$

where

$$\begin{aligned}
\hat{x}_{t|t-1} &= \int a(x_{t-1}, w_t) \mathcal{N}(x_{t-1}; \hat{x}_{t-1|t-1}, \hat{P}_{t-1|t-1}) \mathcal{N}(w_t; \mathbf{0}, Q) dx_{t-1} dw_t, &(2.16a)\\
\hat{P}_{t|t-1} &= \int [a(x_{t-1}, w_t) - \hat{x}_{t|t-1}][a(x_{t-1}, w_t) - \hat{x}_{t|t-1}]^\top\\
&\qquad\qquad \mathcal{N}(x_{t-1}; \hat{x}_{t-1|t-1}, \hat{P}_{t-1|t-1}) \mathcal{N}(w_t; \mathbf{0}, Q) dx_{t-1} dw_t, &(2.16b)\\
N^o_{t|t-1} &= \int [x_{t-1} - x^o_{t-1|t-1}][a(x_{t-1}, w_t) - x^o_{t|t-1}]^\top\\
&\qquad\qquad \mathcal{N}(x_{t-1}; \hat{x}_{t-1|t-1}, \hat{P}_{t-1|t-1}) \mathcal{N}(w_t; \mathbf{0}, Q) dx_{t-1} dw_t &(2.16c)
\end{aligned}$$

and

$$\hat{p}(x_t, y_t|y_{1:t-1}) = \mathcal{N}\left(\begin{bmatrix} x_t \\ y_t \end{bmatrix}; \begin{bmatrix} \hat{x}_{t|t-1} \\ \hat{y}_{t|t-1} \end{bmatrix}, \begin{bmatrix} \hat{P}_{t|t-1} & \hat{M}_{t|t-1} \\ \hat{M}^\top_{t|t-1} & \hat{R}_{t|t-1} \end{bmatrix}\right), \tag{2.17}$$

```
Algorithm 8 The Gaussian filter
A. Initial step: ($t = 1$)
    1. Set ($\hat{x}_{1|0}, \hat{P}_{1|0}$) and use it in (2.18) and (2.19) to compute ($\hat{x}_{1|1}, \hat{P}_{1|1}$).
B. Recursive step: ($t = 2, \dots, T$)
    1. Use ($\hat{x}_{t-1|t-1}, \hat{P}_{t-1|t-1}$) in (2.16a) and (2.16b) to compute ($\hat{x}_{t|t-1}, \hat{P}_{t|t-1}$).
    2. Use ($\hat{x}_{t|t-1}, \hat{P}_{t|t-1}$) in (2.18) and (2.19) to compute ($\hat{x}_{t|t}, \hat{P}_{t|t}$).
```

where

$$\hat{y}_{t|t-1} = \int b(x_t, v_t)\mathcal{N}(x_t; \hat{x}_{t|t-1}, \hat{P}_{t|t-1})\mathcal{N}(v_t; \mathbf{0}, R)dx_t dv_t, \tag{2.18a}$$

$$\hat{M}_{t|t-1} = \int [x_t - \hat{x}_{t|t-1}][b(x_t, v_t) - \hat{y}_{t|t-1}]^\top \mathcal{N}(x_t; \hat{x}_{t|t-1}, \hat{P}_{t|t-1})\mathcal{N}(v_t; \mathbf{0}, R)dx_t dv_t, \tag{2.18b}$$

$$\hat{R}_{t|t-1} = \int [b(x_t, v_t) - \hat{y}_{t|t-1}][b(x_t, v_t) - \hat{y}_{t|t-1}]^\top \mathcal{N}(x_t; \hat{x}_{t|t-1}, \hat{P}_{t|t-1})\mathcal{N}(v_t; \mathbf{0}, R)dx_t dv_t; \tag{2.18c}$$

and then approximating these expectations with a suitably chosen method. The approximate filtering density is simply obtained by writing the conditional density of (2.17),

$$\hat{p}(x_t|y_{1:t}) = \mathcal{N}(x_t; \hat{x}_{t|t}, \hat{P}_{t|t}),$$

where

$$K = \hat{M}_{t|t-1}\hat{R}_{t|t-1}^{-1}, \tag{2.19a}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K(y_t - \hat{y}_{t|t-1}), \tag{2.19b}$$

$$\hat{P}_{t|t} = \hat{P}_{t|t-1} - K\hat{R}_{t|t-1}K^\top, \tag{2.19c}$$

see Lemma A.8 for details. The Gaussian filter is now summarized in Algorithm 8. A similar approach can also be used to derive Student's t filter, see Lemma A.9 for the necessary conditional and marginal densities, in order to increase robustness against the outliers [185].

Cubature Rules and Different Forms of Gaussian Filters

Algorithm 8 is generic and can assume different forms based on the character of the integrated functions. For example, when $a(X, W) = AX + W$ and $b(X, V) = CX + V$, we recover the basic Kalman filter. In such a case, we propagate the exact moments and—if the noise variables W and V are Gaussian—the optimum of (2.8) is zero. In a different scenario, if we choose to expand the nonlinear functions a and b into the Taylor series of a certain order, then we obtain the extended Kalman filter of the corresponding order. The design of extended Kalman filters usually requires some effort when computing associated partial derivatives. This has motivated the development of derivative-free algorithms.

A possible approach is to approximate the integrals (2.16) and (2.18) by a set of Monte-Carlo or quasi-Monte Carlo samples [120, 87]. In the Monte Carlo case, as presented in Theorem 1.1, when the number of particles tends to infinity, we attain the exact values of the intractable moments. Consequently, we obtain the optimal approximation (2.12) and thus diminish the error arising from approximating the moment integrals. However, even then, this optimal solution of the optimization problem (2.8) makes the Kullback-Leibler divergence nonzero, which is caused by the Gaussian assumptions on the feasible density. This is the key example, showing two sources of errors, which justifies the assumed density construction presented above, a feature which is commonly missing (or not made obvious) in the literature.

The derivative-free integration methods can also be constructed based on polynomial interpolation. The divided difference Kalman filter [162] and central difference Kalman filter [92] are typical representatives.

Importantly, (2.16) and (2.18) form a specific class of Gaussian integrals for which there exists a variety of quadrature rules that are commonly referred to as the Gaussian quadrature rules [201]. These are constructed so that they are exact for monomials of a certain degree. For instance, the Gauss-Hermite quadrature rule approximates the integrals based on evaluation points generated as the roots of Nth-order Hermite polynomial. The rule is exact for monomials up to order $2N-1$. The advantage is that the roots can simply be computed as the eigenvalues of a certain tridiagonal matrix [81]. The application of this type of quadrature rule in Algorithm 8 results in the Gauss-Hermit Kalman filter [92, 9], which has the computational complexity scaling with $\mathcal{O}(N)$ operations. The main disadvantage of this approach is that the Gauss-Hermit *cubature* rules are constructed as the product rules. The computational complexity then scales exponentially with the dimension n according to $\mathcal{O}(N^n)$, which can be extremely demanding in high-dimensional scenarios. The distinguishing feature of both the Gaussian density and the integration domain in (2.16) and (2.18) is their symmetry. Therefore, we can exploit a special class of quadrature rules which are referred to as fully symmetric quadrature rules [41]. The key idea here is to generate the evaluation points based on the fully symmetric generators. The evaluation points are computed as the zeros of an orthogonal polynomial of degree $k+1$ for $k \in \mathbb{N}_+$. These rules are exact for monomials of degree $2k+1$. The minimal number of evaluation points, and thus the computational complexity, of the fully symmetric cubature rules scales according to $\mathcal{O}((2n)^k/k!)$, see [150] for details. To obtain the evaluation points, we are required to solve a system of nonlinear equations, which can be difficult when requiring high precision and usually restricts us up to degree 11. Adopting this principle with degree 3 in Algorithm 8 leads to the cubature Kalman filter [225]. A different way of deriving the cubature Kalman filter is to transform the integrals into the spherical-radial coordinate system [10]. This principle has been utilized in [95] to generalize the cubature Kalman filter for monomials of an arbitrary degree. An interesting connection is that the unscented Kalman filter [102] can be seen as a generalization of the degree 3 cubature Kalman filter [225].

Recently, there has been an increased interest in approximating integrals based on Gaussian process quadrature rules that take advantage of the Gaussian processes [179] to interpret the integration as Bayesian estimation problem, rather than seeing it from the classical—frequentist—view point of the basic Gaussian quadrature rules. The Gaussian process quadrature rules provide us with a tool for representing our beliefs about the result of the integration operation. For the relation between these more advanced and classical quadrature rules, see [191]. The main disadvantage of the Gaussian process quadrature rules is that they normally require us to compute the inverse of a matrix with the dimension given by the number of function evaluations N, which scales painfully with $\mathcal{O}(N^3)$ operations (per single dimension). Therefore, the use of product rules in this context is rather unthinkable. This has recently stimulated a substantial research effort towards finding efficient ways for spreading the points throughout the space. The methods can be preferred in the sense that they can be applied to a broader class of integrated functions compared to Gaussian quadrature rules.

Although there is a large body of work on various forms of integration in the context of Gaussian filtering, the weak Gaussian assumptions can prove to be inappropriate in some practical situations. The well-known disadvantage of the Gaussian filters is that they experience difficulties with approximating complex densities.

To see this statement on a specific case, we apply the extended Kalman filter [2], unscented Kalman filter [102], and Gauss-Hermite Kalman filter [92] to Examples 2.1-2.3 and present the resulting state estimation performance in the first, second, and third row of Fig. 2.2, respectively. The initial statistics of all the compared filters are $\hat{x}_{1|0} = 0$ and $\hat{P}_{1|0} = 5$. The number of sigma points of the Gauss-Hermite Kalman filter is 20, and the scaling parameter (κ, see [102]) of the unscented Kalman filter is 2.

2.2.2 Particle Filters

Various particle filtering algorithms can be seen as special cases of the generic SMC procedure presented in Algorithm 2. In this section, we briefly discuss some of its features in the context of state-space models.

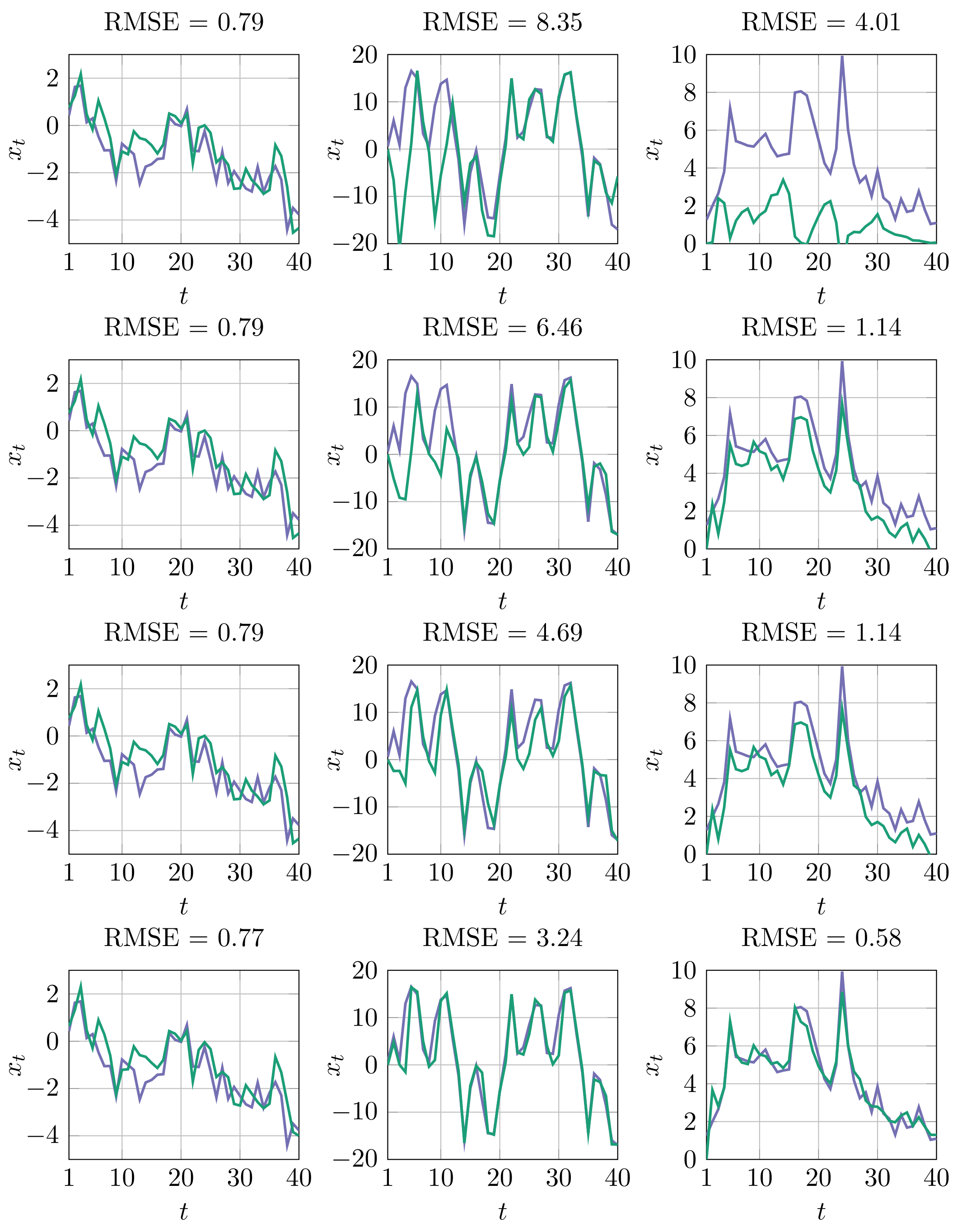

Fig. 2.2: The true state trajectory (—) and its estimate (—) versus the number of observations for the extended Kalman filter [2] (first row), unscented Kalman filter [102] (second row), Gauss-Hermite Kalman filter [92] (third row), and bootstrap particle filter [57] (fourth row). The columns correspond to the models given in Example 2.1 with $(\mu_w, \sigma_w^2, \mu_v, \sigma_v^2) = (1, 1, 1, 1)$ (first column), Example 2.2 with $(\mu_w, \sigma_w^2, \mu_v, \sigma_v^2) = (1, 1, 1, 1)$ (middle column), and Example 2.3 with $(\alpha, \beta, \mu_v, \sigma_v^2) = (1, 1, 1, 1)$ (last column). The initial state is distributed according to $\mu(x_1) = \mathcal{N}(x_1; 0, 5)$ in all the cases.

The Particle Filter

The standard particle filter is simply obtained from the generic SMC framework of Section 1.6 by setting $\pi_t(x_{1:t}) \coloneqq p(x_{1:t}|y_{1:t})$ and $\gamma_t(x_{1:t}) \coloneqq p(x_{1:t}, y_{1:t})$ for $t = 1, \ldots, T$. Then, under the conditional independence assumptions of the state-space models, the unnormalized importance weight function (1.49) becomes

$$v_t(x_{t-1:t}) \coloneqq \frac{g(y_t|x_t)f(x_t|x_{t-1})}{m_t(x_t|x_{t-1})},$$

for $t = 2, \ldots, T$, and

$$v_1(x_1) \coloneqq \frac{g(y_1|x_1)\nu(x_1)}{m_1(x_1)},$$

for $t = 1$. The procedure follows exactly the steps of Algorithm 2. We will refer to this type of particle filter as the sequential importance resampling (SIR) filter [56]. This method approximates the sequence of target densities, $(p(x_{1:t}|y_{1:t}))_{t=1}^T$, by the corresponding sequence of empirical measures, $(p^N(dx_{1:t}|y_{1:t}))_{t=1}^T$, represented by the weighted particle systems, $(W_t^{1:N}, X_{1:t}^{1:N})_{t=1}^T$. Thus, the SIR filter addresses the forward smoothing task. However, as discussed in Section 1.6, the path degeneracy problem makes the approximation $p^N(dx_{1:t}|y_{1:t})$ unreliable as the number of unique particles of the marginals $p^N(dx_m|y_{1:t})$ is low for $m \ll t$. Moreover, implementing the algorithm in this way imposes ever increasing memory requirements. The approximation of the filtering density (2.7a) is given by the marginal measure $p^N(dx_t|y_{1:t})$, which is simply obtained by discarding the past state trajectories, $X_{1:t-1}^{1:N}$. This avoids the need for increasing memory. The key feature, however, is that the diversity of the current set of particles $X_t^{1:N}$ is unreduced, thus providing reliable estimates of the associated quantities of interest.

Algorithm 2 performs the resampling operation at each iteration. Alternatively, triggering the resampling only when the effective sample size decreases below certain value can lead to a particle filter which suffers less from the path degeneracy problem. The effective sample size, see Section 1.3.1, achieves values close to N when the variance of the importance weights is low, motivating the use of variance reduction techniques. Similarly as before, we will refer to this type of particle filter as the sequential importance sampling and resampling (SISR) filter.

To use these particle filters, we have to specify—in addition to state-space model densities (μ, f, g)—the sequence of the proposal densities $(m_t)_{t=1}^T$. The choice of the proposal density can have a significant impact on the variance of the importance weights, as discussed in Section 1.4. The optimal proposal density [57], which minimizes the variance of the importance weights in the context of state-space models, is written as

$$m_t(x_t|x_{t-1}) \coloneqq \frac{g(y_t|x_t)f(x_t|x_{t-1})}{p(y_t|x_{t-1})}. \tag{2.20}$$

This proposal allows the sampling mechanism to take advantage of information provided by the current observation Y_t and the associated model g, which can provide us with samples $X_t^{1:N}$ that are distributed in more interesting parts of X. For the optimal proposal density, the unnormalized importance function becomes $p(y_t|x_{t-1})$, which is advantageously independent of x_t. Unfortunately, (2.20) is intractable in most practical situations and the approximations are needed. A popular but sub-optimal choice of the proposal density, which leads to higher variance of the importance weights, is to simply use the state transition model $m_t(\cdot|x_{t-1}) := f(\cdot|x_{t-1})$. The associated particle filter is then commonly referred to as the bootstrap particle filter [84]. When the observations are not too informative, this choice can lead to satisfactory results. This sampling mechanism is most commonly applied for its simplicity, especially in scenarios where f is intractable or hard to evaluate. Even this sub-optimal choice can provide better performance than the standard procedures based on Gaussian filters, especially when the nonlinearities are severe and/or the state space model is non-Gaussian.

To demonstrate this assertion, we implement the bootstrap particle filter [84] in the context of Examples 2.1-2.3, with the number of particles being set to $N = 100$. The results are presented in the last row of Fig. 2.2.

The Auxiliary Particle Filter

The auxiliary particle filter is another instance of the generic SMC framework of Section 1.6. The key idea behind this type of particle filter is to take advantage of the current observation Y_t to reweight particles before entering the resampling step. The particles that are consistent with this additional observation information have an increased chance to proceed to the next iteration, potentially also increasing the diversity of the particle system and thus counteracting the path degeneracy problem. The original formulation of the auxiliary particle filter involves defining a specific target density on an extended space and utilizing auxiliary variables [173]. However, it was later demonstrated in [99] that this method can be seen as a specific example of the generic SMC procedure when using $\pi_t(x_{1:t}) := p(x_{1:t}|y_{1:t+1})$ and $\gamma_t(x_{1:t}) := p(x_{1:t}, y_{1:t+1})$ for $t = 1, \ldots, T$; with the auxiliary variables playing the role of the ancestor indices. In this case, the unnormalized importance weight function (1.49) is defined by

$$v_t(x_{t-1:t}) := \frac{g(y_t|x_t)f(x_t|x_{t-1})p(y_{t+1}|x_t)}{p(y_t|x_{t-1})m_t(x_t|x_{t-1})},$$

for $t = 2, \ldots, T$, and

$$v_1(x_1) := \frac{g(y_1|x_1)\nu(x_1)p(y_2|x_1)}{p(y_t)m_1(x_1)},$$

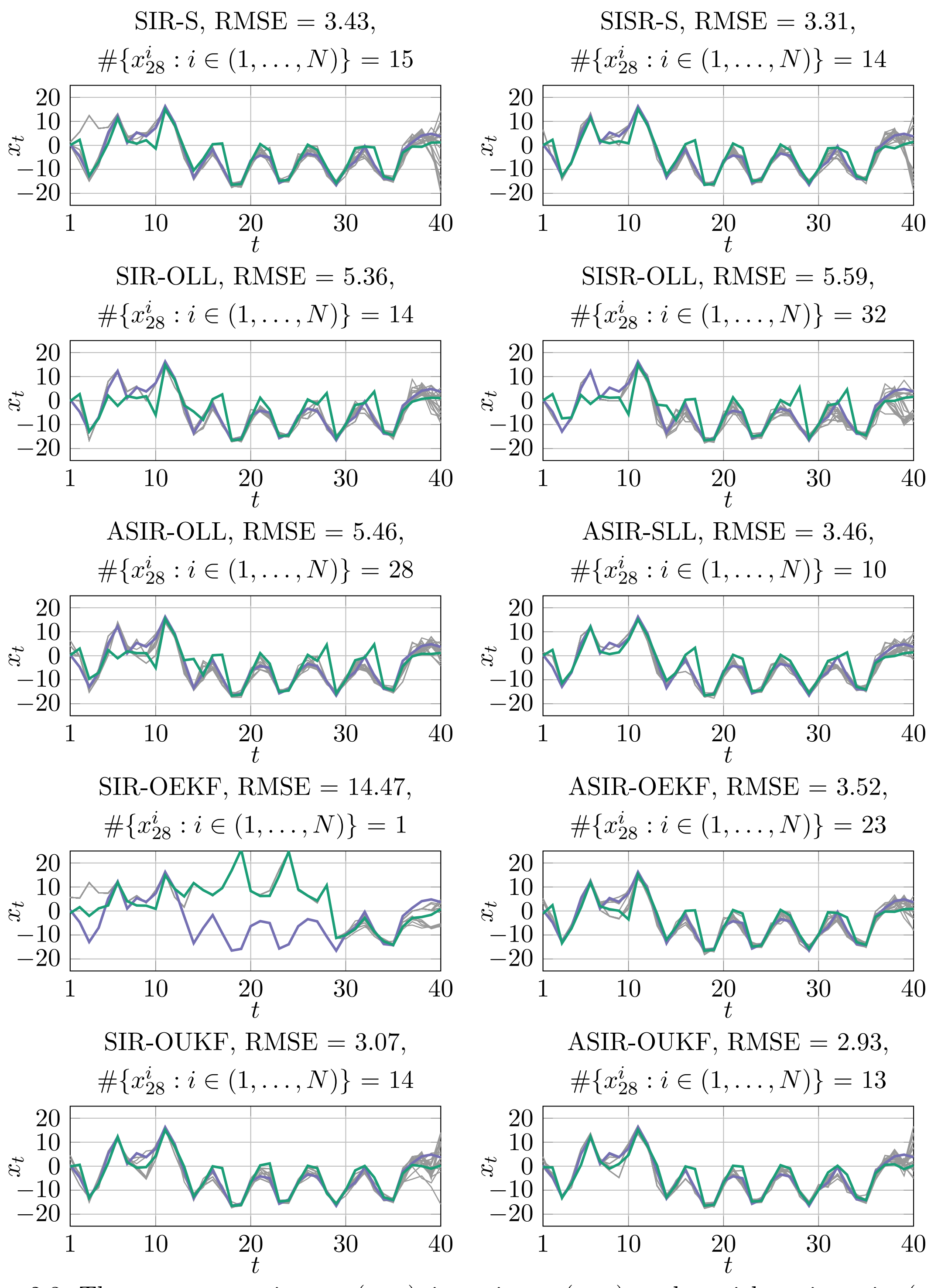

Fig. 2.3: The true state trajectory (—), its estimate (—), and particle trajectories (—) of the SIR, SISR, ASIR, and ASISR filters for sub-optimal (S) and optimal (O) proposal densities. The optimal density is approximated by local linearization (LL) [54], extended Kalman filter (EKF), and unscented Kalman filter (UKF) [209]. We consider Example 2.2 with $(\mu_w, \sigma_w^2, \mu_v, \sigma_v^2) = (1, 1, 1, 1)$ and $\mu(\cdot) = \mathcal{N}(\cdot; 0, 1)$. The particle filters run with $N = 400$ and $N_{\text{th}} = N/3$.

for $t = 1$. To facilitate practical implementation, the computation of these weights is usually split into two stages so that only Y_t is used during one iteration of the algorithm, see, e.g., [172] for details. Note that for $p(y_t|x_{t-1}) := 1$, we recover the standard particle filter. The auxiliary particle filter can be implemented in the SIR and SISR settings, which are here referred to as ASIR and ASISR, respectively. The proposal density is chosen in the same way as discussed with the standard particle filter. For the optimal proposal density, the unnormalized importance weights become equal, and the filter is then commonly referred to as being fully adapted [173].

The interpretation of the auxiliary particle filter as a special case of the generic SMC procedure allows us to apply a wide range of theoretical results given in [154]. Empirical evidence often suggests that the auxiliary particle filter outperforms the particle filter. However, this is not always the case, as it is obvious from the comparison of the asymptotic variances of the associated estimators [99].

We present Fig. 2.3 to investigate the impact of the above discussed particle filter implementations and various approximations of the optimal proposal density on the path degeneracy. We can observe that the ASIR with the optimal proposal density approximated by the local linearization approach is less affected by the path degeneracy problem. However, the improvement is rather weak.

2.3 Forward-Filtering Backward-Smoothing

The purpose of the forward-filtering backward-smoothing is to compute the joint state posterior density, $p(x_{1:T}|y_{1:T})$. In other words, the aim is to use all available information not to only estimate the current state—as in the case of the state filtering—but also the past states. The fact that we use additional information makes resulting state trajectory estimates more precise, hence the name smoothing. In other words, every past state, X_t, where $1 \leq t < T$, benefits from the future observations and therefore has increased precision compared to the case of using only the information up to the current time step t. This obviously does not hold for the final state, X_T, as it cannot benefit from any future observations.

The joint state posterior density $p(x_{1:T}|y_{1:T})$ can be computed based on the backward recursion given by

$$p(x_{t:T}|y_{1:T}) = p(x_t|x_{t+1}, y_{1:t})p(x_{t+1:T}|y_{1:T}), \tag{2.21}$$

where we apply $p(x_t|x_{t+1}, y_{1:t}) = p(x_t|x_{t+1}, y_{1:T})$. That is, $y_{t+1:T}$ does not provide any further information about X_t when x_{t+1} is given. This results from the application of the Markov property of the transition kernel f. Here,

$$p(x_t|x_{t+1}, y_{1:t}) = \frac{f(x_{t+1}|x_t)p(x_t|y_{1:t})}{\int_{\mathsf{X}} f(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t} \tag{2.22}$$

is the backward transition kernel, which is generally time inhomogeneous. We see from (2.22) that—to facilitate computation of (2.21)—we need the sequence of the state filtering densities $(p(x_t|y_{1:t}))_{t=1}^T$ and the forward transition kernel f. Therefore, we first need to run the forward filtering recursion (2.7). The backward recursion (2.21) is then initialized with the filtering density from the final iteration, $p(x_T|y_{1:T})$. The computational procedures based on this principle are then commonly associated with the name *forward-filtering backward-smoothing.* Obviously, the backward smoothing recursion is computationally more expensive than the filtering one. An important implication of the backward smoothing is that it facilitates a more precise estimation of the initial state X_1.

The recursion for computing the marginal smoothing destiny $p(x_t|y_{1:T})$ can simply be obtained from (2.21) by marginalizing over $X_{t+1:T}$,

$$p(x_t|y_{1:T}) = \int_{\mathsf{X}} p(x_t|x_{t+1}, y_{1:t})p(x_{t+1}|y_{1:T})dx_{t+1}. \tag{2.23}$$

Note we can also define the fixed-interval smoothing density in a similar way.

The backward transition kernel (2.22) is generally intractable. Hence, we review some approximation strategies for computing (2.21) and (2.23).

2.3.1 Gaussian Smoothers

The Gaussian forward-filter backward-smoother [188] aims at computing the marginal smoothing density (2.23) for general nonlinear and non-Gaussian state-space models given by the functional form (2.11). Similarly to the Gaussian filtering, the underlying idea of this approach lies in the assumed density framework discussed in Section 2.2.1, with all advantages and disadvantages mentioned there.

The key object in devising the smoother is the joint marginal smoothing density

$$p(x_t, x_{t+1}|y_{1:T}) = p(x_t|x_{t+1}, y_{1:t})p(x_{t+1}|y_{1:T}), \tag{2.24}$$

whose integral over X_{t+1} is (2.23). The complexity of this joint density grows as we proceed backwards in time whenever dealing with state-space models outside the linear Gaussian or discrete-valued structures. The main reason is that the backward transition kernel (2.22) is intractable. We therefore use the assumed density approach to find its approximation.

Consider we have a sequence of filtering densities $(\mathcal{N}(x_t; \hat{x}_{t|t}, \hat{P}_{t|t}))_{t=1}^T$ computed by one of the previously discussed Gaussian filters. The construction of the backward transition kernel (2.22) generally follows from the joint density $p(x_t, x_{t+1}|y_{1:t}) = f(x_{t+1}|x_t)p(x_t|y_{1:t})$. If we simply substitute the Gaussian filtering densities in this formula, we do not obtain any tractable solution due to the incompatibility of $f(x_{t+1}|x_t)$ and $p(x_t|y_{1:t})$. The assumed density framework suggests to approximate

```
Algorithm 9 The Gaussian smoother
A. Initial step: $(t = T)$
    * Set $(\tilde{x}_T, \tilde{P}_T)$ to $(\hat{x}_{T|T}, \hat{P}_{T|T})$.
B. Recursive step: $(t = T-1, \dots, 1)$
    * Use $(\hat{x}_{t|t}, \hat{P}_{t|t})$ and $(\tilde{x}_{t+1}, \tilde{P}_{t+1})$ in (2.16) and (2.27) to compute $(\tilde{x}_t, \tilde{P}_t)$.
```

the exact density $p(x_t, x_{t+1}|y_{1:t})$ by the optimal approximate density (2.12). However, as discussed before, the moments of this optimal solution are given by the exact moments, and we therefore adopt the coarser approximation (2.15). The conditional density of (2.15) corresponds to the sought Gaussian approximation of the backward transition kernel (2.22),

$$\hat{p}(x_t|x_{t+1}, y_{1:t}) = \mathcal{N}(x_t; \mu_t, \Sigma_t), \tag{2.25}$$

where

$$L = \hat{N}_{t+1|t}\hat{P}_{t+1|t}^{-1}, \tag{2.26a}$$

$$\mu_t = \hat{x}_{t|t} + L(x_{t+1} - \hat{x}_{t+1|t}), \tag{2.26b}$$

$$\Sigma_t = \hat{P}_{t|t} - L\hat{P}_{t+1|t}L^\top, \tag{2.26c}$$

which follows from applying Lemma A.8. We continue by assuming $\hat{p}(x_{t+1}|y_{1:T}) = \mathcal{N}(x_{t+1}; \tilde{x}_{t+1}, \tilde{P}_{t+1})$, then after substituting this density along with (2.25) into (2.24), we obtain

$$\hat{p}(x_t, x_{t+1}|y_{1:T}) = \mathcal{N}\left(\begin{bmatrix} x_{t+1} \\ x_t \end{bmatrix}; \begin{bmatrix} \tilde{x}_{t+1} \\ \mu_t \end{bmatrix}, \begin{bmatrix} \tilde{P}_{t+1} & \tilde{P}_{t+1}L^\top \\ L\tilde{P}_{t+1} & L\tilde{P}_{t+1}L^\top + \Sigma_t \end{bmatrix}\right).$$

Finally, after using Lemma A.8, the marginal smoothing density becomes

$$\hat{p}(x_t|y_{1:T}) = \mathcal{N}(x_t; \tilde{x}_t, \tilde{P}_t),$$

with

$$L = \hat{N}_{t+1|t}\hat{P}_{t+1|t}^{-1}, \tag{2.27a}$$

$$\tilde{x}_t = \hat{x}_{t|t} + L(\tilde{x}_{t+1} - \hat{x}_{t+1|t}), \tag{2.27b}$$

$$\tilde{P}_t = \hat{P}_{t|t} + L(\tilde{P}_{t+1} - \hat{P}_{t+1|t})L^\top. \tag{2.27c}$$

We can now summarize this Gaussian smoother in Algorithm 9.

As in the case of Gaussian filtering, various forms of this algorithm are obtained depending on a concrete approximation of the integrals (2.16), see the discussion on various forms of the cubature rules in Section 2.2.1. In particular, if the model (2.11) is linear and Gaussian, we recover the standard Rauch-Tung-Striebel smoother [180].

In the first row of Fig. 2.4, we apply the Gauss-Hermite smoother [188] to Examples 2.1-2.3, with the number of sigma points and initial statistics of the forward filtering sweep, $(\hat{x}_{1|0}, \hat{P}_{1|0})$, being set to 20 and $(0, 5)$, respectively.

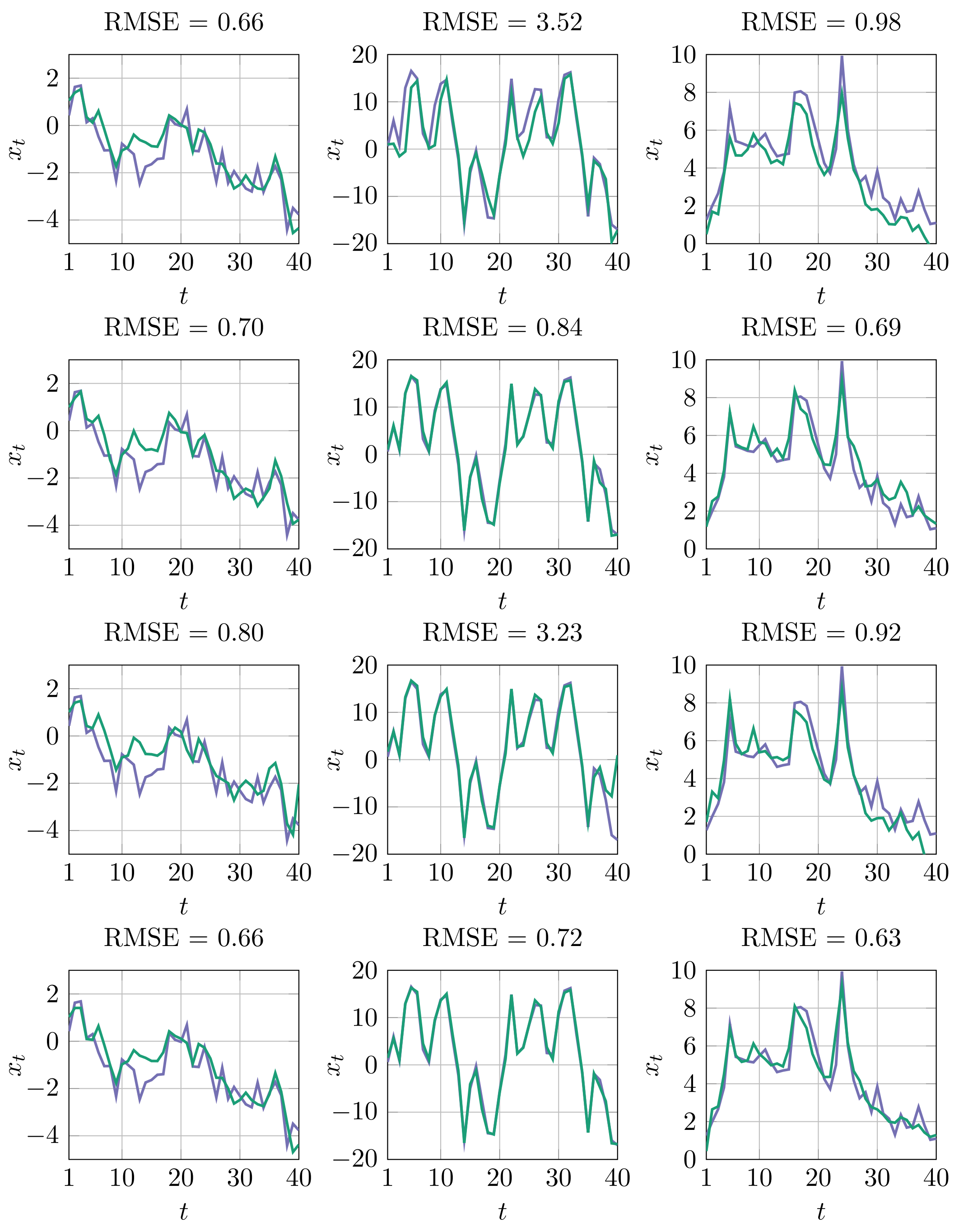

Fig. 2.4: The true state trajectory (—) and its estimate (—) versus the number of observations for the Gauss-Hermite smoother [188] (first row), forward-filter backward-simulator [80] (second row), two-filter particle smoother [26] (third row), and particle Gibbs with ancestor sampling kernel-based smoother [132] (fourth row). The columns correspond to the models given in Example 2.1 with $(\mu_w, \sigma_w^2, \mu_v, \sigma_v^2) = (1, 1, 1, 1)$ (first column), Example 2.2 with $(\mu_w, \sigma_w^2, \mu_v, \sigma_v^2) = (1, 1, 1, 1)$ (middle column), and Example 2.3 with $(\alpha, \beta, \mu_v, \sigma_v^2) = (1, 1, 1, 1)$ (last column). The initial state is distributed according to $\mu(x_1) = \mathcal{N}(x_1; 0, 5)$ in all the cases.

2.3.2 Particle Smoothers

The forward-filter backward-simulator [80] is a particular instance of the generic backward simulation approach presented in Section 1.7 with $\pi_T(x_{1:T}) \coloneqq p(x_{1:T}|y_{1:T})$ and $\gamma_T(x_{1:T}) \coloneqq p(x_{1:T}, y_{1:T})$. Consider that we have already applied one of the particle filters discussed in Section 2.2.2 to produce the sequence of the forward filtering distributions, $(p^N(dx_t|y_{1:t}))_{t=1}^T$. The forward-filter backward-simulator utilizes these distributions to approximate the backward transition kernel (2.22) by

$$p^N(dx_t|\tilde{x}_{t+1}, y_{1:t}) = \sum_{i=1}^{N} W_{t|T}^i \delta_{X_t^i}(dx_t), \tag{2.28}$$

where

$$W_{t|T}^i \propto W_t^i f(\tilde{x}_{t+1}|X_t^i).$$

The backward-simulator then samples from this approximate backward transition kernel in the same way as presented in Algorithm 3.

This method realizes sampling according to the recursion 2.21, thus targeting the joint state posterior density $p(x_{1:T}|y_{1:T})$. To approximate the marginal smoothing density (2.23) or the joint marginal smoothing density (2.24), we only need to discard the superfluous particles from the empirical distribution (1.56).

Note that the backward kernel does not depend on the full trajectory $X_{1:t}$. Therefore, we can use only the filtering distributions that are represented by the weighted particle systems $(\mathbf{W}_t, \mathbf{X}_t)_{t=1}^T$ with unreduced particle diversity. The Markov property of f is thus of key importance here since the backward sampling based on (2.28) overcomes the difficulties associated with the path degeneracy problem. As discussed in Section 1.7, this does not hold for the general backward transition kernel (1.55) which requires the full trajectories, making the kernel poorly approximated. Consequently, applying the Rao-Blackwellization in the context of the backward simulation can be difficult.

The backward simulator preserves the $\mathcal{O}(TMN)$ computational complexity of the generic procedure in Algorithm 3. However, in the context of state-space models, the rejection sampling-based implementation can be used to reduce the computational requirements to scale with $\mathcal{O}(TN)$ operations [53]. In particular, when restricting to situations where we compute expectations of a test function with a certain convenient structure, such as the smoothed additive functionals [30], the implementation can be made in the purely online manner [48]. The computational complexity of this forward-only implementation scales with the original $\mathcal{O}(TMN)$ operations. It was proposed in [164], that this can also be improved in the sense of the rejection sampling as presented in [53], thus leading to the $\mathcal{O}(TN)$ computational burden. These improvements are commonly used to perform the online maximum likelihood parameter inference in state-space models.

The accuracy of the approximate joint state posterior distribution (1.56) strongly depends on the quality of the approximate backward transition kernel (2.28). We can compute a precise approximation of this kernel if we choose a high number of forward particles N. Then, empirical evidence often suggests that it is enough to use a number of backward particles M which is markedly lower than N, to have reasonable approximations of desired quantities. The forward-filter backward-simulator provides strongly consistent and asymptotically Gaussian [49, 53] estimates when both M and N tend to infinity.

The second row of Fig. 2.4 demonstrates the forward-filter backward-simulator [80] on Examples 2.1-2.3. The forward filtering sweep is implemented in the bootstrap proposal setting. The number of forward and backward particles is set to $N = 100$ and $M = 10$, respectively.

2.4 Backward Information Filtering

The goal of the backward information filtering [148] is to compute the joint density of the observation sequence—from the current to the final time step—given the current state, $p(y_{t:T}|x_t)$. The time-reverse character of the backward information filtering predetermines its applicability to offline scenarios only. Expect the basic settings, such as linear Gaussian state-space models, the backward information filter is rather intricate to implement.

The backward information filtering density is generally computed as the marginal of the joint density of the states and observations from T to t, which factorizes in the same way as in (2.2), that is, $p(y_{t:T}|x_t) \propto \int p(y_{t:T}, x_{t:T})dx_{t+1:T}$. Consequently, under the conditional independence assumptions of state-space models, the backward information filtering recursion is

$$p(y_{t:T}|x_t) = g(y_t|x_t)p(y_{t+1:T}|x_t), \tag{2.29a}$$

$$p(y_{t+1:T}|x_t) = \int_{\mathsf{X}} f(x_{t+1}|x_t)p(y_{t+1:T}|x_{t+1})dx_{t+1}. \tag{2.29b}$$

Analogously to the forward filtering, (2.29a) and (2.29b) are referred to as the data and time step, respectively. At the initial step $t = T$, we start the computation of the recursion (2.29) with $p(y_T|x_T) = g(y_T|x_T)$. Note that the backward filtering recursion does not require knowledge of the initial density μ.

For analytically tractable cases, $p(y_{t:T}|x_t)$ can be computed under a closed-form solution. However, for general nonlinear and non-Gaussian state-space models, we commonly need to resort to approximate techniques, such as the Gaussian or SMC-based methods. In these situations, the main difficulty with implementing the backward information filter consists in that $p(y_{t:T}|x_t)$ is not a bounded function with respect to x_t, and therefore its integral may not be finite. In other words, $p(y_{t:T}|x_t)$ is not a probability density function over x_t. This has led to the formulation of the generalized backward information filter [26]. To ensure that $p(y_{t:T}|x_t)$ is integrable with respect to x_t, the generalized filter introduces an artificial prior density $\xi_t(x_t)$ to enable formulation of an auxiliary backward filtering and prediction densities, $\tilde{p}(x_t|y_{t:T}) \propto p(y_{t:T}|x_t)\xi_t(x_t)$ and $\tilde{p}(x_t|y_{t+1:T}) \propto p(y_{t+1:T}|x_t)\xi_t(x_t)$, respectively, which are properly normalized and thus allow us to apply the approximate techniques. The auxiliary densities can then be used to form the generalized backward information filtering recursion as

$$\tilde{p}(x_t|y_{t:T}) \propto g(y_t|x_t)\tilde{p}(x_t|y_{t+1:T}), \tag{2.30a}$$

$$\tilde{p}(x_t|y_{t+1:T}) = \int_{\mathsf{X}} \tilde{p}(x_{t+1}|y_{t+1:T}) \frac{f(x_{t+1}|x_t)\xi_t(x_t)}{\xi_{t+1}(x_{t+1})} dx_{t+1}. \tag{2.30b}$$

For a detailed discussion on the selection of the artificial prior density, see [26].

The backward information filter is most often utilized as a part of more advanced state inference techniques, rather than a purely standalone procedure. The methods of this type are exclusively related to the two-filter smoothing [69, 24, 113], including the two-filter smoother itself, Rao-Blackwellized forward-filter backward-simulator [131], particle Gibbs with ancestor sampling [132], etc.

For details regarding implementation of the generalized backward information filter in the context of the Gaussian and particle approximations, see [25, 26, 131]. The particle-based implementation of the generalized backward information filter approximates a sequence of the artificial backward filtering densities $(\tilde{p}(x_{t:T}|y_{t:T}))_{t=1}^{T}$ by the corresponding sequence of empirical measures $(\tilde{p}^N(dx_{t:T}|y_{t:T}))_{t=1}^{T}$. For how to select the optimal proposal density, which minimizes the variance of the unnormalized importance weights in the context of the particle-based implementation of the generalized backward information filter, see [26]. The auxiliary backward information particle filter is formulated in [66]. An experimental evaluation presented in [25] demonstrates that the particle implementation of the generalized backward information filter can provide improved estimation accuracy over the forward particle filter with the optimal proposal density approximated by the unscented transform.

2.5 Backward-Filtering Forward-Smoothing

The backward-filtering forward-smoothing is an alternative approach for computing the joint state posterior density, $p(x_{1:T}|y_{1:T})$. The name suggests that this approach is a time-reversed version of the forward-filtering backward-smoothing.

The joint state posterior density $p(x_{1:T}|y_{1:T})$ is computed with a forward recursion given by

$$p(x_{1:t}|y_{1:T}) = p(x_t|x_{t-1}, y_{t:T})p(x_{1:t-1}|y_{1:T}), \tag{2.31}$$

where we use $p(x_t|x_{t-1}, y_{1:T}) = p(x_t|x_{t-1}, y_{t:T})$. Thus, based on the Markov property of the transition kernel f, $y_{1:t-1}$ provide no further information about X_t given x_{t-1}. We refer to $p(x_t|x_{t-1}, y_{t:T})$ as the forward transition kernel, for which it holds that

$$p(x_t|x_{t-1}, y_{t:T}) = \frac{p(y_{t:T}|x_t)f(x_t|x_{t-1})}{\int_{\mathsf{X}} p(y_{t:T}|x_t)f(x_t|x_{t-1})dx_t}. \tag{2.32}$$

The presence of $y_{t:T}$ makes this kernel time inhomogeneous. We see from (2.32) that the sequence of backward information filtering densities $(p(y_{t:T}|x_t))_{t=1}^{T}$ is required to enable the computation of (2.31). Hence, it is necessary to first apply the backward information filtering recursion (2.29). The initial step of (2.31) then computes $p(x_1|y_{1:T}) \propto p(y_{1:T}|x_1)\mu(x_1)$, where $p(y_{1:T}|x_1)$ is the backward information filtering density from the final step (taking the time reverse perspective). The fact that we first apply the backward filtering recursion (2.29) and then the forward smoothing recursion (2.31) justifies the name *backward-filtering forward-smoothing*.

To compute the marginal smoothing density $p(x_t|y_{1:T})$, we need to marginalize (2.31) over $X_{1:t-1}$, which yields

$$p(x_t|y_{1:T}) = \int_{\mathsf{X}} p(x_t|x_{t-1}, y_{t:T})p(x_{t-1}|y_{1:T})dx_{t-1}. \tag{2.33}$$

A more commonly used—and equivalent—form of (2.33) is given by

$$p(x_t|y_{1:T}) = \frac{p(y_{t:T}|x_t)p(x_t|y_{1:t-1})}{p(y_{t:T}|y_{1:t-1})}, \tag{2.34}$$

where we see that one needs to compute $p(y_{t:T}|x_t)$ with the backward information filtering recursion (2.29) and $p(x_t|y_{1:t-1})$ with the forward filtering recursion (2.7). Hence, the computational strategies following this idea are commonly known under the name *two-filter smoothing*. Interestingly, the above construction of (2.34) reveals that the two-filter smoothing is a special case of the backward-filtering forward-smoothing. The consequence of marginalization (2.33) is that it is no longer important whether we start computations in the forward or backward direction.

2.5.1 Two-Filter Gaussian Smoothers

Although a rigorous derivation of the Gaussian two-filter smoothers was provided in [25], it seems that these algorithms have not yet been as widely deployed as the forward-filters backward-smoothers. The main reason possibly lies in that the Gaussian forward-filtering backward-smoothing is (i) more easy to implement (no need to design a rescaled backward recursion in the generalized backward information filter) and (ii) computationally less demanding, especially when the state dimension is high (this can be recognized from computations of the mean and information matrix associated with the marginal smoothing density of the Gaussian two-filter smoother). However, the Gaussian two-filter approach has an important methodological position in devising more sophisticated methods, mainly the design of Rao-Blackwellized techniques in the context of MCMC [55], backward simulation [131], and particle MCMC [218]. The precision and computational complexity scale according to specific cubature rules adopted for approximating the associated integrals. Practically all the methods discussed in Section 2.2.1 can be used.

2.5.2 Two-Filter Particle Smoothers

The two-filter particle smoother has recorded a fair portion of popularity over the last years. Similarly as with the forward-backward approach, the two-filter smoother also overcomes the path degeneracy problem. The computational complexity of the algorithm scales with $\mathcal{O}(NMT)$ operations, where N and M is the number of particles associated with the forward and backward filter, respectively. However, under restrictive assumptions on the state-space model, an implementation of the two-filter smoother with improved $\mathcal{O}(NT)$ computational complexity can be designed [66], where N is the number of particles for both the forward and backward filters. This improved implementation offers a substantial improvement in the trade-off between the estimation accuracy and computational budged. Moreover, [66] suggests an approach to overcome difficulties arising in situations when f is degenerate, that is, $f(x_t|x_{t-1})$ is zero for a range of values of x_t and x_{t-1}. The advantage of the two-filter over the forward-backward particle smoother is that the sampling in the forward and backward filters can be performed in parallel. Similarly as with the forward-filter backward-simulator, the particle implementation of the generalized two-filter smoother offers strongly consistent and asymptotically Gaussian estimators [159], and unbiased estimates of the marginal likelihood [170].

The third row of Fig. 2.4 shows the two-filter particle smoother [26] on Examples 2.1-2.3. The forward and backward filters are applied with the bootstrap proposal density, having the number of particles $N = 100$ and $M = 10$, respectively. The computation of the artificial prior density $\xi_t(x_t)$ is based on the unscented Kalman filter [102] where the initial statistics are $\hat{x}_{1|0} = 0$ and $\hat{P}_{1|0} = 5$, and the scaling parameter (κ, see [102]) is 2. We can see that implementing $\xi_t(x_t)$ in this way causes the performance of the two-filter particle smoother to be worse compared to the forward-filter backward simulator. For alternative choices of $\xi_t(x_t)$, see [26].

2.6 Particle MCMC Smoothing

A particle MCMC smoother can be seen as a Markov transition kernel $\mathcal{K}$ defined on X^T which leaves the joint state posterior density $p(x_{1:T}|y_{1:T})$ invariant. Such a kernel can be assembled based on the CSMC method presented in Algorithm 6. The sampling from a Markov kernel constructed in this way is performed as follows: Consider we have a state trajectory from the previous iteration $x_{1:T}[k-1]$, and we utilize it to condition Algorithm 6 for $t = 1, \dots, T$. After finishing the computation of the last time step, we sample $x_{1:T}[k] := x^k_{1:T}$ by drawing $k \sim \mathbf{W}_T$. The sampled state trajectory is then used to condition Algorithm 6 in the next iteration. Repeating this procedure for R iterations generates the Markov chain $(x_{1:T}[k])_{k=1}^R$ which enables the estimation of quantities of interest that are associated with $p(x_{1:T}|y_{1:T})$. As discussed in Sections 1.6 and 1.9.1, the CSMC method suffers from the path degeneracy problem, which negatively influences the mixing properties of the kernel.

The implementation details follow the same guidelines as in Section 2.2.2, including design of the auxiliary particle filter-based proposal density which might potentially improve the mixing properties of the kernel. However, as presented in Fig. 2.3, one can expect this modification to not have any significant impact in this respect. A substantial improvement on this issue is obtained by implementing the kernel based on the backward simulation [216] or the ancestor sampling [132].

In the last row of Fig. 2.4, we present the particle MCMC smoother with the particle Gibbs with ancestor sampling kernel [132] on Examples 2.1-2.3, where the number of particles and iterations are set to $N = 10$ and $R = 100$, respectively.

2.7 Frequentist Parameter Estimation

2.7.1 The EM Algorithm

The expectation maximization (EM) algorithm [51] is a widely applicable tool for addressing the maximum likelihood parameter inference problem (2.5) in incomplete data models, including state-space models. The EM procedure belongs to the class of data augmentation strategies—where the likelihood $p_\theta(y_{1:T})$ is augmented by the latent state sequence, $p_\theta(y_{1:T}, x_{1:T})$—and is generally implementable in either online or offline manner. The characteristic feature of this method is that it allows us to split the maximum likelihood problem into two separated and presumably more easily tractable steps known as the expectation and maximization.

Principle

The nature of the EM algorithm follows from establishing the relation between the complete $p_\theta(x_{1:T}, y_{1:T})$ and incomplete $p_\theta(y_{1:T})$ data likelihood according to

$$l(\theta) = \log p_\theta(x_{1:T}, y_{1:T}) - \log p_\theta(x_{1:T}|y_{1:T}), \tag{2.35}$$

which simply results from the logarithm of Bayes' rule. Here, we use the shorthand notation $l(\theta) \coloneqq \log p_\theta(y_{1:T})$. Note that maximizing the log-likelihood $l(\theta)$ is equivalent to maximizing the likelihood $p_\theta(y_{1:T})$ due to the fact that $p_\theta(y_{1:T})$ is always non-negative. The complete data likelihood is constructed by augmenting the observed data sequence with auxiliary latent variables that are—in the context of state-space modes—given by the latent state sequence. The complete data likelihood is here understood as an idealized version of the incomplete data likelihood. The key motivation for the initial design step (2.35) lies in that it is often more convenient to deal with the complete rather than incomplete data likelihood. We continue by taking the expected value of (2.35) with respect to the joint state posterior density $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$ evaluated at $\theta[k-1] \in \Theta$,

$$l(\theta) = \mathcal{Q}_k(\theta) + \mathcal{H}_k(\theta), \tag{2.36}$$

where we introduce

$$\mathcal{Q}_k(\theta) \coloneqq \quad \mathsf{E}_{\theta[k-1]}[\log p_\theta(X_{1:T}, y_{1:T})], \tag{2.37}$$
$$\mathcal{H}_k(\theta) \coloneqq -\mathsf{E}_{\theta[k-1]}[\log p_\theta(X_{1:T}|y_{1:T})]. \tag{2.38}$$

This step follows from the fact that the unobserved state sequence is unknown, and we therefore apply marginalization with respect to $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$. The expected value of the complete data log-likelihood (2.37) is often referred to as the *intermediate quantity* and plays the key role in the algorithm workflow, which we show next. The expected value of the logarithm of the joint state posterior density (2.38) is the entropy of this density. Note that—despite the marginalization—the formula (2.36) preserves the value of the incomplete data likelihood but provides its different interpretation. The purpose of the EM algorithm is to locate θ which maximizes $l(\theta)$. Let us therefore investigate the difference between two consecutive values of (2.36) evaluated at $\theta[k]$ and $\theta[k-1]$,

$$l(\theta[k]) - l(\theta[k-1]) = \Big(\mathcal{Q}_k(\theta[k]) - \mathcal{Q}_k(\theta[k-1])\Big) + \Big(\mathcal{H}_k(\theta[k]) - \mathcal{H}_k(\theta[k-1])\Big). \tag{2.39}$$

The second term on the r.h.s of (2.39) is equivalent to $\mathcal{D}(p_{\theta[k-1]}||p_{\theta[k]})$—the Kullback-Leibler divergence from $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$ to $p_{\theta[k]}(x_{1:T}|y_{1:T})$—which is always non-negative. With this in mind, the first term on the r.h.s. of (2.39) reveals the essential

Algorithm 10 Expectation maximization (EM)

```
A. Initial step: $(k = 0)$
    1. Set $\theta[0] \in \Theta$, and $\mathcal{Q}_0(\theta) \coloneqq 0$.
B. Recursive step: $(k = 1, \ldots, R)$
    1. Compute $\mathcal{Q}_k(\theta)$ according to (2.37).
    2. Compute $\theta[k] = \arg\max_{\theta \in \Theta} \mathcal{Q}_k(\theta)$.
```

idea of the EM algorithm: if we choose a new estimate $\theta[k]$ such that the value of the intermediate quantity is greater than or equal to its previous value, $\mathcal{Q}_k(\theta[k]) \geq \mathcal{Q}_k(\theta[k-1])$, then the incomplete data log-likelihood is also greater than or equal to its previous value, $l(\theta[k]) \geq l(\theta[k-1])$. This allows us to formulate the fundamental inequality of the EM algorithm [30]

$$l(\theta[k]) - l(\theta[k-1]) \geq \mathcal{Q}_k(\theta[k]) - \mathcal{Q}_k(\theta[k-1]). \tag{2.40}$$

Therefore, we can maximize $p_\theta(y_{1:T})$ by iteratively maximizing $\mathcal{Q}_k(\theta)$. In other words, we can iteratively refine θ until reaching a stationary point of the likelihood $p_\theta(y_{1:T})$ by following the above prescription. Consequently, this allows us to assemble the EM algorithm as a procedure divided into two parts: (i) the expectation (E) step which computes the intermediate quantity (2.37) and (ii) the maximization (M) step which maximizes this intermediate quantity. We summarize this method in Algorithm 10. The inequality (2.40) implies that the sequence $(l(\theta[k]))_{k=1}^{R}$ produced by this procedure is monotonically increasing.

Properties

The EM algorithm is useful mainly in situations where the maximization of $\mathcal{Q}_k(\theta)$ is easier than direct maximization of $p_\theta(y_{1:T})$. Under regularity assumptions delineated in [223], the sequence of the parameter estimates $(\theta[k])_{k=0}^{R}$ produced by the EM algorithm converges towards a stationary point of the likelihood $p_\theta(y_{1:T})$ for $R \to \infty$. However, it is important to note that the EM algorithm is a local deterministic optimization procedure which may converge towards a local maximum or saddle point. As explained above, the sequence of log-likelihood evaluations $(l(\theta[k]))_{k=0}^{R}$ is non-decreasing, and the EM algorithm therefore embodies a monotone optimization procedure. The effectiveness of the EM algorithm is conditioned on our ability to properly choose the auxiliary variables. In the context of state-space models, the choice is commonly undertaken such that the auxiliary variables are the latent states, but an alternative choice given by the latent disturbances [208] is also possible. Contrary to gradient-based techniques, we do not need to compute the gradient of the likelihood function to apply the EM algorithm, and there is thus no need to impose any direct assumptions on the smoothness of $p_\theta(y_{1:T})$. Another advantage of the EM algorithm lies in that it is numerically robust [124], overcomes difficulties with parameterization choices (as there are no parameters to tune, which is also the reason why the EM methods converge slower compared to gradient-based techniques), and offers the possibility to estimate initial conditions. In the case of linear Gaussian state-space models, the method is robust to high-dimensional state spaces [77]. The convergence rate of the EM algorithm is rather slow [223]. However, this has not prevented the method from being applied in a wide-range of practical cases. An often encounter situation is that the M-step facilitates computations under a closed form. Indeed, the complete data likelihood is commonly expressed by a probability density function belonging to the exponential family [12]. Such densities ensure the intermediate quantity to be concave w.r.t. the argument θ, which is especially convenient as the EM algorithm is then guaranteed to converge to the global maximizer of the likelihood. Moreover, the EM algorithm enables us to simply involve constraints on estimated parameters.

Related Methods

The EM algorithm for linear Gaussian state-space models, where both the steps can be computed under an explicit solution, is presented in [196, 224, 77]. This fundamental setup of the EM algorithm have found numerous applications in more advanced versions of this method, including learning of jump Markov linear models [202]. A substantial attention has also been focused on approximating the integral in the E-step based on Gaussian filters [75, 82, 71]. The consequence of applying Gaussian filters in this context carries over all their advantages and disadvantages, mainly the fact that when the nonlinearities become severe the algorithms usually provide a poor estimation performance [118]. Although there is a large body of work related to the EM algorithm, here we only refer to the basic scenarios—connected to the tractable and assumed Gaussian density settings—and leave more advanced algorithm constructions for the following sections. These are primarily related to situations with nonlinear and non-Gaussian state-space models, where the intermediate quantity (2.37) is intractable.

2.7.2 The Monte Carlo EM Algorithm

The Monte Carlo EM (MCEM) algorithm [151, 215] is a simulation-based version of the basic EM procedure which is suitable in situations where it is not feasible to compute the E-step under a closed-form solution. The MCEM method approaches this problem by approximating the E-step based on Monte Carlo sample averages.

Algorithm 11 Monte Carlo expectation maximization (MCEM)

```
A. Initial step: ($k = 0$)
    1. Set $\theta[0] \in \Theta$.
B. Recursive step: ($k = 1, \dots, R$)
    1. Sample $(X^i_{1:T}[k])^N_{i=1} \sim p_{\theta[k-1]}(\cdot|y_{1:T})$.
    2. Compute $\mathcal{Q}^N_k(\theta)$ according to (2.41).
    3. Compute $\theta[k] = \arg\max_{\theta\in\Theta} \mathcal{Q}^N_k(\theta)$.
```

Principle

The basic idea of the MCEM algorithm is to first simulate a set of IID sample trajectories $(X^i_{1:T}[k])^N_{i=1}$ from the joint state posterior distribution $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$ parameterized by the previous estimate $\theta[k-1]$ at each iteration k, and then use the samples to approximate the intermediate quantity (2.37). This approximation is simply obtained by substituting the empirical approximation $p^N_{\theta[k-1]}(dx_{1:T}|y_{1:T})$ into (2.37), that is,

$$\mathcal{Q}^N_k(\theta) := \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(X^i_{1:T}[k], y_{1:T}). \tag{2.41}$$

The rest of the procedure follows basically the same steps as in the fundamental EM algorithm. We present this procedure in Algorithm 11.

Properties

The MCEM algorithm is a stochastic optimization procedure which offers (a very basic) ability to escape local maxima or saddle points. Contrary to the basic EM algorithm, since we approximate $\mathcal{Q}_k(\theta)$ by $\mathcal{Q}^N_k(\theta)$, it is no longer possible to guarantee that the log-likelihood increases monotonically. To ensure that the method behaves in the convergent manner—thus approaches a stationary point of the likelihood—we need the number of particles N to increase with the number of iterations R. (In practical situations, it is often enough to use a low number of particles N to drive the parameter estimates towards important parts of the likelihood function and then increase this number in later iterations of the optimization process.) Indeed, empirical examples commonly indicate that the estimates produced by the MCEM algorithm suffer from a substantial bias and variance that diminish only when increasing the number of particles N. Therefore, under the conditions delineated in [68], including the requirement that the complete data likelihood belongs to the exponential family [12], the method requires both the number of particles N and iterations R to tent to infinity to converge. This approach is thus double asymptotic, which is its main disadvantage. Moreover, as can be seen in Algorithm 11, the simulated trajectories are wastefully discarded at every iteration, implying a high computational complexity. This can be particularly inefficient in situations where we decide to sample from a

Markov kernel which leaves the joint state posterior density $p_\theta(x_{1:T}|y_{1:T})$ invariant. In this case, at each iteration, we need to wait to go over the transient phase of the chain to obtain samples approximately distributed according to $p_\theta(x_{1:T}|y_{1:T})$. This design step also implies that there would be two nested layers of iterations, and the computational cost would be substantially high. Moreover, the dependent samples complicate theoretical analysis of such algorithms.

Related Methods

There exists a number of different methods to sample from the joint state posterior density $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$. A possible way is to use the generic particle smoothing EM (PSEM) framework [194]. This approach considers that the sampling can be made by various different forms of particle smoothers, including those discussed in Section 2.3.2, such as forward-filter backward-smoother [54, 194], forward-filter backward-simulator [80], generalized SMC two-filter smoother [25, 66], and fixed-lag smoother [163]. Nevertheless, as noted before, the PSEM approach does not reuse the samples over the iterations and thus requires a notably higher amount of particles at each iteration to obtain reliable estimates of the intermediate quantity.

2.7.3 The Stochastic Approximation EM Algorithm

The main difficulty with the MCEM algorithm is the requirement that both the number of particles at each iteration and the number of iterations itself have to tend to infinity for the algorithm to converge. The stochastic approximation EM (SAEM) algorithm [50, 207] overcomes this problem by implementing the E-step based on the stochastic approximation [182] which reuses the samples over the iterations and therefore avoids the need for the number of particles to tend to infinity. This is an important methodological concept which states that we can estimate the parameters even by iterating over imprecisely approximated intermediate quantities.

Principle

The underlying idea of the SAEM algorithm is to reuse the a set of IID sample trajectories $(X^i_{1:T}[k])_{i=1}^N$—drawn from the joint state posterior distribution $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$ evaluated at the previous estimate $\theta[k-1]$—over multiple iterations. This can be accomplished by averaging (2.41) according to

$$\mathcal{Q}_k^N(\theta) := \frac{1}{k}\frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{N}\log p_\theta(X^j_{1:T}[i], y_{1:T}). \tag{2.42}$$

Algorithm 12 Stochastic approximation expectation maximization (SAEM)

```
A. Initial step: ($k = 0$)
    1. Set $x_{1:T}[0] \in \mathsf{X}^T$, $\theta[0] \in \Theta$, and $\widehat{\mathcal{Q}}_0(\theta) := 0$.
B. Recursive step: ($k = 1, \dots, R$)
    1. Sample $(X^i_{1:T}[k])_{i=1}^N \sim p_{\theta[k-1]}(\cdot|y_{1:T})$.
    2. Compute $\widehat{\mathcal{Q}}_k(\theta)$ according to (2.44).
    3. Compute $\theta[k] = \arg\max_{\theta\in\Theta} \widehat{\mathcal{Q}}_k(\theta)$.
```

A simple rearrangement of (2.42) leads to

$$\begin{aligned}\mathcal{Q}^N_k(\theta) &= \frac{1}{k}\frac{1}{N}\sum_{j=1}^{N}\log p_\theta(X^j_{1:T}[k], y_{1:T}) + \left(1-\frac{1}{k}\right)\frac{1}{k-1}\frac{1}{N}\sum_{i=1}^{k-1}\sum_{j=1}^{N}\log p_\theta(X^j_{1:T}[i], y_{1:T})\\ &:= \alpha_k\frac{1}{N}\sum_{j=1}^{N}\log p_\theta(X^j_{1:T}[k], y_{1:T}) + (1-\alpha_k)\mathcal{Q}^N_{k-1}(\theta), \qquad (2.43)\end{aligned}$$

where we define $\alpha_k := \frac{1}{k}$. The step-size sequence $(\alpha_k)_{k\geq 1}$ can follow different schedules. The common requirement is that α_k satisfies the constraints $\alpha_k \in [0, 1]$ and

$$\sum_{k=1}^{\infty}\alpha_k = \infty, \qquad \sum_{k=1}^{\infty}\alpha_k^2 < \infty.$$

In usual situations, we select the step size as $\alpha_k = ck^{-\lambda}$ with $\lambda \in (0.5, 1]$ and $c > \mathbb{R}_+$. It is also possible that we approximate the intermediate quantity with only $N = 1$. In such a case, (2.43) becomes

$$\widehat{\mathcal{Q}}_k(\theta) = (1-\alpha_k)\widehat{\mathcal{Q}}_{k-1}(\theta) + \alpha_k \log p_\theta(X_{1:T}[k], y_{1:T}). \qquad (2.44)$$

From (2.43), we see that all simulated trajectories are reused across the iterations, but the older ones are continuously discounted by the forgetting factor related to the step size. Similarly as before, this alternative approach for computing the intermediate quantity is the only modification to the basic structure of the EM algorithm, which allows us to summarize the method in Algorithm 12.

The problem we encounter in numerous practical applications is that we are unable to directly sample from the joint state smoothing density $p_\theta(x_{1:T}|y_{1:T})$. A possible solution to this issue is that one can couple the SAEM algorithm with the MCMC framework [121]. The idea behind this approach is to draw the samples from a Markov kernel $\mathcal{K}_{\theta[k-1]}$ defined on X^T which admits the joint state smoothing density $p_\theta(x_{1:T}|y_{1:T})$ as its unique stationary density. The step B1 in Algorithm 12 is then replaced by: Sample $X_{1:T}[k] \sim \mathcal{K}_{\theta[k-1]}(\cdot|X_{1:T}[k-1])$.

Properties

The SAEM approach is another instance of a stochastic optimization procedure. Under regularity assumptions presented in [121], including uniform ergodicity of the transition kernel $\mathcal{K}_{\theta[k-1]}$ and the complete data likelihood $p_\theta(x_{1:T}, y_{1:T})$ belonging to the exponential family, the sequence of the maximum likelihood estimates, $(\theta[k])_{k=0}^R$, produced by the MCMC version of the SAEM algorithm, converges to a maximizer of $p_\theta(y_{1:T})$ for $R \to \infty$. From (2.44), the choice of the step size α_k significantly affects the reusability of the previously sampled trajectories. On the one hand, for α_k close to one, there is almost no reuse of the previously sampled trajectories, and we mostly rely on new information coming from the second term on the r.h.s. of (2.44). On the other hand, for α_k close to zero, we largely reuse the previously sampled trajectories, and there is only a diminishing amount of new information coming from the second term on the r.h.s. of (2.44). The former case is often used during the initial iterations to allow the algorithm to quickly attain the important areas of the likelihood surface. The latter case is desirable after a high amount of the iterations—when the algorithm already learned the important information—and there is no longer a need for any significant improvements of the estimated parameters. The values between these two extreme cases affect the convergence speed and variance of the sequence of iterates $(\theta[k])_{k=0}^R$. For the previously mentioned step size, $\alpha_k = ck^{-\lambda}$, choosing λ towards 0.5 speeds up the convergence but increases the variance, while setting λ towards 1 makes the convergence slower but also decreases the variance. The choice of the step size also influences the ability of this algorithm to escape local stationary points of the likelihood surface. The fact the SAEM algorithm accumulates the sample trajectories over the iterations substantially reduces its computational requirements compared to the basic MCEM approach. Regarding the use of MCMC kernels: As discussed previously, if a transition kernel were used in the MCEM algorithm, there would be the need to reach the stationary regime of the produced Markov chain at each iteration. Here, however, the transition kernel $\mathcal{K}_{\theta[k-1]}$ generates the Markov chain over the iterations $k = 1, \dots, R$. In other words, this does not require the chain to reach the stationary regime at every single iteration but rather over the natural course of multiple iterations, which substantially saves the computational time.

Related Methods

A wide range of various implementations of the SAEM algorithm can be formulated. The main difference among such methods commonly lies in the way we produce the sample trajectories from the joint state posterior density in the step B1 of Algorithm 12. A possible approach is to utilize the particle smoothing methods discussed in the case of the MCEM algorithm in Section 2.7.2. Another way is to design a particle MCMC smoother producing the sample trajectories from a particle MCMC kernel which leaves $p_\theta(x_{1:T}|y_{1:T})$ invariant. Concretely, one can resort to a particle independent Metropolis-Hastings [7] or particle Gibbs with ancestor sampling [130, 132], the latter of which has been demonstrated to provide a remarkably efficient trade-off between the estimation precision and computational cost.

2.8 Bayesian Parameter Estimation

2.8.1 The Gibbs Sampler

The Gibbs sampler [72] has become a standard tool for addressing the Bayesian parameter inference problem (2.6) in state-space models [33]. The algorithm can be categorized as an offline data augmentation strategy, where the posterior density of the parameters $p(\theta|y_{1:T})$ is augmented by the latent state sequence, $p(\theta|y_{1:T}, x_{1:T})$. The main motivation for this augmentation is that the latter is often tractable in the state-space model setting, while the former is not.

Principle

The basic ideas of this approach remain the same as with the generic Gibbs sampler presented in Section 1.8.1. Here, we briefly comment on specific instances of this method when dealing with tractable and intractable state-space models. In this context, the sampler targets the joint density of the states and parameters conditioned on the observed data sequence, $p(x_{1:T}, \theta|y_{1:T})$. The algorithm alternately samples, for $k = 1, \ldots, R$, from the conditional factors

$$\Theta[k] \sim p(\theta|x_{1:T}[k-1], y_{1:T}), \tag{2.45a}$$

$$X_{1:T}[k] \sim p_{\Theta[k]}(x_{1:T}|y_{1:T}). \tag{2.45b}$$

This procedure generates the Markov chain $(\Theta[k], X_{1:T}[k])_{k=1}^{R}$ which—if the assumptions of Theorem 1.10 are satisfied—can be used to estimate expectations under the target density, or, its marginal $p(\theta|y_{1:T})$ by simply discarding the state trajectories.

Properties

To sample from the second factor (2.45b), we need to address the previously discussed joint smoothing problem. A possible way is to sample the states individually from $p_\theta(x_t|x^{-t}, y_{1:T})$, where $x^{-t} := (x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_T)$, by utilizing the Metropolis-Hastings algorithm [74]. As mentioned before, such an approach may suffer from poor mixing when the individual state components are strongly correlated. An improved performance can be obtained when sampling the full state trajectory $X_{1:T}$ at once. This strategy may substantially improve mixing properties of the algorithm. Compared to the Metropolis-Hastings algorithm, the advantage lies in that there is no need to design a proposal density with the Gibbs sampler. Under the assumptions given in [183], the Gibbs sampler converges in the sense of (also the stronger version of) Theorem 1.10.

Related Methods

The sampling from the second factor (2.45b) can be performed by one of the particle smoothing methods discussed in Section 2.3.2. However, the most often encountered case is to use the forward-filter backward-simulator [80]. Concretely, we are required to first run a forward filtering algorithm and then use its output to run a backward sampling algorithm. A specific implementation of this strategy differs according to the character of a state-space model under study. For linear Gaussian state-space models, the forward filtering algorithm is embodied by the Kalman filter [103], and the backward transition kernel (2.22) can then be computed under a closed-form solution. This kernel is then used to produce the sample trajectories [70, 33]. For nonlinear non-Gaussian state-space models, we proceed by considering severity of nonlinearities. If the nonlinearities are mild, we can design the forward filtering procedure by means of the Gaussian filters, such as the previously mentioned unscented, cubature, or Gauss-Hermite Kalman filters. The backward transition kernel is then approximated based on the Gaussian approximation (2.25), see also [190]. In certain situations, closed-form updating formulas associated with the Gaussian assumed density methods may suffer from numerical problems that are related to computations of the involved matrix inversions. To address this problem, a numerically robust implementation of the Gibbs sampler can be found in [221]. However, perhaps a still open question is whether the Gibbs sampler converges when applying these Gaussian-based approximate techniques. If the nonlinearities are severe, the forward filtering algorithm is assembled by means of the particle filter, and the backward transition kernel is approximated by the corresponding sequence of particle systems (2.28). A recent strategy to address the problem of sampling from the second factor is to use the particle Gibbs kernel which leaves $p_{\Theta[k]}(x_{1:T}|y_{1:T})$ invariant, resulting in the particle Gibbs sampler, as discussed in Section 1.9.1. The particle Gibbs sampler converges for any number of particles $N \geq 2$ and the number of iterations $R \to \infty$ [4].

The often encountered situation is that the complete data likelihood $p_\theta(x_{1:T}, y_{1:T})$ belongs to the exponential family [12]. If we choose the conjugate prior $p(\theta)$, then the first factor (2.45a) admits a closed-form solution and can be computed by means of finite dimensional sufficient statistics. Otherwise, one can utilize the Metropolis-Hastings algorithm to produce the Markov chain $(\Theta[k])_{k=1}^{R}$ which admits the first factor (2.45a) as its stationary density [74].

3 A PROJECTION-BASED PARTICLE FILTER TO ESTIMATE STATIC PARAMETERS IN CONDITIONALLY CONJUGATE STATE-SPACE MODELS

Particle filters constitute today a well-established class of techniques for state filtering in non-linear state-space models. However, online estimation of static parameters under the same framework represents a difficult problem. The solution can be found to some extent within a category of state-space models allowing us to perform parameter estimation in an analytically tractable manner, while still considering non-linearities in data evolution equations. Nevertheless, the well-known particle path degeneracy problem complicates the computation of the statistics that are required to estimate the parameters. The present chapter proposes a simple and efficient method which is experimentally shown to suffer less from this issue.

3.1 Introduction

3.1.1 Context

A state-space model (SSM, [30]) embodies a popular statistical tool for describing dynamical systems in diverse application areas such as signal processing, econometrics, and bioinformatics. This model is especially useful for defining the relation between observed data, latent (unobserved) data, and unknown static parameters. The estimation of the states and parameters based on the observations is the primary task in the aforementioned application areas. A rather general class of state-space models is formed when they contain a *tractable* substructure characterizing the parameters and an *intractable* substructure describing nonlinear, and possibly non-Gaussian, data (observed and unobserved). Such models are herein referred to as the conditionally conjugate SSMs (CCSSMs). Their key feature is that the tractable substructure facilitates recursive updates of statistics related to the posterior distribution of the parameters, but the intractable substructure requires us to use approximate inference to make the parameter estimation feasible. This chapter considers particle filters (PFs, [56]) to perform the approximate inference.

A number of PF-based methods for estimating static parameters in the considered class of models have been developed in the last years [200, 62, 34]. These algorithms utilize the tractable substructure to compute a set of the posterior statistics based on the observations and latent state trajectories simulated by a PF. However, the trajectories are known to suffer from the particle path degeneracy [6], if they are constructed in a single forward pass of a PF. This issue also affects the computation of the posterior statistics, and methods relying on such a principle therefore usually deliver poor performance.

So far, we have only referred to methods that are most relevant to the algorithm proposed in the present chapter. For a thorough overview of PF-based parameter estimation, we refer the reader to a series of recent survey papers [104, 93, 65]. Importantly, there has recently been an increased interest in designing methods based on particle smoothing [133] or particle Markov chain Monte Carlo [4], which are efficient in dealing with the degeneracy issue. However, these procedures are offline, processing repeatedly a fixed batch of data, and since this chapter is concerned with the online estimation, such algorithms are not of a particular interest herein.

3.1.2 Contributions

The main contribution of the present chapter consists in designing an algorithm for estimating parameters in the CCSSMs. The proposed approach shares the similarities with the aforementioned methods in the sense that it also computes the posterior statistics. The design of the method includes two ideas. First, we take advantage of the tractable substructure to integrate out the parameters and thus utilize the Rao-Blackwellization [36]. Second, based on the Kullback-Leibler divergence (KLD, [123]) principle, we formulate an update-project-update cycle to compute the posterior statistics. It is shown that the parameter estimation is then less degenerate.

3.2 Background

3.2.1 Problem Formulation

Let us consider a discrete-time SSM in the form

$$p_\theta(y_t, x_t|x_{t-1}) = g_\theta(y_t|x_t)f_\theta(x_t|x_{t-1}), \tag{3.1}$$

where $x_t \in \mathsf{X} \subseteq \mathbb{R}^{n_x}$ and $y_t \in \mathsf{Y} \subseteq \mathbb{R}^{n_y}$ label the state and observation variables, respectively. The model is characterized by the probability densities $g_\theta(\cdot)$ and $f_\theta(\cdot)$, with $\theta \in \Theta \subseteq \mathbb{R}^{n_\theta}$ denoting some unknown static parameters. At the initial time step, the state and parameter variables are distributed according to $x_1 \sim p_\theta(x_1)$ and $\theta \sim p_0(\theta)$. We restrict ourselves to SSMs that allow us to express (3.1) by the exponential family (EF, [12]) density

$$\begin{aligned} p_\theta(y_t, x_t|x_{t-1}) = \exp\{&\langle \eta(\theta), s_t(x_{t-1}, x_t, y_t)\rangle \\ &- \zeta(\theta) + \log h(x_{t-1}, x_t, y_t)\}, \end{aligned} \tag{3.2}$$

Algorithm 13 Particle Filter (PF)

```
A. Initial step: $(t = 1)$
    1. Sample $x_1^i \sim q_1(\cdot)$.
    2. Compute $w_1^i \propto W_1(x_1^i)$.
B. Recursive step: $(t = 2, \dots, T)$
    1. Sample $a_t^i$ with $\mathbb{P}(a_t^i = j) = w_{t-1}^j$.
    2. Sample $x_t^i \sim q_t(\cdot|x_{1:t-1}^{a_t^i})$ and set $x_{1:t}^i := (x_t^i, x_{1:t-1}^{a_t^i})$.
    3. Compute $w_t^i \propto W_t(x_{1:t}^i)$ according to (3.6).
```

where η and ζ are respectively the matrix and scalar-valued functions defined on Θ, s_t and h constitute respectively the matrix and scalar-valued functions defined on $\mathsf{X}^2 \times \mathsf{Y}$, and $\langle \cdot, \cdot \rangle$ represents the inner product. The SSM delineated by (3.2) is herein referred to as the CCSSM. The name follows from the fact that (3.2) is analytically tractable with respect to the parameters but intractable with respect to the presumably nonlinear functions s_t and h. More specifically, the model (3.2) facilitates analytical computation of the posterior density of the parameters, if we choose the conjugate prior density according to

$$p(\theta|\nu_{t-1}, V_{t-1}) = \exp\{\langle \eta(\theta), V_{t-1} \rangle - \nu_{t-1}\zeta(\theta) \\ - \log \mathcal{I}(\nu_{t-1}, V_{t-1})\}, \tag{3.3}$$

where V_{t-1} denotes the extended information matrix, ν_{t-1} labels the number of degrees of freedom, and $\mathcal{I}$ defines the normalizing constant. The posterior density $p(\theta|\nu_t, V_t)$ then reproduces the form of (3.3), and its statistics can be updated under the closed-form formulae

$$V_t = V_{t-1} + s_t(x_{t-1}, x_t, y_t), \tag{3.4a}$$

$$\nu_t = \nu_{t-1} + 1. \tag{3.4b}$$

The model (3.2) is also known as the conditionally conjugate latent process model [212, 187]. The generic form (3.2) acknowledges standard probability densities such as Poisson, Gaussian, exponential, etc.

The objective of this chapter is to design an online method for computing the posterior density $p(x_t, \theta|y_{1:t})$ while assuming (3.2), where $y_{1:t} := (y_1, \dots, y_t)$. Nevertheless, the nonlinear functions s_t and h prevent us from computing the posterior analytically. To resolve this problem, we need to resort to approximate techniques. For the ability to deal with almost any nonlinear non-Gaussian SSM, we choose PFs to handle the approximate inference.

3.2.2 Particle Filters

A PF is a sequential Monte Carlo algorithm [56] suitable for sequentially approximating probability densities of the form $p(x_{1:t}|y_{1:t})$. At each time step t, the method produces approximation given by the empirical measure

$$p^N(dx_{1:t}|y_{1:t}) = \sum_{i=1}^{N} w_t^i \delta_{x_{1:t}^i}(dx_{1:t}), \tag{3.5}$$

which is represented by the *weighted particle system* $(x_{1:t}^i, w_t^i)_{i=1}^N$, where $x_{1:t}^i$ denotes a particle trajectory, w_t^i labels a normalized importance weight which assesses the significance of the associated trajectory, and δ_x is the Dirac measure located at x. A common particle filtering procedure, which is known as the sequential importance resampling [56], is summarized in Algorithm 13, where all operations are performed for $i = 1, \dots, N$.

The *initial step* of Algorithm 13 is made of standard importance sampling. Thus, we first draw the particles from an initial proposal density q_1 in line A1 and then calculate the normalized importance weights using $W_1(x_1) := p(x_1, y_1)/q_1(x_1)$ in line A2.

The *recursive step* of Algorithm 13 is a combination of sequential importance sampling and resampling. Assume we have the previously generated particle system $(x_{1:t-1}^i, w_{t-1}^i)_{i=1}^N$. The recursion starts with the resampling procedure, which is equivalent to drawing ancestor indices $a_t \in (1, \dots, N)$ in line B1. The indices are then applied in the sequential importance sampling approach given by lines B2 and B3. First, the particles are generated from the proposal density q_t and used to extended a previous trajectory to a current one. The indices here determine the parent trajectory $x_{1:t-1}$ for the offspring particle x_t and offspring trajectory $x_{1:t}$. Second, the normalized importance weights are computed with

$$W_t(x_{1:t}) := \frac{p(y_t, x_t|x_{1:t-1}, y_{1:t-1})}{q_t(x_t|x_{1:t-1})}. \tag{3.6}$$

After performing the sequence of operations B1-B3, we acquire a newly generated particle system $(x_{1:t}^i, w_t^i)_{i=1}^N$. For a detailed introduction to particle filtering, see [57].

3.2.3 Projection-Based Approximation of Probability Densities

The projection-based approach for approximating probability densities [107] is useful in situations where we have a complex density $\tilde{p}(\theta)$ which needs to be replaced by a more simple, approximate, one $\widehat{p}(\theta)$. Contrary to the particle filtering, the projection-based approach is an instance of deterministic approximate inference. The approximate density $\widehat{p}(\theta)$ is sought as the minimizer of the KLD between the complex density $\tilde{p}(\theta)$ and a feasible density $p(\theta) \in \mathsf{P}$, that is, we solve the optimization problem

$$\widehat{p} := \operatorname*{argmin}_{p \in \mathsf{P}} d(\tilde{p}, p) = \operatorname*{argmin}_{p(\theta) \in \mathsf{P}} \int_{\Theta} \tilde{p}(\theta) \log\left(\frac{\tilde{p}(\theta)}{p(\theta)}\right) d\theta, \tag{3.7}$$

where P is a designer-selected set of feasible densities.

Let us consider a specific instance of the discussed approach, which will be needed later on in this chapter. Suppose the density we intend to approximate is given by the mixture form

$$\tilde{p}(\theta) := p(\theta|\tilde{\nu}, \tilde{V}) = \sum_{i=1}^{N} w^i p(\theta|\nu^i, V^i),$$

and the feasible density is chosen as a member of the EF, having the same functional form as (3.3), $p(\theta) := p(\theta|\nu, V)$. If we set the gradient of $d(\tilde{p}, p)$ with respect to the feasible statistics ν and V to zero, we find out that the KLD is minimized by equating the expectations

$$\mathsf{E}[\eta(\theta)|\widehat{\nu}, \widehat{V}] = \sum_{i=1}^{N} w^i \mathsf{E}[\eta(\theta)|\nu^i, V^i], \tag{3.8a}$$

$$\mathsf{E}[\zeta(\theta)|\widehat{\nu}, \widehat{V}] = \sum_{i=1}^{N} w^i \mathsf{E}[\zeta(\theta)|\nu^i, V^i]. \tag{3.8b}$$

The approximate density $\widehat{p}(\theta) := p(\theta|\widehat{\nu}, \widehat{V})$ containing the statistics computed from the above expectations is the optimal solution of the problem (3.7). In this particular case of choosing $p(\theta)$ as a member of the EF, the approach is also known as the moment matching [21], a basic principle being at the core of expectation propagation algorithms [153].

3.3 A Projection-Based Rao-Blackwellized Particle Filter

The proposed method is based on factorizing the joint posterior density of the states and parameters according to

$$p(x_{1:t}, \theta|y_{1:t}) = p(\theta|x_{1:t}, y_{1:t})p(x_{1:t}|y_{1:t}). \tag{3.9}$$

The factorization (3.9) is advantageous since the considered class of models contains the algebraic substructure related to the parameters. The substructure allows us to perform two design steps. First, we integrate out the parameters and apply the PF framework to approximate only the marginal factor $p(x_{1:t}|y_{1:t})$, rather than the full posterior. Second, we compute the conditional factor $p(\theta|x_{1:t}, y_{1:t})$ analytically based on the recursive formulae (3.4) supplied with the observations and samples produced by the PF. These two steps characterize construction of an RBPF, see [55, 192, 166] for a different application context. The motivation behind integrating out a part of latent variables is to design estimators with the variance which is lower than—or at least the same as—we would obtain without the integration [36].

The RBPF approximates (3.9) by

$$p^N(dx_{1:t}, \theta|y_{1:t}) = \sum_{i=1}^{N} w_t^i p(\theta|\nu_t^i, V_t^i)\delta_{x_{1:t}^i}(dx_{1:t}), \tag{3.10}$$

which can be obtained by simply inserting (3.5) into (3.9). The algorithmic construction of the RBPF follows basically the same steps as delineated in Algorithm 13. The extra steps consist of computing (i) the statistics representing the conditional factor $p(\theta|x_{1:t}, y_{1:t}) \coloneqq p(\theta|\nu_t, V_t)$ according to

$$p(\theta|\nu_t^i, V_t^i) \propto p_\theta(y_t, x_t^i|x_{t-1}^{a_t^i})p(\theta|\nu_{t-1}^{a_t^i}, V_{t-1}^{a_t^i}), \tag{3.11a}$$

and (ii) the marginal density $p(y_t, x_t|x_{1:t-1}, y_{1:t-1}) \coloneqq p(y_t, x_t|\nu_{t-1}, V_{t-1})$ in the numerator of (3.6),

$$p(y_t, x_t^i|\nu_{t-1}^{a_t^i}, V_{t-1}^{a_t^i}) = \int_\Theta p_\theta(y_t, x_t^i|x_{t-1}^{a_t^i})p(\theta|\nu_{t-1}^{a_t^i}, V_{t-1}^{a_t^i})d\theta. \tag{3.11b}$$

The approximation of $p(x_t, \theta|y_{1:t})$ is then obtained by simply discarding the past trajectories $(x_{1:t-1}^i)$ in (3.10). Then, the procedure becomes recursive, while the information from the trajectories will be kept in the finite dimensional sufficient statistics (V_t^i). A number of methods that update the statistics based on (3.11a) has been proposed [200, 62, 34]. However, as widely discussed in [6, 38, 5], such methods are known to suffer from the particle path degeneracy [94]. Thus, successful resampling steps decrease the number of unique particle trajectories in the subset $(x_{1:k}^i)$ of $(x_{1:t}^i)$ for some $k < t$. This issue spoils the computation of the statistics, and the parameter estimates then usually experience high variance over multiple simulation runs.

The present chapter proposes to counteract this problem by first formulating the marginal density of (3.10) given by

$$p^N(\theta|y_{1:t}) = \sum_{i=1}^{N} w_t^i p(\theta|\nu_t^i, V_t^i), \tag{3.12}$$

and then use it in the next time step to replace the prior in (3.11a), thus, using $p^N(\theta|y_{1:t-1})$. However, such an approach would lead to an exponentially increasing number of the components of the mixture density (3.12). Therefore, at each time step, we find the approximation $p(\theta|\widehat{\nu}_t, \widehat{V}_t)$ of (3.12) by applying the previously presented projection-based approach. Consequently, we utilize the approximate density with the statistics computed from (3.8) to replace the prior, that is,

$$p(\theta|\nu_t^i, V_t^i) \propto p_\theta(y_t, x_t^i|x_{t-1}^{a_t^i})p(\theta|\widehat{\nu}_{t-1}, \widehat{V}_{t-1}), \tag{3.13a}$$

which also implies

$$p(y_t, x_t^i|\widehat{\nu}_{t-1}, \widehat{V}_{t-1}) = \int_\Theta p_\theta(y_t, x_t^i|x_{t-1}^{a_t^i})p(\theta|\widehat{\nu}_{t-1}, \widehat{V}_{t-1})d\theta. \tag{3.13b}$$

Algorithm 14 Projection-Based RBPF (PBRBPF)

```
A. Initial step: $(t = 1)$
    1. Set $\widehat{\nu}_0$ and $\widehat{V}_0$.
    2. Sample $x_1^i \sim q_1(\cdot)$.
    3. Compute $w_1^i \propto W_1(x_1^i)$.
B. Recursive step: $(t = 2, \dots, T)$
    1. Sample $a_t^i$ with $\mathbb{P}(a_t^i = j) = w_{t-1}^j$.
    2. Sample $x_t^i \sim q_t(\cdot|x_{1:t-1}^{a_t^i})$.
    3. Compute $w_t^i \propto W_t(x_{1:t}^i)$ according to (3.6).
C. Common step: $(t \geq 1)$
    1. Compute $\nu_t^i = \widehat{\nu}_{t-1} + 1$ and $V_t^i = \widehat{V}_{t-1} + s_t(x_{t-1}^{a_t^i}, x_t^i, y_t)$.
    2. Compute $\widehat{\nu}_t$ and $\widehat{V}_t$ as the solution of (3.8).
```

Let us now summarize the proposed method in Algorithm 14, where we use the convention $s_1(x_0, x_1, y_1) \coloneqq s_1(x_1, y_1)$. Specifically, lines C1-C2-C1 define the update-project-update cycle, which effectively avoids the resampling of the statistics—c.f. (3.11) and (3.13)—and it is empirically shown that it reduces the variance of the estimated parameters over multiple simulation runs. Contrary to standard RBPFs, there is no need to enlarge the particle system (x_t^i, w_t^i) by the set of statistics (ν_t^i, V_t^i) as the method only keeps the approximate statistics $\widehat{\nu}_t$ and $\widehat{V}_t$ between the iterations, thus having lower memory requirements.

3.4 Estimating Gaussian Noise Parameters

This section shows how to use the proposed method for estimating parameters of additive Gaussian noise variables in an SSM given by

$$\underbrace{\begin{bmatrix} x_t \\ y_t \end{bmatrix}}_{\xi_t} = \underbrace{\begin{bmatrix} a(x_{t-1}) \\ b(x_t) \end{bmatrix}}_{\Phi(x_{t-1:t})} + \underbrace{\begin{bmatrix} v_t \\ w_t \end{bmatrix}}_{e_t}, \tag{3.14}$$

where ξ_t, $\Phi(x_{t-1:t})$, and e_t embody vectors composed of the state and observation variables, nonlinear state transition and observation functions, and state and observation noise variables, respectively. Furthermore, e_t is an independent and identically distributed (IID) Gaussian noise variable $e_t \overset{IID}{\sim} \mathcal{N}(\mu, \Sigma)$ with the mean vector μ and covariance matrix Σ. The objective is to estimate $\theta = (\mu, \Sigma)$.

The probability density describing the above SSM is expressed by the Gaussian density in the form

$$p_\theta(y_t, x_t|x_{t-1}) = \mathcal{N}(\cdot; \Phi(x_{t-1:t}) + \mu, \Sigma), \tag{3.15}$$

which is a direct consequence of applying the change of variables formula [21] to the density $\mathcal{N}(e_t; \mu, \Sigma)$. The natural choice of the conjugate prior for estimating the unknown mean and covariance of a Gaussian density is the Gauss-inverse-Wishart (GiW) density

$$p(\theta|\widehat{\nu}_{t-1}, \widehat{V}_{t-1}) = \mathcal{N}i\mathcal{W}(\cdot; \widehat{\nu}_{t-1}, \widehat{R}_{t-1}, \widehat{\mu}_{t-1}, \widehat{\Lambda}_{t-1}). \tag{3.16}$$

For a later reference (see, Lemma 3.4), we introduce the following lemma, which determines the functions (η, ζ, s, h) of (3.2) for the specific case (3.15). However, we do not present the direct relation between the extended information matrix V of (3.3) and the statistics (R, μ, Λ) of (3.16) as it would not lead to any conceptual shift at this moment.

Lemma 3.1. *Let us consider the density (3.2) is given by (3.15); then, the functions (η, ζ, s, h) yield*

$$\eta(\theta) = \begin{bmatrix} \Sigma^{-1} & \Sigma^{-1}\mu \\ \mu^\top\Sigma^{-1} & \mu^\top\Sigma^{-1}\mu \end{bmatrix}, \qquad \zeta(\theta) = \frac{1}{2}\log|\Sigma|. \tag{3.17a}$$

$$s_t(x_{t-1:t}, y_t) = \begin{bmatrix} e_t \\ 1 \end{bmatrix} \begin{bmatrix} e_t \\ 1 \end{bmatrix}^\top, \quad h(x_{t-1:t}, y_t) = (2\pi)^{-\frac{n_e}{2}}. \tag{3.17b}$$

Proof. The results follow from simple rearrangements and the fact that the trace operator in the exponent of (3.15) is invariant under the cyclic permutations. □

To make the elements of Algorithm 14 concrete for the considered problem, we introduce the following three lemmas that *respectively* specify the computations required in lines (A3, B3), C1, and C2.

Lemma 3.2. *Let the densities (3.2) and (3.3) be defined by (3.15) and (3.16), respectively; then, the marginal density (3.13b) becomes the Student's t density given by $p(y_t, x_t|\widehat{\nu}_{t-1}, \widehat{V}_{t-1}) = \mathrm{St}(\xi_t; \bar{\mu}, \bar{\Lambda}, \bar{\nu})$, where*

$$\bar{\mu} = \Phi(x_{t-1:t}) + \widehat{\mu}_{t-1}, \tag{3.18a}$$

$$\bar{\Lambda} = \tfrac{1+\widehat{R}_{t-1}}{\widehat{\nu}_{t-1}-n_e+1}\widehat{\Lambda}_{t-1}, \tag{3.18b}$$

$$\bar{\nu} = \widehat{\nu}_{t-1} - n_e + 1, \tag{3.18c}$$

denote the mean value, scale matrix, and number of degrees of freedom, respectively, and $n_e = n_x + n_y$.

Proof. The result is derived in Lemma A.10. □

Lemma 3.3. *Let the densities (3.2) and (3.3) be defined by (3.15) and (3.16), respectively; then, the posterior density (3.13a) is* $p(\theta|\nu_t, V_t) = \mathcal{G}i\mathcal{W}(\cdot; \nu_t, R_t, \mu_t, \Lambda_t)$*, with the statistics being updated as*

$$\nu_t = \widehat{\nu}_{t-1} + 1, \tag{3.19a}$$

$$R_t = \widehat{R}_{t-1}(\widehat{R}_{t-1} + 1)^{-1}, \tag{3.19b}$$

$$\mu_t = \widehat{\mu}_{t-1} + R_t \epsilon_t, \tag{3.19c}$$

$$\Lambda_t = \widehat{\Lambda}_{t-1} + \epsilon_t \epsilon_t^\top (\widehat{R}_{t-1} + 1)^{-1}, \tag{3.19d}$$

where $\epsilon_t = \xi_t - \Phi(x_{t-1:t}) - \widehat{\mu}_{t-1}$.

Proof. For a detailed derivation of these equations, see Lemma A.10. □

Lemma 3.4. *Assume the components of the mixture density (3.12) are given by* $p(\theta|\nu_t, V_t) = \mathcal{G}i\mathcal{W}(\cdot; \nu_t, R_t, \mu_t, \Lambda_t)$*; then, the approximate density is* $p(\theta|\widehat{\nu}_t, \widehat{V}_t) = \mathcal{G}i\mathcal{W}(\cdot; \widehat{\nu}_t, \widehat{R}_t, \widehat{\mu}_t, \widehat{\Lambda}_t)$*, and its statistics are computed according to*

$$\widehat{\Lambda}_t = \Omega_t^{-1} \widehat{\nu}_t, \tag{3.20a}$$

$$\widehat{\mu}_t = \Omega_t^{-1} \Big(\sum_{i=1}^{N} w_t^i \nu_t^i (\Lambda_t^i)^{-1} \mu_t^i \Big), \tag{3.20b}$$

$$\widehat{R}_t = \sum_{i=1}^{N} w_t^i \Big(R_t^i + \tfrac{1}{n_e} (\mu_t^i - \widehat{\mu}_t)^\top \nu_t^i (\Lambda_t^i)^{-1} (\mu_t^i - \widehat{\mu}_t) \Big), \tag{3.20c}$$

$$\textit{find } \widehat{\nu}_t \textit{ as the solution of } \log \tfrac{\widehat{\nu}_t}{2} - \sum_{k=1}^{n_e} \Psi\big(\tfrac{\widehat{\nu}_t + 1 - k}{2}\big) = \Xi_t, \tag{3.20d}$$

introducing the intermediate quantities

$$\Omega_t = \sum_{i=1}^{N} w_t^i \nu_t^i (\Lambda_t^i)^{-1},$$

$$\Xi_t = \sum_{i=1}^{N} w_t^i \Big(\log |\Lambda_t^i| - \sum_{k=1}^{n_e} \Psi\big(\tfrac{\nu_t^i + 1 - k}{2}\big) \Big) + \log \Big| \tfrac{\Omega_t}{2} \Big|,$$

where $|\cdot|$ *denotes the matrix determinant, and* $\Psi(\cdot)$ *is the digamma function.*

Proof. To obtain the approximate statistics (3.20), one first needs to derive the expected values of the unique entries in (3.17a), which yields

$$\mathsf{E}[\Sigma^{-1}|\tilde{\nu}, \tilde{V}] = \tilde{\nu} \tilde{\Lambda}^{-1}, \tag{3.21a}$$

$$\mathsf{E}[\Sigma^{-1}\mu|\tilde{\nu}, \tilde{V}] = \tilde{\nu} \tilde{\Lambda}^{-1} \tilde{\mu}, \tag{3.21b}$$

$$\mathsf{E}[\mu^\top \Sigma^{-1} \mu|\tilde{\nu}, \tilde{V}] = \tilde{R} n_e + \tilde{\mu}^\top \tilde{\nu} \tilde{\Lambda}^{-1} \tilde{\mu}, \tag{3.21c}$$

$$\mathsf{E}[\log |\Sigma||\tilde{\nu}, \tilde{V}] = \log |\tilde{\Lambda}| - n_e \log 2 - \sum_{k=1}^{n_e} \Psi\big(\tfrac{\tilde{\nu} + 1 - k}{2}\big). \tag{3.21d}$$

These expectations are proven in Propositions A.1 and A.2. After substituting (3.21) for the corresponding terms on the both sides of (3.8), we obtain (3.20). □

The presence of $\Psi(\cdot)$ in (3.20d) prevents us from finding an explicit expression for computing $\widehat{\nu}_t$. However, since (3.20d) is a convex function of $\widehat{\nu}_t$, the standard Newton-Raphson (NR) method is sufficient for finding the root. If v_t and w_t are mutually independent, we can choose the prior (3.16) as the product of two GiW densities with the statistics $(\nu_w, R_w, \mu_w, \Lambda_w)$ and $(\nu_v, R_v, \mu_v, \Lambda_v)$. The posterior and approximate versions of these statistics are then computed in the same way as with (3.19) and (3.20), respectively. Moreover, (3.11b) factorizes into the product of two Student's t densities $\mathrm{St}(x_t; \bar{\mu}_w, \bar{\Lambda}_w, \bar{\nu}_w)$ and $\mathrm{St}(y_t; \bar{\mu}_v, \bar{\Lambda}_v, \bar{\nu}_v)$, where the statistics are also computed as in (3.18). See also [167] for similar computations.

3.5 Experiments and Results

The present section demonstrates the performance of the proposed PBRBPF compared to a number of selected particle-based approaches for the parameter estimation in non-linear state-space models. We evaluate the estimation precision of the states and parameters by computing the root-mean-squared error (RMSE) and root-mean-norm-squared error (RMNSE) according to

$$\mathrm{RMSE} = \left(\tfrac{1}{T}\textstyle\sum_{t=1}^{T}(x_{t|t}^{M} - x_{t|t}^{N})^2\right)^{1/2}, \tag{3.22}$$

$$\mathrm{RMNSE} = \left(\tfrac{1}{T}\textstyle\sum_{t=1}^{T}||\theta_{t|t}^{M} - \theta_{t|t}^{N}||^2\right)^{1/2}, \tag{3.23}$$

with T denoting the amount of data samples and $||\cdot||$ labeling the Euclidean norm. Furthermore, $x_{t|t}^{N}$ and $\theta_{t|t}^{N}$ are the state and parameter estimates, respectively, obtained by the compared algorithm, and $x_{t|t}^{M}$ and $\theta_{t|t}^{M}$ are the corresponding 'ground truth' estimates computed by a more precise, offline, estimation method.

We compare the following methods: (i) the Rao-Blackwellized particle filter with linear computational complexity (RBPFN) [200]; (ii) the Rao-Blackwellized particle filter with quadratic computational complexity (RBPFN^2), also known as the Rao-Blackwellized marginal particle filter [134]; (iii) the projection-based RBPF (PBRBPF) proposed in Algorithm 14; (iv) the particle-based EM algorithm with linear computational complexity (PFEMN), the so-called path-based implementation [29]; (v) the particle-based EM algorithm with quadratic computational complexity (PFEMN^2) [48], the so-called forward-only implementation; (vi) the state augmentation particle filter with linear computational complexity (SAPFN), known as the Liu and West filter [136]; and (vii) the state augmentation particle filter with quadratic computational complexity (SAPFN^2), referred to as the nested particle filter [43].

3.5.1 Simulation Settings

The methods are compared on the standard benchmark SSM given by

$$x_t = 0.5x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}} + 8\cos(1.2t) + w_t, \tag{3.24}$$

$$y_t = 0.05x_t^2 + v_t, \tag{3.25}$$

where the variables $w_t \overset{IID}{\sim} \mathcal{N}(\cdot;\mu_w,\sigma_w^2)$ and $v_t \overset{IID}{\sim} \mathcal{N}(\cdot;\mu_v,\sigma_v^2)$ are assumed to be mutually independent. We aim to estimate μ_w, σ_w^2, μ_v, and σ_v^2, whose true values are 1, 2, 1, and 2, respectively. The initial state variable is distributed according to $x_1 \sim \mathcal{N}(\cdot;0,1)$. The amount of observations is $T = 2 \cdot 10^4$, and the number of particles is $N = 500$. The simulation is repeated 20 times with different observation sequences. For both noise variables, the initial statistics $(\nu_0, R_0, \mu_0, \Lambda_0)$ for the RBPFN, RBPFN^2, and PBRBPF algorithms are uniformly sampled from the respective intervals ($[8,12]$, $[2,4]$, $[-2,2]$, $[15,25]$). The NR procedure of the PBRBPF method is implemented with 10 iterations. The initial parameter estimates of $\theta = (\mu_w, \mu_v, \sigma_w^2, \sigma_v^2)$ for the PFEMN, PFEMN^2, SAPFN, and SAPFN^2 techniques are uniformly sampled from the respective intervals ($[-2,2]$, $[-2,2]$, $[0,4]$, $[0,4]$). The step size of the PFEMN and PFEMN^2 procedures satisfies $t^{-0.8}$. The parameter estimates of these EM approaches remain unchanged during the first 25 time steps. The SAPFN and SAPFN^2 algorithms are implemented with the kernel density-based proposal for parameter sampling [136]. All the compared algorithms are implemented in their bootstrap proposal setting (including the SAPFN algorithm). To compute the reference estimates in (3.22) and (3.23), we apply the particle Gibbs with ancestor sampling algorithm [132] with $M = 32$ particles and 200 iterations.

3.5.2 Results

The time evolution of the parameter estimates over the independent simulation runs is displayed in Figs. 3.1-3.3. The results indicate that the proposed PBRBPF algorithm outperforms the RBPFN method due to its lower bias and variance of the parameter estimates over the multiple simulation runs. From this observation, we can state that the proposed approach is less affected by the particle path degeneracy problem. The average time required to process all the observations with the PBRBPF and RBPFN algorithms was approximately 4.44 and 4.53 seconds, respectively. The PBRBPF algorithm delivers slightly higher variance than the RBPFN^2 procedure. Nevertheless, the bias provided by the PBRBPF algorithm is lower than the one of the RBPFN^2 technique. Given the fact that the RBPFN^2 approach is computationally highly demanding, we can expect that a small increase in the number of particles of the PBRBPF method can easily compensate for this slightly higher variance. The PFEMN algorithm is more competitive to the PBRBPF approach, albeit it still provides a higher bias and variance for most of the estimated parameters. The PFEMN^2 algorithm has very similar, and sometimes even lower, variance than the PBRBPF method. However, the bias provided by the PFEMN^2 technique is higher. The bias and variance offered by the SAPFN and SAPFN^2 algorithms is significantly worse compared to the remaining methods. We provide an explanation for this in the next section. Fig. 3.4 presents the trade-off between the estimation precision and computational time of the compared algorithms, demonstrating that the proposed PBRBPF algorithm achieves higher estimation accuracy compared to the remaining methods.

3.6 Discussion

The RBPFN approach possesses no strategy for counteracting the particle path degeneracy problem, expect relaying on suitable forgetting properties of the state-space model when computing the sufficient statistics. The RBPFN^2 method, however, is completely free of this problem, as it is based on the marginal particle filter-like approach [116] for computing the statistics associated with the posterior distribution of the parameters. Nevertheless, this approach still suffers from the error accumulation caused by computing the approximations similar to those presented in Lemma 3.4. The proposed PBRBPF method can be seen as sort of compromise between the RBPFN and RBPFN^2 procedures.

The PFEMN and PFEMN^2 algorithms can generally be seen as specific instances of particle-based methods that compute the smoothed additive functionals [30]. The convergence results presented in [175, 48] demonstrate that the asymptotic bias and variance of the path-based approximation of the smoothed additive functionals—as applied in the PFEMN method—satisfy the linear and quadratic growth with the iterations t, respectively. Similarly, it is shown in [48] that the asymptotic bias of the forward-only approximation of the smoothed additive functionals—as implemented in the PFEMN^2 approach—grows linearly, as in the case of the PFEMN procedure. However, the asymptotic variance of the PFEMN^2 algorithm grows also only linearly with t. See [104] for an empirical study on this matter. Indeed, a closer look at Fig. 3.2 reveals that the bias provided by the PFEMN and PFEMN^2 techniques is very similar, while the variance of the PFEMN^2 algorithm is markedly lower compared to the PFEMN counterpart. To the best of the author's knowledge, theoretical results of this type for the RBPFN and RBPFN^2 are still missing.

Although the performance of the SAPFN and SAPFN^2 algorithms is unsatisfactory with the given number of particles, their main advantage lies in that they can be applied to state-space models without any specific structure. The poor output of the SAPFN approach is caused by the well-known problem with the particle depletion. The empirical evidence often indicates that the SAPFN method can offer a substantially improved performance when $N \gg T$. This requirement is, however, inappropriate for purely online scenarios. The SAPFN^2 approach utilizes two nested layers of particle filters: an upper layer for drawing N parameter samples, and a lower layer formed by N local particle filters that are computed conditionally on each parameter sampled in the upper layer. The weights in the upper layer are computed based on the empirical approximation of the predictive likelihood $p(y_t|\theta, y_{1:t-1})$. In the present simulation scenario, with the given number of particles, this approximation suffers from a substantial bias and variance, and it therefore makes the estimation precision of the SAPFN^2 approach rather poor.

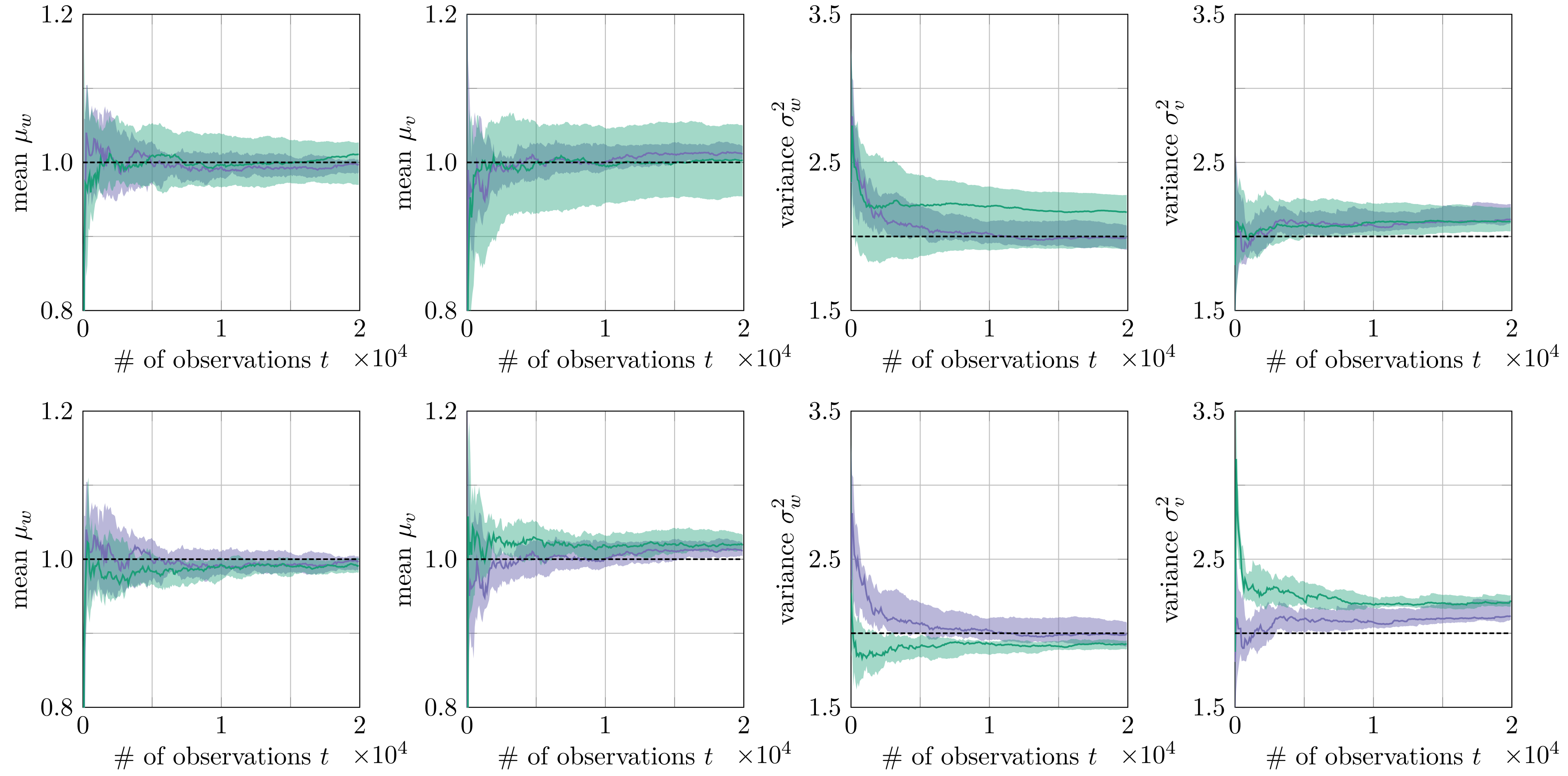

Fig. 3.1: The parameter estimates versus the number of observations. Top: PBRBPF (—) and RBPFN (—) [200]. Bottom: PBRBPF (—) and RBPFN^2 (—) [134]. The results are averaged over 20 independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range. The true parameter values are indicated with the dashed line (------).

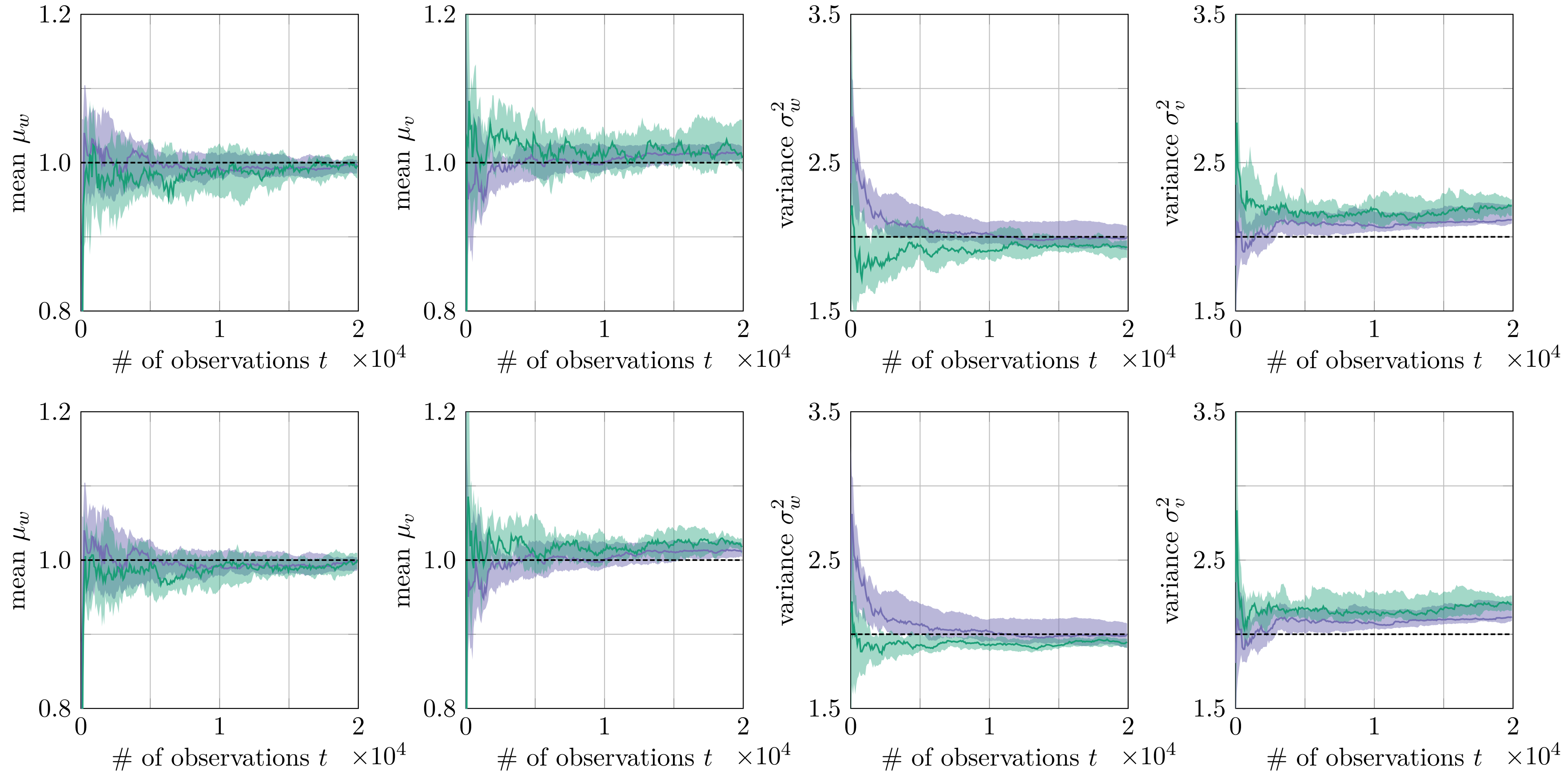

Fig. 3.2: The parameter estimates versus the number of observations. Top: PBRBPF (—) and PFEMN (—) [29]. Bottom: PBRBPF (—) and PFEMN^2 (—) [48]. The results are averaged over 20 independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range. The true parameter values are indicated with the dashed line (------).

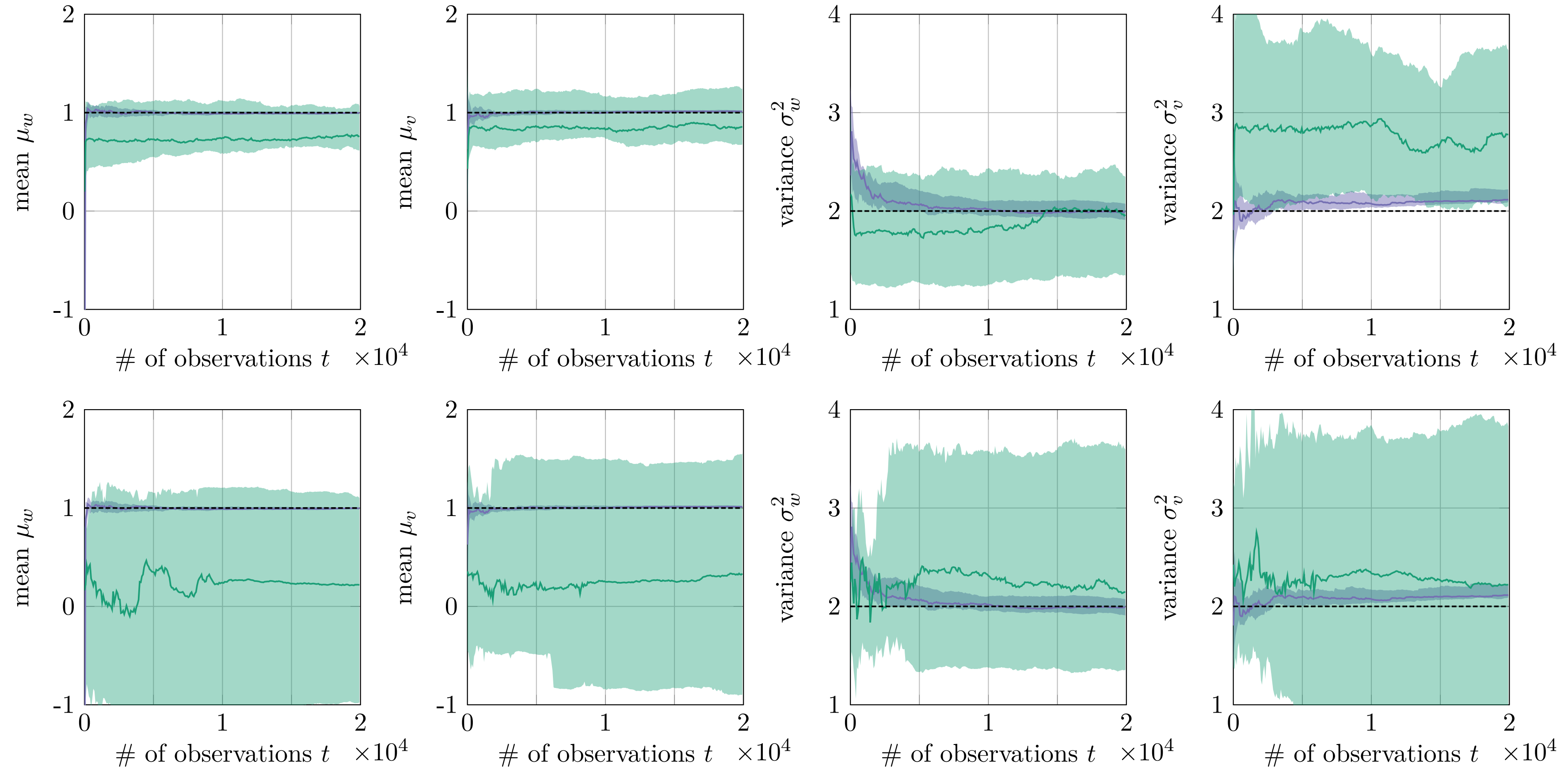

Fig. 3.3: The parameter estimates versus the number of observations. Top: PBRBPF (——) and SAPFN (——) [136]. Bottom: PBRBPF (——) and SAPFN^2 (——) [43]. The results are averaged over 20 independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range. The true parameter values are indicated with the dashed line (------).

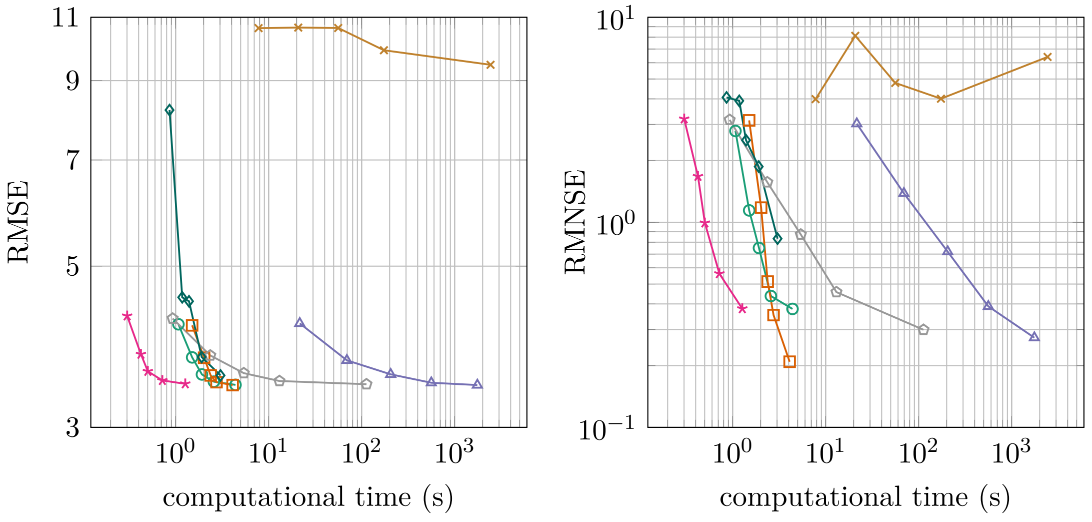

Fig. 3.4: The state RMSE (3.22) and the parameter RMNSE (3.23) versus the computational time (in seconds). The compared algorithm are RBPFN (—○—), PBRBPF (—□—), RBPFN^2 (—△—), PFEMN (—∗—), PFEMN^2 (—⬠—), SAPFN (—◆—), and SAPFN^2 (—×—). The number of particles N takes values in $(32, 64, 128, 256, 512)$. The results are averaged over 20 independent simulation runs, with the solid line being the median.

4 A PARTICLE FILTER TO ESTIMATE TIME-VARYING PARAMETERS IN CONDITIONALLY CONJUGATE STATE-SPACE MODELS

The identification of slowly-varying parameters in dynamical systems constitutes a practically important task in a wide range of applications. The present chapter addresses this problem based on the Bayesian learning and sequential Monte Carlo (SMC) methodology. The proposed approach utilizes an algebraic structure of a specific class of nonlinear and non-Gaussian state-space models in order to enable Rao-Blackwellization of the parameters, thus involving a finite-dimensional sufficient statistic for each particle trajectory into the resulting algorithm. However, relying on basic SMC methods, such techniques are known to suffer from the particle path degeneracy problem. We propose to use alternative stabilized forgetting, which not only allows us to deal with the slowly-varying parameters but also to counteract the degeneracy problem. An experimental study proves the efficiency of the introduced Rao-Blackwellized particle filter compared to some related approaches.

4.1 Introduction

4.1.1 Context

The task of *online* SMC parameter estimation in non-linear state-space models has attracted substantial attention in the last years. Considerable effort has been devoted to maximum likelihood methods, where the aim is to maximize the likelihood $p_\theta(y_{1:t})$ of observed data sequence $y_{1:t} \coloneqq (y_1, \ldots, y_t)$ with respect to some fixed parameterization θ. An algorithmic solution in such cases commonly relies on the computation of expected values of smoothed additive functionals [30], which requires the complete data likelihood $p_\theta(x_{1:t}, y_{1:t})$ to belong to the exponential family [12], where $x_{1:t}$ denotes an unobserved state sequence. The main stream of research in this respect includes the gradient ascent [175] and expectation maximization (EM) methods [28]. However, these SMC-based approaches suffer from the particle path degeneracy problem [6, 94]. Recently, it was recognized in [48] that the forward smoothing algorithm can overcome this issue at the cost of $\mathcal{O}(N^2)$ operations, where N stands for the number of particles. The results of [164] show that the forward smoothing can actually be performed with $\mathcal{O}(N)$ operations by adapting the accept-reject backward sampling of [53].

Bayesian methods interpret unknown parameters as random variables and provide their full description in terms of the posterior density $p(\theta|y_{1:t})$. From this perspective, the earliest SMC approaches apply a particle filter to an augmented state variable $\bar{x}_t = (x_t, \theta)$ while considering a constant model of parameter variations $\theta_t = \theta_{t-1}$. Since the model of constant parameter variations lacks any forgetting properties [30], the diversity of the particle population representing θ decreases with successful resampling steps. The problem is commonly treated by introducing a jittering noise into the model of parameter evolutions [84]. However, a straightforward application of jittering can make the posterior density $p(\theta|y_{1:t})$ unnecessarily diffused. This was addressed in [115] by systematically decreasing the noise variance and later improved by alleviating the artificial variance inflation in [136]. But the simple addition of a jittering noise with a decreasing variance is not always efficient, as it may be difficult to guess a compromise between the number of particles being used and the rate at which the variance should decrease. The advantage of state augmentation techniques is that they can be applied to models without a specific structure. Considering a model with parameters respecting some structure in such a manner that the density $p(\theta|x_{1:t}, y_{1:t})$ is algebraically tractable, the paper [200] proposes to integrate out the parameters and to run a particle filter only for the marginal density $p(x_{1:t}|y_{1:t})$. For each particle trajectory, sampled from this marginal, the density $p(\theta|x_{1:t}, y_{1:t})$ is evaluated in terms of updating the sufficient statistics, which then serves for the parameter estimation. However, this online approach, too, suffers from the particle path degeneracy problem, resulting in a poor approximation of the posterior $p(\theta|x_{1:t}, y_{1:t})$. The related paper [167] imposes exponential forgetting [122] into this algorithm in order to facilitate the estimation of time-varying parameters and counteract the degenerate behavior.

4.1.2 Contributions

This chapter proposes a sequential Monte Carlo-based algorithm which exploits the algebraically tractable substructure of a special class of nonlinear state-space models, here referred to as conditionally conjugate state-space models. A characteristic feature of these models consists in that they facilitate the computation of $p(\theta|x_{1:t}, y_{1:t})$ under a close-form solution. The algorithm is—in its basic structure—similar to the one proposed in [200] but offers an ability to trace time-varying parameters. However, compared to the similar work [167], we accomplish this by utilizing a different forgetting strategy which is known as the alternative stabilized forgetting [108]. We demonstrate that the proposed algorithm outperforms this previous approach in terms of estimation accuracy and computational time.

4.2 Background

4.2.1 Problem Formulation

In this chapter, we are concerned with discrete-time state-space models (SSMs) given by the joint probability density

$$p_{\theta_t}(y_t, x_t|x_{t-1}) = g_{\theta_t}(y_t|x_t)f_{\theta_t}(x_t|x_{t-1}), \tag{4.1}$$

where $x_t \in \mathsf{X} \subseteq \mathbb{R}^{n_x}$ and $y_t \in \mathsf{Y} \subseteq \mathbb{R}^{n_y}$ denote the state and observation variables, respectively. Furthermore, g_{θ_t} and f_{θ_t} constitute observation and state-transition models, with $\theta_t \in \Theta \subseteq \mathbb{R}^{n_\theta}$ being some unknown time-varying parameters. The initial step assumes that the state and parameter variables are distributed as $x_1 \sim p_{\theta_1}(x_1)$ and $\theta_1 \sim p(\theta_1)$. We are particularly interested in a specific class of SSMs which allows us to express (4.1) by the exponential family [12] density

$$\begin{aligned} p_{\theta_t}(y_t, x_t|x_{t-1}) = \exp\{&\langle \eta(\theta_t), s_t(x_{t-1}, x_t, y_t)\rangle \\ &- \zeta(\theta_t) + \log h(x_{t-1}, x_t, y_t)\}, \end{aligned} \tag{4.2}$$

where (η, ζ) and (s_t, h) are functions of appropriate dimensions, defined on Θ and $\mathsf{X}^2 \times \mathsf{Y}$, respectively, and $\langle \cdot, \cdot \rangle$ is the inner product. Due to the fact that (4.2) is analytically intractable with respect to the nonlinear functions (s_t, h) but tractable with respect to the parameter functions (η, ζ), we refer to (4.2) as the conditionally conjugate state-space model (CCSSM), alternatively known as the conditionally conjugate latent process model [212, 187]. The key characteristic of (4.2) consists in that, if we choose the conjugate prior density

$$\begin{aligned} p(\theta_t|\nu_{t|t-1}, V_{t|t-1}) = \exp\{&\langle \eta(\theta_t), V_{t|t-1}\rangle - \nu_{t|t-1}\zeta(\theta_t) \\ &- \log \mathcal{I}(\nu_{t|t-1}, V_{t|t-1})\}; \end{aligned} \tag{4.3}$$

then, we can compute the posterior density, $p(\theta_t|x_{1:t}, y_{1:t}) \coloneqq p(\theta_t|\nu_{t|t}, V_{t|t})$, analytically. In (4.3), $V_{t|t-1}$ is the extended information matrix, $\nu_{t|t-1}$ is the number of degrees of freedom, and $\mathcal{I}$ denotes the normalizing constant. Under this choice, the posterior density $p(\theta_t|\nu_{t|t}, V_{t|t})$ reproduces the form of (4.3), with the statistics being updated according to the closed-form formulae

$$V_{t|t} = V_{t|t-1} + s_t(x_{t-1}, x_t, y_t), \tag{4.4a}$$

$$\nu_{t|t} = \nu_{t|t-1} + 1. \tag{4.4b}$$

Fundamental probability densities, including Poisson, Gaussian, and exponential, fit into the generic form delineated by (4.2).

The objective of this chapter consists in designing an online algorithm for computing the joint posterior density $p(x_t, \theta_t|y_{1:t})$ while assuming the model (4.2). There are, however, two main obstacles in achieving this goal: (i) the nonlinear functions (s_t, h) prevent us from computing the joint posterior density analytically, and (ii) the parameter time-evolution model $p(\theta_t|\theta_{t-1})$ is unknown. To deal with the first problem, we apply the particle filters, as they constitute a theoretically [217] and practically [57] well-established tool for approximating highly nonlinear probability densities. To resolve the second one, we incorporate—for the first time—the concept of alternative stabilized forgetting [108] into the context of particle filter-based estimation of slowly-varying parameters.

4.2.2 Sequential Monte Carlo Methods

The SMC methodology [47] embodies a versatile approach for approximating a flow of densities $(\pi_t)_{t=1}^T$ defined on a sequence of spaces of increasing dimensions $(\mathsf{X}^t)_{t=1}^T$. These densities are assumed to be known only up to the normalization constant, $\pi_t(x_{1:t}) \propto \gamma_t(x_{1:t})$. At the previous time instance, $t-1$, an SMC algorithm targets π_{t-1} by a weighted particle system $(x_{1:t-1}^i, w_{t-1}^i)_{i=1}^N$, where w_{t-1}^i is a non-negative importance weight and $x_{1:t-1}^i$ is an associated particle trajectory. The weighted particle system is propagated to time t by combining the sequential importance sampling and resampling techniques. Specifically, when a certain condition is fulfilled (e.g., the effective sample size [30] is below a specified threshold), resampling is performed by drawing ancestor indexes as $a_t^i \sim \mathbb{P}(a_t^i = j) = w_{t-1}^j$. After that, the weights $w_{t-1}^{1:N}$ are set to $1/N$. If the condition is not fulfilled, we assign $a_t^i = i$ and keep the weights unmodified. Subsequently, based on sampling from a proposal density $x_t^i \sim q_t(\cdot|x_{1:t-1}^{a_t^i})$, the previous particle trajectories are extended, $x_{1:t}^i := (x_t^i, x_{1:t-1}^{a_t^i})$. The iteration is completed by updating the importance weights

$$w_t^i \propto W_t(x_{1:t}^i) w_{t-1}^i, \tag{4.5}$$

where

$$W_t(x_{1:t}) := \frac{\gamma_t(x_{1:t})}{\gamma_{t-1}(x_{1:t-1}) q_t(x_t|x_{1:t-1})}. \tag{4.6}$$

These operations facilitate the sequential construction of an empirical distribution approximating π_t, that is,

$$\pi_t^N(dx_{1:t}) = \sum_{i=1}^N w_t^i \delta_{x_{1:t}^i}(dx_{1:t}),$$

where δ_x stands for the Dirac delta measure located at x. The algorithm starts by sampling from the initial proposal density $x_1^i \sim q_1(x_1)$ and calculating the weights according to $w_1^i \propto \gamma_1(x_1^i)/q_1(x_1^i)$. Considering the filtering context, we choose $\pi_t(x_{1:t})$ and $\gamma_t(x_{1:t})$ to represent the joint state posterior density of the states $p(x_{1:t}|y_{1:t})$ and the complete data density $p(x_{1:t}, y_{1:t})$, respectively. For a more thorough description of SMC methods, see [56].

4.2.3 Decision-Making Approximation of Probability Densities

Let us assume that there exists an exact density $p(\theta)$ which is supposed to contain all available information about our model. The objective consists in finding its approximation $\widehat{p}(\theta)$. Addressing this task in terms of a statistical decision-making problem [46], we define parameter space to coincide with the original parameter space Θ, and also determine decision space as a set of feasible densities P. To assess a loss incurred by taking a particular decision $\bar{p}(\theta) \in \mathsf{P}$, with the value of θ being materialized, we introduce a loss function $l(\theta, \bar{p}(\theta)) : \Theta \times \mathsf{P} \to \mathbb{R}_{\geq 0}$, representing a mapping from the product space $\Theta \times \mathsf{P}$ to the set of non-negative reals $\mathbb{R}_{\geq 0}$. However, the latent nature of the parameters requires us to integrate over their possible values and thus introduce the expected loss. The expectation shall be taken with respect to the exact density $\mathsf{E}[l(\theta, \bar{p}(\theta))] = \int p(\theta) l(\theta, \bar{p}(\theta)) d\theta$ in order to be honest in terms of reporting our beliefs. The rationale behind this idea is that the expected loss should attain its minimum only if the decision stands for the exact density $\bar{p}(\theta) = p(\theta)$. Additionally, we require the loss function to depend on $\bar{p}(\theta)$ only through its realized value $l(\theta, \bar{p}(\cdot)) = l(\theta, \bar{p}(\theta))$. As demonstrated in [16], the logarithmic loss $l(\theta, \bar{p}(\theta)) = -\ln \bar{p}(\theta)$ preserves these properties. The above construction of the expected loss yields the Kerridge inaccuracy [111],

$$d_K(p, \bar{p}) = -\mathsf{E}[\ln \bar{p}(\theta)] = \int_\Theta p(\theta) \ln \bar{p}(\theta) d\theta. \tag{4.7}$$

Our intuition about the exact density is reflected by the expectation that $p(\theta)$ can represent a finite number of user-designed possibilities $p_j(\theta)$, where $j = 0, \dots, M$. Therefore, considering $p(\theta)$ random and taking the expected value of (4.7) over the law $\mathbb{P}(p = p_j) = \lambda_j$, we obtain

$$\mathsf{E}[d_K(p, \bar{p})] = -\sum_{j=0}^{M} \lambda_j \mathsf{E}_j[\ln \bar{p}(\theta)] = -\sum_{j=0}^{M} \lambda_j \int_\Theta p_j(\theta) \ln \bar{p}(\theta) d\theta. \tag{4.8}$$

Now, given the above construction, the aim is to minimize the loss (4.8) with respect to $\bar{p}$. The resulting minimizer $\widehat{p}$ is an approximation computed as a compromise among a number of possible densities p_j. We want to approximate the exact density $p(\theta)$ by a density which belongs to the exponential family, and therefore, we define $\bar{p}(\theta) := p(\theta|\bar{\nu}, \bar{V})$. Consequently, we need to solve the optimization problem

$$\widehat{\nu}, \widehat{V} \in \operatorname*{argmin}_{\bar{\nu}, \bar{V}} d_K\Big(\sum_{j=0}^{M} \lambda_j p_j(\theta), \exp\{\langle \eta(\theta), \bar{V} \rangle - \bar{\nu}\zeta(\theta) - \ln \mathcal{I}(\bar{\nu}, \bar{V})\}\Big).$$

A solution for the approximate statistics $\widehat{\nu}$ and $\widehat{V}$ is obtained, respectively, from the formulae

$$\mathsf{E}[\eta(\theta)|\widehat{\nu},\widehat{V}] = \sum_{j=0}^{M} \lambda_j \mathsf{E}_j[\eta(\theta)], \tag{4.9a}$$

$$\mathsf{E}[\zeta(\theta)|\widehat{\nu},\widehat{V}] = \sum_{j=0}^{M} \lambda_j \mathsf{E}_j[\zeta(\theta)], \tag{4.9b}$$

which result from taking the partial derivatives of d_K with respect to the statistics $(\bar{\nu}, \bar{V})$ and equating them to zero. Note the densities p_j have not yet been specified into any family.

4.3 A Rao-Blackwellized Particle Filter with Alternative Stabilized Forgetting

The design of the method is based on factorizing the joint posterior density of the parameters and states according to

$$p(\theta_t, x_{1:t}|y_{1:t}) = p(\theta_t|x_{1:t}, y_{1:t})p(x_{1:t}|y_{1:t}). \tag{4.10}$$

The above rearrangement is convenient in situations where the parameters of a state-space model admit algebraically tractable substructures, which is exactly the case of the model class considered here. The conditional factor $p(\theta_t|x_{1:t}, y_{1:t})$ can then be evaluated under a closed-form solution. However, due to the presence of the nonlinear functions in (4.2), the marginal factor $p(x_{1:t}|y_{1:t})$ is intractable—possibly describing a highly non-linear dynamic—and we therefore seek a proper approximation. The factorization (4.10) is the common setup for a specific class of SMC methods referred to as Rao-Blackwellized particle filters [55], where we split the model into algebraically *tractable* and *intractable* part.

4.3.1 The Basic Structure

The tractable part—the conditional factor in (4.10)—can simply be computed based on the law of conditional probability

$$p(\theta_t|\nu_{t|t}, V_{t|t}) \propto p_{\theta_t}(y_t, x_t|x_{t-1})p(\theta_t|\nu_{t|t-1}, V_{t|t-1}), \tag{4.11}$$

where we apply the definition $p(\theta_t|x_{1:t}, y_{1:t}) := p(\theta_t|\nu_{t|t}, V_{t|t})$ due to our restriction on the specific model class. Here, $\propto$ denotes the equality up to proportionality factor which is given by the joint predictive density of the states and observations,

$$p(y_t, x_t|\nu_{t|t-1}, V_{t|t-1}) = \int_{\Theta} p_{\theta_t}(y_t, x_t|x_{t-1})p(\theta_t|\nu_{t|t-1}, V_{t|t-1})d\theta_t. \tag{4.12}$$

The predictive density of the parameters, containing the unknown parameter time-evolution model, $p(\theta_t|\theta_{t-1})$, is expressed as

$$p(\theta_t|\nu_{t|t-1}, V_{t|t-1}) = \int_{\Theta} p(\theta_t|\theta_{t-1})p(\theta_{t-1}|\nu_{t-1|t-1}, V_{t-1|t-1})d\theta_{t-1}. \tag{4.13}$$

If we choose the state-space model and prior density in (4.11) according to (4.2) and (4.3), then the computations associated with (4.11) reduce to only updating the statistics (4.4). The formulae (4.11) and (4.13) are usually referred to as the data and time step, respectively.

The intractable part—the marginal factor in (4.10)—is approximated by the SMC framework discussed in the previous section. This only requires us to specify $\pi_t(x_{1:t}) \coloneqq p(x_{1:t}|y_{1:t})$, from which it follows that $\gamma(x_{1:t}) \coloneqq p(x_{1:t}, y_{1:t})$. Then, in the present context, the weight function (4.6) becomes

$$W_t(x_{1:t}) \coloneqq \frac{p(y_t, x_t|\nu_{t|t-1}, V_{t|t-1})}{q_t(x_t|x_{1:t-1})}, \tag{4.14}$$

where the marginal density in the numerator is given by (4.12). From (4.14) and (4.5), we see that the basic flow of the SMC algorithm requires us to compute the predictive density and involved statistics for $i = 1, \ldots, N$. These requirements are the only additional steps to the basic structure of the SMC method presented in Section 4.2.2. The rest of the operations of this algorithm remains unchanged.

Remark 4.1. *In the case of computing the static parameters, we choose* $p(\theta_t|\theta_{t-1}) \coloneqq \delta_{\theta_{t-1}}(\theta_t)$*, which simplifies (4.13) to* $p(\theta_t|\nu_{t|t-1}, V_{t|t-1}) = p(\theta_t|\nu_{t-1|t-1}, V_{t-1|t-1})$*, and the statistics are then simply constant in the time step,* $\nu_{t|t-1} = \nu_{t-1|t-1}$ *and* $V_{t|t-1} = V_{t-1|t-1}$*. Such an approach coincides with that of [200], which was further elaborated in [34] and subjected to a recent examination within [38]. The reason for this consists in that standard SMC methods provide a poor approximation of the joint state posterior density* $p(x_{1:t}|y_{1:t})$*, which is the consequence of the problem known as the particle path degeneracy [94]. Specifically, successive resampling steps decrease the diversity of the particle system so that* $p(x_{1:m}|y_{1:t})$ *is—for a large enough difference* $n - m$*—approximated by only a single particle [6]. Since the computation of sufficient statistics in the above mentioned approach relies on* $p(x_{1:t}|y_{1:t})$*, the estimates of* θ_t *usually experience considerable variance over multiple simulation runs.*

4.3.2 Alternative Stabilized Forgetting

In the case of estimating time-varying parameters, the lack of knowledge of the parameter time-evolution model during the design of estimation techniques is more the rule than the exception. This fact renders the computation of the predictive density (4.13) problematic. A pragmatic approach consists in choosing the time-evolution model to be the identity kernel $p(\theta_t|\theta_{t-1}) \coloneqq \delta_{\theta_{t-1}}(\theta_t)$, ignoring the slowly-varying character of the parameters. However, such a choice makes the algorithm insensitive to the parameter changes. Instead of having the time-evolution model at disposal, we propose to use alternative stabilized forgetting [108], which addresses the absence of the model in terms of finding a compromise between possible candidates of the predictive density (4.13).

In the alternative stabilized forgetting, we have two possible representations of the predictive density. The first one is based on the time-evolution model of constant parameters $p(\theta_t|\theta_{t-1}) \coloneqq \delta_{\theta_{t-1}}(\theta_t)$, i.e., it constitutes the posterior density from the previous time step, $p_0(\theta_t) \coloneqq p(\theta_t|\nu_{t-1|t-1}, V_{t-1|t-1})$. The second one embodies the alternative to the case of constant parameters and represents our knowledge about their assumed changes, e.g., the worst case scenario. The alternative predictive density is supposed to be in the exponential family, $p_1(\theta_t) \coloneqq p(\theta_t|\nu_A, V_A)$, and can be either time-variant or invariant. Here, we choose the invariant case for simplicity. These densities constitute two hypotheses about the exact predictive density. We assign a probability to each one of these, reflecting our beliefs in the possibility that they represent the exact predictive density. The probability λ is associated with the hypothesis that the parameters do not change and the complementary probability $1-\lambda$ with the alternative hypothesis. The probability λ is called the forgetting factor [122]. Consequently, we have the mixture density

$$p(\theta_t|\nu_{t|t-1}, V_{t|t-1}) \coloneqq \lambda p(\theta_t|\nu_{t-1|t-1}, V_{t-1|t-1}) + (1-\lambda)p(\theta_t|\nu_A, V_A). \tag{4.15}$$

The statistics $(\nu_{t|t-1}, V_{t|t-1})$ cannot be computed exactly in the case of (4.15). Therefore, we apply the framework from Section 4.2.3, where we make the choice $\bar{p}(\theta_t) \coloneqq p(\theta_t|\nu_{t|t-1}, V_{t|t-1})$, leading to

$$\mathsf{E}[\eta(\theta_t)|\widehat{\nu}^i_{t|t-1}, \widehat{V}^i_{t|t-1}] = \lambda\mathsf{E}[\eta(\theta_t)|\nu^i_{t-1|t-1}, V^i_{t-1|t-1}] + (1-\lambda)\mathsf{E}[\eta(\theta_t)|\nu_A, V_A], \tag{4.16a}$$
$$\mathsf{E}[\zeta(\theta_t)|\widehat{\nu}^i_{t|t-1}, \widehat{V}^i_{t|t-1}] = \lambda\mathsf{E}[\zeta(\theta_t)|\nu^i_{t-1|t-1}, V^i_{t-1|t-1}] + (1-\lambda)\mathsf{E}[\zeta(\theta_t)|\nu_A, V_A], \tag{4.16b}$$

thus being a result which follows from (4.9) for $M = 2$. The statistics $(\widehat{\nu}^i_{t|t-1}, \widehat{V}^i_{t|t-1})$—computed as the solution of (4.16)—can now be used in (4.4) for $i = 1, \ldots, N$.

The resulting method is summarized in Algorithm 15, where all i-dependent operations are performed for $i = 1, \ldots, N$. Additionally, we use the convention that $s_1(x_0, x_1, y_1) \coloneqq s_1(x_1, y_1)$.

Remark 4.2. *As discussed in Remark 4.1, choosing the parameter time-evolution as the identity kernel $p(\theta_t|\theta_{t-1}) \coloneqq \delta_{\theta_{t-1}}(\theta_t)$ makes the resulting algorithmic solution too sensitive to the particle path degeneracy. An additional benefit in utilizing the alternative stabilized forgetting consists in that it counteracts the particle path degeneracy problem, as will be demonstrated in the experimental part of this chapter.*

Algorithm 15 RBPF with Alternative Stabilized Forgetting (RBPFASF)

```
A. Initial step: $(t = 1)$
    1. Set $(\widehat{\nu}^i_{1|0}, \widehat{V}^i_{1|0}, \nu_A, V_A)$, and $\lambda$.
    2. Sample $x^i_1 \sim q_1(\cdot)$.
    3. Compute $w^i_1 \propto W_1(x^i_1)$.
B. Recursive step: $(t = 2, \dots, T)$
    1. If $N_{\text{eff}} \leq N_{\text{th}}$, sample $a^i_t$ with $\mathbb{P}(a^i_t = j) = w^j_{t-1}$ and set $\bar{w}^i_{t-1} = 1/N$.
       Else, set $a^i_t = i$ and $\bar{w}^i_{t-1} = w^i_{t-1}$.
    2. Sample $x^i_t \sim q_t(\cdot|x^{a^i_t}_{1:t-1})$.
    3. Compute $w^i_t \propto W_t(x^i_{1:t})\bar{w}^i_{t-1}$ using (4.14).
C. Common step: $(t \geq 1)$
    1. Compute $\nu^i_{t|t} = \widehat{\nu}^{a^i_t}_{t|t-1} + 1$ and $V^i_t = \widehat{V}^{a^i_t}_{t|t-1} + s_t(x^{a^i_t}_{t-1}, x^i_t, y_t)$.
    2. Use $(\nu^i_{t|t}, V^i_{t|t})$, $(\nu_A, V_A)$, and $\lambda$ in (4.16) to compute $(\widehat{\nu}^i_{t+1|t}, \widehat{V}^i_{t+1|t})$.
```

The rationale behind this idea is that the alternative stabilized forgetting discounts the past values in the statistics (ν, V) so that their computation is based on a more recent and more satisfactorily approximated part of $p(x_{1:t}|y_{1:t})$. Different forgetting strategies were adopted, e.g., in [29] and [167], which we discus later on in this chapter.

4.4 Estimating Time-Varying Gaussian Noise Parameters

This section specifies the generic structure of Algorithm 15 to the case of estimating time-varying parameters of a nonlinear state-space model with additive Gaussian noise variables. The resulting algorithm can therefore be understood as an adaptive Rao-Blackwellized particle filter. We consider a state-space model given in the form

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} a(x_{t-1}) \\ b(x_t) \end{bmatrix} + \begin{bmatrix} v_t \\ w_t \end{bmatrix} \Leftrightarrow \xi_t = \Phi(x_{t-1:t}) + e_t, \tag{4.17}$$

where the vectors ξ_t, $\Phi(x_{t-1:t})$, and e_t contain the state and observation variables, nonlinear state-transition and observation functions, and state and observation noise variables, respectively. The composite noise vector e_t is a Gaussian, independent and identically distributed (IID), random variable $e_t \overset{IID}{\sim} \mathcal{N}(\mu_t, \Sigma_t)$ with the mean vector μ_t and covariance matrix Σ_t. Our aim is to estimate the parameters $\theta_t = (\mu_t, \Sigma_t)$.

In the context of model (4.17), the densities (4.2) and (4.3) are given by

$$p_{\theta_t}(y_t, x_t|x_{t-1}) = \mathcal{N}(\cdot; \Phi(x_{t-1:t}) + \mu_t, \Sigma_t), \tag{4.18}$$

$$p(\theta_t|\widehat{\nu}_{t|t-1}, \widehat{V}_{t|t-1}) = \mathcal{N}i\mathcal{W}(\cdot; \widehat{\nu}_{t|t-1}, \widehat{R}_{t|t-1}, \widehat{\mu}_{t|t-1}, \widehat{\Lambda}_{t|t-1}), \tag{4.19}$$

where—as discussed previously—we use the notation for the approximate statistics as they replace the original statistics in the step C2 in Algorithm 15. The first density results from a direct application of the change of variables formula [21] to the density of the noise term, $\mathcal{N}(e_t;\mu,\Sigma)$. The second density—the Gauss-inverse-Wishart (GiW) density—is the conjugate prior for the case of estimating the mean and covariance of the Gaussian density.

To establish the link between (4.18) and (4.2), and to prepare grounds for the subsequent developments, we present the following lemma. For the relation between the extended information matrix V of (4.3) and the statistics (R,μ,Λ) of (4.19) we refer the reader to Lemma A.7.

Lemma 4.1. *For the CCSSM (4.2) defined by (4.18), the functions (η,ζ,s,h) are expressed according to*

$$\eta(\theta_t)=\begin{bmatrix}\Sigma_t^{-1} & \Sigma_t^{-1}\mu_t\\ \mu_t^\top\Sigma_t^{-1} & \mu_t^\top\Sigma_t^{-1}\mu_t\end{bmatrix},\qquad \zeta(\theta_t)=\frac{1}{2}\log|\Sigma_t|. \tag{4.20a}$$

$$s_t(x_{t-1:t},y_t)=\begin{bmatrix}e_t\\1\end{bmatrix}\begin{bmatrix}e_t\\1\end{bmatrix}^\top,\quad h(x_{t-1:t},y_t)=(2\pi)^{-\frac{n_e}{2}}. \tag{4.20b}$$

Proof. The result can simply be obtained by utilizing the fact that the trace operator in the exponent of (4.18) is invariant under the cyclic permutations. □

The formulae that specify the updating of the statistics in step C1 of Algorithm 15 are presented in the next lemma.

Lemma 4.2. *Let the densities (4.2) and (4.3) be given by (4.18) and (4.19), respectively; then, the posterior density (4.11) becomes $p(\theta_t|\nu_{t|t},V_{t|t})=\mathcal{G}i\mathcal{W}(\cdot;\nu_{t|t},R_{t|t},\mu_{t|t},\Lambda_{t|t})$, where the statistics are computed according to*

$$\nu_{t|t}=\widehat{\nu}_{t|t-1}+1, \tag{4.21a}$$
$$R_{t|t}=\widehat{R}_{t|t-1}(\widehat{R}_{t|t-1}+1)^{-1}, \tag{4.21b}$$
$$\mu_{t|t}=\widehat{\mu}_{t|t-1}+R_{t|t}\epsilon_t, \tag{4.21c}$$
$$\Lambda_{t|t}=\widehat{\Lambda}_{t|t-1}+\epsilon_t\epsilon_t^\top(\widehat{R}_{t|t-1}+1)^{-1}, \tag{4.21d}$$

where $\epsilon_t=\xi_t-\Phi(x_{t-1:t})-\widehat{\mu}_{t|t-1}$.

Proof. The formulae are derived in Lemma A.10. □

Similarly, the formulae implementing the alternative stabilized forgetting in step C2 of Algorithm 15 are introduced in the following lemma.

Lemma 4.3. *Let us consider the components of the mixture density (4.15) are defined by*

$$p(\theta_t|\nu_{t-1|t-1},V_{t-1|t-1})=\mathcal{G}i\mathcal{W}(\cdot;\nu_{t-1|t-1},R_{t-1|t-1},\mu_{t-1|t-1},\Lambda_{t-1|t-1}),$$
$$p(\theta_t|\nu_A,V_A)=\mathcal{G}i\mathcal{W}(\cdot;\nu_A,R_A,\mu_A,\Lambda_A);$$

then, the alternative stabilized forgetting at the current time step is performed by computing the approximate statistics $(\widehat{\nu}, \widehat{V})$ *based on (4.16), which results in*

$$\widehat{\Lambda} = \Omega^{-1}\widehat{\nu}, \tag{4.22a}$$

$$\widehat{\mu} = \Omega^{-1}\Big(\lambda\nu\Lambda^{-1}\mu + (1-\lambda)\nu_A\Lambda_A^{-1}\mu_A\Big), \tag{4.22b}$$

$$\begin{aligned}\widehat{R} = \lambda\Big(R_A + \tfrac{1}{n_e}(\mu_A - \widehat{\mu})^\top \nu_A\Lambda_A^{-1}(\mu_A - \widehat{\mu})\Big)&\\ + (1-\lambda)\Big(R + \tfrac{1}{n_e}(\mu - \widehat{\mu})^\top \nu\Lambda^{-1}(\mu - \widehat{\mu})\Big),&\end{aligned} \tag{4.22c}$$

$$\textit{find } \widehat{\nu} \textit{ as the solution of } \log\tfrac{\widehat{\nu}}{2} - \sum_{k=1}^{n_e}\Psi\big(\tfrac{\widehat{\nu}+1-k}{2}\big) = \Xi, \tag{4.22d}$$

where we define the quantities

$$\Omega = \lambda\nu\Lambda^{-1} + (1-\lambda)\nu_A\Lambda_A^{-1},$$

$$\Xi = \lambda\Big(\log|\Lambda| - \sum_{k=1}^{n_e}\Psi\big(\tfrac{\nu+1-k}{2}\big)\Big) + (1-\lambda)\Big(\log|\Lambda_A| - \sum_{k=1}^{n_e}\Psi\big(\tfrac{\nu_A+1-k}{2}\big)\Big) + \log\Big|\tfrac{\Omega}{2}\Big|,$$

with $|\cdot|$ *and* $\Psi(\cdot)$ *denoting the matrix determinant and the digamma function, respectively. Moreover,* $n_e = n_x + n_y$*.*

Proof. The approximate statistics (4.22) are derived based on evaluating the expected values of the unique entries in (4.20a) given by

$$\mathsf{E}[\Sigma^{-1}|\tilde{\nu}, \tilde{V}] = \tilde{\nu}\tilde{\Lambda}^{-1}, \tag{4.23a}$$

$$\mathsf{E}[\Sigma^{-1}\mu|\tilde{\nu}, \tilde{V}] = \tilde{\nu}\tilde{\Lambda}^{-1}\tilde{\mu}, \tag{4.23b}$$

$$\mathsf{E}[\mu^\top\Sigma^{-1}\mu|\tilde{\nu}, \tilde{V}] = \tilde{R}n_e + \tilde{\mu}^\top\tilde{\nu}\tilde{\Lambda}^{-1}\tilde{\mu}, \tag{4.23c}$$

$$\mathsf{E}[\log|\Sigma||\tilde{\nu}, \tilde{V}] = \log|\tilde{\Lambda}| - n_e\log 2 - \sum_{k=1}^{n_e}\Psi\big(\tfrac{\tilde{\nu}+1-k}{2}\big). \tag{4.23d}$$

The proof of the expected values (4.23) can be found in Propositions A.1 and A.2. If we plug (4.23) for the corresponding elements on the both sides of (4.16), the result (4.22) follows. □

The presence of Ψ in (4.22d) prevents us from finding an explicit expression for computing $\widehat{\nu}_{t|t-1}$. However, it turns out that approximating (4.22d) based on the first three terms of the Taylor series expansion of Ψ is sufficient [45], that is,

$$\widehat{\nu}_{t|t-1} \approx \Big(1 + \sqrt{1 + 4/3\Xi}\Big)\Big/2\Xi.$$

In the final lemma, we present the specific form of the predictive density (4.12), which is utilized for computing the weights in the steps A3 and B3 in Algorithm 15.

Lemma 4.4. *If we define the densities (4.2) and (4.3) by (4.18) and (4.19), respectively; then, the marginal density (4.12) becomes the Student's t density given by* $p(y_t, x_t|\widehat{\nu}_{t|t-1}, \widehat{V}_{t|t-1}) = \mathrm{St}(\xi_t; \bar{\mu}, \bar{\Lambda}, \bar{\nu})$*, where*

$$\bar{\mu} = \Phi(x_{t-1:t}) + \widehat{\mu}_{t|t-1}, \tag{4.24a}$$

$$\bar{\Lambda} = \tfrac{1+\widehat{R}_{t|t-1}}{\widehat{\nu}_{t|t-1}-n_e+1}\widehat{\Lambda}_{t|t-1}, \tag{4.24b}$$

$$\bar{\nu} = \widehat{\nu}_{t|t-1} - n_e + 1, \tag{4.24c}$$

denote an intermediate mean value, scale matrix, and number of degrees of freedom, respectively.

Proof. The proof of this result is presented in Lemma A.10. □

The resulting estimates can be obtained by taking the expected values of μ_t and Σ_t with respect to $p^N(\theta_t|y_{1:t}) = \sum_{i=1}^N w_t^i p(\theta_t|\nu_{t|t}^i, V_{t|t}^i)$, which gives

$$\mathsf{E}^N[\mu_t|y_{1:t}] = \sum_{i=1}^N w_t^i \mu_{t|t}^i,$$
$$\mathsf{E}^N[\Sigma_t|y_{1:t}] = \sum_{i=1}^N w_t^i \frac{\Lambda_{t|t}^i}{\nu_{t|t}^i - m_e - 1},$$

where the necessary expectations are presented in Propositions A.1 and A.2.

The special case of the above framework consists in the mutual independence of the noise variables v_t and w_t. In such situations, it is convenient to choose the prior (4.19) as the product of two GiW densities with the statistics $(\nu_w, R_w, \mu_w, \Lambda_w)$ and $(\nu_v, R_v, \mu_v, \Lambda_v)$, both being computed separately according to (4.21) for the updating and (4.22) for the forgetting step. Another consequence of this independence assumption consists in that (4.12) factorizes into the product of two Student's t densities $\mathrm{St}(x_t; \bar{\mu}_w, \bar{\Lambda}_w, \bar{\nu}_w)$ and $\mathrm{St}(y_t; \bar{\mu}_v, \bar{\Lambda}_v, \bar{\nu}_v)$, with the associated computations being handled with (4.24). Similar formulae were used in, e.g., [167].

4.5 Experiments and Results

This section illustrates the behavior of the proposed method compared to a number of selected techniques for the parameter estimation in non-linear state-space models. In a simulation scenario based on synthetic data, we compare the algorithms in terms of the estimation precision and computational complexity. We restrict ourselves to the case of scalar-valued state and observation variables. To assess the estimation precision of the states and parameters, we compute, respectively, the root-mean-squared error (RMSE) and root-mean-norm-squared error (RMNSE) according to

$$\text{RMSE} = \left(\tfrac{1}{T}\sum_{t=1}^{T}(x_t - x_{t|t}^N)^2\right)^{1/2}, \tag{4.25}$$

$$\text{RMNSE} = \left(\tfrac{1}{T}\sum_{t=1}^{T}||\theta_t - \theta_{t|t}^N||^2\right)^{1/2}, \tag{4.26}$$

where T stands for the amount of data samples, and $||\cdot||$ denotes the Euclidean norm. The computational complexity is evaluated as the average time needed to process all observations. We compare the following algorithms: the particle filter combined with the expectation maximization algorithm (PFEM) [28], the Rao-Blackwellized particle filter for static parameter estimation (RBPF) [200], the RBPF with exponential forgetting (RBPFEF) [167], and the RBPF with alternative stabilized forgetting (RBPFASF) proposed in Algorithm 15.

4.5.1 Simulation Settings

We generate $T = 4000$ observations from the univariate non-stationary growth model, which is commonly used to benchmark various state and parameter estimation techniques,

$$\begin{aligned} x_t &= \frac{x_{t-1}}{2} + \frac{25x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + w_t, \\ y_t &= \frac{x_t^2}{20} + v_t, \end{aligned}$$

where $w_t \overset{IID}{\sim} \mathcal{N}(\cdot;\mu_w,\sigma_w^2)$ and $v_t \overset{IID}{\sim} \mathcal{N}(\cdot;\mu_v,\sigma_v^2)$ are mutually independent Gaussian noise variables. The initial value of the state variable is distributed as $x_1 \sim \mathcal{N}(0,1)$. To be comparative, we follow the pattern of parameter changes outlined in [167]; thus, we have $\mu_{w,1} = 1$, $\Sigma_{w,1} = 2$, $\mu_{v,1} = 3$, $\Sigma_{v,1} = 4$, and $\mu_{w,4000} = 2$, $\Sigma_{w,4000} = 4$, $\mu_{v,4000} = 1$, $\Sigma_{v,4000} = 7$ for the initial and final steps, respectively. The changes are executed between the times 1500 and 2500, see Fig. 4.1. The initial statistics $(\nu_{w,1|0}, R_{w,1|0}, \mu_{w,1|0}, \Lambda_{w,1|0})$ and $(\nu_{v,1|0}, R_{v,1|0}, \mu_{v,1|0}, \Lambda_{v,1|0})$ are set according to $(5, 0.2, 3, 9)$ and $(5, 0.2, 1, 27)$, respectively, which holds for the RBPF, RBPFEF, and RBPFASF. Specifically, for the RBPFASF, the statistics of the alternative hypothesis about the parameter evolution $(\nu_{w,A}, R_{w,A}, \mu_{w,A}, \Lambda_{w,A})$ and $(\nu_{v,A}, R_{v,A}, \mu_{v,A}, \Lambda_{v,A})$ are given by $(5, 0.1, 0, 50)$ and $(5, 0.1, 0, 30)$. Furthermore, the forgetting factors of the RBPFEF and RBPFASF are respectively set to $\lambda = 0.98$ and $\lambda = 0.9998$. Concerning the PFEM algorithm, the initial parameter estimates are defined by $\widehat{\theta}_1 = (\widehat{\mu}_{w,1}, \widehat{\mu}_{v,1}, \widehat{\sigma}_{w,1}^2, \widehat{\sigma}_{v,1}^2) = (3, 1, 3, 9)$, the step size satisfies $t^{-0.6}$, and the parameter estimates are not altered during processing of the first 25 observations. The resampling is triggered whenever the effective sample size drops below the threshold

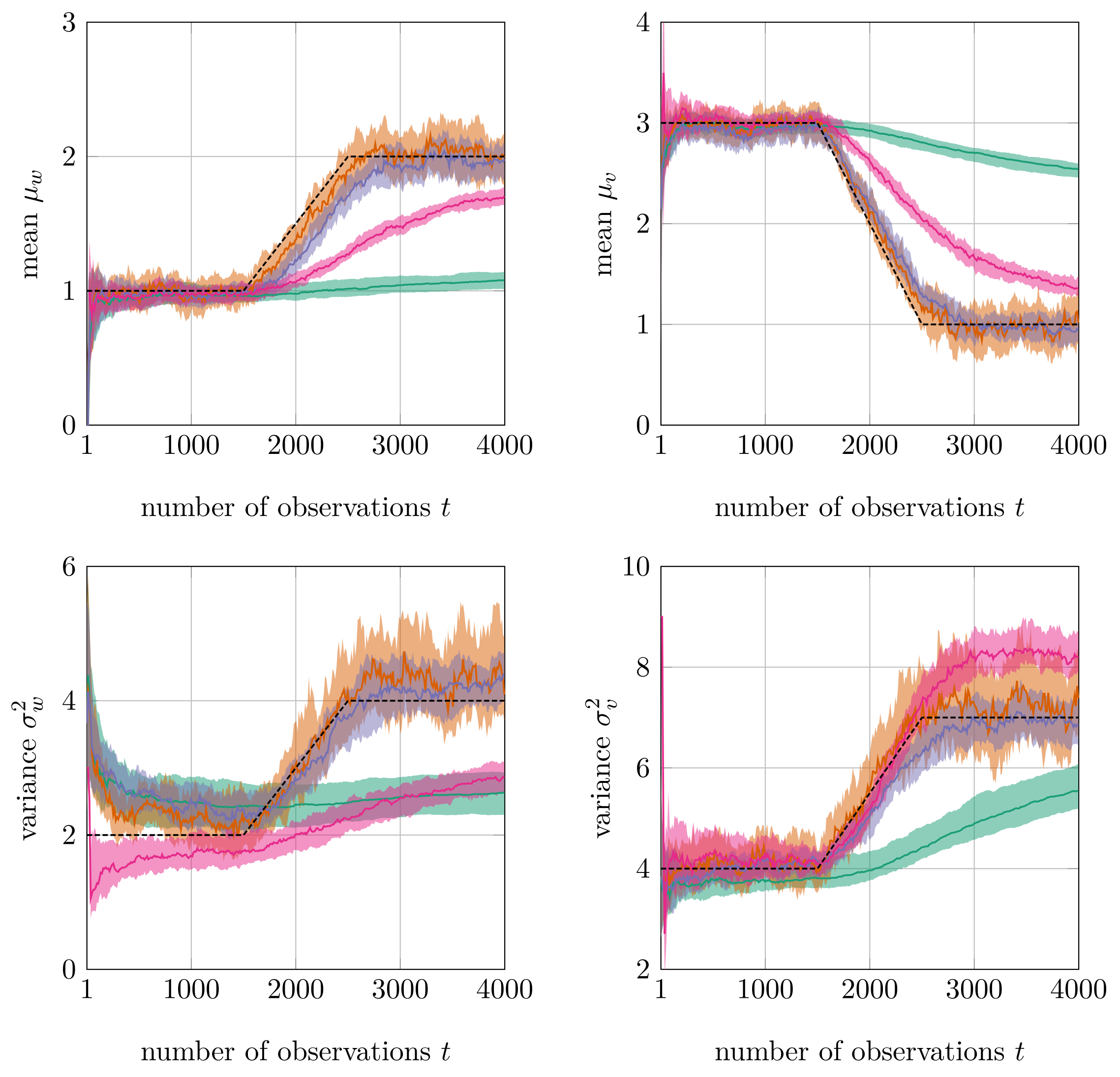

Fig. 4.1: The parameter estimates against the number of observations. The compared methods are PFEM (——) [28], RBPF (——) [200], RBPFEF (——) [167], and RBPFASF (——) Algorithm 15. The number of particles is $N = 512$. The results are averaged over 50 independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range. The true parameter values are indicated with the dashed line (------).

$N_{\text{th}} = N/3$. Finally, all methods are implemented in their corresponding bootstrap proposal setting.

4.5.2 Results

The resulting parameter estimates versus the number of observations are depicted in Fig. 4.1. The PFEM algorithm exhibits relatively good performance in terms of learning the static parameters; unfortunately, after the parameters start to change,

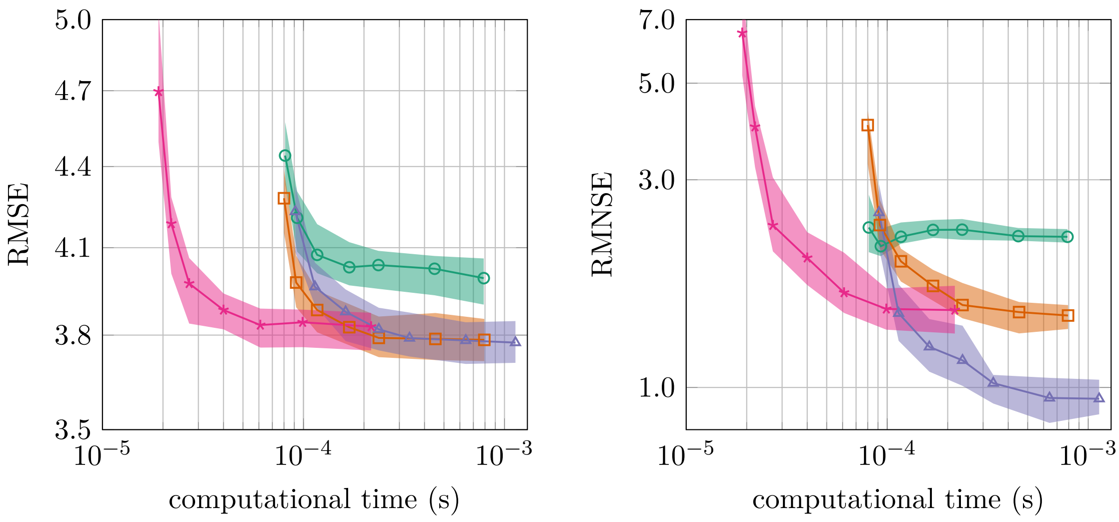

Fig. 4.2: The state RMSE (4.25) and the parameter RMNSE (4.26) versus the computational time (in seconds). The compared methods are PFEM (——) [28], RBPF (——) [200], RBPFEF (——) [167], and RBPFASF (——) Algorithm 15. The number of particles N takes values in $(32, 64, 128, 256, 512, 1024, 2048)$. The results are averaged over 50 independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range.

we can observe that the adaptability of this methods is rather poor. A similar finding holds also for the RBPF, albeit the parameter estimates are obviously more biased. The RBPFEF offers better capability of tracking the changes. This is, however, achieved at the cost of higher variance of the estimated values. In addition, the estimates of σ_v^2 are considerably more biased. The proposed RBPFASF performs more favorably, mostly providing estimates with lower bias and variance compared to the other algorithms.

The state RMSE indicator (4.25) versus the computational time in seconds—for different settings of the number of particles N—is presented in the left part of Fig. 4.2. The RBPFEF and RBPFASF algorithms perform very similarly, with the RBPFASF method being slightly better for low N. Although the PFEM algorithm is computationally more efficient, its RMSE does not approach the level attained by the RBPFEF and RBPFASF procedures (in the average behavior). In other words, this algorithm also does not improve any further for $N > 512$, which is caused by its lower ability to adapt the parameter changes. The RBPF method is significantly worse compared to the other algorithms. The reason for this consists in that this approach is not equipped with any ability to trace the parameter variations.

The parameter RMNSE indicator (4.26) versus the computational time is demonstrated in the right part of 4.2. The proposed RBPFASF algorithm is computationally more expensive at each N. However, we can observe that the RMNSE of the proposed method with $N = 128$ safely outperforms the other algorithms that run even with $N = 2048$. Thus, the proposed method surpasses the other procedures by almost one order of magnitude lower computational time. In other words, this observation reveals that the proposed RBPFASF approach is significantly more efficient in terms of the computational resources. Moreover, while the other algorithms improve their performance only slightly when increasing N, the proposed approach still improves the parameter estimation precision as N grows. Although the PFEM algorithm converges rather nicely as the number of particles increases, we must note that this favorable output of the PFEM algorithm is mainly caused by the fact that this method does not alter the parameter estimates for the first 25 steps, which provides an advantage when evaluating (4.26). The RBPF provides poor estimation accuracy, which is—as mentioned before—caused by the fact that this algorithm does not posses any ability to deal with the parameter variations. There is also another reason for this, which we discuss next.

4.6 Discussion

We can observe from the first 1500 steps in Fig. 4.1 that the basic RBPF converges to wrong values over multiple simulation runs. This behavior is explained by the particle path degeneracy problem. However, the RBPFEF and the proposed RBPFASF are significantly more robust in this respect, as can be seen from the lower bias of the resulting estimates. This is caused by the presence of the forgetting strategies in these algorithms, which allows us to deal with the time-varying character of the parameters and also to suppress—to a certain degree—the path degeneracy problem, as discussed in Remarks 4.1 and 4.2.

The RBPFEF procedure influences its forgetting properties by only tuning the forgetting factor λ. The proposed RBPFASF algorithm, on the other hand, is more versatile in this respect. Specifically, it enables us to tune the amount of forgotten information for each of the estimated parameters by setting the statistics of the alternative density. However, this makes the RBPFASF approach more difficult to tune.

Note that the PFEM algorithm is not made to deal with the parameter changes, and we include it exclusively to determine how much impact it can have when we use such an algorithm and the parameters undergo slow changes. This method, similarly to those derived from it [48, 164], also relies on an imposed type of forgetting. The formula for computing the smoothed additive functional in this approach contains the step size $t^{-\alpha}$, which can be seen as a time-varying forgetting factor. For the relation between the step size of the PFEM algorithm (or its variant [48]) and the forgetting factor of the RBPFEF procedure, see [198]. The functionality of this step-size-based forgetting strategy differs from that characterizing the approach developed in this chapter. The value $t^{-\alpha}$ progressively discounts the current sufficient statistics $s_t(x_{t-1:t}, y_t)$ of the state-space model (4.2), while the complement $1 - t^{-\alpha}$ prefers the past values accumulated in $V_{t|t-1}$. Naturally, such a mechanism can be efficient in learning static parameters. Nevertheless, poor adaptability has to be expected if parameter variations arrive too late.

5 RAO-BLACKWELLIZED PARTICLE GIBBS KERNELS FOR SMOOTHING IN JUMP MARKOV NONLINEAR MODELS

Jump Markov nonlinear models (JMNMs) characterize a dynamical system by a finite number of presumably nonlinear and possibly non-Gaussian state-space configurations that switch according to a discrete-valued hidden Markov process. In this context, the smoothing problem—the task of estimating fixed points or sequences of hidden variables given all available data—is of key relevance to many objectives of statistical inference, including the estimation of static parameters. The present chapter proposes a particle Gibbs with ancestor sampling (PGAS)-based smoother for JMNMs. The design methodology relies on integrating out the discrete process in order to increase the efficiency through Rao-Blackwellization. The experimental evaluation illustrates that the proposed method achieves higher estimation accuracy in less computational time compared to the original PGAS procedure.

5.1 Introduction

5.1.1 Context

Particle Markov chain Monte Carlo (PMCMC) methods [4] have recently emerged as an efficient tool to perform statistical inference in general state-space models (SSMs, [30]). These algorithms apply sequential Monte Carlo (SMC, [56]) to tackle the issue of constructing high-dimensional proposal kernels in MCMC [3]. This makes them particularly well suited for addressing the smoothing problem in jump Markov nonlinear models (JMNMs). The particle Gibbs with ancestor sampling (PGAS) kernel [132], which can be seen as a PMCMC smoother, has proved to be a serious competitor to the prominent SMC-based smoothing strategies such as the backward simulator [80] and generalized SMC two-filter smoother [25]. For a thorough review of existing SMC-based smoothers, see [133] and references therein.

The development in this chapter is motivated by the recent progress in constructing PG kernels specifically tailored for jump Markov linear models (JMLMs) [218, 202]. The methods therein exploit the linear Gaussian substructure of the model to increase their efficiency through Rao-Blackwellization. This is achieved by using the Kalman filter (KF) to design the conditional variants of either the discrete particle filter [64] or Rao-Blackwellized particle filter (RBPF, [55]). A common aspect of these PG methods lies in that the backward information filter (BIF, [148]) is used to further increase the effect of Rao-Blackwellization and to improve the mixing properties [3] via ancestor sampling or backward simulation.

5.1.2 Contributions

The problem with JMNMs is that their nonlinear character prevents us from applying Rao-Blackwellization in the same sense as with JMLMs; nevertheless, there is still a tractable substructure to exploit. The present chapter is concerned with the design of a Rao-Blackwellized PGAS (RBPGAS) kernel that takes advantage of the hierarchical structure formed by the discrete latent process. The method builds on the RBPF proposed in [166], which is similar to that introduced in [55] except it replaces the above-discussed KF with a finite state-space filter; conversely, the particle filter (PF) focuses on the remaining (continuous-valued) part of the latent process. However, the design of a finite state-space BIF turns out to be more intricate in this context as it requires us to introduce a sequence of artificial probability distributions to change the scale of the associated backward recursion.

5.2 Background

5.2.1 Jump Markov Nonlinear Models

The generic form of the discrete-time JMNM considered in the present chapter is defined by

$$c_t|c_{t-1} \sim p(c_t|c_{t-1}), \tag{5.1a}$$

$$z_t|c_t, z_{t-1} \sim f(z_t|c_t, z_{t-1}), \tag{5.1b}$$

$$y_t|c_t, z_t \sim g(y_t|c_t, z_t), \tag{5.1c}$$

where the states and measurements are denoted by $z_t \in \mathsf{Z} \subseteq \mathbb{R}^{n_z}$ and $y_t \in \mathsf{Y} \subseteq \mathbb{R}^{n_y}$, respectively. The activity of the current regime of the model is indicated by the discrete *mode* variable $c_t \in \mathsf{C} := \{1, \ldots, K\}$. We assume to have access only to the measurements y_t, while the state z_t and mode c_t variables are considered hidden. Furthermore, for all $c_t \in \mathsf{C}$, the model is characterized by its state transition and observation probability densities $f(\cdot)$ and $g(\cdot)$, respectively. The switching between the modes is governed by the conditional probability distribution $p(\cdot)$. At the initial time step, the hidden variables are distributed according to $z_1 \sim \mu(z_1|c_1)$ and $c_1 \sim p(c_1)$. For a graphical representation of a JMNM, see Fig. 5.1.

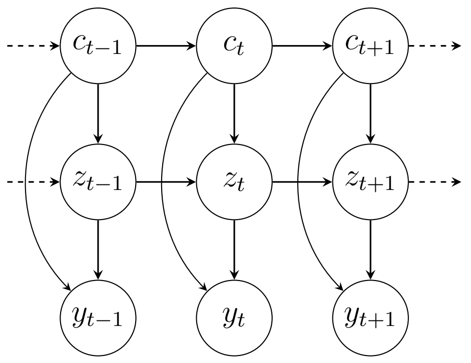

Fig. 5.1: Graphical model of a jump Markov nonlinear model.

5.2.2 Problem Formulation

Let $x_{1:T} \coloneqq (x_1, \dots, x_T)$ denote a generic sequence of variables defined on some product space X^T, for an integer $T > 0$ denoting the final time point. The aim of this study is to design an efficient PMCMC smoother targeting the joint smoothing density given by

$$p(c_{1:T}, z_{1:T}|y_{1:T}) = \frac{p(c_{1:T}, z_{1:T}, y_{1:T})}{p(y_{1:T})}. \tag{5.2}$$

However, the density (5.2) is intractable even in situations where (5.1b) and (5.1c) are linear and Gaussian. The reason consists in that the marginal likelihood $p(y_{1:T})$ contains summation over K^T values, which is always impossible to compute exactly, except for small data sets. Despite this, we consider (5.1b) and (5.1c) nonlinear and non-Gaussian, making the situation even more difficult as the integral over Z^T in the marginal likelihood $p(y_{1:T})$ cannot be evaluated either.

5.2.3 Sequential Monte Carlo

SMC [56] refers to a general class of algorithms suitable for approximating a sequence of (intractable) target densities $\{\pi_t(x_{1:t})\}_{t=1}^T$. We consider each of these densities to be defined on a product space X^t and to have the form

$$\pi_t(x_{1:t}) = \gamma_t(x_{1:t})/Z_t, \tag{5.3}$$

where $\gamma_t(x_{1:t})$ and $Z_t = \int \gamma_t(x_{1:t}) dx_{1:t}$ most often constitute the complete data likelihood and the marginal likelihood, respectively, with $Z_t > 0$ for all $t = 1, \dots, T$. The SMC approximation embodies an empirical measure represented by

$$\hat{\pi}_t(dx_{1:t}) = \sum_{i=1}^N w_t^i \delta_{x_{1:t}^i}(dx_{1:t}), \tag{5.4}$$

which is completely specified by the weighted particle system $\{x_{1:t}^i, w_t^i\}_{i=1}^N$. Here, the samples $\{x_{1:t}^i\}_{i=1}^N$ are termed particle trajectories and represent a hypothetical evolution of the true trajectory $x_{1:t}$, while the weights $\{w_t^i\}_{i=1}^N$ assess the contribution of the corresponding particle trajectories to the resulting approximation.

SMC methods are based on the repetitive use of sequential importance sampling and resampling in order to propagate (5.4) in time. Let us assume we have the previously generated particle system $\{x_{1:t-1}^i, w_{t-1}^i\}_{i=1}^N$. The *recursive step* of an SMC method begins with resampling. This procedure consists of sampling a set of ancestor indices $\{a_t^i\}_{i=1}^N$ from

$$\mathbb{P}(a_t^i = j) = w_{t-1}^j, \qquad j = 1, \dots, N. \tag{5.5}$$

The indices are then applied in the sequential importance sampling approach. This proceeds by first drawing the particles $\{x_t^i\}_{i=1}^N$ from the proposal density as

$$x_t^i \sim q_t(\cdot | x_{1:t-1}^{a_t^i}), \tag{5.6}$$

where the index a_t^i is used to assign the parent trajectory to the offspring particle x_t^i. The set of ancestors $\{a_{1:t}^i\}_{i=1}^N$ thus serves for tracing the genealogy of the particles. Subsequently, we extend the previous trajectories according to

$$x_{1:t}^i \coloneqq \{x_{1:t-1}^{a_t^i}, x_t^i\}. \tag{5.7}$$

The recursive step is concluded by computing the normalized importance weights $w_t^i \propto W_t(x_{1:t}^i)$ for $i = 1, \dots, N$, where the unnormalized weight function is defined by

$$W_t(x_{1:t}) \coloneqq \frac{\gamma_t(x_{1:t})}{\gamma_{t-1}(x_{1:t-1}) q_t(x_t | x_{1:t-1})}. \tag{5.8}$$

The *initial step* applies the standard importance sampling, that is, we first draw the particles $\{x_1^i\}_{i=1}^N$ from the initial proposal density $x_1^i \sim q_1(\cdot)$, and then calculate the importance weights $w_1^i \propto W_1(x_1^i)$ for $i = 1, \dots, N$, where $W_1(x_1) = \gamma_1(x_1)/q_1(x_1)$.

5.2.4 Particle Markov Chain Monte Carlo Smoothing

MCMC [3] constitutes a family of methods appropriate for approximating a target density $\pi(x)$ defined on some space X. The idea underlying MCMC is to simulate a Markov chain $\{x[k]\}_{k=1}^R$ according to

$$x[k] \sim \mathcal{K}(\cdot | x[k-1]),$$

where $\mathcal{K}$ is a Markov transition kernel on X, and R denotes the number of MCMC iterations. The chain is initialized by sampling from an initial density $x[1] \sim \nu(x)$. If the kernel $\mathcal{K}$ is ergodic and admits the target density $\pi(x)$ as its stationary density, then the chain $\{x[k]\}_{k=1}^{R}$ can be used to approximate expectations in the sense

$$\frac{1}{R}\sum_{k=1}^{R}\varphi(x[k]) \xrightarrow{a.s.} \mathbb{E}_{\pi}\left[\varphi(x)\right] \tag{5.9}$$

for $R \to \infty$, where $\xrightarrow{a.s.}$ labels almost sure convergence, φ is a suitable function on X, and $\mathbb{E}_{\pi}\left[\varphi(x)\right]$ is the expected value with respect to $\pi(x)$.

An MCMC smoother respects these ideas with the only difference being that the target density π is defined on X^T. However, it is mostly difficult to efficiently sample from the kernel $\mathcal{K}$ when the dimension T is large. This issue is addressed here on the basis of a specific PMCMC method known as the PG [4]. The procedure relies on the *conditional* SMC (CSMC) update, which is nearly the same as the original SMC method described above, except one particle trajectory $x'_{1:T}$ is specified in advance. We refer to $x'_{1:T}$ as the reference trajectory. The consequence of this change is that one samples from (5.5) and (5.6) only for $i = 1, \ldots, N-1$, while the remaining ancestor index and particle are set as $a_t^N := N$ and $x_t^N := x'_t$, respectively. The rest of the operations remain unchanged. After finishing the run of the CSMC method, a new reference trajectory is received by sampling k with $\mathbb{P}(k = i) = w_T^i$ and selecting $x_{1:T}^k$ from $\{x_{1:T}^i\}_{i=1}^N$. The fact that the technique requires a state trajectory as the input and returns another state trajectory as the output defines a Markov kernel on X^T, which is called the PG kernel [132]. It is shown in [4] that the kernel is ergodic and admits $\pi_T(x_{1:T})$ as the stationary density.

5.2.5 Particle Gibbs with Ancestor Sampling Kernel

The mixing properties of the basic PG kernel may be poor when the number of particles N is low or the dimension T is high. The reason consists in that the (C)SMC algorithm is quite inefficient in updating the past values of the particle trajectories (5.7). In fact, for some time points not too far from the current time t, the number of identical particle trajectories is often very close or equal to N. Such a loss in diversity is commonly known as particle path degeneracy [94]. The consequence of the phenomenon is that the consecutive trajectories generated by the PG kernel will be highly correlated, and the ability of the PMCMC smoother to explore the space of trajectories X^T will thus become unsatisfactory.

To improve the mixing properties, let us adopt the method referred to as PGAS [132]. The procedure enhances the original PG kernel by introducing an additional sampling step that generates new values of the ancestor indices a_t^N, instead of just setting them as $a_t^N := N$. The Nth trajectory is constructed, for all $t \geq 2$, by concatenating one of the historical trajectories with a future part of the reference

Algorithm 16 PGAS Kernel [132]

```
    **Inputs:** $x'_{1:T} = x_{1:T}[k-1]$.
    **Outputs:** $x_{1:T}[k]$ and $\{x^i_{1:T}, w^i_T\}_{i=1}^N$.
A. **Initial step:** $(t = 1)$
    1. Sample $x^i_1 \sim q_1(\cdot)$ for $i = 1, \ldots, N-1$ and set $x^N_1 := x'_1$.
    2. Compute $w^i_1 \propto W_1(x^i_1)$ for $i = 1, \ldots, N$.
B. **Recursive step:** $(t = 2, \ldots, T)$
    1. Sample $a^i_t$ with $\mathbb{P}(a^i_t = j) = w^j_{t-1}$ for $i = 1, \ldots, N-1$.
    2. Sample $x^i_t \sim q_t(\cdot|x^{a^i_t}_{1:t-1})$ for $i = 1, \ldots, N-1$.
    3. Sample $a^N_t$ using (5.10) and set $x^N_t = x'_t$.
    4. Set $x^i_{1:t} := \{x^i_t, x^{a^i_t}_{1:t-1}\}$ for $i = 1, \ldots, N$.
    5. Compute $w^i_t \propto W_t(x^i_{1:t})$ according to (5.8), for $i = 1, \ldots, N$.
C. **Final step:**
    1. Sample $k$ with $\mathbb{P}(k = i) = w^i_T$ and set $x_{1:T}[k] = x^k_{1:T}$.
```

trajectory according to

$$x^N_{1:T} := \{x^{a^N_t}_{1:t-1}, x'_{t:T}\},$$

where the connection between the two partial paths is determined by the ancestor index a^N_t. After the complete pass through the data, the Nth trajectory no longer coincides with the reference trajectory, as in the case of the fundamental PG kernel, but rather becomes fragmented into pieces. The consecutive sampled trajectories are then significantly less correlated, thus providing a substantial improvement of the mixing properties. The ancestor a^N_t is sampled from

$$\mathbb{P}(a^N_t = i) = w^i_{t-1|T}, \qquad i = 1, \ldots, N, \tag{5.10}$$

where

$$w^i_{t-1|T} \propto w^i_{t-1} \frac{\gamma_T(\{x^i_{1:t-1}, x'_{t:T}\})}{\gamma_{t-1}(x^i_{1:t-1})} \tag{5.11}$$

is the probability of connecting the ith historical trajectory with the future one, see [132] for the derivation of (5.11) and the proof of ergodicity. The method is recalled in Algorithm 16. Note the CSMC procedure is represented by steps A and B, and the basic PG kernel is obtained by setting $a^N_t := N$ in step B3.

5.3 Smoother Design

5.3.1 Design Objectives

A possible approach to design the algorithm consists in directly targeting the join smoothing density (5.2). This would require us to define $\gamma_t(x_{1:t}) := p(z_{1:t}, c_{1:t}, y_{1:t})$ and $Z_t := p(y_{1:t})$ in the above framework. The CSMC procedure in Algorithm 16 would then be represented by the conditional PF [4] operating with the composite state variable $x_t := (c_t, z_t)$. However, it was noticed in [58] that the PFs sampling this joint state may suffer from the degeneracy of the mode variables when the mixing properties of the transition kernel (5.1a) are poor. To suppress this problem, an RBPF that marginalizes out the mode variable was proposed in [166]. The development in the present chapter is based on introducing a conditional version of this RBPF.

Hence, a better solution is to exploit the tractable substructure of the model and thus factorize (5.2) as

$$p(c_{1:T}, z_{1:T}|y_{1:T}) = p(c_{1:T}|z_{1:T}, y_{1:T})p(z_{1:T}|y_{1:T}).$$

The main goal is to design a Rao-Blackwellized PG (RBPG) kernel for approximating the second factor. Then, the sampled trajectories are used in the first factor to analytically compute a finite state-space smoother.

5.3.2 Conditional Rao-Blackwellized Particle Filter

To show the derivation of the conditional RBPF (CRBPF), let us factorize the extended target density $p(c_t, z_{1:t}|y_{1:t})$ as

$$p(c_t, z_{1:t}|y_{1:t}) = p(c_t|z_{1:t}, y_{1:t})p(z_{1:t}|y_{1:t}). \tag{5.12}$$

The first factor embodies the posterior distribution of the mode variable given the sequence $z_{1:t}$, which is computable analytically through the standard filtering recursion based on the forward prediction

$$p(c_t|z_{1:t-1}, y_{1:t-1}) = \sum_{c_{t-1} \in \mathsf{C}} p(c_t|c_{t-1})p(c_{t-1}|z_{1:t-1}, y_{1:t-1}) \tag{5.13}$$

and the forward update

$$p(c_t|z_{1:t}, y_{1:t}) \propto p(y_t, z_t|c_t, z_{t-1})p(c_t|z_{1:t-1}, y_{1:t-1}), \tag{5.14}$$

forming together a finite state-space filter.

As the second factor is supposed to describe the analytically intractable part of the model, we resort to the above framework to find the approximation. Thus, the CSMC method now targets $p(z_{1:t}|y_{1:t})$, which requires us to define $\gamma_t(x_{1:t}) := p(z_{1:t}, y_{1:t})$ while Z_t remains the same. The CSMC procedure in Algorithm 16 therefore operates with the marginal state variable $x_t := z_t$. In consequence of these changes, the weight function (5.8) becomes

$$W_t(z_{1:t}) = \frac{p(y_t, z_t|z_{1:t-1}, y_{1:t-1})}{q_t(z_t|z_{1:t-1})}, \tag{5.15}$$

where the marginal density in the numerator is

$$p(y_t, z_t|z_{1:t-1}, y_{1:t-1}) = \sum_{c_t \in \mathsf{C}} p(y_t, z_t|c_t, z_{t-1})p(c_t|z_{1:t-1}, y_{1:t-1}). \tag{5.16}$$

To compute the weights (5.15) for all the trajectories $\{z^i_{1:t}\}^N_{i=1}$, the recursion given by (5.13) and (5.14) needs to be computed for each $i = 1, \dots, N$; this is crucial to note as some of the steps in Algorithm 16 are computed just for $N-1$ particles. Importantly, too, the posterior distributions (5.14) extend the original particle system, yielding $\{z^i_{1:t}, p(c_t|z^i_{1:t}, y_{1:t}), w^i_t\}^N_{i=1}$. Construction of the bootstrap proposal density for $q_t(z_t|z_{1:t-1})$ is outlined in [166].

5.3.3 Ancestor Sampling Weights

A possible way to acquire an RBPG kernel would be to use the CRBPF in the basic PG kernel. While this is easy to realize, the mixing properties of the resulting sampler may be poor, as discussed above. Hence, let us derive the ancestor sampling to improve upon this issue. Considering we have $\gamma_T(x_{1:T}) \coloneqq p(z_{1:T}, y_{1:T})$, the weight (5.11) transforms into

$$w^i_{t-1|T} \propto w^i_{t-1} p(y_{t:T}, z'_{t:T}|z^i_{1:t-1}, y_{1:t-1}). \tag{5.17}$$

The predictive densities in (5.17) can be computed by running one finite state-space filter for each of the historical state trajectories $z^i_{1:t-1}$ over the time range from $t-1$ to T. However, if such a straightforward implementation were used, the total cost of computing the weights (5.17) would amount to $\mathcal{O}(TK^2N)$ operations per time step t. The issue can be circumvented as demonstrated through the design of the Rao-Blackwellized particle smoothers (RBPSs) presented within [218, 189], which consists in rewriting the predictive density as

$$p(y_{t:T}, z'_{t:T}|z^i_{1:t-1}, y_{1:t-1}) = \sum_{c_{t-1} \in \mathsf{C}} p(y_{t:T}, z'_{t:T}|z^i_{t-1}, c_{t-1})p(c_{t-1}|z^i_{1:t-1}, y_{1:t-1}). \tag{5.18}$$

The summand in (5.18) is similar to the two-filter smoothing formula [24] that is based on running one filter forward and the other backward in time. In our situation, the forward filter has already been recalled in (5.13) and (5.14). The backward filter is designed below.

5.3.4 Finite State-Space Backward Information Filter

The BIF [148] facilitates the computation of the likelihood term in the summand of (5.18) under a closed-form solution. In the present context, the recursive step of such a filter iterates, for $t = T-1, \dots, 1$, over the backward prediction

$$p(y_{t+1:T}, z_{t+1:T}|z_t, c_t) = \sum_{c_{t+1} \in \mathsf{C}} p(y_{t+1:T}, z_{t+2:T}|z_{t+1}, c_{t+1})p(z_{t+1}, c_{t+1}|c_t, z_t) \tag{5.19}$$

Algorithm 17 Finite State-Space BIF

```
    Inputs: $z'_{1:T}$ and $\{\xi_t(c_t)\}_{t=1}^T$.
    Outputs: $\{\tilde{p}(c_t|y_{t:T}, z'_{t:T})\}_{t=1}^T$.
A. Initial step: $(t = T)$
    1. Compute $\tilde{p}(c_T|y_T, z'_T) \propto p(y_T|c_T, z'_T)\xi_T(c_T)$.
B. Recursive step: $(t = T-1, \dots, 1)$
    1. Compute $\tilde{p}(c_t|y_{t+1:T}, z'_{t:T})$ using (5.22), $\xi_t(c_t)$, and $\xi_{t+1}(c_{t+1})$.
    2. Compute $\tilde{p}(c_t|y_{t:T}, z'_{t:T})$ using (5.23) and $\tilde{p}(c_t|y_{t+1:T}, z'_{t:T})$.
```

and the backward update

$$p(y_{t:T}, z_{t+1:T}|z_t, c_t) = p(y_t|c_t, z_t)p(y_{t+1:T}, z_{t+1:T}|z_t, c_t). \tag{5.20}$$

The initial step computes only the observation density (5.1c) at $t = T$. A similar recursive form has recently been used to develop an RBPS for mixed linear/nonlinear SSMs in [131].

The difficulty encountered with (5.19) and (5.20) lies in that neither of these is a probability density in the arguments z_t and c_t. Therefore, integrating with respect to these variables might lead to results which are not finite. To overcome this issue, Briers et al. [26] proposed to design a sequence of *artificial* probability distributions to change the scale of such problematic measures. Here, this approach is applied only to the mode variable c_t as the state variable z_t is fixed at this stage of the design. The sought recursion is introduced below.

Proposition 5.1 (Rescaled backward recursion). *Let $\xi_t(c_t)$ denote the artificial (prior) distribution related to an auxiliary (posterior) distribution $\tilde{p}(c_t|y_{t:T}, z_{t:T})$ according to*

$$\tilde{p}(c_t|y_{t:T}, z_{t:T}) \propto p(y_{t:T}, z_{t+1:T}|z_t, c_t)\xi_t(c_t). \tag{5.21}$$

Then, the rescaled version of the backward prediction (5.19) is

$$\tilde{p}(c_t|y_{t+1:T}, z_{t:T}) \propto \sum_{c_{t+1}\in \mathsf{C}} \tilde{p}(c_{t+1}|y_{t+1:T}, z_{t+1:T}) \frac{p(z_{t+1}, c_{t+1}|c_t, z_t)\xi_t(c_t)}{\xi_{t+1}(c_{t+1})}, \tag{5.22}$$

and, similarly, the backward update (5.20) becomes

$$\tilde{p}(c_t|y_{t:T}, z_{t:T}) \propto p(y_t|c_t, z_t)\tilde{p}(c_t|y_{t+1:T}, z_{t:T}). \tag{5.23}$$

Proof. The result follows immediately from multiplying both sides of (5.19) and (5.20) by $\xi_t(c_t)$ and noticing that a definition analogous to (5.21) holds also for $\tilde{p}(c_t|y_{t+1:T}, z_{t:T})$. □

To compute the predictive density by means of this reformulated backward recursion, we first substitute from (5.21) into (5.19) to receive

$$p(y_{t:T}, z'_{t:T}|z^i_{t-1}, c_{t-1}) \propto \sum_{c_t\in \mathsf{C}} \frac{\tilde{p}(c_t|y_{t:T}, z'_{t:T})}{\xi_t(c_t)} p(z'_t, c_t|c_{t-1}, z^i_{t-1}), \tag{5.24}$$

and, subsequently, we plug this result into (5.18). Note that $\tilde{p}(c_t|y_{t:T}, z'_{t:T})$ needs to be computed only for the reference trajectory as it does not contain any part of the historical particle trajectories. Thus, the distributions $\tilde{p}(c_t|y_{t:T}, z'_{t:T})$ can be precomputed in a single backward pass through the data. It is also important to mention the total cost of computing (5.17) becomes $\mathcal{O}(K^2N)$ operations in a single time step t. The finite state-space BIF is summarized in Algorithm 17.

The artificial distribution is introduced into the backward recursion such that its choice can be made arbitrarily. Multiple recommendations regarding how to define $\xi_t(c_t)$ are contained in [26]; here, the marginal and independent prior is chosen, namely, $\xi_t(c_t) \coloneqq p(c_t) = p(c_t|z_t)$, where

$$p(c_t) = \sum_{c_{t-1} \in \mathsf{C}} p(c_t|c_{t-1})p(c_{t-1}). \tag{5.25}$$

Note this choice simplifies the ratio in (5.22) to the product $p(z_{t+1}|c_t, z_t)p(c_t|c_{t+1})$, where $p(c_t|c_{t+1})$ is the backward transition kernel of the mode variable. Other choices of $\xi_t(c_t)$ are discussed later in the text.

The proposed RBPGAS kernel is summarized in Algorithm 18, where the convention $z_{1:0} \coloneqq \varnothing$ and $y_{1:0} \coloneqq \varnothing$ holds in step A4. Although not explicitly expressed in Algorithm 18, the posterior distributions $p(c_{t-1}|z^i_{1:t-1}, y_{1:t-1})$ are resampled, together with the particle trajectories $z^i_{1:t-1}$, by applying the ancestor indices a^i_t before their use in (5.13).

5.3.5 Finite State-Space Smoother

To obtain the smoothed estimates of the mode trajectory, we resort to a finite state-space forward-backward smoother conditioned on the state trajectory $z_{1:T}$. The recursive step computes, for $t = T-1, \dots, 1$, the marginal smoothing distribution according to

$$p(c_t|z_{1:T}, y_{1:T}) = \sum_{c_{t+1} \in \mathsf{C}} \frac{p(c_{t+1}|c_t)p(c_t|z_{1:t}, y_{1:t})}{p(c_{t+1}|z_{1:t}, y_{1:t})} p(c_{t+1}|z_{1:T}, y_{1:T}), \tag{5.26}$$

where $p(c_t|z_{1:t}, y_{1:t})$ is produced by the forward recursion formed by (5.13) and (5.14). For the initial step, the filtering and marginal smoothing distributions are identical. After computing (5.26) for each of the trajectories $\{z_{1:T}[k]\}_{k=1}^{R}$ and averaging the results with (5.9), the maximum *a posteriori* sequence is taken to represent the smoothed estimate.

An alternative approach is to use the auxiliary distributions $\tilde{p}(c_t|y_{t:T}, z'_{t:T})$ in the two-filter smoother [25].

Algorithm 18 RBPGAS Kernel for JMNMs

```
**Inputs:** $z'_{1:T} = z_{1:T}[k-1]$.
**Outputs:** $z_{1:T}[k]$ and $\{z^i_{1:T}, \{p(c_t|z^i_{1:t}, y_{1:t})\}^T_{t=1}, w^i_T\}^N_{i=1}$.
A. **Initial step:** $(t = 1)$
    1. Compute the sequence $\{\xi_t(c_t)\}^T_{t=1}$.
    2. Use $z'_{1:T}$ and $\{\xi_t(c_t)\}^T_{t=1}$ as the input for Algorithm 17 to produce $\{\tilde{p}(c_t|y_{t:T}, z'_{t:T})\}^T_{t=1}$.
    3. Sample $z^i_1 \sim q_1(\cdot)$ for $i = 1, \dots, N-1$ and set $z^N_1 := z'_1$.
    4. Compute $p(c_1|z^i_1, y_1)$ and $p(y_1, z^i_1)$ according to (5.14) and (5.16), respectively, for $i = 1, \dots, N$.
    5. Compute $w^i_1 \propto W_1(z^i_1)$ for $i = 1, \dots, N$.
B. **Recursive step:** $(t = 2, \dots, T)$
    1. Sample $a^i_t$ with $\mathbb{P}(a^i_t = j) = w^j_{t-1}$ for $i = 1, \dots, N-1$.
    2. Sample $z^i_t \sim q_t(\cdot|z^{a^i_t}_{1:t-1})$ for $i = 1, \dots, N-1$.
    3. Compute $w^i_{t-1|T}$ according to (5.17-5.18) and (5.24), using $\xi_t(c_t)$ and $\tilde{p}(c_t|y_{t:T}, z'_{t:T})$, for $i = 1, \dots, N$.
    4. Sample $a^N_t$ with $\mathbb{P}(a^N_t = i) = w^i_{t-1|T}$ and set $z^N_t := z'_t$.
    5. Set $z^i_{1:t} := \{z^i_t, z^{a^i_t}_{1:t-1}\}$ for $i = 1, \dots, N$.
    6. Compute $p(c_t|z^i_{1:t}, y_{1:t})$ and $p(y_t, z^i_t|z^{a^i_t}_{1:t-1}, y_{1:t-1})$ according to (5.13-5.14) and (5.16), respectively, for $i = 1, \dots, N$.
    7. Compute $w^i_t \propto W_t(z^i_{1:t})$ according to (5.15), for $i = 1, \dots, N$.
C. **Final step:**
    1. Sample $k$ with $\mathbb{P}(k = i) = w^i_T$ and set $z_{1:T}[k] := z^k_{1:T}$.
```

5.4 Numerical Illustration

This section demonstrates the performance of the proposed RBPGAS kernel (Algorithm 20) in comparison to the PG [4], PGAS [132], RBPG (Algorithm 20 with setting $a^N_t := N$ in step B4), and RBPGASnr (Algorithm 20 with the non-rescaled recursion) kernels. Let us consider the nonlinear benchmark model given by

$$z_t = 0.5 z_{t-1} + 25\frac{z_{t-1}}{1+z^2_{t-1}} + 8\cos(1.2t) + v_t, \tag{5.27a}$$

$$y_t = 0.05 z^2_t + w_t, \tag{5.27b}$$

where, for $c_t \in \mathsf{C} := \{1, 2\}$, $w_t \sim \mathcal{N}(\mu_{c_t}, \sigma^2_{c_t})$ denotes a mode-dependent Gaussian noise variable with the mean μ_{c_t} and variance $\sigma^2_{c_t}$. Furthermore, $v_t \sim \mathcal{N}(0, 1)$ is an independent and identically distributed Gaussian noise variable with zero mean and unit variance. The kernel (5.1a) is parameterized by the transition probability matrix Π according to $p(c_t = j|c_{t-1} = i) := \Pi_{ij}$ with $i, j \in \mathsf{C}$. The diagonal entries of this matrix are set as $\Pi_{11} = 0.6$ and $\Pi_{22} = 0.8$. The mode-dependent means and variances are defined by $\mu_1 = 0$, $\mu_2 = 7$ and $\sigma^2_1 = 4$, $\sigma^2_2 = 1$. The state prior density is mode-independent and Gaussian, $\mu(z_1|c_1) := \mathcal{N}(z_1; 0, 1)$. Further, the prior distribution of the mode variable $p(c_1)$ is parameterized by the vector λ with the relation $p(c_1 = i) := \lambda_i$, where the first entry is $\lambda_1 = 0.5$; the artificial distribution corresponds to (5.25). All the algorithms use the associated bootstrap proposal density.

The experiment assesses the performance of the compared methods when the number of particles changes according to $N = 2^n$ for $n = 1, \ldots, 9$. For all these values, forty independent runs of the model (5.27) were performed, each producing the measurement sequence of the length $T = 100$. The algorithms subjected to comparison were tested with the number of iterations $R = 500$. To evaluate resulting estimates, the proposed RBPGAS method with $N = 1024$ particles (a higher number did not lead a significant improvement) was used to compute 'exact' state and mode trajectories for each of the measurement sequences.

As is obvious from Fig. 5.2, the RBPGAS method achieves the best accuracy with the number of particles $N=4$, increasing the value does not provide any noticeable improvement. The PGAS procedure approaches the performance of the RBPGAS algorithm as the number of particles increases, with practically no difference in the RMSE and NNZE indicators for $N \geq 32$. The distinction between the methods is most significant at $N = 2$ particles, where the effect of Rao-Blackwellization, brought by the RBPGAS procedure, is most obvious. The PG and RBPG algorithms attain the precision of the RBPGAS method for a much higher N, which is caused by the absence of the ancestor sampling in these methods. This also implies that the PG and RBPG algorithms require substantially more computational resources. Nevertheless, even in this case, it can be seen that the Rao-Blackwellization may substantially increase the accuracy.

Fig. 5.3 provides a closer look at the situation where the compared algorithms are applied with $N = 2$ particles. We can see that the RMSE of the PGAS and RBPGAS methods is competitive for approximately the first 10^{-1} seconds, with the RBPGAS algorithm starting to be computationally more efficient (in the average behavior) after this time. This is obvious from the median and interquartile range, which decrease more quickly for the RBPGAS procedure. The right part of Fig. 5.3 reveals that the RBPGAS method achieves lower values of the NNZE in a shorter computational time. For example, we can see that the value 10 is there reached after approximately $2 \cdot 10^{-1}$ seconds with the PGAS method, while the same value is attained after approximately $7 \cdot 10^{-2}$ seconds with the RBPGAS algorithm. It is therefore obvious that the RBPGAS procedure is markedly quicker in approaching the ergodic regime. However, the PG and RBPG kernels deliver a very slow convergence, being still very far from the ergodic regime. The reason is, again, the lack of the ancestor sampling. The RBPGASnr method has a higher variance than the RBPGAS procedure, proving the importance of involving the artificial distribution $\xi_t(c_t)$ into the design.

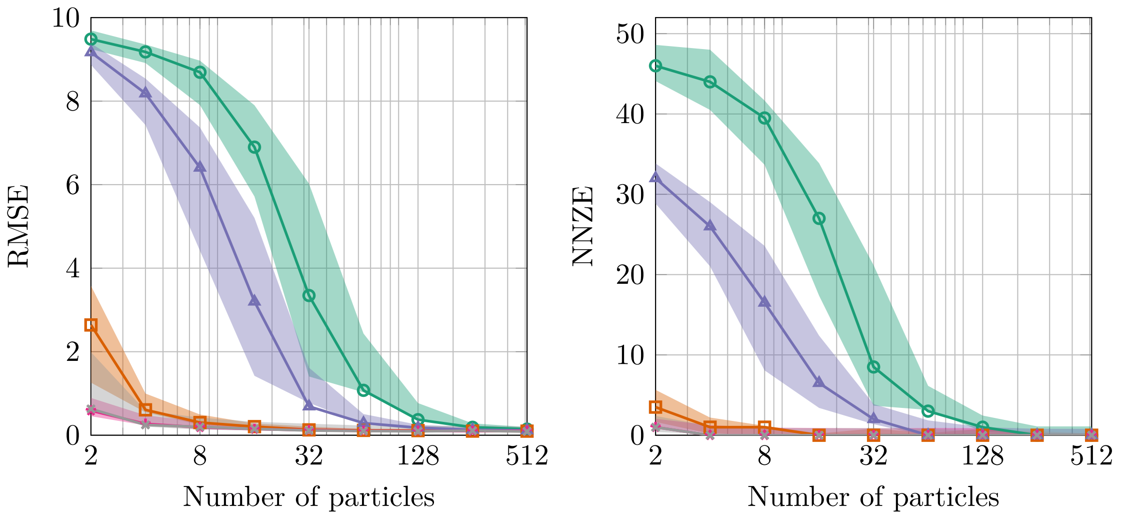

Fig. 5.2: Left: the root-mean-square error (RMSE) between the exact and estimated state trajectories versus the number of particles. Right: the number of non-zero elements (NNZE) in the error sequence between the exact and estimated mode trajectories versus the number of particles. The solid line represents the median, and the shaded area is the interquartile range; both are computed over forty independent runs. The compared algorithms are PG (–○–), PGAS (–□–), RBPG (–▲–), RBPGAS (–★–), and RBPGASnr (–×–).

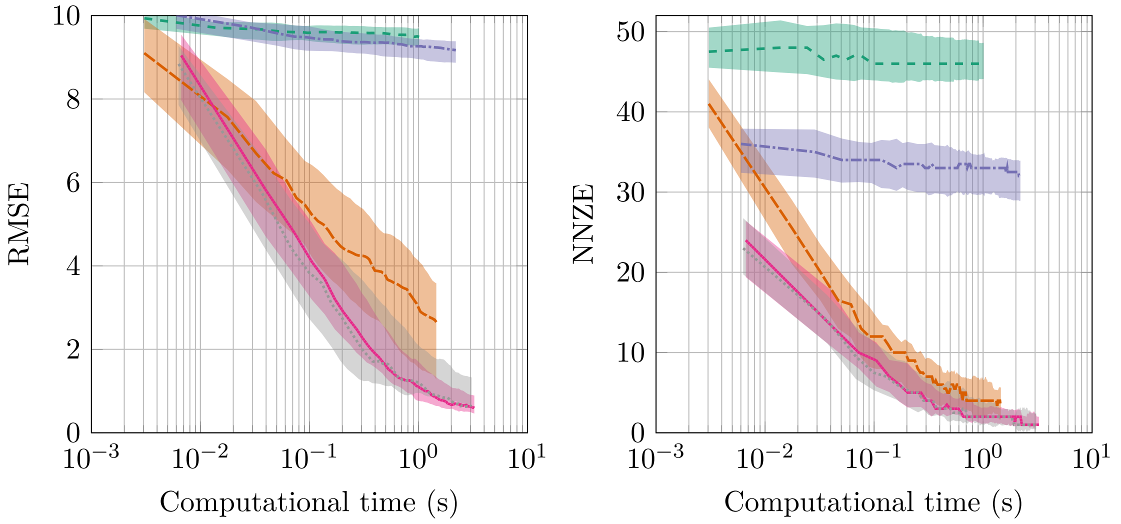

Fig. 5.3: Left: the root-mean-square error (RMSE) between the exact and estimated state trajectories versus the computational time (in seconds). Right: the number of non-zero elements (NNZE) in the error sequence between the exact and estimated mode trajectories versus the computational time. The solid line shows the median, and the shaded area is the interquartile range; both are computed over forty independent runs. The compared methods are PG (- - -), PGAS (– – –), RBPG (-·-·-), RBPGAS (——), and RBPGASnr (········).

5.5 Discussion

The uniform prior λ and stationary distribution calculated as the left eigenvector of the matrix Π were used to represent the artificial distribution $\xi_t(c_t)$. Both these options provided results similar to the situation considering (5.25), with only a minor difference in the variance of the smoothed estimates, indicating insensitivity to various settings of $\xi_t(c_t)$.

If the transition kernel (5.1a) has poor mixing properties, the proposed RBPGAS method offers a higher robustness compared to the PGAS algorithm. The trade-off between the estimation accuracy and computational time then starts to be more pronounced, with the proposed RBPGAS method still being the more successful one. The reason lies, as discussed previously, in that the PGAS procedure suffers from the degeneracy around the mode changes [58]. As shown in the experiments, the effectiveness is mainly achieved in terms of a shorter time required to reach the ergodic regime.

There are various particle filters for jump Markov nonlinear models that can be used to develop the PMCMC kernel. Therefore, a preliminary research on assessing different types of particle filters was performed before deciding which one is the most suitable basis for developing the PMCMC kernel. A number of well-established strategies were compared, while also proposing some alternative solutions. The comparison comprises of the following methods: (i) the basic particle filter (PF) which jointly samples both the latent variables; (ii) the Rao-Blackwellized particle filter (RBPF) [166] which marginalizes out the discrete regime variable; (iii) the interacting multiple model particle filter (IMMPF1) with the 'mixing stage' [58]; (iv) the interacting multiple model particle filter (IMMPF2) with the 'interaction resampling' [22]; (v) the exact approximate Rao-Blackwellized particle filter (EARBPF) [100], where two nested layers of particle filters are utilized: the upper layer samples the discrete regime variables while the lower layer runs a regime-conditioned particle filter for each sample from the upper layer; and (vi) the exact approximate discrete particle filter (EADPF) which modifies the EARBPF method in the sense of utilizing the optimal resampling [126] in a similar manner as with the discrete particle filter [64]. Specifically, the idea of designing the EADPF method was first mentioned in [100]; however, to the best of the author's knowledge, it is here—in the present thesis—where such an approach is first realized and exemplified.

The computational complexity of the PF, RBPF, IMMPF1, IMMPF2, EARBPF, and EADPF algorithms scales according to $\mathcal{O}(N)$, $\mathcal{O}(K^2N)$, $\mathcal{O}(KN)$, $\mathcal{O}(KN)$, $\mathcal{O}(MN)$, and $\mathcal{O}(KMN)$, respectively, where K is the number of discrete regime variables, N is the number of lower layer particles, and M is the number of upper layer particles. This comparison of computational complexity is only qualitative as

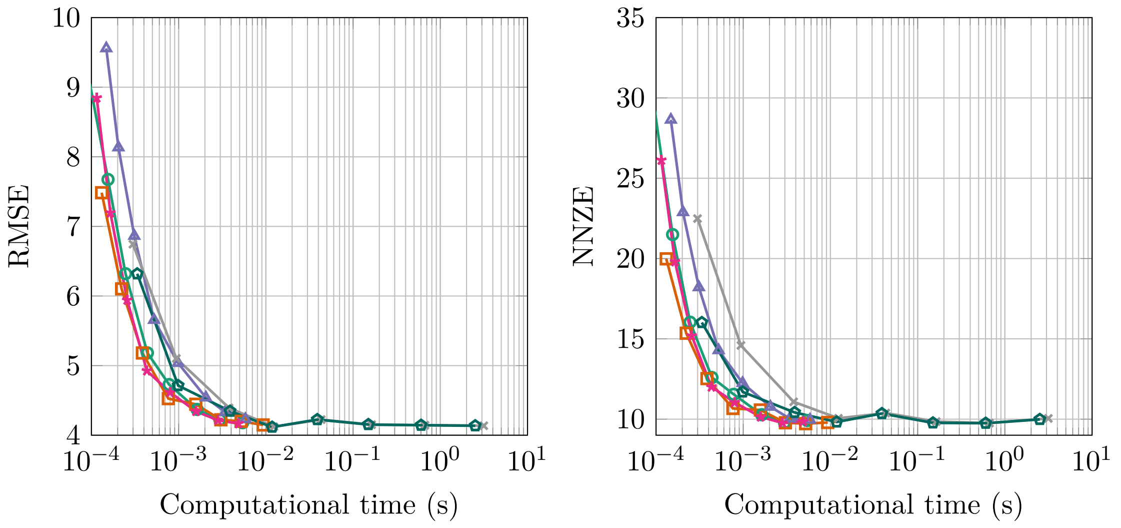

Fig. 5.4: Left: the root-mean-square error (RMSE) between the true and estimated state trajectories versus the computational time (in seconds). Right: the number of non-zero elements (NNZE) in the error sequence between the true and estimated mode trajectories versus the computational time. The compared algorithms are PF (—○—), RBPF (—□—), IMMPF1 (—▲—), IMMPF2 (—✱—), EARBPF (—×—), and EADPF (—⬠—). The number of particles, N, takes values in $(4, 8, 16, 32, 64, 128, 256, 512)$.

there can be important implementation details creating substantial computational differences among the considered techniques. Therefore, to justify the decision on which of the methods is the most suitable one, we evaluate the trade-off between the computational time and estimation precision. We compare the above procedures based on the simulation scenario from Section 5.4. The experiment is implemented in C with a precise decomposition of common subroutines among the compared methods in order to ensure a fair assessment of the computational time. In this experiment, we set $M = N$ and apply the algorithms in their respective bootstrap proposal setting (in both the layers regarding EARBPF and EADPF). The results are presented in Fig. 5.4.

Although the EARBPF and EADPF approaches provide the best estimation precision at the lowest N (with the EADPF procedure being better), their hierarchical implementation makes them computationally more demanding compared to the remaining—conceptually simpler—algorithms. The RBPF method offers the most favorable computational time and estimation precision for low N. However, this preferred behavior starts to be less pronounced with increasing N. For high N, the IMMPF2 approach seems to provide the lowest computational time at approximately the same estimation precision compared to the remaining methods. The IMMPF1 algorithm reveals rather poor performance than the non-nested methods, as its RMSE and computational time are higher for almost all N.

To conclude, the best choice for developing the PMCMC kernel for jump Markov nonlinear models is the RBPF method [166] since—as obvious from Fig. 5.4—it achieves the best estimation precision and computational time at the lowest number of particles. The justification for this conclusion lies in that the PGAS kernels are often very efficient with a low number of particles [130]. Therefore, we need to base their development on particle filters that are most precise, and least computationally demanding, at the lowest number of particles. From this point of view, the IMMPF1, IMMPF2, EARBPF, and EADPF techniques are inappropriate as they rely on computing the approximation of the mode-conditioned predictive likelihood, $p(y_t|c_t, y_{1:t-1})$, which suffers from non-zero bias and substantial variance when the number of particles is low. Moreover, if the EARBPF or EADPF algorithms were chosen to proceed with the design, the resulting PGAS kernel would be conceptually rather difficult. There would be the need to impose the conditioning on the reference trajectory over the two nested layers. Such an approach would most likely be computationally very demanding due to the fact that the PGAS kernel has—on its own—slightly more intricate algorithmic flow, and it is thus technically more demanding to process by computing devices.

6 A PARTICLE SAEM ALGORITHM TO IDENTIFY JUMP MARKOV NONLINEAR MODELS

The identification of static parameters in jump Markov nonlinear models (JMNMs) poses a key challenge in explaining nonlinear and abruptly changing behavior of dynamical systems. This chapter introduces a stochastic approximation expectation maximization algorithm to facilitate offline maximum likelihood parameter estimation in JMNMs. The method relies on the construction of a particle Gibbs kernel that takes advantage of the inherent structure of the model to increase the efficiency through Rao-Blackwellization. Numerical examples illustrate that the proposed solution outperforms related approaches.

6.1 Introduction

6.1.1 Context

Jump Markov nonlinear models (JMNMs) can be seen as a particular class of nonlinear and non-Gaussian state-space models (SSMs, [30]) where the observation variable is related to the *latent* state variable that contains a continuous and discrete-valued part. While the continuous part describes the dynamics of a system, the discrete one indicates the switching of different dynamical *modes*.

The expectation maximization (EM) algorithm by [51] has become a standard tool to address the maximum likelihood (ML) parameter estimation in SSMs. The method is favored especially for its inherent feature of splitting the ML problem into two more conveniently tractable steps known as expectation and maximization. In the model class considered here, the expectation step is intractable and requires us to solve the nonlinear smoothing problem [133]. The particle Markov chain Monte Carlo (PMCMC) methods [4], which rely on sequential Monte Carlo (SMC, [56]) to facilitate the construction of high-dimensional proposal *kernels* in (MCMC, [3]), embody an efficient tool to address the issue. The paper [132] recently elaborated on the PMCMC idea and suggested to combine their particle Gibbs with ancestor sampling (PGAS) kernel and the stochastic approximation EM (SAEM) algorithm of [50] to obtain the particle SAEM (PSAEM) procedure. The related paper [202] then extended this design to propose a Rao-Blackwellized PSAEM (RBPSAEM) algorithm for jump Markov linear models by utilizing their linear Gaussian substructure.

A recent EM approach specifically tailored for JMNMs was proposed by [11] who extended the particle smoothing EM (PSEM) framework of [194]. The method proposed herein differs from this approach mainly in using stochastic approximation, Rao-Blackwellization, and PMCMC-based smoothing. Another EM solution was developed by [166] who introduced a Rao-Blackwellized forward smoother, which differs from the present method also in the smoothing methodology but shares similarities with the specific type of Rao-Blackwellization.

6.1.2 Contributions

The contribution of this chapter consists in developing an RBPSAEM method for JMNMs which exploits the substructure related to the discrete state. This is achieved by formulating a conditional version of the RBPF proposed by [166]. To facilitate the ancestor sampling, a finite state-space variant of the backward information filter (BIF, [148]) is proposed, requiring us to change the scale of the associated backward recursion. The experimental evidence indicates that the proposed method offers a higher estimation accuracy compared to competing approaches.

6.2 Background

6.2.1 Problem Formulation

Consider the discrete-time JMNM given by

$$\begin{aligned} c_t &\sim p(c_t|c_{t-1}), && (6.1\text{a}) \\ z_t &\sim f(z_t|c_t, z_{t-1}; \theta_{c_t}), && (6.1\text{b}) \\ y_t &\sim g(y_t|c_t, z_t; \theta_{c_t}), && (6.1\text{c}) \end{aligned}$$

where the continuous states and observations are denoted by $z_t \in \mathsf{Z} \subseteq \mathbb{R}^{n_z}$ and $y_t \in \mathsf{Y} \subseteq \mathbb{R}^{n_y}$, respectively. The discrete state $c_t \in \mathsf{C} := \{1, \dots, K\}$ indicates the currently active mode, with K being the total number of the modes. We assume that each mode is described by the probability densities $f(\cdot; \theta_{c_t})$ and $g(\cdot; \theta_{c_t})$, where θ_{c_t} is the associated parameter set. The probability distribution $p(\cdot)$ governs the switching between the modes and is parameterized by the $K \times K$ transition probability matrix Π with the entries

$$\Pi_{mn} := \mathbb{P}(c_t = n | c_{t-1} = m) = p(n|m). \qquad (6.2)$$

The set of all unknown parameters, $\theta \in \Theta \subseteq \mathbb{R}^{n_\theta}$, is defined by $\theta := \{\Pi, \{\theta_n\}_{n=1}^K\}$. At the initial time instance, the latent states are distributed according to $z_1 \sim \mu(z_1|c_1)$ and $c_1 \sim \nu(c_1)$; both μ and ν are assumed to be known.

We search for the parameter estimate maximizing the likelihood of the observed data sequence $y_{1:T} \coloneqq (y_1, \dots, y_T)$, with T denoting its length, that is,

$$\widehat{\theta}_{\mathrm{ML}} = \underset{\theta \in \Theta}{\arg\max}\, p_\theta(y_{1:T}). \tag{6.3}$$

In the present class of models, the computation of $p_\theta(y_{1:T})$ cannot be conducted exactly, as it contains the summation over K^T possible values, which is infeasible to perform even for a moderate T. Additionally, the integration over Z^T required for evaluating $p_\theta(y_{1:T})$ cannot be performed either, as the model is supposed to contain nonlinearities.

6.2.2 EM and SAEM Algorithms

The EM algorithm [51] is a popular tool to facilitate maximum likelihood estimation in models that contain latent data $x_{1:T}$, such as $x_t \coloneqq (z_t, c_t)$. The method relies on that the expected value of the complete data log-likelihood, the so-called *intermediate quantity*

$$\mathcal{Q}(\theta, \theta') \coloneqq \int \log p_\theta(x_{1:T}, y_{1:T}) p_{\theta'}(x_{1:T}|y_{1:T}) dx_{1:T}, \tag{6.4}$$

can be used to locate the maximizer of the incomplete data likelihood $p_\theta(y_{1:T})$. The maximizer is found indirectly in the two-step iterative procedure which alternates between the expectation (E) and maximization (M) according to

$$\begin{aligned}
&\text{(E) compute } \mathcal{Q}(\theta, \theta[k-1]),\\
&\text{(M) compute } \theta[k] = \underset{\theta \in \Theta}{\arg\max}\, \mathcal{Q}(\theta, \theta[k-1]),
\end{aligned}$$

starting with some $\theta[0] \in \Theta$. The main motivation behind postulating (6.4) is to simplify, or at least more conveniently reformulate, the original problem (6.3).

Under mild regularity assumptions, [223] proved that the produced sequence of the parameter estimates $\{\theta[k]\}_{k \geq 0}$ converges towards a stationary point of $p_\theta(y_{1:T})$. Additionally, the method embodies a monotone optimization procedure as the corresponding sequence of log-likelihood evaluations $\{\log p_{\theta[k]}(y_{1:T})\}_{k \geq 0}$ is non-decreasing.

When no explicit solution is available for implementing the E-step, one can resort to the SAEM algorithm proposed by [50]. The basic idea is to redefine the E-step so that we first simulate the samples $x_{1:T}[k]$ from the joint smoothing density $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$ and then use them in the stochastic approximation [182] of the intermediate quantity (6.4) given by

$$\widehat{\mathcal{Q}}_k(\theta) = (1 - \alpha_k)\widehat{\mathcal{Q}}_{k-1}(\theta) + \alpha_k \log p_\theta(x_{1:T}[k], y_{1:T}), \tag{6.6}$$

where $\alpha_k \in [0, 1]$ is a positive step size satisfying the constraints $\sum_{k \geq 1} \alpha_k = \infty$ and $\sum_{k \geq 1} \alpha_k^2 < \infty$. However, for complicated models, direct sampling from the

Algorithm 19 Stochastic Approximation Expectation Maximization (SAEM)

```
A. Initial step: $(k = 0)$
    1. Set $x_{1:T}[0] \in \mathsf{X}^T$, $\theta[0] \in \Theta$, and $\widehat{\mathcal{Q}}_0(\theta) := 0$.
B. Recursive step: $(k = 1, \ldots, R)$
    1. Sample $x_{1:T}[k] \sim \mathcal{K}_{\theta[k-1]}(\cdot|x_{1:T}[k-1])$.
    2. Compute $\widehat{\mathcal{Q}}_k(\theta)$ according to (6.6).
    3. Compute $\theta[k] = \arg\max_{\theta\in\Theta} \widehat{\mathcal{Q}}_k(\theta)$.
```

joint smoothing density $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$ is often infeasible. In such situations, we can utilize the MCMC framework [121] and thus draw the samples from a Markov kernel $\mathcal{K}_{\theta[k-1]}$ admitting the joint smoothing density as its unique stationary density. We summarize this MCMC version of the SAEM method in Algorithm 19.

Under the requirement of uniform ergodicity of the kernel $\mathcal{K}_{\theta[k-1]}$, and with some regularity assumptions, [121] have proven that the sequence $\{\theta[k]\}_{k\geq 0}$ converges to a maximizer of $p_\theta(y_{1:T})$ when the complete data likelihood belongs to the exponential family.

An alternative way of sampling from $p_{\theta[k-1]}(x_{1:T}|y_{1:T})$ is to utilize the PSEM framework by [194], which covers the alternative SMC-based smoothing strategies previously used in the EM context, including those presented by [163], [66]. Nevertheless, the PSEM approach does not rely on the stochastic approximation to reuse the samples over the iterations and thus requires a notably higher computational budget.

However, efficient construction of the kernel $\mathcal{K}_{\theta[k-1]}$ may be difficult with high T. [132] addressed this problem by designing the PGAS kernel which, when combined with SAEM, can be used to form the PSAEM algorithm (see also [130]). The development in the present work is based on such ideas. To simplify the subsequent exposition, let us describe the basic PG kernel, leaving the AS modification to another section.

6.2.3 Particle Gibbs Kernel

The conditional SMC (CSMC) update introduced by [4] offers a possible approach for constructing a Markov kernel on X^T. To describe the method, we first briefly present the standard SMC sampler and then incorporate the features that establish the conditional version.

SMC methods [56] constitute a general class of procedures appropriate for approximating a sequence of *target* densities $\{p_\theta(x_{1:t}|y_{1:t})\}_{t=1}^T$. The SMC approximation embodies the empirical measure

$$\widehat{p}_\theta(dx_{1:t}|y_{1:t}) = \sum_{i=1}^N w_t^i \delta_{x_{1:t}^i}(dx_{1:t}), \tag{6.7}$$

which is fully determined by the *weighted particle system* $\{x_{1:t}^i, w_t^i\}_{i=1}^N$, where $x_{1:t}^i$ denotes a particle trajectory and w_t^i labels a weight that assesses the contribution of the associated trajectory to the resulting approximation.

The *initial step* of an SMC sampler is assembled of standard importance sampling. Thus, we first draw the particles $\{x_1^i\}_{i=1}^N$ from the initial proposal density $x_1^i \sim q_\theta(\cdot|y_1)$ and then calculate the importance weights $w_1^i \propto W_{\theta,1}(x_1^i)$ for $i = 1, \ldots, N$, where $W_{\theta,1}(x_1) = p_\theta(x_1, y_1)/q_\theta(x_1|y_1)$.

The *recursive step* combines sequential importance sampling and resampling in order to propagate (6.7) recursively in time. Assume we have the previously generated particle system $\{x_{1:t-1}^i, w_{t-1}^i\}_{i=1}^N$. The recursion starts with the resampling procedure, which is equivalent to drawing a set of ancestor indices $\{a_t^i\}_{i=1}^N$ according to

$$\mathbb{P}(a_t^i = j) = w_{t-1}^j, \qquad j = 1, \ldots, N. \tag{6.8}$$

Then, we make use of the indices in the sequential importance sampling approach. First, this procedure involves generating the particles $\{x_t^i\}_{i=1}^N$ from the proposal density

$$x_t^i \sim q_\theta(\cdot|x_{1:t-1}^{a_t^i}, y_{1:t}), \tag{6.9}$$

where a_t^i determines the parent trajectory for the offspring particle x_t^i. The set of ancestor indices $\{a_{1:t}^i\}_{i=1}^N$ contains information about the genealogy of the particles. Second, the previous trajectories are extended as

$$x_{1:t}^i \coloneqq \{x_{1:t-1}^{a_t^i}, x_t^i\},$$

and, finally, we conclude the recursive step by computing the normalized importance weights $w_t^i \propto W_{\theta,t}(x_{1:t}^i)$ for $i = 1, \ldots, N$, where

$$W_{\theta,t}(x_{1:t}) \coloneqq \frac{p_\theta(y_t, x_t|x_{1:t-1}, y_{1:t-1})}{q_\theta(x_t|x_{1:t-1}, y_{1:t})}. \tag{6.10}$$

The basic idea of the CSMC update is to modify the above described SMC sampler so that one particle trajectory $x'_{1:T}$ – called *reference trajectory* – can be specified in advance. This is achieved by sampling from (6.8) and (6.9) only for $i = 1, \ldots, N-1$, whereas the remaining particle and ancestor index are set as $x_t^N \coloneqq x'_t$ and $a_t^N \coloneqq N$, respectively. After the procedure has completed, we find a new reference trajectory by drawing k with $\mathbb{P}(k = i) = w_T^i$ and selecting $x_{1:T}^k$ from $\{x_{1:T}^i\}_{i=1}^N$. The fact that the method requires a state trajectory as the input and returns another state trajectory as the output defines a Markov kernel on X^T, which is referred to as the PG kernel.

6.3 Expectation

A straightforward way to implement the E-step consists in designing the kernel $\mathcal{K}_\theta$ such that it produces the samples $x_{1:T}[k] \coloneqq (z_{1:T}, c_{1:T})[k]$ from the joint smoothing density $p_\theta(c_{1:T}, z_{1:T}|y_{1:T})$. However, applying the samples directly in the stochastic approximation (6.6) would not exploit the partial analytical tractability of the model. As an alternative solution, we use the decomposition

$$p_\theta(c_{1:T}, z_{1:T}|y_{1:T}) = p_\theta(c_{1:T}|z_{1:T}, y_{1:T})p_\theta(z_{1:T}|y_{1:T}) \tag{6.11}$$

and construct the kernel $\mathcal{K}_\theta$ so that it generates the samples $z_{1:T}[k]$ from the marginal density $p_\theta(z_{1:T}|y_{1:T})$, while the first factor in (6.11) is solved analytically. The stochastic approximation can therefore be rewritten as

$$\widehat{\mathcal{Q}}_k(\theta) = (1-\alpha_k)\widehat{\mathcal{Q}}_{k-1}(\theta) + \alpha_k \mathbb{E}_{\theta[k-1]}\Big[\log p_\theta(c_{1:T}, z_{1:T}[k], y_{1:T})\Big|z_{1:T}[k], y_{1:T}\Big], \tag{6.12}$$

where the expected value is taken w.r.t. $c_{1:T}$. To simplify the notation, we omit the parameters θ in the remaining part of this section.

6.3.1 Conditional Rao-Blackwellized Particle Filter

The development of the sought kernel is herein based on the RBPF introduced by [166], which applies Rao-Blackwellization in a way identical with that demonstrated in (6.11). With the help of the CSMC framework presented in the previous section, we can directly cast this RBPF into the conditional RBPF (CRBPF), having in mind the whole CSMC update now operates with z_t instead of x_t. This conversion is actually simple and comprises of only two actions. First, we compute the numerator in $W_{\theta,t}(z_{1:t})$ (cf. (6.10)) via the marginalization given by

$$p(y_t, z_t|z_{1:t-1}, y_{1:t-1}) = \sum_{c_t \in \mathsf{C}} p(y_t, z_t|c_t, z_{t-1})p(c_t|z_{1:t-1}, y_{1:t-1}), \tag{6.13}$$

where the first factor of the summand is represented by the product of (6.1c) and (6.1b). To compute (6.13), we need to introduce a finite state-space filter. This consists of the filtering recursion formed by the forward update

$$p(c_t|z_{1:t}, y_{1:t}) \propto p(y_t, z_t|c_t, z_{t-1})p(c_t|z_{1:t-1}, y_{1:t-1}) \tag{6.14a}$$

and the forward prediction

$$p(c_t|z_{1:t-1}, y_{1:t-1}) = \sum_{c_{t-1} \in \mathsf{C}} p(c_t|c_{t-1})p(c_{t-1}|z_{1:t-1}, y_{1:t-1}) \tag{6.14b}$$

which embodies the second factor in the summand of (6.13). Second, since $W_{\theta,t}(z^i_{1:t})$ has to be evaluated for all $i = 1, \ldots, N$ (implying the recursion given by (6.14a) and (6.14b) is also necessary to be computed for each $z^i_{1:t}$), we need to enlarge the memory allocated for the original particle system to incorporate the distributions $p(c_t|z^i_{1:t}, y_{1:t})$ as

$$\{z^i_{1:t}, p(c_t|z^i_{1:t}, y_{1:t}), w^i_t\}^N_{i=1}.$$

Both the changes (6.13) and (6.14b) are usually undertaken in interpreting PFs as RBPFs. See, e.g., [192] for a different context. Other details, such as the construction of the bootstrap proposal density, $p(z_t|z_{1:t-1}, y_{1:t})$, are discussed in [166].

6.3.2 Ancestor Sampling Weights

The above-described construction of the CRBPF inherits the well-known drawback of the standard SMC sampler, namely, the particle path degeneracy problem (see [94]). In consequence of this phenomenon, the sought kernel may deliver poor mixing. To improve upon this issue, [132] suggested to sample new values of the ancestor indices a^N_t according to

$$\mathbb{P}(a^N_t = i) = w^i_{t-1|T}, \qquad i = 1, \ldots, N, \tag{6.15}$$

where (in the present context)

$$w^i_{t-1|T} \propto w^i_{t-1} p(y_{t:T}, z'_{t:T}|z^i_{1:t-1}, y_{1:t-1}) \tag{6.16}$$

is the probability of connecting the ith historical trajectory with the future part of the reference one; see [132] for the derivation of (6.16) and the proof that the PGAS kernel is uniformly ergodic.

To increase the impact of Rao-Blackwellization, in addition to that brought by the CRBPF, we express the predictive density in (6.16) by the marginalization

$$p(y_{t:T}, z'_{t:T}|z^i_{1:t-1}, y_{1:t-1}) = \sum_{c_{t-1} \in \mathsf{C}} p(y_{t:T}, z'_{t:T}|z^i_{t-1}, c_{t-1}) p(c_{t-1}|z^i_{1:t-1}, y_{1:t-1}). \tag{6.17}$$

A similar approach for computing the predictive density has recently proved to be a key element in designing Rao-Blackwellized particle smoothers (RBPSs), see [218, 189]. The summand in (6.17) is reminiscent of the original two-filter smoothing formula [24], which is based on running one filter forward and the other backward in time. In our situation, we exploit the forward filter represented by (6.14a) and (6.14b). The backward filter is designed below.

6.3.3 Finite State-Space Backward Information Filter

The BIF [148] allows us to compute the likelihood term in the summand of (6.17). In the above context, the recursive step of such a filter iterates, for $t = T-1, \ldots, 1$, over the backward prediction

$$p(y_{t+1:T}, z_{t+1:T}|z_t, c_t) = \sum_{c_{t+1}\in\mathsf{C}} p(y_{t+1:T}, z_{t+2:T}|z_{t+1}, c_{t+1})p(z_{t+1}, c_{t+1}|c_t, z_t) \tag{6.18a}$$

and the backward update

$$p(y_{t:T}, z_{t+1:T}|z_t, c_t) = p(y_t|c_t, z_t)p(y_{t+1:T}, z_{t+1:T}|z_t, c_t). \tag{6.18b}$$

The initial step calculates only the observation density (6.1c) at $t = T$. A similar recursive form was recently used to develop an RBPS for mixed linear/nonlinear SSMs in [131].

However, the difficulty encountered with (6.18a) and (6.18b) consists in the fact that neither of these is a probability distribution in the argument c_t. Therefore, computing the likelihood term in (6.17) according to the recursion (6.18) may result in numerical instability [25]. To deal with this issue, we need to change the scale of the recursion (6.18). The following two propositions show respectively the rescaled version of (6.18) and how to use it for computing the likelihood term in (6.17).

Proposition 6.1. *Let $\alpha_t(c_t|z_{t:T})$ and $\alpha_{t+1}(c_t|z_{t:T})$ denote the rescaled backward filtering and predictive distributions, respectively. Then, the backward update (6.18b) can be rewritten as*

$$\alpha_t(c_t|z_{t:T}) \propto p(y_t|c_t, z_t)\alpha_{t+1}(c_t|z_{t:T}), \tag{6.19a}$$

and, similarly, the backward prediction (6.18a) becomes

$$\alpha_{t+1}(c_t|z_{t:T}) \propto \sum_{c_{t+1}\in\mathsf{C}} \alpha_{t+1}(c_{t+1}|z_{t+1:T})p(z_{t+1}, c_{t+1}|c_t, z_t), \tag{6.19b}$$

for $t = T-1, \ldots, 1$. At the initial time step $t = T$, the recursion starts with $\alpha_T(c_T|z_T) \propto p(y_T|c_T, z_T)$.

Proof. See Section 6.7. □

Proposition 6.2. *Consider we applied Proposition 6.1 to obtain the rescaled backward filtering distribution $\alpha_t(c_t|z'_{t:T})$, where $z'_{t:T}$ is the partial reference trajectory. Then, the likelihood term in (6.17) can be computed according to*

$$p(y_{t:T}, z'_{t:T}|z^i_{t-1}, c_{t-1}) \propto \sum_{c_t\in\mathsf{C}} \alpha_t(c_t|z'_{t:T})p(z'_t, c_t|c_{t-1}, z^i_{t-1}), \tag{6.20}$$

for $i = 1, \ldots, N$.

Proof. See Section 6.7. □

The above derived results can now be used to summarize the proposed RBPGAS kernel in Algorithm 20. Here, the knowledge of $z'_{1:T} = z_{1:T}[k-1]$ allows us to run the BIF in the initial step (A1) of Algorithm 20. The precomputed distributions $\alpha_t(c_t|z'_{t:T})$ are then used in the recursive step (B3) to obtain the ancestor sampling weights.

Algorithm 20 RBPGAS Kernel for JMNMs

```
**Inputs:** $z'_{1:T} = z_{1:T}[k-1]$.
**Outputs:** $z_{1:T}[k]$ and $\{z^i_{1:T}, \{p(c_t|z^i_{1:t}, y_{1:t})\}^T_{t=1}, w^i_T\}^N_{i=1}$.
A. **Initial step:** $(t = 1)$
  1. Use $z'_{1:T}$ in (6.19) to compute $\{\alpha_t(c_t|z'_{t:T})\}^T_{t=1}$.
  2. Sample $z^i_1 \sim q_1(\cdot)$ for $i = 1, \dots, N-1$ and set $z^N_1 := z'_1$.
  3. Compute $p(c_1|z^i_1, y_1)$ and $p(y_1, z^i_1)$ according to (6.14a) and (6.13), respectively, for $i = 1, \dots, N$.
  4. Compute $w^i_1 \propto W_1(z^i_1)$ for $i = 1, \dots, N$.
B. **Recursive step:** $(t = 2, \dots, T)$
  1. Sample $a^i_t$ with $\mathbb{P}(a^i_t = j) = w^j_{t-1}$ for $i = 1, \dots, N-1$.
  2. Sample $z^i_t \sim q(\cdot|z^{a^i_t}_{1:t-1}, y_{1:t})$ for $i = 1, \dots, N-1$.
  3. Compute $w^i_{t-1|T}$ according to (6.16), (6.17), and (6.20), utilizing $\alpha_t(c_t|z'_{t:T})$, for $i = 1, \dots, N$.
  4. Sample $a^N_t$ with $\mathbb{P}(a^N_t = i) = w^i_{t-1|T}$ and set $z^N_t := z'_t$.
  5. Set $z^i_{1:t} := \{z^i_t, z^{a^i_t}_{1:t-1}\}$ for $i = 1, \dots, N$.
  6. Compute $p(y_t, z^i_t|z^{a^i_t}_{1:t-1}, y_{1:t-1})$ and $p(c_t|z^i_{1:t}, y_{1:t})$ according to (6.13) and (6.14), respectively, for $i = 1, \dots, N$.
  7. Compute $w^i_t \propto W_t(z^i_{1:t})$ for $i = 1, \dots, N$.
C. **Final step:**
  1. Sample $k$ with $\mathbb{P}(k = i) = w^i_T$ and set $z_{1:T}[k] := z^k_{1:T}$.
```

6.3.4 Finite State-Space Forward-Backward Smoother

To facilitate the computation of the expected value in (6.12), we prepare a finite state-space variant of the forward-backward smoother conditioned on the trajectory $z_{1:T}$; see, e.g., section 5 of [194] for the derivation. In the backward recursion, for $t = T-1, \dots, 1$, the algorithm computes the marginal smoothing distributions

$$p(c_t|z_{1:T}, y_{1:T}) = \sum_{c_{t+1} \in \mathsf{C}} p(c_t, c_{t+1}|z_{1:T}, y_{1:T}) \tag{6.21}$$

and

$$p(c_t, c_{t+1}|z_{1:T}, y_{1:T}) = \frac{p(c_{t+1}|c_t)p(c_t|z_{1:t}, y_{1:t})}{p(c_{t+1}|z_{1:t}, y_{1:t})} p(c_{t+1}|z_{1:T}, y_{1:T}), \tag{6.22}$$

where $p(c_t|z_{1:t}, y_{1:t})$ is produced by the forward recursion formed by (6.14b) and (6.14a). At the initial step $t = T$, the smoothing distribution (6.21) agrees with the filtering one.

6.4 Maximization

To implement the M-step, one needs to be more concrete about the densities (6.1b) and (6.1c). Let us therefore consider the example of estimating the parameters of additive Gaussian noise variables in the following JMNM:

$$z_t = a(z_{t-1}, c_t) + v_t, \qquad v_t \sim \mathcal{N}(\mu_{v,c_t}, \Sigma_{v,c_t}), \tag{6.23a}$$

$$y_t = b(z_t, c_t) + w_t, \qquad w_t \sim \mathcal{N}(\mu_{w,c_t}, \Sigma_{w,c_t}), \tag{6.23b}$$

where v_t and w_t are the mutually independent Gaussian noise variables with the mode-dependent mean vector $\mu_{\cdot,c_t}$ and covariance matrix $\Sigma_{\cdot,c_t}$. The terms $a(\cdot)$ and $b(\cdot)$ denote some known nonlinear functions. For all $c_t \in \mathsf{C}$, we intend to estimate the parameter set $\theta_{c_t} = \{\mu_{v,c_t}, \Sigma_{v,c_t}, \mu_{w,c_t}, \Sigma_{w,c_t}\}$, along with the matrix Π.

The important feature of (6.23) consists in that it allows us to simplify (6.12) into a closed-form algebraic solution, as demonstrated by the following lemma.

Lemma 6.1. *For the model (6.23), with its unknown parameters, the computation of the stochastic approximation (6.12) reduces to recursively updating the sufficient statistics*

$$\mathbb{S}_k = (1 - \alpha_k)\mathbb{S}_{k-1} + \alpha_k S_k,$$

where $S_k = (S_k^{(1)}, \ldots, S_k^{(5)})$ is assembled of the vectors containing the entries

$$S^{(1)}_{mn,k} = \sum_{t=2}^{T} \mathbb{E}_{\theta[k-1]}\Big[\mathbb{1}(c_t = n, c_{t-1} = m)\Big| z_{1:T}[k], y_{1:T}\Big], \tag{6.24a}$$

$$S^{(2)}_{n,k} = \sum_{t=2}^{T} \mathbb{E}_{\theta[k-1]}\Big[\mathbb{1}(c_t = n)\Big| z_{1:T}[k], y_{1:T}\Big], \tag{6.24b}$$

$$S^{(3)}_{n,k} = \sum_{t=2}^{T} \mathbb{E}_{\theta[k-1]}\Big[s^{(3)}(z_{t-1:t}[k], c_t = n)\Big| z_{1:T}[k], y_{1:T}\Big], \tag{6.24c}$$

$$S^{(4)}_{n,k} = \sum_{t=1}^{T} \mathbb{E}_{\theta[k-1]}\Big[\mathbb{1}(c_t = n)\Big| z_{1:T}[k], y_{1:T}\Big], \tag{6.24d}$$

$$S^{(5)}_{n,k} = \sum_{t=1}^{T} \mathbb{E}_{\theta[k-1]}\Big[s^{(5)}(y_t, z_t[k], c_t = n)\Big| z_{1:T}[k], y_{1:T}\Big]. \tag{6.24e}$$

Here, $(m, n) \in \mathsf{C}^2$ and

$$s^{(3)}(z_{t-1:t}, c_t = n) = \mathbb{1}(c_t = n)[(z_t - a(z_{t-1}, c_t))^\top \ 1]^\top[\cdot],$$
$$s^{(5)}(y_t, z_t, c_t = n) = \mathbb{1}(c_t = n)[(y_t - b(z_t, c_t))^\top \quad 1]^\top[\cdot],$$

with $\mathbb{1}(\cdot)$ denoting the indicator function.

Proof. See Section 6.7. □

The expected values in (6.24a) and (6.24b-6.24e) are taken w.r.t. the previously prepared smoothing distributions (6.22) and (6.21), respectively. For a newly sampled reference trajectory $z_{1:T}[k]$, the smoothing distributions need to be computed only once per iteration k. Furthermore, to facilitate the estimation of the unknown parameters, we have to maximize (6.12). The resulting maximizers can also be computed in a closed form and are presented in the following lemma for completeness. The result can also be found in, e.g., [166].

Lemma 6.2. *At iteration k, the stochastic approximation (6.12) is maximized w.r.t. Π_{mn}, and $\mu_{v,n}$, $\Sigma_{v,n}$ by*

$$\Pi_{mn}[k] = \frac{\mathbb{S}^{(1)}_{mn,k}}{\sum_{l=1}^{K} \mathbb{S}^{(1)}_{ml,k}}, \qquad \forall (m,n) \in \mathsf{C}^2.$$

and

$$\mu_{v,n}[k] = \frac{\mathbb{S}^{(3)}_{n,k}\langle 2\rangle}{\mathbb{S}^{(2)}_{n,k}}, \qquad\qquad \Sigma_{v,n}[k] = \frac{\mathbb{S}^{(3)}_{n,k}\langle 1\rangle}{\mathbb{S}^{(2)}_{n,k}} - \mu_{v,n}[k]\mu_{v,n}[k]^\top,$$

respectively, where we use the partitioning

$$\mathbb{S}^{(3)}_{n,k} = \begin{bmatrix} \mathbb{S}^{(3)}_{n,k}\langle 1\rangle & \mathbb{S}^{(3)}_{n,k}\langle 2\rangle \\ \cdot & \cdot \end{bmatrix}.$$

Proof. See Section 6.7. □

The same approach as with $\mu_{v,n}[k]$ and $\Sigma_{v,n}[k]$ holds also for $\mu_{w,n}[k]$ and $\Sigma_{w,n}[k]$. Using Algorithm 20 to implement Step B1 in Algorithm 19, and replacing Steps B2 and B3 with the expressions from Lemmas 6.1 and 6.2, respectively, leads to the estimation procedure for the model (6.23).

6.5 Numerical Illustration

This section illustrates the performance of the proposed RBPSAEM algorithm compared to the PSAEM [130] and PSEM [194] procedures.

Let us consider that the functions in (6.23) are given by

$$a(z_{t-1}, c_t) = 0.5 z_{t-1} + 25 \frac{z_{t-1}}{1 + z_{t-1}^2} + 8\cos(1.2t), \tag{6.25a}$$

$$b(z_t, c_t) = 0.05 z_t^2, \tag{6.25b}$$

and the total number of modes is $K = 2$. The state noise parameters are known and mode-independent, with $\mu_v = 0$ and $\Sigma_v = 1$. We aim to estimate the remaining parameters $\mu_{w,1}$, $\mu_{w,2}$, $\Sigma_{w,1}$, $\Sigma_{w,2}$, Π_{11}, and Π_{22}, whose true values are 0, 8, 5, 1, 0.98, and 0.8, respectively. All the compared procedures are applied with the bootstrap proposal and 300 iterations. The number of sampled trajectories for the PSAEM and RBPSAEM algorithms is $N = 4$. For the PSEM approach, we simulate 90 forward and 10 backward trajectories. The experiments are performed on 20 different observation sequences of the length $T = 1000$. The remaining portion of the simulation settings is included in Section 6.8.

Fig. 6.1 shows that the proposed method surpasses the PSAEM procedure because of the lower (or very similar) bias and variance of the estimated parameters.

Although the PSEM technique is better in estimating the transition probabilities, the remaining estimates converge to incorrect values. The main reason then consists in that the PSEM algorithm does not rely on the stochastic approximation and thus requires a higher number of particles to perform similarly to the remaining procedures. Moreover, as the probability Π_{11} is close to its upper bound, the method suffers from the degeneracy around the mode changes [58]. Nevertheless, both the PSAEM and RBPSAEM algorithms seem to be more robust in this respect. In the present experiment, the time required to compute the proposed RBPSAEM procedure is approximately two times higher than the PSAEM method but forty times lower than the PSEM approach.

6.6 Discussion

Importantly, the proposed method is not limited only to situations with an analytically tractable M-step but can be combined with, e.g., the conditional M-step [149] or gradient-based search techniques.

The idea suggested by [130] that consists in averaging over the index k in the log part of (6.6) by using all the trajectories from one iteration of the RBPGAS kernel was also investigated. The results were practically the same as those presented here, and thus this strategy is left out in the present chapter. However, the approach may be more useful when a better proposal density is designed.

The total cost of computing one iteration with the PSEM, PSAEM, and RBPSAEM procedures scales according to $\mathcal{O}(TMN)$, $\mathcal{O}(TN)$, and $\mathcal{O}(TK^2N)$, respectively, where M is the number of backward trajectories used by the PSEM method [194]. The extra computational complexity of the RBPSAEM algorithm over the PSAEM algorithm, given by the K^2 term, is caused by the two levels of marginalization required to compute the ancestor sampling weights. Nevertheless, the lower variance and faster convergence rate of the estimates provided by the RBPSAEM approach may increase its computational efficiency over the PSAEM technique. The experiments supporting this statement are presented in Section 6.8.

6.7 Proofs

6.7.1 Proof of Proposition 1

The proof of Proposition 6.1 carries out by first formulating a basic BIF and then its rescaled version.

The derivation of the formula for computing the backward filtering distribution $p(c_t|y_{t:T}, z_{t:T})$ begins with the application of the law of conditional probability

$$\begin{aligned} p(c_t|y_{t:T}, z_{t:T}) &= \frac{p(c_t, y_{t:T}, z_{t:T})}{p(y_{t:T}, z_{t:T})} \\ &= \frac{p(y_t|c_t, y_{t+1:T}, z_{t:T})p(c_t|y_{t+1:T}, z_{t:T})}{p(y_t|y_{t+1:T}, z_{t:T})}. \end{aligned} \tag{6.26}$$

By using the fact that, given c_t and z_t, there is no further information about y_t contained in $y_{t+1:T}$ and $z_{t+1:T}$, the first term in the numerator of (6.26) becomes the observation model (6.1c),

$$p(y_t|c_t, y_{t+1:T}, z_{t:T}) = p(y_t|c_t, z_t),$$

which allows us to rewrite (6.26) according to

$$p(c_t|y_{t:T}, z_{t:T}) \propto p(y_t|c_t, z_t)p(c_t|y_{t+1:T}, z_{t:T}), \tag{6.27}$$

where the equality up to the proportionality constant $\propto$ is used.

The formula for calculating the backward predictive distribution $p(c_t|y_{t+1:T}, z_{t:T})$ can be derived by applying the marginalization

$$p(c_t|y_{t+1:T}, z_{t:T}) = \sum_{c_{t+1}\in \mathsf{C}} p(c_t, c_{t+1}|y_{t+1:T}, z_{t:T}).$$

Here, we continue by expressing the summand in the sense of the law of conditional probability

$$\begin{aligned} p(c_t, c_{t+1}|y_{t+1:T}, z_{t:T}) &= \sum_{c_{t+1}\in \mathsf{C}} \frac{p(c_t, c_{t+1}, y_{t+1:T}, z_{t:T})}{p(y_{t+1:T}, z_{t:T})} \\ &= \sum_{c_{t+1}\in \mathsf{C}} \frac{p(c_t, z_t|c_{t+1}, y_{t+1:T}, z_{t+1:T})p(c_{t+1}|y_{t+1:T}, z_{t+1:T})}{p(z_t|y_{t+1:T}, z_{t+1:T})}. \end{aligned} \tag{6.28}$$

For the first term in the numerator of (6.28), we perform the sequence of rearrangements given by

$$\begin{aligned} p(c_t, z_t|c_{t+1}, y_{t+1:T}, z_{t+1:T}) &= \frac{p(c_t, c_{t+1}, y_{t+1:T}, z_{t:T})}{p(c_{t+1}, y_{t+1:T}, z_{t+1:T})} \\ &= \frac{p(y_{t+1:T}, z_{t+2:T}|c_t, z_t, c_{t+1}, z_{t+1})p(c_t, z_t, c_{t+1}, z_{t+1})}{p(c_{t+1}, y_{t+1:T}, z_{t+1:T})} \\ &= \frac{p(y_{t+1:T}, z_{t+2:T}|c_t, z_t, c_{t+1}, z_{t+1})p(c_t, z_t|c_{t+1}, z_{t+1})}{p(y_{t+1:T}, z_{t+2:T}|c_{t+1}, z_{t+1})} \\ &= p(c_t, z_t|c_{t+1}, z_{t+1}), \end{aligned} \tag{6.29}$$

where, to obtain the last line, we apply

$$p(y_{t+1:T}, z_{t+2:T}|c_t, z_t, c_{t+1}, z_{t+1}) = p(y_{t+1:T}, z_{t+2:T}|c_{t+1}, z_{t+1}).$$

That is, utilizing the Markov property of transition models (6.1a) and (6.1b), we can see that, given c_{t+1} and z_{t+1}, there is no more information about $y_{t+1:T}$ and $z_{t+2:T}$ contained in c_t and z_t. After substituting (6.29) in (6.28), we obtain

$$p(c_t|y_{t+1:T}, z_{t:T}) \propto \sum_{c_{t+1}\in \mathsf{C}} p(c_t, z_t|c_{t+1}, z_{t+1})p(c_{t+1}|y_{t+1:T}, z_{t+1:T}). \tag{6.30}$$

The formula (6.30) closes the functional recursion of the sought BIF via the backward filtering distribution from the previous time step, $p(c_{t+1}|y_{t+1:T}, z_{t+1:T})$. The BIF is then formed by (6.27) and (6.30).

To rescale the filter, we first rewrite the backward transition kernel as

$$p(c_t, z_t|c_{t+1}, z_{t+1}) = \frac{p(c_{t+1}, z_{t+1}|c_t, z_t)p(c_t, z_t)}{p(c_{t+1}, z_{t+1})},$$

and plug it back in (6.30), which yields

$$\begin{aligned} p(c_t|y_{t+1:T}, z_{t:T}) &\propto \sum_{c_{t+1}\in \mathsf{C}} \frac{p(c_{t+1}, z_{t+1}|c_t, z_t)p(c_t, z_t)}{p(c_{t+1}, z_{t+1})} p(c_{t+1}|y_{t+1:T}, z_{t+1:T}) \\ &\propto \sum_{c_{t+1}\in \mathsf{C}} \frac{p(c_{t+1}, z_{t+1}|c_t, z_t)p(c_t|z_t)}{p(c_{t+1}|z_{t+1})} p(c_{t+1}|y_{t+1:T}, z_{t+1:T}). \end{aligned} \tag{6.31}$$

Subsequently, we divide both sides of (6.27) and (6.31) by the conditional prior density $p(c_t|z_t)$, resulting in

$$\frac{p(c_t|y_{t:T}, z_{t:T})}{p(c_t|z_t)} \propto p(y_t|c_t, z_t)\frac{p(c_t|y_{t+1:T}, z_{t:T})}{p(c_t|z_t)}, \tag{6.32a}$$

$$\frac{p(c_t|y_{t+1:T}, z_{t:T})}{p(c_t|z_t)} \propto \sum_{c_{t+1}\in \mathsf{C}} p(c_{t+1}, z_{t+1}|c_t, z_t)\frac{p(c_{t+1}|y_{t+1:T}, z_{t+1:T})}{p(c_{t+1}|z_{t+1})}. \tag{6.32b}$$

By introducing the notation

$$\alpha_t(c_t|z_{t:T}) := \frac{p(c_t|y_{t:T}, z_{t:T})}{p(c_t|z_t)}, \tag{6.33a}$$

$$\alpha_{t+1}(c_t|z_{t:T}) := \frac{p(c_t|y_{t+1:T}, z_{t:T})}{p(c_t|z_t)}, \tag{6.33b}$$

the recursion (6.32) can finally be rewritten into the form given by (6.19). □

6.7.2 Proof of Proposition 2

The requirement is to compute the likelihood term in (6.17) for a single partial reference trajectory $z'_{t:T}$ and multiple particles z^i_{t-1}, where $i = 1, \dots, N$. Therefore, it is convenient to rewrite it as

$$p(y_{t:T}, z'_{t:T}|z^i_{t-1}, c_{t-1}) = \sum_{c_t\in \mathsf{C}} p(y_{t:T}, z'_{t+1:T}|z'_t, c_t)p(z'_t, c_t|c_{t-1}, z^i_{t-1}). \tag{6.34}$$

This formulation allows us to compute $p(y_{t:T}, z_{t+1:T}|z_t, c_t)$ in a single backward pass through the data. However, computing $p(y_{t:T}, z_{t+1:T}|z_t, c_t)$ directly according to (6.18) may result in numerical instability. To resolve this issue, we use the fact that the backward filtering distribution is proportional to the product of the likelihood $p(y_{t:T}, z_{t+1:T}|z_t, c_t)$ and the conditional prior $p(c_t|z_t)$, that is,

$$\begin{aligned} p(c_t|y_{t:T}, z_{t:T}) &= \frac{p(c_t, y_{t:T}, z_{t:T})}{p(y_{t:T}, z_{t:T})} \\ &= \frac{p(y_{t:T}, z_{t+1:T}|z_t, c_t)p(c_t, z_t)}{p(y_{t:T}, z_{t:T})} \\ &= \frac{p(y_{t:T}, z_{t+1:T}|z_t, c_t)p(c_t|z_t)}{p(y_{t:T}, z_{t+1:T}|z_t)} \\ &\propto p(y_{t:T}, z_{t+1:T}|z_t, c_t)p(c_t|z_t). \end{aligned} \tag{6.35}$$

After dividing both sides of (6.35) by $p(c_t|z_t)$, we obtain

$$\frac{p(c_t|y_{t:T}, z_{t:T})}{p(c_t|z_t)} \propto p(y_{t:T}, z_{t+1:T}|z_t, c_t), \tag{6.36}$$

which can be used to replace the first term in the summand of (6.34). Here, we can notice that it is more convenient to find a recursive relation for a direct propagation of the ratio, than computing the conditional prior $p(c_t|z_t)$ and use it to divide the backward filtering distribution $p(c_t|y_{t:T}, z_{t:T})$ at each time step. This motivates the derivation of the rescaled BIF recursion given in Proposition 6.1.

The result (6.20) is then simply obtained by applying (6.33a) in (6.36) and substituting it for (6.34). □

6.7.3 Proof of Lemma 3

The proof of Lemma 6.1 starts with applying the chain rule to factorize the complete data log likelihood

$$\begin{aligned} \log p_\theta(c_{1:T}, z_{1:T}, y_{1:T}) = &\sum_{t=2}^{T} \log p(c_t|c_{t-1}) + \\ &\sum_{t=2}^{T} \log f(x_t|c_t, x_{t-1}; \theta_{c_t}) + \sum_{t=1}^{T} \log g(y_t|c_t, x_t; \theta_{c_t}). \end{aligned} \tag{6.37}$$

As mentioned in Section 6.2.1, the initial terms $\mu(z_1|c_1)$ and $\nu(c_1)$ are known and therefore excluded in (6.37). (This is also the reason why (6.24a-6.24c) start to sum at $t = 2$.) To find a concrete form of the expected value in (6.12), let us write the model (6.1) for the specific case (6.23) as

$$p(c_t|c_{t-1}) = \prod_{m=1}^{K}\prod_{n=1}^{K} \Pi_{mn}^{\mathbb{1}(c_t=n, c_{t-1}=m)}, \tag{6.38a}$$

$$f(z_t|c_t, z_{t-1}; \theta_{c_t}) \propto \prod_{n=1}^{K} |\Sigma_{v,n}|^{-0.5\mathbb{1}(c_t=n)} \times \exp\Big\{ -0.5\,\mathrm{tr}\,\Big(H_n^v s^{(3)}(z_{t-1:t}, c_t=n)\Big)\Big\}, \tag{6.38b}$$

$$g(y_t|c_t, z_t; \theta_{c_t}) \propto \prod_{n=1}^{K} |\Sigma_{w,n}|^{-0.5\mathbb{1}(c_t=n)} \times \exp\Big\{ -0.5\,\mathrm{tr}\,\Big(H_n^w s^{(5)}(y_t, z_t, c_t=n)\Big)\Big\}, \tag{6.38c}$$

where we introduce the parameter-dependent terms

$$H_n^v = [I \;\; \mu_{v,n}]^\top \Sigma_{v,n}^{-1}\,[\cdot],$$
$$H_n^w = [I \;\; \mu_{w,n}]^\top \Sigma_{w,n}^{-1}\,[\cdot],$$

with the statistics $s^{(3)}$ and $s^{(5)}$ being given as in Lemma 6.1. After substituting (6.38) for the corresponding terms in (6.37), and using the linearity of expectation operator, we obtain

$$\mathbb{E}_{\theta[k-1]}\Big[\log p_\theta(c_{1:T}, z_{1:T}[k], y_{1:T})\Big| z_{1:T}[k], y_{1:T}\Big] \coloneqq \sum_{m=1}^{K}\sum_{n=1}^{K} S_{mn,k}^{(1)} \log \Pi_{mn} - \frac{1}{2}\sum_{n=1}^{K}\Big(\log|\Sigma_{v,n}| S_{n,k}^{(2)} + \mathrm{tr}(H_n^v S_{n,k}^{(3)})\Big) - \frac{1}{2}\sum_{n=1}^{K}\Big(\log|\Sigma_{w,n}| S_{n,k}^{(4)} + \mathrm{tr}(H_n^w S_{n,k}^{(5)})\Big), \tag{6.39}$$

where the statistics (6.24) are introduced. Note we ignore the constant terms in (6.39). The expected value in the form (6.39) can conveniently be written as the inner product

$$\mathbb{E}_{\theta[k-1]}\Big[\log p_\theta(c_{1:T}, z_{1:T}[k], y_{1:T})\Big| z_{1:T}[k], y_{1:T}\Big] = \langle S_k, \psi(\theta)\rangle, \tag{6.40}$$

with the vector $S_k = (S_k^{(1)}, \dots, S_k^{(5)})$, whose entries are given by those defined in (6.24), and function $\psi(\theta)$ which summarizes the corresponding parameter-dependent terms. The form of the r.h.s. of (6.40) is reproduced across all iterations k, which enables us to rewrite (6.12) according to

$$\widehat{\mathcal{Q}}_k(\theta) = \langle \mathbb{S}_k, \psi(\theta)\rangle = \langle (1-\alpha_k)\mathbb{S}_{k-1} + \alpha_k S_k, \psi(\theta)\rangle. \tag{6.41}$$

Now, as the dependence between the corresponding entries of the statistics $\mathbb{S}_k$ and parameter-dependent terms in $\psi(\theta)$ is linear, the maximization of (6.41) w.r.t. a given parameter will result in an estimator which is only a function of the statistics (c.f. Lemma 6.2). This statement concludes the proof of Lemma 6.1. □

6.7.4 Proof of Lemma 4

To better clarify the maximization of (6.12), let us reuse the derivations of the proof of Lemma 6.1 for writing (6.41) as

$$\begin{aligned}\langle \mathbb{S}_k, \psi(\theta)\rangle = \sum_{m=1}^{K}\sum_{n=1}^{K} \mathbb{S}^{(1)}_{mn,k} \log \Pi_{mn} - \frac{1}{2}\sum_{n=1}^{K}\Big(\log|\Sigma_{v,n}|\mathbb{S}^{(2)}_{n,k} + \operatorname{tr}(H^v_n \mathbb{S}^{(3)}_{n,k})\Big) \\ - \frac{1}{2}\sum_{n=1}^{K}\Big(\log|\Sigma_{w,n}|\mathbb{S}^{(4)}_{n,k} + \operatorname{tr}(H^w_n \mathbb{S}^{(5)}_{n,k})\Big). \qquad (6.42)\end{aligned}$$

We first perform the maximization w.r.t. Π_{mn}. Therefore, we leave the Π_{mn}-independent terms in (6.42), which reduces the optimization problem to

$$\begin{aligned}&\underset{\Pi_{mn}}{\text{maximize}} \sum_{m=1}^{K}\sum_{n=1}^{K} \mathbb{S}^{(1)}_{mn,k} \log \Pi_{mn}, \\ &\text{subject to} \sum_{n=1}^{K} \Pi_{mn} = 1,\ m \in \mathsf{C},\ \Pi_{mn} \geq 0,\ \forall (m,n) \in \mathsf{C}^2.\end{aligned}$$

The solution can be found by formulating the Lagrangian

$$\sum_{n=1}^{K} \mathbb{S}^{(1)}_{mn,k} \log \Pi_{mn} + \lambda(1 - \sum_{l=1}^{K} \Pi_{ml})$$

for each $m \in \mathsf{C}$, where λ is the Lagrange multiplier. Since the optimized function is concave, taking the partial derivatives w.r.t. Π_{mn} and λ leads to the first-order necessary and sufficient conditions to find the global maximizer $\Pi_{mn}[k]$.

Subsequently, we find the maximizers w.r.t. $\mu_{v,n}$ and Σ_{mn}. Ignoring the $\mu_{v,n}$ and Σ_{mn}-independent terms in (6.42), the optimization problem is reduced to

$$\underset{\mu_{v,n};\Sigma_{v,n}}{\text{maximize}}\ -\log|\Sigma_{v,n}|\mathbb{S}^{(2)}_{n,k} - \operatorname{tr}(H^v_n \mathbb{S}^{(3)}_{n,k}).$$

Similarly to the previous case, taking the partial derivatives w.r.t. $\mu_{v,n}$ and $\Sigma^{-1}_{v,n}$ leads to the first-order necessary and sufficient conditions for finding the global maximizers $\mu_{v,n}[k]$ and $\Sigma_{v,n}[k]$. (The required matrix derivatives are $\frac{\partial}{\partial X}\log|X| = X^{-\top}$, $\frac{\partial}{\partial X}\operatorname{tr}(AXB) = A^\top B^\top$, $\frac{\partial}{\partial X}\operatorname{tr}(AX^{-1}B) = -X^{-\top}A^\top B^\top X^{-\top}$.) $\square$

6.8 Additional Results

This section presents additional experiments with the jump Markov nonlinear model (6.25). We assess the compared algorithms in terms of the computational time and estimation accuracy of the hidden state and parameter trajectories.

To facilitate reproducibility of the numerical illustrations, we first provide the rest of the simulation settings presented in Section 6.5. The continuous state prior is defined as mode-independent $\mu(z_1|c_1) \coloneqq \mathcal{N}(z_1; 0, 1)$, and the discrete state prior $\nu(c_1)$ is fixed to the vector $(0.5, 0.5)$. The step-size sequence is generated as $\alpha_k = 1$ for $k \leq 50$ and $\alpha_k = (k - 50)^{-0.7}$ for $k > 50$. The compared algorithms are randomly (using uniform distribution) initialized from $[0.5\theta, 1.5\theta]$, excepting the diagonal entries of Π initialized from $[0, 1]$.

In the first experiment, we evaluate the computational time of the compared algorithms across multiple iterations, with the same number of particles as considered before. To assess the estimation accuracy, the 'ground truth' continuous state, discrete state, and parameter trajectories were generated by the RBPSAEM method with $N = 1024$ particles. Fig. 6.2 illustrates that the RMSE of the continuous state trajectories is comparable for the PSAEM and RBPSAEM algorithms until approximately the first second of the simulation run. After this point, the RBPSAEM approach starts to outperform the PSAEM procedure by reaching lower values of the RMSE in less computational time. The NNZE of the discrete state trajectory is more favorable for the RBPSAEM algorithm, which is obvious from that its values are lower compared to the PSAEM method for almost full duration of the experiment. The performance in terms of the NNZE begins to be comparative for the PSAEM and RBPSAEM techniques as they approach ergodic regime. A similar behavior can also be observed for the RMSE of the parameters, where the proposed RBPSAEM algorithm again surpasses its PSAEM counterpart. Specifically, the RBPSAEM method achieves a lower RMSE in less computational time. We can see, for instance, that RMSE $= 1$ is reached by the RBPSAEM procedure after approximately one second, while the same RMSE is attained by the PSAEM method after roughly two seconds. The PSEM algorithm fails to compete with the remaining methods due to its performance indicators converging to high values at the cost of extra computational time.

In the second experiment, the aforementioned performance indicators are recorded with the number of particles being changed according to $N = 2^i$ for the PSAEM and RBPSAEM algorithms, and respecting $N = 50 + 10 \cdot 2^i$ for the PSEM procedure, where $i = 1, \ldots, 5$. The number of backward trajectories for the PSEM method remains unchanged. The 'ground truth' trajectories are again computed by the RBPSAEM algorithm with $N = 1024$ particles. Fig. 6.3 reports that the proposed method achieves lower values of the RMSE for all N. Specifically, one can notice that the RBPSAEM technique is upper bounded by its PSAEM counterpart, making the effect of Rao-Blackwellization evident. Although the performance of the PSEM method is poor again, we can notice that increasing the number of particles makes the assessment criteria lower. However, to make the performance of the PSEM algorithm comparative to the remaining methods, its number of particles needs to be substantially increased.

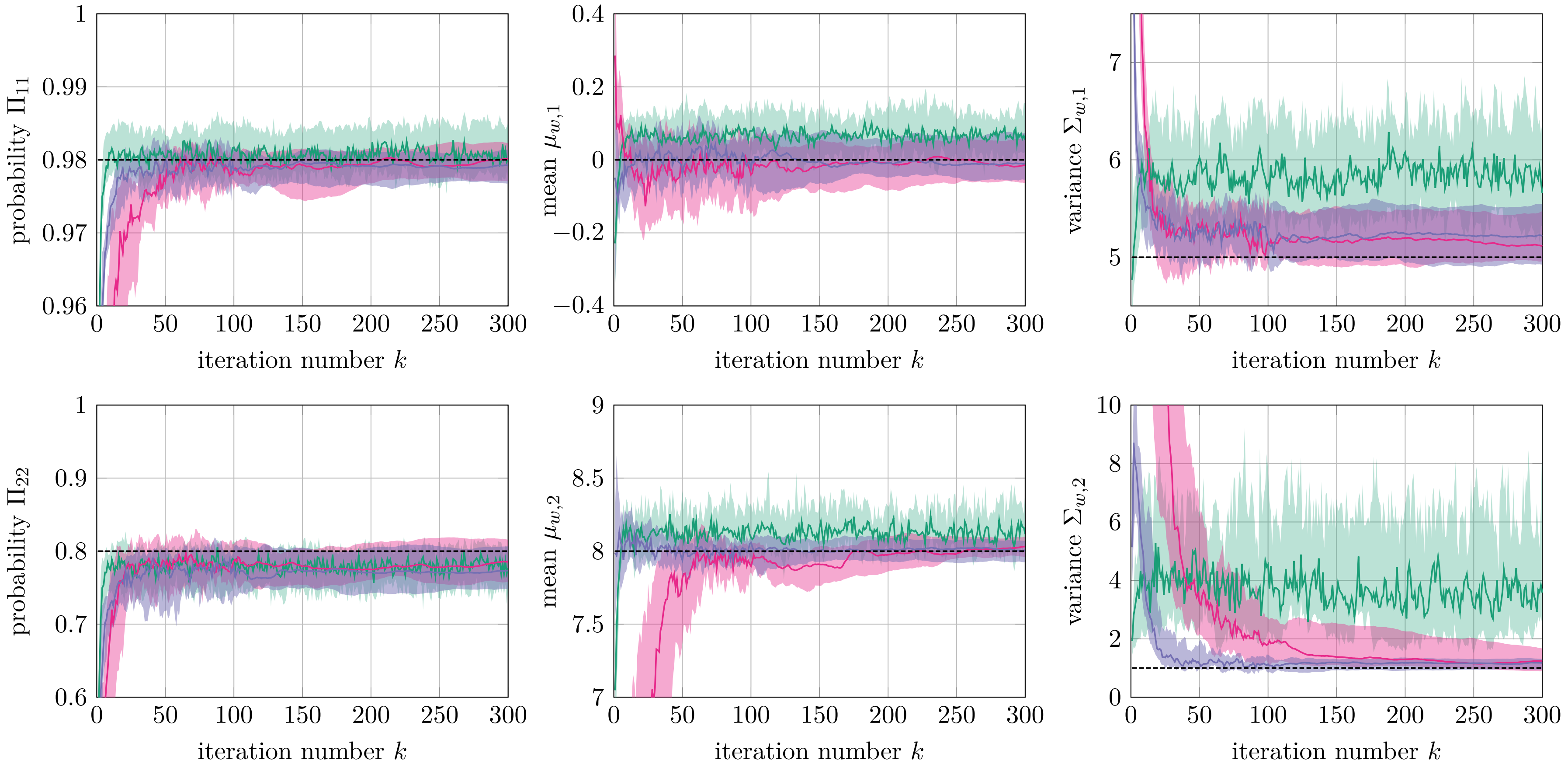

Fig. 6.1: The resulting parameter estimates versus the number of iterations for PSEM (—), PSAEM (—), and RBPSAEM (—). The results are averaged over twenty independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range. The true parameter values are indicated with the dashed line (------).

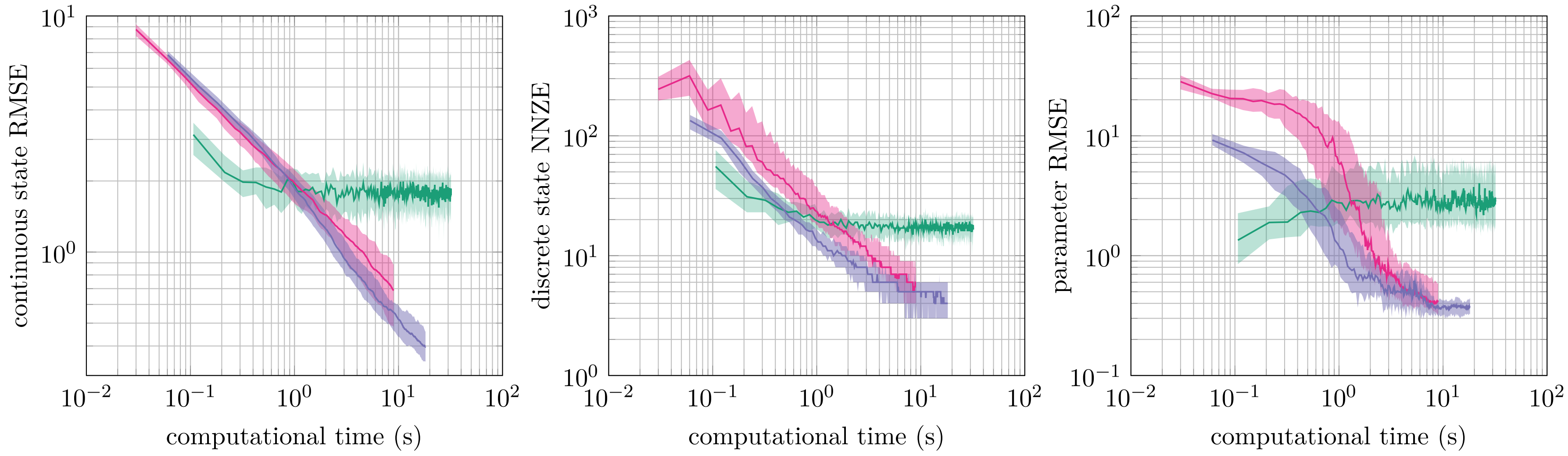

Fig. 6.2: Left: the RMSE between the ground truth continuous state trajectory and the smoothed estimate versus the computational time in seconds. Middle: the number of non-zero elements (NNZE) in the error sequence between ground truth discrete state trajectory and the smoothed estimate versus the computational time in seconds. Right: the RMSE between the ground truth parameter trajectory and the maximum likelihood estimate versus the computational time in seconds. The solid line represents the median, and the shaded area is the interquartile range; both are computed over twenty independent runs. The compared algorithms are PSEM (—), PSAEM (—), and RBPSAEM (—).

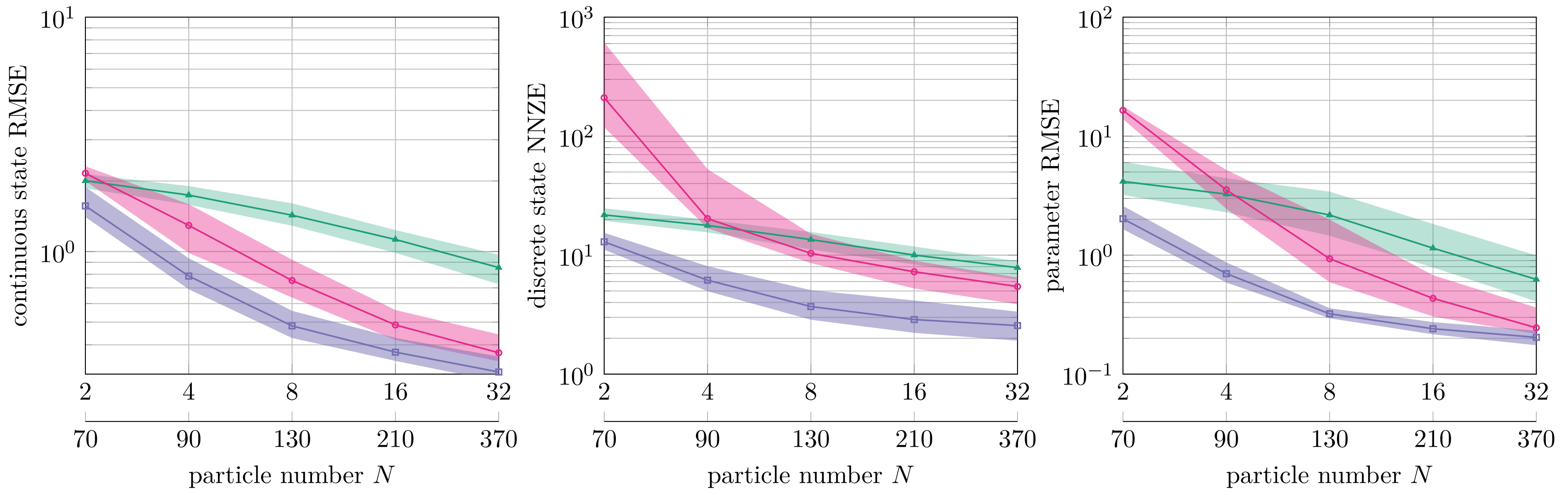

Fig. 6.3: Left: the RMSE between the ground truth continuous state trajectory and the smoothed estimate versus the number of particles. Middle: the number of non-zero elements (NNZE) in the error sequence between ground truth discrete state trajectory and the smoothed estimate versus the number of particles. Right: the RMSE between the ground truth parameter trajectory and the maximum likelihood estimate versus the number of particles. The solid line represents the median, and the shaded area is the interquartile range; both are computed over twenty independent runs. The compared algorithms are PSEM (—▲—), PSAEM (—○—), and RBPSAEM (—□—). The top horizontal axis displays the number of particles for PSAEM and RBPSAEM, and the bottom one shows the number of particles for PSEM.

7 DYNAMIC BAYESIAN KNOWLEDGE TRANSFER BETWEEN A PAIR OF KALMAN FILTERS

The research in this chapter was conducted under
the supervision of Dr. Anthony Quinn.

Transfer learning is a framework that includes—among other topics—the design of knowledge transfer mechanisms between Bayesian filters. Transfer learning strategies in this context typically rely on a complete stochastic dependence structure being specified between the participating learning procedures (filters). This chapter proposes a method that does not require such a restrictive assumption. The solution in this *incomplete modelling* case is based on the fully probabilistic design of an unknown probability distribution which conditions on knowledge in the form of an externally supplied distribution. We are specifically interested in the situation where the external distribution accumulates knowledge dynamically via Kalman filtering. Simulations illustrate that the proposed algorithm outperforms alternative methods for transferring this dynamic knowledge from the external Kalman filter.

7.1 Introduction

7.1.1 Context

Transfer learning [168] has become a key research direction in statistical machine learning [156]. The basic principle of transfer learning is to utilize the experience of an external learning agent (source task) to improve the learning of a primary agent (target task). Transfer learning has recently witnessed substantial attention in a multitude of theoretically and practically oriented scientific fields, such as reinforcement learning [204], deep learning [14], autonomous driving [91], computer vision [169], sensor networks [211], etc. This chapter focuses on a specific transfer learning context referred to as Bayesian transfer learning and its deployment in statistical signal processing. We are specifically interested in developing a procedure for probabilistic knowledge transfer in sensor networks where each knowledge-bearing node constitutes a Bayesian filter acting on its associated state-space model.

The conventional approach to Bayesian transfer learning involves replacing the prior distribution of standard Bayesian learning with a distribution conditioned on the transferred knowledge [206]. The methods based on this principle differ in the way the knowledge-conditioned prior is elicited [17]. An alternative principle is to define the joint posterior distribution of both source and target quantities of interest given source and target data, and then to compute the posterior distribution of the target quantity by marginalization [105]. Hierarchical Bayesian learning provides another formalization of Bayesian transfer learning [222], where the knowledge is transferred by means of a hyper-prior. However, it seems that a widely accepted consensus on Bayesian transfer learning is missing. This chapter seeks to fill this gap.

7.1.2 Contributions

The common aspect of the above approaches is that they assume existence of an explicit model of all unknown quantities of interest, enabling Bayes' rule to accommodate transfer learning, which we call here the *complete modelling* case. In the present chapter, we are concerned with a scenario where there is not enough knowledge to construct such a model explicitly. We refer to this particular situation as the *incomplete modelling* case. The previous work in this respect [67] involved a static Bayesian knowledge transfer for a pair of Kalman filters, where the external knowledge is transferred in the form of a marginal distribution defined at a single time-step. The present chapter extends this work by designing a mechanism for transferring distributions defined over multiple time-steps, thus achieving dynamic and on-line Bayesian knowledge transfer.

7.2 Knowledge Transfer Between a Pair of Bayesian Filters

7.2.1 Problem Formulation

Let us consider a state-space model given by

$$x_i \sim f(x_i|x_{i-1}), \tag{7.1a}$$

$$z_i \sim f(z_i|x_i), \tag{7.1b}$$

where $x_i \in \mathsf{X} \subseteq \mathbb{R}^{n_x}$ and $z_i \in \mathsf{Z} \subseteq \mathbb{R}^{n_z}$ are respectively the state and observation variables defined at the discrete-time instants $i = 1, \ldots, n$. The state-space model (7.1) is fully determined by the state transition (7.1a) and observation (7.1b) probability densities, with all their parameters being known. At the initial time step ($i = 1$), the state variable is distributed according to $x_1 \sim f(x_1)$. The time-evolution of the state-space model (7.1) is characterized by the joint augmented model

$$f(\mathbf{x}_n, \mathbf{z}_n) = f(\mathbf{z}_n|\mathbf{x}_n) f(\mathbf{x}_n) \equiv \prod_{i=1}^{n} f(z_i|x_i) f(x_i|x_{i-1}), \tag{7.2}$$

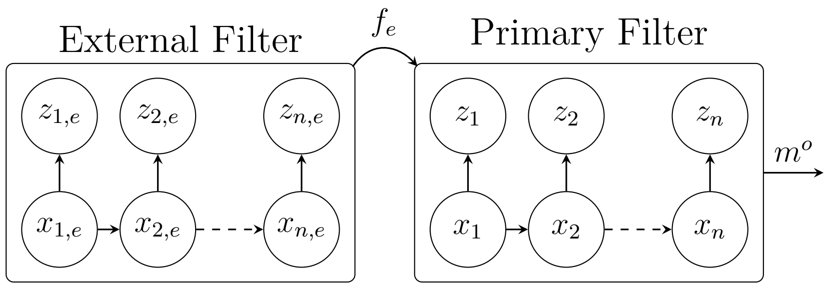

Fig. 7.1: A pair of Bayesian filters acting on their state-space models. The external filter provides the density f_e summarizing knowledge of the quantities (states or observations) gathered over the whole run of the filter. The primary filter makes use of this external knowledge to improve state inference over the corresponding time interval.

where $f(\mathbf{z}_n|\mathbf{x}_n)$ and $f(\mathbf{x}_n)$ define the joint observation model and joint state pre-prior model, respectively. In (7.2), we respect the convention $x_0 \equiv \varnothing$ and use the boldface notation $\mathbf{v}_n \equiv (v_1, \dots, v_n)$ to denote a sequence of variables $v_i \in \mathsf{V}$, for $i = 1, \dots, n$. Moreover, we use the symbols m and f to denote unspecified (variational form) and specified (fixed form) densities, respectively.

We are concerned with the problem of optimally transferring knowledge from an external Bayesian filter (source task) to a primary one (target task). The filters operate on their respective models, processing their local observations, and estimating their local states (Fig. 1). The conditional independence structure between the variables in each model is as specified in (7.2). However, an explicit conditioning mechanism describing dependence between $(\mathbf{x}_n, \mathbf{z}_n)$ and $(\mathbf{x}_{n,e}, \mathbf{z}_{n,e})$ is assumed missing. Note that there is no edge between these node sets in the graphical model in Fig. 1. The common modelling approach based on a joint density of the external and primary variables is therefore unavailable. This incomplete modelling scenario is addressed here as a problem of optimal design of an unknown probability density, processing the external (distributional) knowledge as a constraint. Specifically, we design a *dynamic* Bayesian knowledge transfer method, where knowledge is transferred in the form of a joint probability density, f_e, describing a sequence of external quantities, either $\mathbf{x}_{n,e}$ or $\mathbf{z}_{n,e}$.

7.2.2 Fully Probabilistic Design

A central concern of probabilistic inference is to design (i.e. infer) a stochastic model representing our beliefs about an unknown quantity of interest, $v \in \mathsf{V}$. The construction of such a model is naturally performed by processing our knowledge, k (from physical laws, empirical facts etc.), about the modelled quantity in some way. However, such knowledge is usually insufficient to determine the model completely.

Thus, an explicit density, $m(k|v)$, quantifying our beliefs about k given v is unavailable, and we therefore cannot compute $m(v|k)$ directly by application of Bayes' rule. The model is then sought within a user-specified set of possible models, $m(v|k) \in \mathbf{M}$, that are compatible with k. To complete the decision-making set-up, we specify our preferences about the unknown model, $m(v|k)$, by defining its ideal prescription, $m_{\mathsf{I}}(v)$. Fully probabilistic design (FPD, [106]) is a principled and axiomatically justified [110] approach for optimally choosing $m \in \mathbf{M}$ while taking into account our knowledge and preferences. The optimal model (i.e. design) provides a compromise between the knowledge, k, and the ideal prescription, m_{I}. It is found as the density that is closest to $m_{\mathsf{I}}(v)$ in the minimum Kullback-Leibler divergence (KLD, [123]) sense, while respecting the set-based knowledge constraint, $m \in \mathbf{M}$:

$$m^o(v|k) \equiv \underset{m \in \mathbf{M}}{\operatorname{argmin}}\, \mathcal{D}(m||m_{\mathsf{I}}),$$

where $\mathcal{D}(m||m_{\mathsf{I}})$ is the KLD from m to m_{I}, given as

$$\mathcal{D}(m||m_{\mathsf{I}}) = \mathsf{E}_m \left[\ln \left(\frac{m}{m_{\mathsf{I}}} \right) \right],$$

with E_m denoting the expected value with respect to m. The density $m^o(v|k) \in \mathbf{M}$ is also consistent with k and is referred to as the FPD-optimal design. Typically, $m_{\mathsf{I}} \notin \mathbf{M}$. The case where $m_{\mathsf{I}} \in \mathbf{M}$ implies that the knowledge constraint is inactive, leading to $m^o = m_{\mathsf{I}}$.

In common with the minimum cross-entropy (MXE) principle [195], the FPD framework is a *deterministic* approach for designing an unknown density. A recent extension of FPD leading to a stochastic design of the unknown density has been provided in [177], conferring measures of uncertainty on the designed density. The key feature that distinguishes FPD from the MXE principle is that FPD allows preferences about the unknown model to be processed. The MXE principle follows the same setting as presented above, but the ideal model, m_{I}, is replaced by a prior model, m_{P}.

7.3 Dynamic Bayesian Knowledge Transfer

This section formalizes dynamic Bayesian knowledge transfer as an FPD-based optimal design of an unknown density and shows its application in Bayesian filtering. A principal purpose of Bayesian filtering is to compute the marginal (filtering) density, $f(x_i|\mathbf{z}_i)$, of the joint state posterior density, $f(\mathbf{x}_i|\mathbf{z}_i)$. Under the conditional independence assumptions adopted in (7.2), this density becomes

$$f(x_i|\mathbf{z}_i) = \frac{f(z_i|x_i) f(x_i|\mathbf{z}_{i-1})}{f(z_i|\mathbf{z}_{i-1})}, \tag{7.3a}$$

with

$$f(x_i|\mathbf{z}_{i-1}) = \int f(x_i|x_{i-1})f(x_{i-1}|\mathbf{z}_{i-1})dx_{i-1}, \tag{7.3b}$$

$$f(z_i|\mathbf{z}_{i-1}) = \int f(z_i|x_i)f(x_i|\mathbf{z}_{i-1})dx_i. \tag{7.3c}$$

(7.3b) and (7.3c) are the one-step-ahead state and observation predictors, respectively.

To solve the transfer learning problem (Fig. 1), we use FPD to choose optimally the unknown joint augmented model of the states and observations, $(\mathbf{x}_n, \mathbf{z}_n)$, conditioned on the external density, f_e. This factorizes as follows:

$$m(\mathbf{x}_n, \mathbf{z}_n|f_e) = m(\mathbf{z}_n|\mathbf{x}_n, f_e)m(\mathbf{x}_n|f_e). \tag{7.4}$$

We express our joint preferences about the quantities $(\mathbf{x}_n, \mathbf{z}_n)$ by defining the ideal joint augmented model as (7.2), that is,

$$m_{\mathsf{I}}(\mathbf{x}_n, \mathbf{z}_n) \equiv f(\mathbf{x}_n, \mathbf{z}_n). \tag{7.5}$$

The FPD-optimal choice, m^o, conditioned on the external knowledge, f_e, is found as the unique minimizer of the KLD from the unknown model (7.4) to the ideal model (7.5):

$$m^o(\mathbf{x}_n, \mathbf{z}_n|f_e) \in \underset{m\in\mathbf{M}}{\operatorname{argmin}} \mathcal{D}(m||m_{\mathsf{I}}). \tag{7.6}$$

The external knowledge—encoded as f_e—is transferred by constraining the set $\mathbf{M}$ in a specific way, as we now show.

7.3.1 Transferring an External Joint Observation Predictor

We choose to transfer the external joint observation predictor, $f_e(\mathbf{z}_{n,e})$. To do so, we must specify exactly how the f_e condition constrains the functional form of m in (7.4). First, we consider the f_e-conditioned joint observation model, which factorizes as

$$m(\mathbf{z}_n|\mathbf{x}_n, f_e) = \prod_{i=1}^{n} m(z_i|\mathbf{x}_i, \mathbf{z}_{i-1}, f_e),$$

and we impose the following conditional independence assumption:

$$m(z_i|\mathbf{x}_i, \mathbf{z}_{i-1}, f_e) \equiv f_e(z_{i,e}|\mathbf{z}_{i-1,e}) \mid_{z_{i,e}=z_i}. \tag{7.7}$$

Here, we have constrained the f_e-conditioned model for the *primary* observations to be the externally supplied one-step-ahead observation predictor. Next, we consider the f_e-conditioned joint state prior model in (7.4), which factorizes as

$$m(\mathbf{x}_n|f_e) = \prod_{i=1}^{n} m(x_i|\mathbf{x}_{i-1}, f_e),$$

and we impose the conventional Markov property:

$$m(x_i|\mathbf{x}_{i-1}, f_e) \equiv m(x_i|x_{i-1}, f_e).$$

Under these specified knowledge constraints, the unknown f_e-conditioned joint augmented model (7.4) becomes

$$m(\mathbf{x}_n, \mathbf{z}_n|f_e) \equiv f_e(\mathbf{z}_n)m(\mathbf{x}_n|f_e). \tag{7.8}$$

With $f_e(\mathbf{z}_n)$ fixed via the external filter, the f_e-conditioned joint state prior factor, $m(\mathbf{x}_n|f_e)$, is the only variational quantity which we can now choose via FPD for the purpose of optimal knowledge transfer. In summary, the f_e-constrained set of candidate models is

$$\begin{aligned}\mathbf{M} \equiv \Big\{&\text{models (7.8) with } f_e(\mathbf{z}_n) \text{ fixed} \\ &\qquad\text{and } m(\mathbf{x}_n|f_e) \text{ variational}\Big\}.\end{aligned} \tag{7.9}$$

The following proposition establishes the fact that $f_e(\mathbf{z}_{n,e})$ is *sequentially* processed into the FPD-optimal joint state prior of the primary filter. This will be key in securing a *recursive*, causal, dynamic Bayesian transfer learning algorithm between a pair of Kalman filters, as we will see in Section 7.3.2.

Proposition 7.1. *The unknown joint augmented model satisfies the knowledge constraint, $m \in \mathbf{M}$(7.9), imposed by transfer of the external joint observation predictor, $f_e(\mathbf{z}_{n,e})$. The ideal model is defined in (7.5), and $D(m||m_I) < \infty, \forall m \in \mathbf{M}$. Then, an FPD-optimal design of m—i.e. a solution of (7.6)—is*

$$m^o(\mathbf{x}_n, \mathbf{z}_n|f_e) = f_e(\mathbf{z}_n)m^o(\mathbf{x}_n|f_e), \tag{7.10}$$

with

$$\begin{aligned} m^o(\mathbf{x}_n|f_e) &= \prod_{i=1}^{n} m^o(x_i|x_{i-1}, f_e) &\text{(7.11a)}\\ &\propto f(\mathbf{x}_n)\prod_{i=1}^{n} \exp\{-\mathcal{D}(f_e||f)\}\gamma(x_i). &\text{(7.11b)}\end{aligned}$$

Here,

$$m^o(x_i|x_{i-1}, f_e) \equiv \frac{f(x_i|x_{i-1})\exp\{-\mathcal{D}(f_e||f)\}\gamma(x_i)}{\gamma(x_{i-1})}, \tag{7.12}$$

$$\mathcal{D}(f_e||f) \equiv \int f_e(z_i|\mathbf{z}_{i-1,e}) \ln \frac{f_e(z_i|\mathbf{z}_{i-1,e})}{f(z_i|x_i)} dz_i, \tag{7.13}$$

$$\begin{aligned}\gamma(x_{i-1}) \equiv \int & f(x_i|x_{i-1}) \\ &\times \exp\{-\mathcal{D}(f_e||f)\}\gamma(x_i)dx_i.\end{aligned} \tag{7.14}$$

The normalization functions, $\gamma(x_i)$, need to be computed via a backward sweep through the recursions (7.14), for $i = n, \ldots, 1$, initialized with $\gamma(x_n) \equiv 1$.

Proof. See Section 7.6.2. □

Proposition 7.1 shows that FPD-optimal Bayesian transfer learning is achieved by updating the pre-prior, $f(\mathbf{x}_n)$, to the prior, $m^o(\mathbf{x}_n|f_e)$. This is achieved via modulation with a product term (7.11b) containing the external knowledge over the full time horizon. Correspondingly, at each time instant, i, the update of the state transition model to the FPD-optimal state transition model is achieved via the modulation (7.12). This optimal joint prior, $m^o(\mathbf{x}_n|f_e)$, can therefore be sequentially processed by the primary filter, via (7.3), since it enjoys the recursive factorization form in (7.11b,7.12). In particular, (7.12) replaces (7.1a) in the standard Bayesian filtering setting (7.3), optimally transferring the external joint observation predictor, $f_e(\mathbf{z}_{n,e})$, in a sequential manner.

7.3.2 Transfer of an External Kalman Filter Observation Predictor

Here, we describe a specific application of Proposition 7.1 to the case of transferring the external Kalman filter joint observation predictor. The Kalman filter is one of the very restricted instances which ensure that the Bayesian filtering equations (7.3) are tractable. Specifically, (7.1) is specialized to the linear-Gaussian case:

$$f(x_i|x_{i-1}) \equiv \mathcal{N}_{x_i}(Ax_{i-1}, Q), \tag{7.15a}$$

$$f(z_i|x_i) \equiv \mathcal{N}_{z_i}(Cx_i, R), \tag{7.15b}$$

and the marginal state pre-prior density has to be chosen as the Gaussian density $f(x_1) \equiv \mathcal{N}_{x_1}(\mu_{1|0}, \Sigma_{1|0})$. Here, $\mathcal{N}_v(\mu, \Sigma)$ denotes the Gaussian density of a (vector) random variable, v, with the mean, μ, and covariance matrix, Σ; and A and C are matrices of appropriate dimensions. Under these assumptions, the densities (7.3) preserve the Gaussian form across all iterations, $i = 1, \ldots, n$,

$$f(x_i|\mathbf{z}_i) = \mathcal{N}_{x_i}(\mu_{i|i}, \Sigma_{i|i}), \tag{7.16a}$$

$$f(x_i|\mathbf{z}_{i-1}) = \mathcal{N}_{x_i}(\mu_{i|i-1}, \Sigma_{i|i-1}), \tag{7.16b}$$

$$f(z_i|\mathbf{z}_{i-1}) = \mathcal{N}_{z_i}(z_{i|i-1}, R_{i|i-1}), \tag{7.16c}$$

with the shaping parameters being computed explicitly and recursively as follows:

$$\mu_{i|i} = \mu_{i|i-1} + K(z_i - z_{i|i-1}), \tag{7.17a}$$

$$\Sigma_{i|i} = \Sigma_{i|i-1} - KR_{i|i-1}K^\top, \tag{7.17b}$$

$$\mu_{i|i-1} = A\mu_{i-1|i-1}, \tag{7.18a}$$

$$\Sigma_{i|i-1} = A\Sigma_{i-1|i-1}A^\top + Q, \tag{7.18b}$$

$$z_{i|i-1} = C\mu_{i|i-1}, \tag{7.19a}$$

$$R_{i|i-1} = C\Sigma_{i|i-1}C^\top + R. \tag{7.19b}$$

Here, $K \equiv \Sigma_{i|i-1}C^\top R_{i|i-1}^{-1}$ and $^\top$ denotes matrix transposition. These formulae follow directly from application of the conditioning and marginalization rules for Gaussian densities containing affine transformations [188].

To support our next proposition, we present the following lemma, which specifies the computation of the normalization function (7.14) in this Kalman context.

Lemma 7.1. *Let the state-space model be defined by (7.15), and the external one-step-ahead observation predictor by (7.16c), i.e.* $f_e(z_{i,e}|\mathbf{z}_{i-1,e}) \equiv \mathcal{N}_{z_{i,e}}(z_{i|i-1,e}, R_{i|i-1,e})$, $i = n, \ldots, 2$*. Then, (7.14) preserves the form*

$$\gamma(x_{i-1}) \propto \exp\Big[-\tfrac{1}{2}(x_{i-1}^\top S_{i-1|i}x_{i-1} - 2x_{i-1}^\top r_{i-1|i})\Big], \tag{7.20}$$

and its explicit computation reduces to the recursion

$$r_{i-1|i} = A^\top(I_{n_x} - L)r_{i|i}, \tag{7.21a}$$

$$S_{i-1|i} = A^\top(I_{n_x} - L)S_{i|i}A, \tag{7.21b}$$

where, for $i = n-1, \ldots, 2$,

$$r_{i|i} = r_{i|i+1} + C^\top R^{-1} z_{i|i-1,e}, \tag{7.22a}$$

$$S_{i|i} = S_{i|i+1} + C^\top R^{-1}C, \tag{7.22b}$$

and, for $i = n$,

$$r_{n|n} = C^\top R^{-1} z_{n|n-1,e}, \tag{7.23a}$$

$$S_{n|n} = C^\top R^{-1}C. \tag{7.23b}$$

Here, $L \equiv S_{i|i}Q^{\frac{1}{2}}(Q^{\frac{\top}{2}}S_{i|i}Q^{\frac{1}{2}} + I_{n_x})^{-1}Q^{\frac{\top}{2}}$, I_{n_x} *is the* $n_x \times n_x$ *identity matrix, and* $Q^{\frac{1}{2}}$ *is the Cholesky factor of* Q.

Proof. See Section 7.6.3. □

Lemma 7.1 demonstrates the connection between the computation of (7.14) and the backward information filter [148] which takes the mean value of the external predictor $z_{i|i-1,e}$ as the observation input. Based on this result, the next proposition furnishes the explicit recursive computation of the FPD-optimal state transition model (7.12).

Proposition 7.2. *Under the conditions of Lemma 7.1, the FPD-optimal state transition model (7.12) is given by*

$$m^o(x_i|x_{i-1}, f_e) = \mathcal{N}_{x_i}(\mu_i^o, \Sigma_i^o), \tag{7.24}$$

Algorithm 21 FPD-optimal processing for dynamic transfer between Kalman filters

```
A. Backward sweep:
    1. For $i = n$,
        $*$ use $z_{n|n-1,e}$ in (7.23) to compute $(r_{n|n}, S_{n|n})$.
        $*$ use $(r_{n|n}, S_{n|n})$ in (7.21) to compute $(r_{n-1|n}, S_{n-1|n})$.
    2. For $i = n-1, \dots, 2$;
        $*$ use $z_{i|i-1,e}$ and $(r_{i|i+1}, S_{i|i+1})$ in (7.22) to compute $(r_{i|i}, S_{i|i})$.
        $*$ use $(r_{i|i}, S_{i|i})$ in (7.21) to compute $(r_{i-1|i}, S_{i-1|i})$.
B. Forward sweep:
    1. For $i = 1$, set $\mu_{1|0}, \Sigma_{1|0}$ and use it in (7.17) to compute $(\mu_{1|1}, \Sigma_{1|1})$.
    2. For $i = 2, \dots, n$;
        $*$ use $(\mu_{i-1|i-1}, \Sigma_{i-1|i-1})$ in (7.27) to compute $(\mu_{i|i-1}, \Sigma_{i|i-1})$.
        $*$ use $(\mu_{i|i-1}, \Sigma_{i|i-1})$ in (7.17) to compute $(\mu_{i|i}, \Sigma_{i|i})$.
```

with the shaping parameters calculated according to

$$\mu_i^o = (I_{n_x} - \Sigma_i^o S_{i|i}) A x_{i-1} + \Sigma_i^o r_{i|i}, \tag{7.25}$$

$$\Sigma_i^o = Q^{\frac{1}{2}} (Q^{\frac{\top}{2}} S_{i|i} Q^{\frac{1}{2}} + I_{n_x})^{-1} Q^{\frac{\top}{2}}. \tag{7.26}$$

Here, $r_{i|i}$ and $S_{i|i}$ are given by (7.22a) and (7.22b), respectively.

Proof. See Section 7.6.4. □

Proposition 7.2 specifies the optimal adaptation of the primary (i.e. target) Kalman filter flow, in order to process transferred knowledge in the form of the external joint observation predictor. If we apply (7.24) in (7.3b), then the one-step-ahead state predictor preserves the Gaussian form of (7.16b). However, the difference is that, now, the shaping parameters (7.18a,7.18b) are replaced with

$$\mu_{i|i-1} = (I_{n_x} - \Sigma_i^o S_{i|i}) A \mu_{i-1|i-1} + \Sigma_i^o r_{i|i}, \tag{7.27a}$$

$$\Sigma_{i|i-1} = (I_{n_x} - \Sigma_i^o S_{i|i}) A \Sigma_{i-1|i-1} A^\top (I_{n_x} - \Sigma_i^o S_{i|i})^\top + \Sigma_i^o, \tag{7.27b}$$

respectively. The resulting filter with FPD-optimal dynamic transfer is summarized in Algorithm 21.

7.4 Experiments

The purpose of this section is to compare the proposed method against alternative approaches. We evaluate the performance of the primary filter when keeping its observation variance R fixed but changing the observation variance of the external filter R_e, which quantifies the confidence of the external knowledge. To assess the resulting state estimates, we use the mean norm squared-error, MNSE $= \frac{1}{n} \sum_{i=1}^n ||x_i - \mu_{i|i}||^2$, with $|| \cdot ||$ denoting the Euclidean norm. We are concerned with a simple

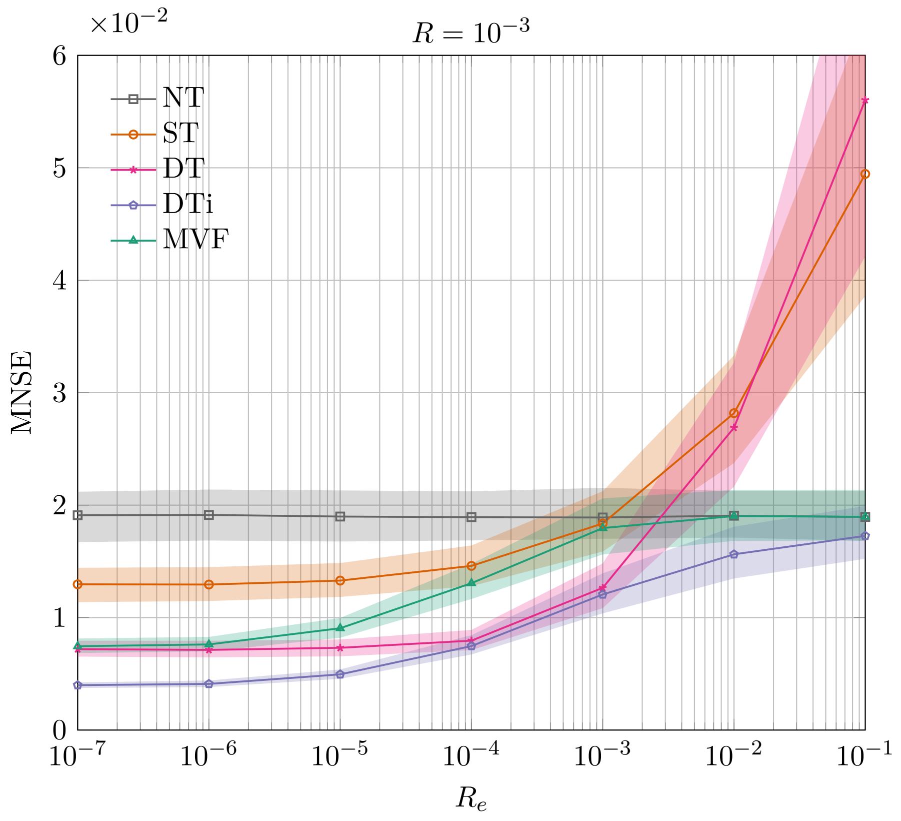

Fig. 7.2: The mean norm squared-error (MNSE) of the primary filter versus the observation variance R_e of the external Kalman filter. The results are averaged over 1000 independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range. The procedures that are compared are (i) the Kalman filter with *No Transfer* (*NT*), (ii) *Static Bayesian knowledge Transfer* (*ST*) [67], (iii) *Dynamic Bayesian knowledge Transfer* (*DT*) given by Algorithm 21, (iv) an *informally adapted version of DT* (*DTi*) which we mention in Section 7.5; and (v) *Measurement Vector Fusion* (*MVF*) [220].

position-velocity state-space model [61] specified by

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad Q = 10^{-5} I_2, \quad R = 10^{-3}.$$

The number of time steps is $n = 50$. The results of the compared algorithms are illustrated in Fig. 7.2.

The MNSE of the NT filter defines a reference level against which the remaining filters are compared. This level is obviously constant as the external observation variance does not enter the standard Kalman filter via (7.19b). The error in the remaining filters varies according to the ratio of the primary and external observation variances. We can observe that the proposed DT filter achieves positive knowledge transfer for $R_e < 3 \times 10^{-3}$, which is evidenced by the fact that the error of the DT filter is lower than that of the NT filter in this range. Moreover, the DT filter outperforms the MVF filter in the same interval, and it also outperforms the ST filter for $R_e < 2 \times 10^{-2}$. The important observation is that the ST and MVF filters meet the performance of the NT filter close to the intersection where $R_e = R$, but the proposed DT filter passes this point with a markedly lower error and meets the NT filter later (i.e. for higher external observation variance). This increased robustness of the DT filter, which now benefits even from external observations that are of a lower quality than the primary ones, is achieved because of its ability to accumulate the external knowledge over multiple time steps via the *dynamic* transfer which is the focus of this chapter. The ST and MVF filters do not have this property, as is evidenced by the fact that their error is, respectively, worse and very similar to the NT filter, above $R_e = R$. However, accumulating external knowledge of increasingly poor quality does lead to a more quickly decreasing performance of the DT filter for $R_e > 2 \times 10^{-2}$.

7.5 Discussion

In common with the ST filter of [67], the DT filter is also insensitive to the transfer of the covariance of the external observation predictor $R_{i|i-1,e}$. The loss of this moment information occurs when evaluating the KLD (7.13) and can informally be resolved by replacing R with $R_{i|i-1,e}$ in (7.22) and (7.23). This simple substitution defines the DTi filter introduced in Section 7.4. The experiments demonstrate that the DTi filter surpasses all the other filters across the full range of values of R_e. This outcome is remarkable as it proves that improved estimation accuracy is achieved by implementing this FPD-optimal Bayesian transfer learning, obviating the need—usually prohibitive—to specify an explicit stochastic dependence structure between the external and primary quantities. It is also important to note that the DT filter offers the same advantage, albeit over a slightly shorter range of values of R_e. However, it seems that the fragile dependence assumptions inherited by the MVF filter undermine its performance. The fact that we do not require these dependence assumptions is a markedly simplifying feature of this FPD-based transfer learning framework, and should ensure its consistency in a wide range of applications. In Section 7.7, we provide evidence that the proposed method also offers more robustness against higher values of the state covariance Q.

A universal Bayesian transfer learning framework has been elusive so far. However, the practical evidence of this chapter—along with the axiomatically driven optimality it provides—supports the assertion that FPD-optimal Bayesian transfer learning can become such a universal framework.

7.6 Proofs

7.6.1 Preliminaries

To make subsequent derivations more compact, we first introduce the following two auxiliary lemmata.

Lemma 7.2. *Let $f(x|\bar{x}) = \mathcal{N}_x(A\bar{x}, Q)$ and $g(x) = \exp\{-\frac{1}{2}(x^\top Sx - 2x^\top r)\}$, where A, Q, S and r, $\bar{x}$ are constant real-valued matrices and vectors of appropriate dimensions, respectively, with Q and S being symmetric and positive definite, then*

$$\begin{aligned}\mathcal{N}_x(A\bar{x}, Q)\exp\Big\{-\tfrac{1}{2}(x^\top Sx - 2x^\top r)\Big\} &= \\ &\mathcal{N}_x(\mu, \Sigma)\exp\Big\{-\tfrac{1}{2}(\bar{x}^\top \bar{S}\bar{x} - 2\bar{x}^\top \bar{r} - a)\Big\}.\end{aligned} \tag{7.28}$$

Here, we introduce the quantities

$$\mu = (I - \Sigma S)A\bar{x} + \Sigma r, \tag{7.29a}$$

$$\Sigma = Q^{\frac{1}{2}}T^{-1}Q^{\frac{\top}{2}}, \tag{7.29b}$$

$$\bar{r} = A^\top(I - SQ^{\frac{1}{2}}T^{-1}Q^{\frac{\top}{2}})r, \tag{7.29c}$$

$$\bar{S} = A^\top(I - SQ^{\frac{1}{2}}T^{-1}Q^{\frac{\top}{2}})SA, \tag{7.29d}$$

$$a = ||r||^2_{Q^{\frac{1}{2}}T^{-1}Q^{\frac{\top}{2}}} - \log|QS + I|, \tag{7.29e}$$

$$T = Q^{\frac{\top}{2}}SQ^{\frac{1}{2}} + I, \tag{7.29f}$$

with $||v||^2_W = v^\top Wv$ and $|W|$ denoting the square of the weighted norm and matrix determinant, respectively.

Proof. We begin with rearranging the product on the l.h.s. of (7.28) according to

$$f(x|\bar{x})g(x) = (2\pi)^{-\frac{n_x}{2}}\exp\Big\{-\tfrac{1}{2}(h(x,\bar{x}) + \log|Q|)\Big\}, \tag{7.30}$$

where

$$h(x,\bar{x}) = ||x - A\bar{x}||^2_{Q^{-1}} + ||x||^2_S - 2x^\top r. \tag{7.31}$$

The proof then continues by separating the x-dependent terms and completing the square in (7.31), which leads to

$$h(x,\bar{x}) = ||x - \mu||^2_{\Sigma^{-1}} - ||\mu||^2_{\Sigma^{-1}} + ||A\bar{x}||^2_{Q^{-1}}, \tag{7.32}$$

using $\mu \equiv \Sigma(r + Q^{-1}A\bar{x})$ and $\Sigma^{-1} \equiv S + Q^{-1}$. The next step concerns only the last two terms in (7.32) where we gather the first and second-order quantities related to $\bar{x}$ as

$$h(x,\bar{x}) = ||x - \mu||^2_{\Sigma^{-1}} + ||A\bar{x}||^2_{Q^{-1}-Q^{-1}\Sigma Q^{-1}} - 2\bar{x}^\top A^\top Q^{-1}\Sigma r - ||r||^2_\Sigma. \tag{7.33}$$

Substituting (7.33) back in (7.30) and extending the exponent by $\log|\Sigma| - \log|\Sigma|$ allows us to write

$$\begin{aligned}(2\pi)^{-\frac{n_x}{2}} \exp\Big\{ -\tfrac{1}{2}(h(x,\bar{x}) + \log|Q| + \log|\Sigma| - \log|\Sigma|)\Big\} \\ = \mathcal{N}_x(\mu,\Sigma)\exp\Big\{ -\tfrac{1}{2}(\bar{x}^\top \bar{S}\bar{x} - 2\bar{x}^\top \bar{r} - a)\Big\}, \qquad (7.34)\end{aligned}$$

where

$$\bar{r} = A^\top Q^{-1}(S+Q^{-1})^{-1} r, \qquad (7.35a)$$
$$\bar{S} = A^\top (Q^{-1} - Q^{-1}(S+Q^{-1})^{-1}Q^{-1})A, \qquad (7.35b)$$
$$a = ||r||^2_{(S+Q^{-1})^{-1}} - \log|QS+I|, \qquad (7.35c)$$

utilizing $S + Q^{-1} = Q^{-1}(QS+I)$ to obtain (7.35c).

However, the form of the shaping parameters (7.35a) and (7.35b) is inconvenient for practical computations, mainly due to the presence of the inverse matrix Q^{-1}. This issue can simply be resolved by taking advantage of the identities

$$Q^{-1}(S+Q^{-1})^{-1} = (I+SQ)^{-1} = I - S(S+Q^{-1})^{-1}, \qquad (7.36a)$$
$$Q^{-1} - Q^{-1}(S+Q^{-1})^{-1}Q^{-1} = (S^{-1}+Q)^{-1} = S - S(S+Q^{-1})^{-1}S, \qquad (7.36b)$$

which both result from the matrix inversion lemma. These formulae leave the inverse matrix Q^{-1} only in the inner term $(S+Q^{-1})^{-1}$. By taking advantage of the factorization $Q = Q^{\frac{\top}{2}}Q^{\frac{1}{2}}$, with $Q^{\frac{1}{2}}$ being the Cholesky factor of Q, we obtain

$$(S+Q^{-1})^{-1} = Q^{\frac{1}{2}}(Q^{\frac{\top}{2}}SQ^{\frac{1}{2}} + I)^{-1}Q^{\frac{\top}{2}} = Q^{\frac{1}{2}}T^{-1}Q^{\frac{\top}{2}}, \qquad (7.37)$$

where we introduce (7.29f). Finally, substituting (7.37) in (7.36), and then plugging (7.36a) and (7.36b) in (7.35a) and (7.35b), respectively, leads to the formulae for computing the shaping parameters (7.29c) and (7.29d). □

Lemma 7.3. *Let $f_e(z) = \mathcal{N}_z(\bar{z},\bar{R})$ and $f(z|x) = \mathcal{N}_z(Cx,R)$, where $\bar{R}$, C, R and $\bar{z}$, x are constant real-valued matrices and vectors of appropriate dimensions, respectively, with $\bar{R}$ and R being symmetric and positive definite, then*

$$\exp\Big\{ -\mathcal{D}(f_e||f)\Big\} = \exp\Big\{ -\tfrac{1}{2}(x^\top C^\top R^{-1}Cx - 2x^\top C^\top R^{-1}\bar{z} - b)\Big\}, \qquad (7.38)$$

where

$$b = -2\mathsf{E}_{f_e}[\log f_e(z)] - n_z\log(2\pi) - \log|R| - \mathsf{E}_{f_e}[||z||^2_{R^{-1}}].$$

Proof. The proof follows straightforwardly from

$$\begin{aligned}\mathcal{D}(f_e||f) &= \mathsf{E}_{f_e}[\log f_e(z)] - \mathsf{E}_{f_e}[\log f(z|x)] \\ &= \mathsf{E}_{f_e}[\log f_e(z)] + \tfrac{1}{2}(n_z\log(2\pi) + \log|R| + \mathsf{E}_{f_e}[||z||^2_{R^{-1}}]) \\ &\qquad + \tfrac{1}{2}(||Cx||^2_{R^{-1}} - 2x^\top C^\top R^{-1}\mathsf{E}_{f_e}[z]). \qquad (7.39)\end{aligned}$$

Applying the basic identity $\int z\mathcal{N}_z(\bar{z}, \bar{R})dz = \bar{z}$ to the third term in (7.39) and plugging the result in $\exp\{-\mathcal{D}(f_e||f)\}$ leads to (7.38) and concludes the proof. Note there is no need to calculate the expected values in the first two terms in (7.39) as they do not depend on x. □

7.6.2 Proof of Proposition 1

To simplify the assertion, let us introduce the shorthand notation $m_n \equiv m(\mathbf{x}_n, \mathbf{z}_n)$, $\bar{m} \equiv m(x_n|x_{n-1}, f_e)$, and $f_n \equiv f(\mathbf{x}_n, \mathbf{z}_n)$. The proof begins in a way similar to the original formulation of the FPD for designing control strategies [106, 109]. We begin with rearranging (7.6) according to

$$\begin{aligned}
\min_{m_n\in\mathbf{M}} \mathcal{D}(m_n||f_n) &= \min_{m_n\in\mathbf{M}} \int f_e(z_n)m(x_n|x_{n-1}, f_e)m_{n-1} \\
&\qquad \times \ln \frac{f_e(z_n)m(x_n|x_{n-1}, f_e)m_{n-1}}{f(z_n|x_n)f(x_n|x_{n-1})f_{n-1}} d\mathbf{x}_n d\mathbf{z}_n \\
&= \min_{m_n\in\mathbf{M}} \Bigg(\int f_e(z_n)m(x_n|x_{n-1}, f_e)m_{n-1} \\
&\qquad \times \ln \frac{m_{n-1}}{f_{n-1}} dx_n dz_n d\mathbf{x}_{n-1} d\mathbf{z}_{n-1} \\
&+ \int f_e(z_n)m(x_n|x_{n-1}, f_e)m_{n-1} \\
&\qquad \times \ln \frac{f_e(z_n)m(x_n|x_{n-1}, f_e)}{f(z_n|x_n)f(x_n|x_{n-1})} dx_n dz_n d\mathbf{x}_{n-1} d\mathbf{z}_{n-1} \Bigg) \\
&= \min_{m_n\in\mathbf{M}} \Bigg(\mathcal{D}(m_{n-1}||f_{n-1}) + \int f_e(z_n)m(x_n|x_{n-1}, f_e)m_{n-1} \\
&\qquad \times \ln \frac{f_e(z_n)m(x_n|x_{n-1}, f_e)}{f(z_n|x_n)f(x_n|x_{n-1})} dx_n dz_n d\mathbf{x}_{n-1} d\mathbf{z}_{n-1} \Bigg), \quad (7.40)
\end{aligned}$$

where we apply the definition of the KL divergence, the independence assumptions given in the models (7.5) and (7.8), and the normalization property of density functions. To describe the minimization of $\mathcal{D}(m_n||f_n)$, let us denote the second term in the last line of (7.40) as

$$C(m_{n-1}, \bar{m}) \equiv \int m_{n-1}B(x_{n-1}, \bar{m})d\mathbf{x}_{n-1}d\mathbf{z}_{n-1}, \quad (7.41)$$

where, to simplify the subsequent rearrangements, we introduce an intermediate quantity

$$B(x_{n-1}, \bar{m}) \equiv \int f_e(z_n)m(x_n|x_{n-1}, f_e) \ln \frac{f_e(z_n)m(x_n|x_{n-1}, f_e)}{\gamma(x_n)f(z_n|x_n)f(x_n|x_{n-1})} dx_n dz_n, \quad (7.42)$$

in which the definition $\gamma(x_n) \equiv 1$ is applied. This inclusion of $\gamma(x_n)$ is required to take care of the normalization functions, as will be obvious later on in the proof.

Since f_e is considered fixed, m_{n-1} and $\bar{m}$ are the only quantities to be optimized in (7.41). After attaining the minimum with respect to $\bar{m}$, we can continue by minimizing the remaining terms with respect to m_{n-1}. These considerations allow us to formulate a *recursive scheme* for the optimization task (7.40) given by

$$\min_{m_n \in \mathbf{M}} \mathcal{D}(m_n||f_n) = \min_{m_{n-1} \in \mathbf{M}} \Big(\mathcal{D}(m_{n-1}||f_{n-1}) + \min_{\bar{m} \in \mathbf{M}} C(m_{n-1}, \bar{m}) \Big). \tag{7.43}$$

To find the minimizer of (7.41) with respect to $\bar{m}$, let us rewrite (7.42) according to

$$B(x_{n-1}, \bar{m}) = \int m(x_n|x_{n-1}, f_e) \left[\ln \frac{m(x_n|x_{n-1}, f_e)}{\gamma(x_n) f(x_n|x_{n-1})} + \mathcal{D}(f_e||f) \right] dx_n, \tag{7.44}$$

where

$$\mathcal{D}(f_e||f) = \int f_e(z_n) \ln \frac{f_e(z_n)}{f(z_n|x_n)} dz_n \tag{7.45}$$

is the KLD from $f_e(z_n)$ to $f(z_n|x_n)$ which is conditioned on x_n. Now, we are in the position which allows us to find the minimizer. Introducing the normalization function

$$\gamma(x_{n-1}) = \int f(x_n|x_{n-1}) \exp\{-\mathcal{D}(f_e||f)\} \gamma(x_n) dx_n$$

into (7.44) provides us with

$$B(x_{n-1}, \bar{m}) = \int m(x_n|x_{n-1}, f_e) \ln \frac{m(x_n|x_{n-1}, f_e)}{\frac{f(x_n|x_{n-1}) \exp\{-\mathcal{D}(f_e||f)\} \gamma(x_n)}{\gamma(x_{n-1})}} dx_n - \ln \gamma(x_{n-1}), \tag{7.46}$$

from which, by applying the basic property of the KL divergence, $D(m||m) = 0$, we obtain

$$m^o(x_n|x_{n-1}, f_e) = \frac{f(x_n|x_{n-1}) \exp\{-\mathcal{D}(f_e||f)\} \gamma(x_n)}{\gamma(x_{n-1})}. \tag{7.47}$$

Finally, substituting this partial minimizer (7.47) into (7.46), applying the attained optimum $B(x_{n-1}, \bar{m}^o) = -\ln \gamma(x_{n-1})$ in (7.41), and plugging the result in (7.43), yields

$$\min_{m_{n-1} \in \mathbf{M}} \Big(D(m_{n-1}||f_{n-1}) - \int m_{n-1} \ln \gamma(x_{n-1}) d\mathbf{x}_{n-1} d\mathbf{z}_{n-1} \Big),$$

where we can see that $\gamma(x_{n-1})$ enters $D(m_{n-1}||f_{n-1})$ in the same way as $\gamma(x_n)$ enters $D(m_n||f_n)$, which allows the procedure to be repeated recursively. Performing the above described minimization for the remaining terms of the recursion leads to the full result $m^o(\mathbf{x}_n, \mathbf{z}_n|f_e)$ and concludes the proof. ■

7.6.3 Proof of Lemma 1

The proof of Lemma 7.1 begins by writing (7.14) as backward time and data updating equations

$$\gamma(x_{i-1}) = \int f(x_i|x_{i-1})\beta(x_i)dx_i, \tag{7.48}$$

$$\beta(x_i) \equiv \exp\{-\mathcal{D}(f_e||f)\}\gamma(x_i). \tag{7.49}$$

The rest of the proof then proceeds by induction. Let us first be concerned with how to derive the initial shaping parameters of (7.49). Thus, for $i = n$, we take $\gamma(x_n) \equiv 1$ from Proposition 7.1 and adopt Lemma 7.3 to express $\exp\{-\mathcal{D}(f_e||f)\}$, which leads to

$$\begin{aligned}\beta(x_n) &= \exp\Big\{-\tfrac{1}{2}(x_n^\top C^\top R^{-1}Cx_n - 2x_n^\top C^\top R^{-1}z_{n|n-1,e} - b_n)\Big\}\\ &\equiv \exp\Big\{-\tfrac{1}{2}(x_n^\top S_{n|n}x_n - 2x_n^\top r_{n|n} - c_{n|n})\Big\},\end{aligned} \tag{7.50a}$$

where $r_{n|n}$ and $S_{n|n}$ are given by (7.23), and $c_{n|n} = b_n$. We continue by seeking formulae that reduce the computation of (7.49) into a closed-form algebraic recursion. For $i = n-1, \ldots, 1$, let us assume

$$\gamma(x_i) = \exp\Big\{-\tfrac{1}{2}(x_i^\top S_{i|i+1}x_i - 2x_i^\top r_{i|i+1} - c_{i|i+1})\Big\},$$

then, after substituting this into (7.49) and applying Lemma 7.3 in order to express $\exp\{-\mathcal{D}(f_e||f)\}$, we obtain

$$\begin{aligned}\beta(x_i) &= \exp\Big\{-\tfrac{1}{2}\Big(x_i^\top(S_{i|i+1} + C^\top R^{-1}C)x_i\\ &\qquad\qquad - 2x_i^\top(r_{i|i+1} + C^\top R^{-1}z_{i|i-1,e}) - c_{i|i+1} - b_i\Big)\Big\}\\ &\equiv \exp\Big\{-\tfrac{1}{2}(x_i^\top S_{i|i}x_i - 2x_i^\top r_{i|i} - c_{i|i})\Big\},\end{aligned} \tag{7.50b}$$

which reveals that $r_{i|i}$ and $S_{i|i}$ are given by (7.22), and $c_{i|i} = c_{i|i+1} + b_i$. The last step in proving Lemma 7.1 consists of finding a closed-form algebraic recursion for updating the shaping parameters of (7.48). For $i = n, \ldots, 2$, this can be achieved by first substituting (7.50b) and $f(x_i|x_{i-1}) = \mathcal{N}_{x_i}(Ax_{i-1}, Q)$ into (7.48) and then making use of Lemma 7.2 to write

$$\begin{aligned}\gamma(x_{i-1}) &= \int \exp\Big\{-\tfrac{1}{2}(x_i^\top S_{i|i}x_i - 2x_i^\top r_{i|i})\Big\}\mathcal{N}_{x_i}(Ax_{i-1}, Q)dx_i \exp\Big\{\tfrac{1}{2}c_{i|i}\Big\}\\ &\equiv \int \mathcal{N}_{x_i}(\mu_i^o, \Sigma_i^o)dx_i \exp\Big\{-\tfrac{1}{2}(x_{i-1}^\top S_{i-1|i}x_{i-1} - 2x_i^\top r_{i-1|i} - c_{i-1|i})\Big\},\end{aligned}$$

applying $\int \mathcal{N}_{x_i}(\mu_i^o, \Sigma_i^o)dx_i = 1$ leads to the sought result, with $r_{i-1|i}$ and $S_{i-1|i}$ given by (7.21), and $c_{i-1|i} = c_{i|i} + a_i$. ■

7.6.4 Proof of Proposition 2

The proof of Proposition 2 relies on rewriting (7.12) according to

$$m^o(x_i|x_{i-1}, f_e) = \frac{f(x_i|x_{i-1})\beta(x_i)}{\int f(x_i|x_{i-1})\beta(x_i)dx_i}, \tag{7.51}$$

with $\beta(x_i)$ having the same form as (7.49). For $i = n, \dots, 2$, we first substitute $f(x_i|x_{i-1}) = \mathcal{N}_{x_i}(Ax_{i-1}, Q)$ and (7.50) in (7.51) and then use Lemma 7.2 to obtain

$$m^o(x_i|x_{i-1}, f_e) = \frac{\mathcal{N}_{x_i}(\mu_i^o, \Sigma_i^o)\exp\left\{-\frac{1}{2}(x_{i-1}^\top S_{i-1|i}x_{i-1} - 2x_i^\top r_{i-1|i} - c_{i-1|i})\right\}}{\exp\left\{-\frac{1}{2}(x_{i-1}^\top S_{i-1|i}x_{i-1} - 2x_i^\top r_{i-1|i} - c_{i-1|i})\right\}}, \tag{7.52}$$

where we utilize $\int \mathcal{N}_{x_i}(\mu_i^o, \Sigma_i^o)dx_i = 1$ in the denominator. Canceling out the exponential terms then leads to sought result (7.24). ■

7.7 Additional Experiments

We stick to the same simulation example and settings as in Section 4 and only investigate the performance of the compared filters for different values of the observation variance R and state covariance matrix Q.

The MNSE for $R = (10^{-1}, 10^{-2}, 10^{-4}, 10^{-5})$ is depicted in Fig. 7.3 (with the case of $R = 10^{-3}$ being given in Fig. 7.2). The situation where $R = 10^{-1}$ and $R = 10^{-2}$ is similar. For both these settings, we can observe that the DT filter is outperformed by the MVF filter when the values of R_e are low. However, from approximately two orders of magnitude below $R_e = R$, the DT filter starts to dominate the MVF filter. Once again, we need to remind that the DT filter is not designed with dependence assumptions between external and primary quantities being known. The ST filter is closer to the performance of the NT filter but still takes advantage of the external information to improve its performance for most of the values of R_e. The situation changes for increasingly more precise measurements of the primary filter $R = 10^{-4}$ and $R = 10^{-5}$. The DT filter is now better than MVF filter for the values of R_e that are lower than those slightly after the intersection point $R_e = R$. In the case of highly precise measurements $R = 10^{-5}$, we can see that the ST filter no longer benefits from the external information at any value of R_e, and the MVF filter has practically the same performance as the NT filter. The DTi filter surpasses the remaining filters at every combination of R and R_e.

Fig. 7.4 shows the MNSE of the compared methods for $Q = (10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}) \cdot I_2$. (The value of primary observation variance is $R = 10^{-3}$ and the situation for $Q = 10^{-5} \cdot I_2$ coincides with Fig. 7.2.) Describing the situation from $Q = 10^{-4} \cdot I_2$ to $Q = 10^{-1} \cdot I_2$, we can observe that the MNSE of the MVF filter quickly becomes indistinguishable from the NT filter and remains in this state for all the values Q. The MVF filter therefore substantially suffers from higher values of Q. The ST method also performs poorly when increasing Q, where for $Q \geq 10^{-3} \cdot I_2$ the external information does not improve its performance over the NT filter. However, an important observation is that the DT procedure provides better performance compared to the NT and ST filters for most of the values of R_e and Q. For increasing values of Q, we see that the distance between the MNSE of the DT and DTi filters diminishes when decreasing R_e. We can also observe that as the values of Q increase, the MNSE of the DTi filter is increasingly collinear with, and have farther distance from, the reference level delineated by the NT filter for higher values of R_e. Similarly as in the case of changing R, the DTi filter outperforms the rest of the algorithms over the full range of R_e and Q. All in all, we can conclude that the DT and DTi methods are more robust for high values of state covariance matrix Q.

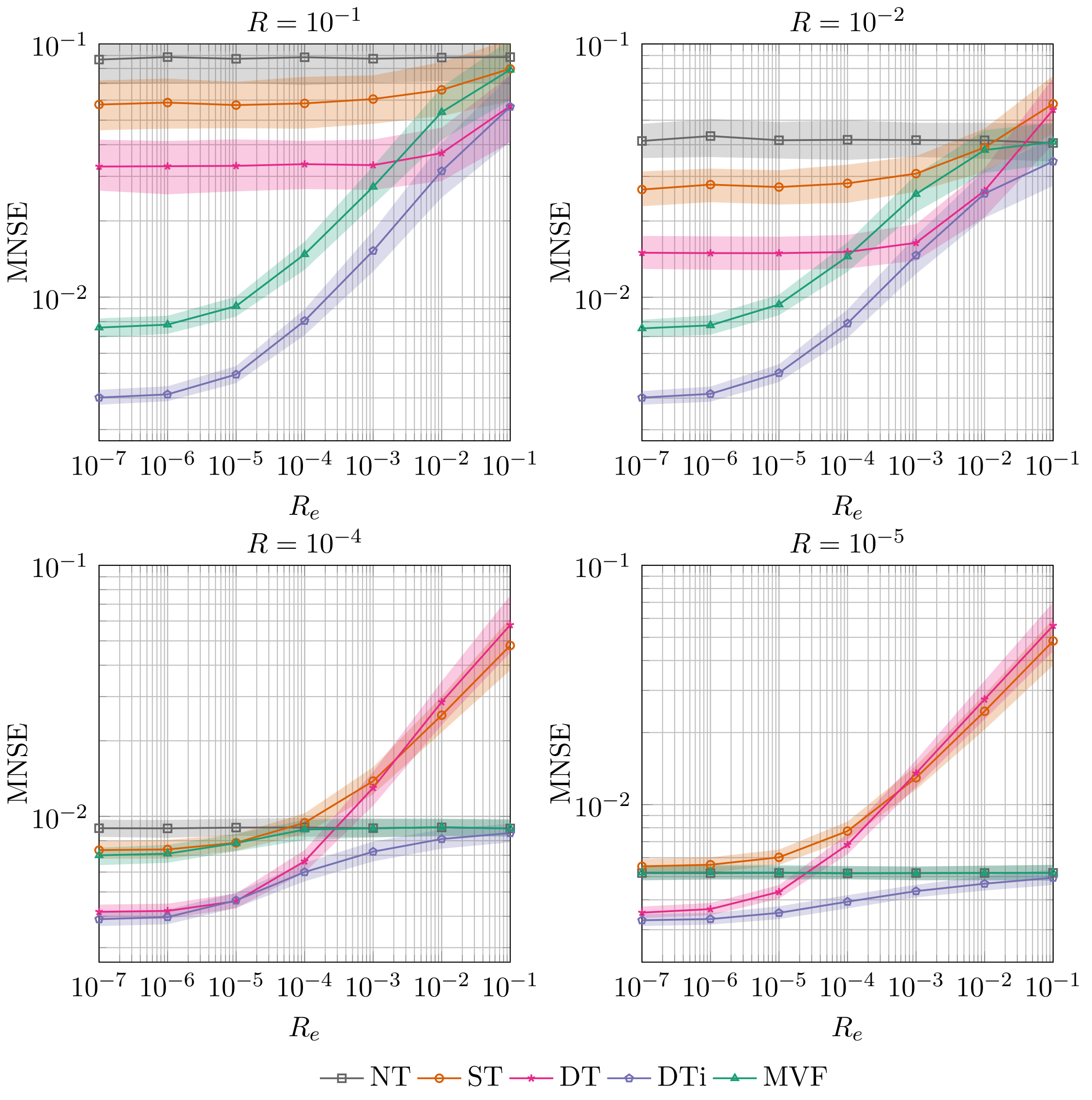

Fig. 7.3: The mean norm squared-error (MNSE) of the primary filter versus the observation variance R_e of the eternal Kalman filter for different settings of the observation variance R of the primary Kalman filter. The results are averaged over 1000 independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range. The procedures that are compared are (i) the Kalman filter with *N*o *T*ransfer (*NT*), (ii) *S*tatic Bayesian knowledge *T*ransfer (*ST*) [67], (iii) *D*ynamic Bayesian knowledge *T*ransfer (*DT*) given by Algorithm 21, (iv) an *i*nformally adapted version of DT (*DTi*) which we mention in Section 7.5; and (v) *M*easurement *V*ector *F*usion (*MVF*) [220].

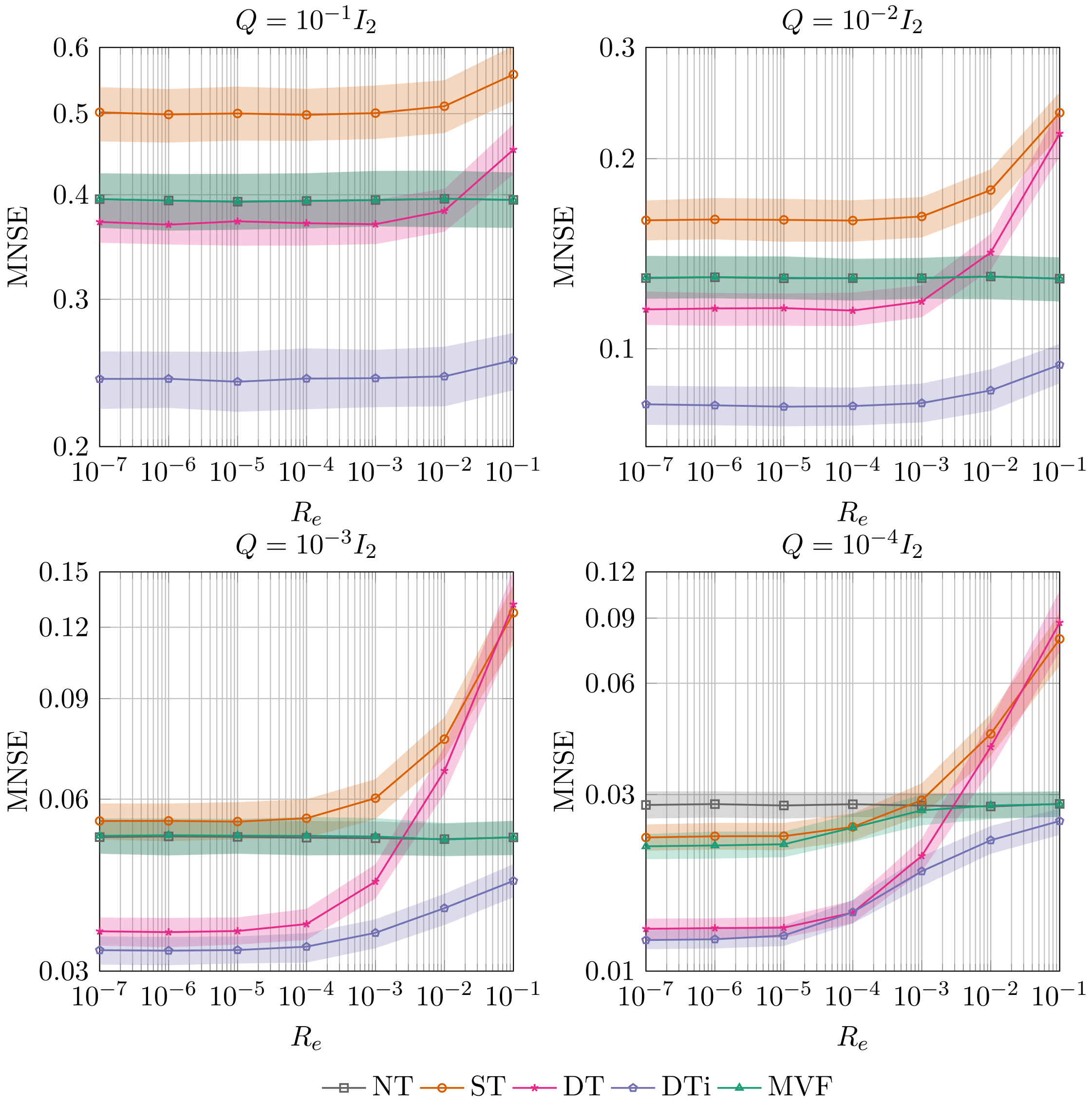

Fig. 7.4: The mean norm squared-error (MNSE) of the primary filter versus the observation variance R_e of the eternal Kalman filter for different settings of the state covariance matrix Q of the primary Kalman filter. The results are averaged over 1000 independent simulation runs, with the solid line being the median and the shaded area delineating the interquartile range. The procedures that are compared are (i) the Kalman filter with *N*o *T*ransfer (*NT*), (ii) *S*tatic Bayesian knowledge *T*ransfer (*ST*) [67], (iii) *D*ynamic Bayesian knowledge *T*ransfer (*DT*) given by Algorithm 21, (iv) an *i*nformally adapted version of DT (*DTi*) which we mention in Section 7.5; and (v) *M*easurement *V*ector *F*usion (*MVF*) [220].

CONCLUSION

Chapter 1 describes the fundamental principles and building blocks of sequential Monte Carlo and particle Markov chain Monte Carlo methods. The presentation in this chapter is made in a generic way in order to focus on only the main principles of the discussed algorithms, without adopting any context specific details.

Chapter 2 discusses implementation aspects and consequences of applying the methods presented in Chapter 1 to the state and parameter inference in state-space models. To support the motivation for the use of SMC-based methods in this thesis, an attention is paid to also discuss alternative approximate strategies—the assumed Gaussian density methods. We do not make any final ranking between these approaches as they both have their pros and cons depending on a specific problem (being the trade-off between the approximation error and computational time, which depends on the severity of nonlinearities in the state-space model). Chapter 2 considers various strategies to perform the filtering and smoothing in state-space models. A possible agenda for future work in this matter is to make an exhaustive and up-to-date experimental comparison of diverse smoothing strategies, as such comparison is currently missing in the literature.

Chapter 3 is concerned with the design of the projection-based Rao-Blackwellized particle filter for estimating static parameters in the conditionally conjugate state-space models. The primary objective was to devise an SMC-based approach which counteracts the particle path degeneracy problem. This was accomplished by formulating the projection-based updates for computing the statistics representing the posterior density of the parameters in order to avoid their resampling and thus make them less affected by the degenerate particle trajectories. The results reveal that the proposed solution indeed decreases the variance of the parameter estimates over multiple simulation runs compared to the plain Rao-Blackwellized particle filter, and it therefore suffers less from the degeneracy problem. Moreover, the proposed approach outperforms a number of alternative techniques for parameter estimation in nonlinear and non-Gaussian state-space models. In the presented experiment, the resulting solution has approximately the same computational complexity as the basic Rao-Blackwellized particle filter but provides an improved estimation precision. Therefore, for the same precision level of both these methods, we obtain a considerable decrease in the computational time in favor of the proposed method. When changing the signal-to-noise ratio in the considered experimental setup, the proposed projection-based Rao-Blackwellized particle filter starts to be more sensitive to the initial setting of the posterior statistics. This increased sensitivity is mainly caused by the adoption of the bootstrap proposal density. Therefore, designing a suitable approximation of the optimal proposal density may provide more robustness in this sense.

The proposed algorithm can be applied to, e.g., Gaussian process-based (Bayesian) optimization [85], seasonal epidemics detection [129], charge estimation of batteries [138], etc.

The idea of computing the projections seems to provide an interesting opportunity for counteracting the particle path degeneracy problem. Therefore, the primary aim of future work should be focused on different strategies for the evolution of the statistics and investigating dependence of the algorithm on the forgetting properties of the state-space model. A possible generalization of the proposed approach is to use an MCMC procedure [78] at each iteration in order to facilitate application to nonlinear and non-Gaussian state-space models without the tractable substructure with respect to the parameters. An increase in the computational complexity of such an algorithm should be expected. Another possibility is to extend the method to allow for the parameter inference in the conditionally conjugate jump Markov models. Such a method could then be applied to, e.g., traffic flow monitoring [174] and evaluation of the stock return sensitivity to macroeconomic news announcement [89]. Alternatively, to enable tracking of time-varying parameters, it is also tempting to extend the estimation procedure by a suitable forgetting strategy [122, 107].

Chapter 4 investigates the possibility of using alternative stabilized forgetting in the context of SMC-based estimation of slowly-varying parameters in conditionally conjugate state-space models. It is demonstrated that the proposed Rao-Blackwellized particle filter outperforms the one introduced in [167]; more concretely, the estimates of the measurement noise variance are less biased, and the approach also reduces the variance of the estimated parameters. This is achieved in a computationally more efficient way. Specifically, in the present experiment, the proposed method reduces the computational time by an order of magnitude. The algorithm offers a fair degree of adaptability by allowing us to tune the forgetting of the past information by the hyper-parameters of the alternative density. This makes the method slightly more difficult to tune (setting the statistic μ_A of the alternative density to zero always substantially simplifies the initial tuning).

There is a multitude of practical problems for which the proposed technique can be utilized, such as estimating parameters of automotive-grade sensors [18], tire radii estimation [142], etc.

The proposed algorithm—similarly to the one from Chapter 3—can also be extended to incorporate the MCMC steps, thus broadening the range of admissible models to completely nonlinear and non-Gaussian state-space models. However, to simplify the applicability of the proposed method, the main direction of future work will consist in facilitating an autonomous adaptation of the hyper-parameters of the alternative density. A possible approach how to solve this requirement lies in the hierarchical Bayesian modeling [17].

Chapter 5 designs Rao-Blackwellized particle Gibbs kernels for smoothing in jump Markov nonlinear models. The experimental evidence shows that the proposed algorithms are computationally more efficient than the competing approaches. An additional investigation of the proposed (ancestor-sampling-based) procedure revealed that the introduction of the artificial prior is redundant. However, changing the scale of the backward information filtering recursion—provided by the associated design step—is necessary. Practically, this means that we can set the artificial prior to one, while leaving the related derivations intact. The necessary change of scale is then still preserved in the algorithm design. A formally more suitable derivation of this part of the algorithm is provided in Chapter 6. In various experiments, the algorithm *without* the change of scale provided poor estimation precision compared to the one *with* this change. In fact, the former version numerically failed several times during the experiments, whereas the latter one always prevailed.

A possible application scenario for the developed smoothing algorithm consists in offline processing of experimental data in indoor localization [160], target classification [8], fault detection [203], etc. In such cases, the proposed method can serve as a generator of reference trajectories for the development and validation of online algorithms.

Chapter 6 proposes the Rao-Blackwellized particle stochastic approximation expectation algorithm for jump Markov nonlinear models, offering a computationally more efficient alternative to the basic formulation which jointly samples both the latent variables. The efficiency depends on the distance between the individual regimes of the jump Markov nonlinear model. On the one hand, if the regime parameters are substantially different, it is easy to detect the changes in the observations and the algorithm provides best efficiency. On the other hand, if the regime parameters are very similar, it is harder to capture the changes in the observations and the method is less efficient. However, in the latter case, it is no more reasonable to use an algorithm which assumes both continuous state and discrete regime variables, it would suffice to use an algorithm which considers only the continuous state variable. The rationale behind this statement is that the changes in the observations become so small that they will be hidden in the noise, and there is thus no need to consider a jump Markov nonlinear model but rather a plain nonlinear non-Gaussian state-space model. Therefore, the best performance can be expected when the changes are clearly distinguishable from the noise. This behavior is common for all algorithms dealing with switching models.

The method is applicable to parameter identification in diverse application areas such as option pricing in financial markets [35], engine performance diagnosis [213], land vehicle positioning [31], etc.

The proposed Rao-Blackwellized particle stochastic approximation expectation algorithm can be seen as an instance of where the Rao-Blackwellized particle Gibbs kernel from Chapter 5 can be utilized. This building block opens up for the design of various identification strategies in jump Markov nonlinear models, including particle Gibbs with ancestor sampling for Bayesian parameter inference [132].

Chapter 7 devises an FPD-based optimal dynamic Bayesian transfer learning approach and shows its application to probabilistic knowledge transfer between a pair of Kalman filters. The resulting experiments demonstrate that FPD offers a potential for building a versatile framework for Bayesian transfer learning. However, there is still the question of dealing with the aforementioned insensitivity to the second moment transfer, as discussed in Section 7.5. A possible answer to this problem may lie in the recently proposed hierarchical FPD-based Bayesian transfer learning [178], which will be the primary aim of future work.

We have focused thusfar on the basic scenario of one-directional knowledge transfer between two nodes. The natural extension of the proposed approach therefore consists of (i) facilitating the knowledge transfer among a greater number of nodes and (ii) making the transfer bi-directional. Specifically, the former point will require us to introduce an optimal weighting mechanism to assess knowledge in a network of nodes. Another possible extension is to replace the Kalman filters with different forms of Gaussian filters [188], requiring us to make slight modifications to the derivations presented in Section 7.3.2. Although the application of sequential Monte Carlo methods [56] may be feasible, the recursive computation of (7.14) may present problems. Finally, one can change the transferred knowledge and conditional independence assumptions specified in (7.8) in order to propose other FPD-based transfer learning options, such as transfer of the external joint state predictor.

BIBLIOGRAPHY

[1] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1965.

[2] B. Anderson and J. B. Moore. *Optimal filtering.* Prentice-Hall, 1979.

[3] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003.

[4] C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[5] C. Andrieu, A. Doucet, S. S. Singh, and V. B. Tadić. Particle methods for change detection, system identification, and control. *Proceedings of the IEEE*, 92(3):423–438, 2004.

[6] C. Andrieu, A. Doucet, and V. Tadic. On-line parameter estimation in general state-space models. In *Proceedings of the 44th IEEE Conference on Decision and Control (CDC)*, pages 332–337, 2005.

[7] C. Andrieu, M. Vihola, et al. Markovian stochastic approximation with expanding projections. *Bernoulli*, 20(2):545–585, 2014.

[8] D. Angelova and L. Mihaylova. Joint target tracking and classification with particle filtering and mixture Kalman filtering using kinematic radar information. *Digital Signal Processing*, 16(2):180–204, 2006.

[9] I. Arasaratnam and S. Haykin. Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature. *Proceedings of the IEEE*, 95(5), 2007.

[10] I. Arasaratnam and S. Haykin. Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009.

[11] T. T. Ashley and S. B. Andersson. A sequential Monte Carlo framework for the system identification of jump Markov state space models. In *Proceedings of 2014 American Control Conference (ACC)*, pages 1144–1149, 2014.

[12] O. Barndorff-Nielsen. *Information and Exponential Families in Statistical Theory.* Wiley, 1978.

[13] V. Bastani, L. Marcenaro, and C. Regazzoni. A particle filter based sequential trajectory classifier for behavior analysis in video surveillance. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages 3690–3694, 2015.

[14] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36, 2012.

[15] J. O. Berger. *Statistical decision theory and Bayesian analysis*. Springer, 2013.

[16] J. M. Bernardo. Expected information as expected utility. *Annals of Statistics*, 7:686–690, 1979.

[17] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. Wiley, 1994.

[18] K. Berntorp and S. D. Cairano. Offset and noise estimation of automotive-grade sensors using adaptive particle filtering. In *2018 Annual American Control Conference (ACC)*, pages 4745–4750, 2018.

[19] A. Beskos, D. Crisan, A. Jasra, et al. On the stability of sequential Monte Carlo methods in high dimensions. *The Annals of Applied Probability*, 24(4):1396–1445, 2014.

[20] P. Billingsley. *Probability and Measure*. Wiley, 1995.

[21] M. C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[22] H. A. Blom and E. A. Bloem. Exact Bayesian and particle filtering of stochastic hybrid systems. *IEEE Transactions on Aerospace and Electronic Systems*, 43(1):55–70, 2007.

[23] A. R. Braga, M. G. S. Bruno, E. Özkan, C. Fritsche, and F. Gustafsson. Cooperative terrain based navigation and coverage identification using consensus. In *Proceedings of 18th International Conference on Information Fusion (Fusion)*, pages 1190–1197, 2015.

[24] Y. Bresler. Two-filter formulae for discrete-time non-linear Bayesian smoothing. *International Journal of Control*, 43(2):629–641, 1986.

[25] M. Briers, A. Doucet, and S. Maskell. Smoothing algorithms for state–space models. Technical Report CUED/F-INFENG/TR.498, Cambridge University, 2004.

[26] M. Briers, A. Doucet, and S. Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61–89, 2010.

[27] J. Bucklew. *Introduction to rare event simulation.* Springer, 2013.

[28] O. Cappé. Online sequential Monte Carlo EM algorithm. In *15th IEEE Workshop on Statistical Signal Processing*, pages 37–40, 2009.

[29] O. Cappé and E. Moulines. On-line expectation-maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

[30] O. Cappé, E. Moulines, and T. Ryden. *Inference in Hidden Markov Models.* Springer, 2005.

[31] F. Caron, M. Davy, E. Duflos, and P. Vanheeghe. Particle filtering for multisensor data fusion with switching observation models: Application to land vehicle positioning. *IEEE Transactions on Signal Processing*, 55(6):2703–2719, 2007.

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

[33] C. K. Carter and R. Kohn. On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994.

[34] C. M. Carvalho, M. S. Johannes, H. F. Lopes, and N. G. Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2010.

[35] C. M. Carvalho and H. F. Lopes. Simulation-based sequential analysis of Markov switching stochastic volatility models. *Computational Statistics & Data Analysis*, 51(9):4526–4542, 2007.

[36] G. Casella and C. P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[37] N. Chopin et al. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.

[38] N. Chopin, A. Iacobucci, J.-M. Marin, K. Mengersen, C. P. Robert, R. Ryder, and C. Schäfer. On particle learning. *arXiv preprint arXiv:1006.0554*, 2010.

[39] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos. Smc2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013.

[40] Y. S. Chow and H. Teicher. *Probability theory: independence, interchangeability, martingales*. Springer, 2012.

[41] R. Cools. Constructing cubature formulae: the science behind the art. *Acta Numerica*, 6:1–54, 1997.

[42] J. Cornebise. Discussion on Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):317–319, 2010.

[43] D. Crisan, J. Miguez, et al. Nested particle filters for online parameter estimation in discrete-time state-space Markov models. *Bernoulli*, 24(4A):3039–3086, 2018.

[44] F. Daum. Exact finite-dimensional nonlinear filters. *IEEE Transactions on Automatic Control*, 31(7):616–622, July 1986.

[45] K. Dedecius, I. Nagy, and M. Kárný. Parameter tracking with partial forgetting method. *International Journal of Adaptive Control and Signal Processing*, 26(1):1–12, 2012.

[46] M. DeGroot. *Optimal Statistical Decisions*. Wiley, 2004.

[47] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

[48] P. Del Moral, A. Doucet, and S. Singh. Forward smoothing using sequential Monte Carlo. *arXiv preprint arXiv:1012.5390*, 2010.

[49] P. Del Moral, A. Doucet, and S. S. Singh. A backward particle interpretation of feynman-kac formulae. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(5):947–975, 2010.

[50] B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94–128, 1999.

[51] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 39(1):1–38, 1977.

[52] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 64–69, Sept 2005.

[53] R. Douc, A. Garivier, E. Moulines, and J. Olsson. Sequential Monte Carlo smoothing for general state space hidden Markov models. *The Annals of Applied Probability*, 21(6):2109–2145, 2011.

[54] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[55] A. Doucet, N. J. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624, 2001.

[56] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2009.

[57] A. Doucet, A. Smith, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[58] H. Driessen and Y. Boers. Efficient particle filter for jump Markov nonlinear systems. *IEE Proceedings - Radar, Sonar and Navigation*, 152(5):323–326, 2005.

[59] V. Dukic, H. F. Lopes, and N. G. Polson. Tracking epidemics with Google flu trends data and a state-space SEIR model. *Journal of the American Statistical Association*, 107(500):1410–1426, 2012.

[60] L. Eeckhout. Is Moore's law slowing down? What's next? *IEEE Micro*, 37(4):4–5, 2017.

[61] R. Faragher. Understanding the basis of the Kalman filter via a simple and intuitive derivation. *IEEE Signal Processing Magazine*, 29(5):128–132, 2012.

[62] P. Fearnhead. Markov chain Monte Carlo, sufficient statistics, and particle filters. *Journal of Computational and Graphical Statistics*, 11(4):848–862, 2002.

[63] P. Fearnhead. Discussion on Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):302–304, 2010.

[64] P. Fearnhead and P. Clifford. On-line inference for hidden Markov models via particle filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(4):887–899, 2003.

[65] P. Fearnhead and H. R. Künsch. Particle filters and data assimilation. *Annual Review of Statistics and Its Application*, 5(1):1–31, 2018.

[66] P. Fearnhead, D. Wyncoll, and J. Tawn. A sequential smoothing algorithm with linear computational cost. *Biometrika*, 97(2):447–464, 2010.

[67] C. Foley and A. Quinn. Fully probabilistic design for knowledge transfer in a pair of Kalman filters. *IEEE Signal Processing Letters*, 25(4):487–490, 2018.

[68] G. Fort and E. Moulines. Convergence of the Monte Carlo expectation maximization for curved exponential families. *The Annals of Statistics*, 31(4):1220–1259, 2003.

[69] D. Fraser and J. Potter. The optimum linear smoother as a combination of two optimum linear filters. *IEEE Transactions on Automatic Control*, 14(4):387–390, 1969.

[70] S. Frühwirth-Schnatter. Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, 15(2):183–202, 1994.

[71] M. Gašperin and Đ. Juričić. Application of unscented transformation in nonlinear system identification. *IFAC Proceedings Volumes*, 44(1):4428–4433, 2011.

[72] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741, 1984.

[73] J. Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica: Journal of the Econometric Society*, pages 1317–1339, 1989.

[74] J. Geweke and H. Tanizaki. Bayesian estimation of state-space models using the Metropolis–Hastings algorithm within Gibbs sampling. *Computational Statistics & Data Analysis*, 37(2):151–170, 2001.

[75] Z. Ghahramani and S. T. Roweis. Learning nonlinear dynamical systems using an EM algorithm. In *Advances in neural information processing systems*, pages 431–437, 1999.

[76] J. Ghosh and R. Ramamoorthi. *Bayesian Nonparametrics.* Springer, 2003.

[77] S. Gibson and B. Ninness. Robust maximum-likelihood estimation of multivariable dynamic systems. *Automatica*, 41(10):1667–1682, 2005.

[78] W. R. Gilks and C. Berzuini. Following a moving target–Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146, 2001.

[79] W. J. Godinez, M. Lampe, P. Koch, R. Eils, B. Muller, and K. Rohr. Identifying virus-cell fusion in two-channel fluorescence microscopy image sequences based on a layered probabilistic approach. *IEEE Transactions on Medical Imaging*, 31(9):1786–1808, 2012.

[80] S. J. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, 99(465):156–168, 2004.

[81] G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15(2):318–334, 1973.

[82] G. C. Goodwin and J. C. Aguero. Approximate EM algorithms for parameter and state estimation in nonlinear stochastic models. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 368–373. Citeseer, 2005.

[83] G. C. Goodwin and R. L. Payne. *Dynamic System Identification: Experiment Design and Data Analysis.* Academic Press, 1977.

[84] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.

[85] R. B. Gramacy and N. G. Polson. Particle learning of Gaussian process models for sequential design and optimization. *Journal of Computational and Graphical Statistics*, 20(1):102–118, 2011.

[86] R. B. Gramacy, M. Taddy, and S. M. Wild. Variable selection and sensitivity analysis using dynamic trees, with an application to computer code performance tuning. *The Annals of Applied Statistics*, 7(1):51–80, 2013.

[87] D. Guo and X. Wang. Quasi-Monte Carlo filtering in nonlinear dynamic systems. *IEEE Transactions on Signal Processing*, 54(6):2087–2098, 2006.

[88] J. E. Handschin and D. Q. Mayne. Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control*, 9(5):547–559, 1969.

[89] T. Hann Law, D. Song, and A. Yaron. Fearing the Fed: How Wall Street reads main street. *SSRN Electronic Journal*, 2017.

[90] J. D. Hol. Resampling in particle filters, 2004.

[91] D. Isele and A. Cosgun. Transferring autonomous driving knowledge on simulated and real intersections. *arXiv preprint arXiv:1712.01106*, 2017.

[92] K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 45(5), 2000.

[93] P. E. Jacob. Sequential Bayesian inference for implicit hidden Markov models and current limitations. *ESAIM: Proceedings and Surveys*, 51:24–48, 2015.

[94] P. E. Jacob, L. M. Murray, and S. Rubenthaler. Path storage in the particle filter. *Statistics and Computing*, 25(2):487–496, 2015.

[95] B. Jia, M. Xin, and Y. Cheng. High-degree cubature Kalman filter. *Automatica*, 49(2):510–518, 2013.

[96] M. Johannes, A. Korteweg, and N. Polson. Sequential learning, predictability, and optimal portfolio returns. *The Journal of Finance*, 69(2):611–644, 2014.

[97] M. Johannes, L. A. Lochstoer, and Y. Mou. Learning about consumption dynamics. *The Journal of Finance*, 71(2):551–600, 2016.

[98] A. M. Johansen. Discussion on Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):326–327, 2010.

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

[100] A. M. Johansen, N. Whiteley, and A. Doucet. Exact approximation of Rao-Blackwellised particle filters. In *Proceedings of the 16th IFAC Symposium on System Identification*, volume 45, pages 488–493, 2012.

[101] G. L. Jones. On the Markov chain central limit theorem. *Probability Surveys*, 1:229–320, 2004.

[102] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

[103] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[104] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin. On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351, 2015.

[105] A. Karbalayghareh, X. Qian, and E. R. Dougherty. Optimal Bayesian transfer learning. *arXiv preprint arXiv:1801.00857*, 2018.

[106] M. Kárný. Towards fully probabilistic control design. *Automatica*, 32(12):1719–1722, 1996.

[107] M. Kárný. Approximate Bayesian recursive estimation. *Information Sciences*, 285(1):100–111, 2014.

[108] M. Kárný and J. Andrýsek. Use of Kullback-–Leibler divergence for forgetting. *International Journal of Adaptive Control and Signal Processing*, 23(10):961–975, 2009.

[109] M. Kárný and T. V. Guy. Fully probabilistic control design. *Systems & Control Letters*, 55(4):259–265, 2006.

[110] M. Kárný and T. Kroupa. Axiomatisation of fully probabilistic design. *Information Sciences*, 186(1):105–113, 2012.

[111] D. F. Kerridge. Inaccuracy and inference. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 23(1):184–194, 1961.

[112] G. Kitagawa. Non-Gaussian state—space modeling of nonstationary time series. *Journal of the American Statistical Association*, 82(400):1032–1041, 1987.

[113] G. Kitagawa. The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, 46(4):605–623, 1994.

[114] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[115] G. Kitagawa. A self-organizing state-space model. *Journal of the American Statistical Association*, 93(443):1203–1215, 1998.

[116] M. Klaas, N. d. Freitas, and A. Doucet. Toward practical N2 Monte Carlo: the marginal particle filter. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 308–315, 2005.

[117] A. Klenke. *Probability Theory: A Comprehensive Course.* Springer, 2013.

[118] J. Kokkala, A. Solin, and S. Särkkä. Sigma-point filtering and smoothing based parameter estimation in nonlinear dynamic systems. *arXiv preprint arXiv:1504.06173*, 2015.

[119] A. Kong. A note on importance sampling using standardized weights. Technical Report 348, University of Chicago, 1992.

[120] J. H. Kotecha and P. M. Djuric. Gaussian sum particle filtering. *IEEE Transactions on Signal Processing*, 51(10):2602–2612, 2003.

[121] E. Kuhn and M. Lavielle. Coupling a stochastic approximation version of EM with an MCMC procedure. *ESAIM: Probability and Statistics*, 8:115–131, 2004.

[122] R. Kulhavý and M. B. Zarrop. On a general concept of forgetting. *International Journal of Control*, 58(4):905–924, 1993.

[123] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[124] K. Lange. A gradient algorithm locally equivalent to the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pages 425–437, 1995.

[125] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling.* Springer, 2009.

[126] T. Li, M. Bolic, and P. M. Djuric. Resampling methods for particle filtering: classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3):70–86, 2015.

[127] W. Li, Y. Jia, J. Du, and J. Zhang. Distributed multiple-model estimation for simultaneous localization and tracking with NLOS mitigation. *IEEE Transactions on Vehicular Technology*, 62(6):2824–2830, 2013.

[128] K. Lidstrom and T. Larsson. Model-based estimation of driver intentions using particle filtering. In *Proceedings of 11th International IEEE Conference on Intelligent Transportation Systems*, pages 1177–1182, 2008.

[129] J. Lin and M. Ludkovski. Sequential Bayesian inference in hidden Markov stochastic kinetic models with application to detection and response to seasonal epidemics. *Statistics and Computing*, 24(6):1047–1062, 2014.

[130] F. Lindsten. An efficient stochastic approximation EM algorithm using conditional particle filters. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6274–6278, 2013.

[131] F. Lindsten, P. Bunch, S. Särkkä, T. B. Schön, and S. J. Godsill. Rao-Blackwellized particle smoothers for conditionally linear Gaussian models. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):353–365, 2016.

[132] F. Lindsten, M. I. Jordan, and T. B. Schön. Particle Gibbs with ancestor sampling. *Journal of Machine Learning Research*, 15(1):2145–2184, 2014.

[133] F. Lindsten and T. B. Schön. Backward simulation methods for Monte Carlo statistical inference. *Foundations and Trends in Machine Learning*, 6(1):1–143, 2013.

[134] F. Lindsten, T. B. Schön, and L. Svensson. A non-degenerate Rao-Blackwellised particle filter for estimating static parameters in dynamical models. *IFAC Proceedings Volumes*, 45(16):1149–1154, 2012.

[135] J. S. Liu. *Monte Carlo strategies in scientific computing.* Springer, 2004.

[136] J. S. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In A. Doucet, N. De Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 10, pages 197–223. Springer, New York, 2001.

[137] J. S. Liu, W. H. Wong, and A. Kong. Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika*, 81(1):27–40, 1994.

[138] X. Liu, Z. Chen, C. Zhang, and J. Wu. A novel temperature-compensated model for power li-ion batteries with dual-particle-filter state of charge estimation. *Applied Energy*, 123:263–272, 2014.

[139] Z. Liu, G. Sun, S. Bu, J. Han, X. Tang, and M. Pecht. Particle learning framework for estimating the remaining useful life of lithium-ion batteries. *IEEE Transactions on Instrumentation and Measurement*, 66(2):280–293, 2017.

[140] L. Ljung. *System identification - Theory for the User.* Prentice-Hall, 1998.

[141] H. F. Lopes and C. M. Carvalho. Factor stochastic volatility with time varying loadings and Markov switching regimes. *Journal of Statistical Planning and Inference*, 137(10):3082 – 3091, 2007.

[142] C. Lundquist, R. Karlsson, E. Ozkan, and F. Gustafsson. Tire radii estimation using a marginalized particle filter. *IEEE Transactions on Intelligent Transportation Systems*, 2(15):663–672, 2014.

[143] J. R. Magnus and H. Neudecker. Symmetry, 0-1 matrices and Jacobians: A review. *Econometric Theory*, 2(2):157–190, 1986.

[144] J. Marcinkiewicz and A. Zygmund. Sur les fonctions indépendantes. *Fundamenta Mathematicae*, 29:60–90, 1937.

[145] L. Martino, V. Elvira, and F. Louzada. Effective sample size for importance sampling based on discrepancy measures. *Signal Processing*, 131:386–401, 2017.

[146] C. Mavroforakis, I. Valera, and M. Gomez-Rodriguez. Modeling the dynamics of learning activity on the web. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1421–1430, 2017.

[147] P. Maybeck. *Stochastic Models, Estimation, and Control*, volume 2. Academic Press, 1982.

[148] D. Q. Mayne. A solution of the smoothing problem for linear dynamic systems. *Automatica*, 4(2):73–92, 1966.

[149] G. McLachlan and T. Krishnan. *The EM algorithm and extensions (2nd Edition).* John Wiley & Sons, 2008.

[150] J. McNamee and F. Stenger. Construction of fully symmetric numerical integration formulas of fully symmetric numerical integration formulas. *Numerische Mathematik*, 10(4):327–344, 1967.

[151] X.-L. Meng and D. B. Rubin. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80(2):267–278, 1993.

[152] S. P. Meyn and R. L. Tweedie. *Markov chains and stochastic stability.* Springer, 2012.

[153] T. P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference on Uncertainty in artificial intelligence*, pages 362–369, 2001.

[154] P. Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications.* Springer, 2004.

[155] C. Mukherjee and M. West. Sequential Monte Carlo in model comparison: Example in cellular dynamics in systems biology. In *JSM Proceedings, Section on Bayesian Statistical Science*, pages 1274–1287, 2009.

[156] K. P. Murphy. *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[157] L. M. Murray, A. Lee, and P. E. Jacob. Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics*, 25(3):789–805, 2016.

[158] C. Nemeth, P. Fearnhead, and L. Mihaylova. Sequential Monte Carlo methods for state and parameter estimation in abruptly changing environments. *IEEE Transactions on Signal Processing*, 62(5):1245–1255, 2014.

[159] T. N. M. Nguyen, S. Le Corff, and E. Moulines. On the two-filter approximations of marginal smoothing distributions in general state-space models. *Advances in Applied Probability*, 50(1):154–177, 2017.

[160] M. Nicoli, C. Morelli, and V. Rampa. A jump markov particle filter for localization of moving terminals in multipath indoor scenarios. *IEEE Transactions on Signal Processing*, 56(8):3801–3809, 2008.

[161] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*, volume 63. SIAM, 1992.

[162] M. NøRgaard, N. K. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36(11):1627–1638, 2000.

[163] J. Olsson, O. Cappé, R. Douc, and E. Moulines. Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli*, 14(1):155–179, 2008.

[164] J. Olsson, J. Westerborn, et al. Efficient particle-based online smoothing in general hidden Markov models: the PaRIS algorithm. *Bernoulli*, 23(3):1951–1996, 2017.

[165] A. B. Owen. *Monte Carlo theory, methods and examples.* 2013.

[166] E. Özkan, F. Lindsten, C. Fritsche, and F. Gustafsson. Recursive maximum likelihood identification of jump Markov nonlinear systems. *IEEE Transactions on Signal Processing*, 63(3):754–765, 2015.

[167] E. Özkan, V. Šmídl, S. Saha, C. Lundquist, and F. Gustafsson. Marginalized adaptive particle filtering for nonlinear models with unknown time-varying noise parameters. *Automatica*, 49(6):1566–1575, 2013.

[168] S. J. Pan. Transfer learning. In *Data Classification: Algorithms and Applications*, pages 537–558. Chapman and Hall/CRC, 2015.

[169] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE Signal Processing Magazine*, 32(3):53–69, 2015.

[170] A. Persing and A. Jasra. Likelihood computation for hidden markov models via generalized two-filter smoothing. *Statistics & Probability Letters*, 83(5):1433–1442, 2013.

[171] V. Peterka. Bayesian approach to system identification. In *Trends and Progress in System Identification*, pages 239–304, 1981.

[172] M. Pitt, R. Silva, P. Giordani, and R. Kohn. Auxiliary particle filtering within adaptive Metropolis-Hastings sampling. *arXiv preprint arXiv:1006.1914*, 2010.

[173] M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[174] N. Polson and V. Sokolov. Bayesian particle tracking of traffic flows. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):345–356, 2018.

[175] G. Poyiadjis, A. Doucet, and S. S. Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011.

[176] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, et al. *Numerical recipes.* Cambridge University Press, 1989.

[177] A. Quinn, M. Kárný, and T. V. Guy. Fully probabilistic design of hierarchical Bayesian models. *Information Sciences*, 369:532–547, 2016.

[178] A. Quinn, M. Kárný, and T. V. Guy. Optimal design of priors constrained by external predictors. *International Journal of Approximate Reasoning*, 84:150–158, 2017.

[179] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. MIT Press, 2005.

[180] H. E. Rauch, C. Striebel, and F. Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

[181] Y.-F. Ren and H.-Y. Liang. On the best constant in Marcinkiewicz–Zygmund inequality. *Statistics & probability letters*, 53(3):227–233, 2001.

[182] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[183] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[184] G. O. Roberts and S. K. Sahu. Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(2):291–317, 1997.

[185] M. Roth, E. Özkan, and F. Gustafsson. A student's t filter for heavy tailed process and measurement noise. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 5770–5774, 2013.

[186] D. B. Rubin. A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when the fraction of missing information is modest: the SIR algorithm (discussion of Tanner and Wong). *Journal of the American Statistical Association*, 82(398):543–546, 1987.

[187] S. Saha, G. Hendeby, and F. Gustafsson. Mixture Kalman filters and beyond. In *Current Trends in Bayesian Methodology with Applications*, pages 537–562, 2015.

[188] S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[189] S. Särkkä, P. Bunch, and S. J. Godsill. A backward-simulation based Rao-Blackwellized particle smoother for conditionally linear Gaussian models. *IFAC Proceedings Volumes*, 45(16):506–511, 2012.

[190] S. Särkkä, J. Hartikainen, I. S. Mbalawata, and H. Haario. Posterior inference on parameters of stochastic differential equations via non-linear Gaussian filtering and adaptive MCMC. *Statistics and Computing*, 25(2):427–437, 2015.

[191] S. Särkkä, J. Hartikainen, L. Svensson, and F. Sandblom. On the relation between Gaussian process quadratures and sigma-point methods. *arXiv preprint arXiv:1504.05994*, 2015.

[192] T. B. Schön, F. Gustafsson, and P.-J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7):2279–2289, 2005.

[193] T. B. Schön, F. Lindsten, J. Dahlin, J. Wågberg, A. C. Naesseth, A. Svensson, and L. Dai. Sequential Monte Carlo methods for system identification. In *Proceedings of the 17th IFAC Symposium on System Identification*, volume 48, pages 775–786, 2015.

[194] T. B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

[195] J. Shore and R. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on Information Theory*, 26(1):26–37, 1980.

[196] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[197] I. Smal, E. Meijering, K. Draegestein, N. Galjart, I. Grigoriev, A. Akhmanova, M. van Royen, A. Houtsmuller, and W. Niessen. Multiple object tracking in molecular bioimaging by Rao-Blackwellized marginal particle filtering. *Medical Image Analysis*, 12(6):764–777, 2008.

[198] V. Šmídl. Forgetting in marginalized particle filtering and its relation to forward smoothing. Technical Report LiTH-ISY-R-3009, Department of Electrical Engineering, Linköping University, 2011.

[199] J. Spanier and E. H. Maize. Quasi-random methods for estimating integrals using relatively small samples. *SIAM Review*, 36(1):18–44, 1994.

[200] G. Storvik. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing*, 50(2):281–289, 2002.

[201] A. H. Stroud and D. Secrest. *Gaussian quadrature formulas*. Prentice-Hall, 1966.

[202] A. Svensson, T. B. Schön, and F. Lindsten. Identification of jump Markov linear models using particle filters. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6504–6509. IEEE, 2014.

[203] S. Tafazoli and X. Sun. Hybrid system state tracking and fault detection using particle filters. *IEEE Transactions on Control Systems Technology*, 14(6):1078–1087, 2006.

[204] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

[205] L. Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, pages 1701–1728, 1994.

[206] L. Torrey and J. Shavlik. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 242–264. IGI Global, 2010.

[207] P. Toulis and E. M. Airoldi. Scalable estimation strategies based on stochastic approximations: classical results and new insights. *Statistics and Computing*, 25(4):781–795, 2015.

[208] J. Umenberger, J. Wågberg, I. R. Manchester, and T. B. Schön. On Identification via EM with Latent Disturbances and Lagrangian Relaxation. In *Proceedings of the 17th IFAC Symposium on System Identification SYSID 2015*, pages 69–74, 2015.

[209] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. A. Wan. The unscented particle filter. In *Advances in neural information processing systems*, pages 584–590, 2001.

[210] A. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 2000.

[211] T. L. M. Van Kasteren, G. Englebienne, and B. J. A. Kröse. Transferring knowledge of activity recognition across sensor networks. In *International Conference on Pervasive Computing*, pages 283–300. Springer, 2010.

[212] P. Vidoni. Exponential family state-space models based on a conjugate latent process. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(1):213–221, 1999.

[213] P. Wang and R. X. Gao. Markov nonlinear system estimation for engine performance tracking. *Journal of Engineering for Gas Turbines and Power*, 138(9):091201, 2016.

[214] Y. Wang, V. Gupta, and P. J. Antsaklis. Stochastic passivity of discrete-time markovian jump nonlinear systems. In *2013 American Control Conference*, pages 4879–4884, 2013.

[215] G. C. Wei and M. A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

[216] N. Whiteley. Discussion on Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):306–307, 2010.

[217] N. Whiteley. Stability properties of some particle filters. *The Annals of Applied Probability*, 23(6):2500–2537, 2013.

[218] N. Whiteley, C. Andrieu, and A. Doucet. Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods. *arXiv preprint arXiv:1011.2437*, 2010.

[219] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, 1994.

[220] D. Willner, C. B. Chang, and K. P. Dunn. Kalman filter algorithms for a multi-sensor system. In *1976 IEEE Conference on Decision and Control including the 15th Symposium on Adaptive Processes*, volume 15, pages 570–574. IEEE, 1976.

[221] A. Wills, T. B. Schön, F. Lindsten, and B. Ninness. Estimation of linear systems using a Gibbs sampler. In *Proceedings of the 16th IFAC Symposium on System Identification*, volume 45, pages 488–493, 2012.

[222] A. Wilson, A. Fern, and P. Tadepalli. Transfer learning in sequential decision problems: A hierarchical Bayesian approach. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 217–227, 2012.

[223] C. J. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[224] L. S.-Y. Wu, J. S. Pai, and J. Hosking. An algorithm for estimating parameters of state-space models. *Statistics & Probability Letters*, 28(2):99–106, 1996.

[225] Y. Wu, D. Hu, M. Wu, and X. Hu. A numerical-integration perspective on Gaussian filters. *IEEE Transactions on Signal Processing*, 54(8):2910–2921, 2006.

[226] C. Zeng, Q. Wang, S. Mokhtari, and T. Li. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2025–2034, 2016.

LIST OF PAPERS

[227] M. Papež and P. Pivoňka. Numerical aspects of inertial navigation. *IFAC-PapersOnLine*, 46(28):262–267, 2013.

[228] M. Papež. On Bayesian decision-making and approximation of probability densities. In *Proceedings of the 38th International Conference on Telecommunications and Signal Processing (TSP)*, pages 499–503, 2015.

[229] J. Dokoupil, M. Papež, and P. Václavek. Comparison of Kalman filters formulated as the statistics of the Normal-inverse-Wishart distribution. In *Proceedings of the 54th IEEE Conference on Decision and Control (CDC)*, pages 5008–5013, 2015.

[230] J. Dokoupil, M. Papež, and P. Václavek. Bayesian comparison of Kalman filters with known covariance matrices. *AIP Conference Proceedings*, 1648(1):070009, 2015.

[231] M. Papež. A Rao-Blackwellized particle filter to estimate the time-varying noise parameters in non-linear state-space models using alternative stabilized forgetting. In *Proceedings of the 16th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 229–234, 2016.

[232] M. Papež. Approximate bayesian inference methods for mixture filtering with known model of switching. In *Proceedings of the 17th International Carpathian Control Conference (ICCC)*, pages 545–551, 2016.

[233] M. Papež. Sequential Monte Carlo estimation of transition probabilities in mixture filtering problems. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 1063–1070, 2016.

[234] M. Papež. A projection-based Rao-Blackwellized particle filter to estimate parameters in conditionally conjugate state-space models. In *Proceedings of the 20th Statistical Signal Processing Workshop (SSP)*, pages 268–272, 2018.

[235] M. Papež. Rao-Blackwellized particle Gibbs kernels for smoothing in jump Markov nonlinear models. In *Proceedings of the 16th European Control Conference (ECC)*, pages 2466–2471, 2018.

[236] M. Papež. A particle stochastic approximation EM algorithm to identify jump Markov nonlinear models. *IFAC-PapersOnLine*, 51(15):676–681, 2018.

[237] M. Papež and A. Quinn. Dynamic Bayesian knowledge transfer between a pair of Kalman filters. In *Proceedings of the 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2018.

LIST OF APPENDICES

A SOME USEFUL STATISTICAL ANALYSIS

A.1 Preliminaries

Lemma A.1 (Block LDU and UDL decompositions)**.** *Let us consider real-valued matrices A, B, C, and D defined on the spaces $\mathbb{R}^{n\times n}$, $\mathbb{R}^{n\times m}$, $\mathbb{R}^{m\times n}$, and $\mathbb{R}^{m\times m}$, respectively, where $n \in \mathbb{N}_{\geq 1}$ and $m \in \mathbb{N}_{\geq 1}$. If the matrices A and D are invertible, then we can find LDU and UDL decompositions as follows:*

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I & O \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & O \\ O & D - CA^{-1}B \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ O & I \end{bmatrix} \tag{A.1a}$$

$$= \begin{bmatrix} I & BD^{-1} \\ O & I \end{bmatrix} \begin{bmatrix} A - BD^{-1}C & O \\ O & D \end{bmatrix} \begin{bmatrix} I & O \\ D^{-1}C & I \end{bmatrix}, \tag{A.1b}$$

where O and I are respectively zero and unit diagonal matrices of appropriate dimensions.

Proof. It is enough to simply multiply the matrices on the r.h.s. of (A.1a) and (A.1b) to make Lemma A.1 proved; however, it is more interesting to simultaneously show how we obtain the Schur's complement, $D - CA^{-1}B$. Hence, we use the quadratic form

$$\begin{bmatrix} a \\ b \end{bmatrix}^\top \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = a^\top Aa + b^\top Ca + a^\top Bb + b^\top Db,$$

which yields, after completing the square,

$$\begin{aligned}(a + A^{-1}C^\top b)^\top A(a + A^{-1}Bb) + b^\top(D - CA^{-1}B)b \\ = \begin{bmatrix} a \\ b \end{bmatrix}^\top \begin{bmatrix} I & O \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & O \\ O & D - CA^{-1}B \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ O & I \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix},\end{aligned} \tag{A.2}$$

where we can see that the second term on the l.h.s. of (A.2) is the Schur's complement, and we simultaneously proof (A.1a). The second equality (A.1b) follows the same approach. □

Lemma A.2 (Inversion of a block lower-triangular matrix)**.** *The lower-triangular, real-valued, block matrix $L \in \mathbb{R}^{n\times n}$ and its inverse L^{-1} are given as follows:*

$$L = \begin{bmatrix} I & O \\ L_{21} & I \end{bmatrix}, \qquad L^{-1} = \begin{bmatrix} I & O \\ -L_{21} & I \end{bmatrix},$$

where L_{21}, O, and I are repsectively arbitrary, zero, and unit diagonal matrices of appropriate dimensions.

Proof. The assertion leads directly from comparing the entries of $LL^{-1} = I_n$. □

Lemma A.3 (Inversion of a block matrix)**.** *Let us consider real-valued matrices A, B, C, and D belonging to the spaces $\mathbb{R}^{n\times n}$, $\mathbb{R}^{n\times m}$, $\mathbb{R}^{m\times n}$, and $\mathbb{R}^{\gamma\times\gamma}$, respectively; then, the inversion of the following matrix holds:*

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} I & -A^{-1}B \\ O & I \end{bmatrix}\begin{bmatrix} A^{-1} & O \\ O & (D-CA^{-1}B)^{-1} \end{bmatrix}\begin{bmatrix} I & O \\ -CA^{-1} & I \end{bmatrix} \tag{A.3a}$$

$$= \begin{bmatrix} A^{-1} + A^{-1}B(D-CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1} \\ -(D-CA^{-1}B)^{-1}CA^{-1} & (D-CA^{-1}B)^{-1} \end{bmatrix} \tag{A.3b}$$

$$= \begin{bmatrix} I & O \\ -D^{-1}C & I \end{bmatrix}\begin{bmatrix} (A-BD^{-1}C)^{-1} & O \\ O & D^{-1} \end{bmatrix}\begin{bmatrix} I & -BD^{-1} \\ O & I \end{bmatrix} \tag{A.3c}$$

$$= \begin{bmatrix} (A-BD^{-1}C)^{-1} & -(A-BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A-BD^{-1}C)^{-1} & D^{-1}C(A-BD^{-1}C)^{-1}BD^{-1} + D^{-1} \end{bmatrix}, \tag{A.3d}$$

if all the inversions are admissible.

Proof. To prove the assertion, we utilize Lemma A.1 and Lemma A.2. □

Lemma A.4 (The matrix inversion lemma)**.** *Let A, B, C, and D be real-valued matrices belonging to the spaces $\mathbb{R}^{n\times n}$, $\mathbb{R}^{n\times m}$, $\mathbb{R}^{m\times n}$, and $\mathbb{R}^{m\times m}$, respectively; then, it holds*

$$(A+BDC)^{-1} = A^{-1} - A^{-1}B(D^{-1} + CA^{-1}B)^{-1}CA^{-1}, \tag{A.4}$$

considering the required inversions exist.

Proof. The proof follows directly from comparing the first entry of (A.3b) and (A.3d) in Lemma A.3. □

Lemma A.5. *Let us consider a scalar-valued variable $v \in \mathbb{R}$ and the function $f(v) = v^n \exp\{-av^2\}$, where $a \in \mathbb{R}_{>0}$ and $n \in \mathbb{N}_{\geq 0}$; then, the integral $J = \int_{-\infty}^{\infty} v^n \exp\{-av^2\}dv$ yields*

(i) $J = 0$ *for odd n,*

(ii) $J = \Gamma\left(\frac{n+1}{2}\right)a^{-\frac{n+1}{2}}$ *for even n,*

(iii) $J = \pi^{\frac{1}{2}}a^{-\frac{1}{2}}$ *for $n = 0$.*

Proof. (i) For odd n, the function $f(v)$ is odd, that is, $f(-v) = -f(v)$, and we therefore have $J = \int_0^\infty f(v)dv + \int_{-\infty}^0 f(v)dv = \int_0^\infty f(v)dv - \int_0^\infty f(v)dv = 0$. (ii) For even n, we introduce the substitution $t = av^2$, from which it leads that $v = (t/a)^{\frac{1}{2}}$ and $dv = 1/2(at)^{-\frac{1}{2}}dt$. Hence, we can write $a^{-\frac{n+1}{2}} \int_0^\infty t^{\frac{n+1}{2}-1}\exp\{-t\}dt$, where we recognize the gamma function [1], $\Gamma(b) = \int_0^\infty t^{b-1}\exp\{-t\}dt$, with $b \in \mathbb{R}_{>0}$. □

Lemma A.6 (Multivariate gamma function)**.** *The real-valued multivariate gamma function is given by*

$$\Gamma_n(a) = \int_{X>0} |X|^{a-\frac{n+1}{2}} \exp\{-\mathrm{tr}(X)\} dv(X) = \pi^{n\frac{n-1}{4}} \prod_{i=1}^{n} \Gamma\Big(a - \frac{i-1}{2}\Big), \qquad \text{(A.5)}$$

where $a > \frac{n-1}{2}$. The integration is taken over the space of symmetric positive definite real-valued matrices $X \in \mathbb{R}^{n\times n}$, with $v(X)$ taking the unique entries of $\mathrm{vec}(X)$ *by removing all supradiagonal elements of X.*

Proof. To prove (A.5), we use the fact that the determinant and the trace of X can be computed according to

$$|X| = x_{11}|\bar{X}_{22}|, \qquad \text{(A.6a)}$$
$$\mathrm{tr}(X) = x_{11} + \mathrm{tr}(\bar{X}_{22}) + \mathrm{tr}(x_{21}x_{11}^{-1}x_{12}), \qquad \text{(A.6b)}$$

where $\bar{X}_{22} = X_{22} - x_{21}x_{11}^{-1}x_{12}$. These formulae can simply be obtained by decomposing X into blocks and applying (A.1a), that is,

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & X_{22} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}_{n-1}^\top \\ x_{21}x_{11}^{-1} & I_{n-1} \end{bmatrix} \begin{bmatrix} x_{11} & \mathbf{0}_{n-1}^\top \\ \mathbf{0}_{n-1} & X_{22} - x_{21}x_{11}^{-1}x_{12} \end{bmatrix} \begin{bmatrix} 1 & x_{11}^{-1}x_{12} \\ \mathbf{0}_{n-1} & I_{n-1} \end{bmatrix}.$$

Consequently, by substituting (A.6) for the respective terms in (A.5), we obtain

$$\Gamma_n(a) = \int_0^\infty x_{11}^{a-\frac{1}{2}(n+1)} \exp\{-x_{11}\} \prod_{i=1}^{n-1} \int \exp\{-x_{11}^{-1}x_{12,i}^2\} dx_{12,i} dx_{11} \\ \times \int_{\bar{X}_{22}>0} |\bar{X}_{22}|^{a-\frac{1}{2}(n+1)} \exp\{-\mathrm{tr}(\bar{X}_{22})\} dv(\bar{X}_{22}),$$

where $x_{12,i}$ denotes the ith entry of the vector x_{12}. The integral w.r.t. $x_{12,i}$ can be computed by utilizing Lemma A.5, the point (iii), with $a = x_{11}^{-1}$, which results in $\int \exp\{-a_{11}^{-1}x_{12,i}^2\} dx_{12,i} = (x_{11}\pi)^{\frac{1}{2}}$. Furthermore, we notice that the integral w.r.t. $\bar{X}_{22}$ is equivalent to $\Gamma_{n-1}\big(a - \frac{1}{2}\big)$. If we put this together, we can write

$$\Gamma_n(a) = \pi^{\frac{n-1}{2}} \int_0^\infty x_{11}^{a-1} \exp\{-x_{11}\} dx_{11} \Gamma_{n-1}\big(a - \tfrac{1}{2}\big) \\ = \pi^{\frac{n-1}{2}} \Gamma(a) \Gamma_{n-1}\big(a - \tfrac{1}{2}\big), \qquad \text{(A.7)}$$

where we use, similarly as in Lemma A.5, the definition of the Gamma integral. It is now obvious that (A.7) defines a recursive formula, which, after unwrapping, yields

$$\Gamma_n(a) = \pi^{\frac{n-1}{2}} \Gamma(a) \pi^{\frac{n-2}{2}} \Gamma\big(a - \tfrac{1}{2}\big) \cdots \pi^{\frac{n-n}{2}} \Gamma\big(a - \tfrac{n-1}{2}\big) \\ = \pi^{n\frac{n-1}{4}} \prod_{i=1}^{n} \Gamma\big(a - \tfrac{i-1}{2}\big). \qquad \text{(A.8)}$$

Here, we use simple identities $\sum_{i=1}^{n} \big(\frac{n-i}{2}\big) = n\frac{n-1}{4}$ and $\sum_{i=1}^{n} i = n\frac{n+1}{2}$. From (A.8), we the r.h.s. of (A.5), and the proof is thus concluded. □

A.2 Common Probability Density Functions

In this section, we prove various properties of the probability density functions that are commonly used in the present document. The proofs are carried out by means of the straightforward matrix and integral calculus, without the need to rely on philosophically deeper constructions, such as the moment generating functions [20].

Proposition A.1 (The matrix Gaussian probability density and its properties)**.** *A matrix-valued random variable $X \in \mathbb{R}^{n\times m}$ is Gaussian distributed if it follows the probability density function in the form*

$$\begin{aligned}\mathrm{vec}(X) &\sim \mathcal{N}(\mathrm{vec}(\hat{X}), Y \otimes Z)\\ &= \mathcal{I}^{-1}\exp\Big\{-\tfrac{1}{2}\Big(\mathrm{vec}(X)-\mathrm{vec}(\hat{X})\Big)^{\top}\Big(Y\otimes Z\Big)^{-1}\Big(\mathrm{vec}(X)-\mathrm{vec}(\hat{X})\Big)\Big\} &&\text{(A.9a)}\\ &= \mathcal{I}^{-1}\exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(Y^{-1}(X-\hat{X})^{\top}Z^{-1}(X-\hat{X})\Big)\Big\}, &&\text{(A.9b)}\end{aligned}$$

where $\hat{X} \in \mathbb{R}^{n\times m}$ labels the mean value matrix, $Y \in \mathbb{R}^{m\times m}$ and $Z \in \mathbb{R}^{n\times n}$ are symmetric, positive-definite matrices, $\otimes$ is the Kronecker product, and $\mathcal{I}$ denotes the proportionality constant. We consider that

a) the proportionality constant is $\mathcal{I} = (2\pi)^{\frac{nm}{2}}|Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}}$;
b) the first non-central moment is $\mathsf{E}(X) = \hat{X}$;
c) the second non-central moment is $\mathsf{E}(XHX^{\top}) = \mathrm{tr}(YH)Z + \hat{X}H\hat{X}^{\top}$, where $H \in \mathbb{R}^{m\times m}$ is an arbitrary matrix;
d) the second non-central moment is $\mathsf{E}(X^{\top}HX) = \mathrm{tr}(ZH)Y + \hat{X}^{\top}H\hat{X}$, where $H \in \mathbb{R}^{n\times n}$ is an arbitrary matrix.

Proof. We start by establishing tools needed to prove the above statements. The proofs are carried out in the sense of the second expression (A.9b). The equivalence between (A.9a) and (A.9b) can be established as

$$\begin{aligned}\mathrm{vec}(X-\hat{X})^{\top}(Y\otimes Z)^{-1}\mathrm{vec}(X-\hat{X}) &= \mathrm{vec}(X-\hat{X})^{\top}\mathrm{vec}(Z^{-1}(X-\hat{X})Y^{-1})\\ &= \mathrm{tr}\Big((X-\hat{X})^{\top}Z^{-1}(X-\hat{X})Y^{-1}\Big),\end{aligned}$$

where the identities $(A\otimes B)^{-1} = A^{-1}\otimes B^{-1}$, $(B^{\top}\otimes A)\mathrm{vec}(C) = \mathrm{vec}(ACB)$, and $\mathrm{vec}(A)^{\top}\mathrm{vec}(B) = \mathrm{tr}(A^{\top}B)$ are used. These three formulae are discussed and proven in [143]. The expression (A.9b) is not very convenient for performing the integration directly. Therefore, we need to resort to the change of variables formula. Thus, we search a proper transformation $X = F(U)$, where $F : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$, and the corresponding Jacobian determinant.

The matrices Y and Z can be decomposed into the product of their square root matrices $Y = (Y^{\frac{1}{2}})^\top Y^{\frac{1}{2}}$ and $Z = (Z^{\frac{1}{2}})^\top Z^{\frac{1}{2}}$, respectively, which allows us to write

$$\begin{aligned}&\mathrm{tr}\Big(Y^{-\frac{1}{2}}(Y^{-\frac{1}{2}})^\top (X-\hat{X})^\top Z^{-\frac{1}{2}}(Z^{-\frac{1}{2}})^\top (X-\hat{X})\Big) \\ &\qquad = \mathrm{tr}\Big[\Big((Z^{-\frac{1}{2}})^\top (X-\hat{X})Y^{-\frac{1}{2}}\Big)\Big((Z^{-\frac{1}{2}})^\top (X-\hat{X})Y^{-\frac{1}{2}}\Big)^\top\Big].\end{aligned}$$

We introduce the substitution

$$U = (Z^{-\frac{1}{2}})^\top (X-\hat{X})Y^{-\frac{1}{2}}, \tag{A.10}$$

from which we obtain the sought transformation

$$X = (Z^{\frac{1}{2}})^\top U Y^{\frac{1}{2}} + \hat{X}. \tag{A.11}$$

Taking differentials of the both sides of (A.11) yields

$$dX = (Z^{\frac{1}{2}})^\top (dU) Y^{\frac{1}{2}},$$

which further provides, after vectorizing,

$$\mathrm{vec}(dX) = \mathrm{vec}((Z^{\frac{1}{2}})^\top (dU) Y^{\frac{1}{2}}) = ((Y^{\frac{1}{2}})^\top \otimes (Z^{\frac{1}{2}})^\top)\mathrm{vec}(dU). \tag{A.12}$$

The Jacobian matrix is therefore given as

$$\frac{\partial\,\mathrm{vec}(F(U))}{\partial(\mathrm{vec}(U))^\top} = ((Y^{\frac{1}{2}})^\top \otimes (Z^{\frac{1}{2}})^\top);$$

accordingly, the Jacobian determinant satisfies

$$\left|\frac{\partial\,\mathrm{vec}(F(U))}{\partial(\mathrm{vec}(U))^\top}\right| = |Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}}, \tag{A.13}$$

where we use the formula $|A \otimes B| = |A|^b|B|^a$, with $A \in \mathbb{R}^{a\times a}$ and $B \in \mathbb{R}^{b\times b}$, see [143]. Now, with the above tools, let us consecutively prove a) − d).

a) From (A.9b) and the normalization property of probability density functions $\mathsf{E}(1) = 1$, we have

$$\mathcal{I} = \int \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(Y^{-1}(X-\hat{X})^\top Z^{-1}(X-\hat{X})\Big)\Big\}dX. \tag{A.14}$$

Using (A.10), (A.12), and (A.13), the integral (A.14) can be rewritten as

$$\mathcal{I} = \int \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(U^\top U\Big)\Big\}|Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}}d\mathrm{vec}(U)$$

and decomposed into the product of simpler integrals $\mathcal{I}_{ij}$ according to

$$\mathcal{I} = |Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}}\prod_{i=1}^{n}\prod_{j=1}^{m}\mathcal{I}_{ij} = |Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}}\prod_{i=1}^{n}\prod_{j=1}^{m}\int \exp\Big\{-\tfrac{1}{2}u_{ij}^2\Big\}du_{ij}, \tag{A.15}$$

where we utilize

$$\mathrm{tr}(U^\top U) = \mathrm{vec}(U)^\top \mathrm{vec}(U) = \sum_{i=1}^{n}\sum_{j=1}^{m} u_{ij}^2.$$

Consequently, using Lemma A.5, the point (iii), with $a = \frac{1}{2}$, in (A.15), gives $\mathcal{I} = (2\pi)^{\frac{nm}{2}}|Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}}$, which concludes the proof of part a).

b) The expected value of X w.r.t. (A.9b) can be written as

$$\mathsf{E}(X) = \mathcal{I}^{-1}\int X \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(Y^{-1}(X-\hat{X})^\top Z^{-1}(X-\hat{X})\Big)\Big\}dX.$$

If we apply (A.11) and (A.13) to the above formula, we obtain

$$\mathsf{E}(X) = (Z^{\frac{1}{2}})^\top J_1 Y^{\frac{1}{2}} + \hat{X}J_2,$$

where

$$J_1 = \mathcal{I}^{-1}\int U \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(U^\top U\Big)\Big\}|Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}}d\mathrm{vec}(U). \tag{A.16}$$

The integral (A.16) can be calculated entry-wise according to

$$J_{1,ij} = (2\pi)^{-\frac{nm}{2}}\int u_{ij}\exp\Big\{-\tfrac{1}{2}u_{ij}^2\Big\}du_{ij} \times \prod_{k=1,k\neq i}^{n}\prod_{l=1,l\neq j}^{m}\int \exp\Big\{-\tfrac{1}{2}u_{kl}^2\Big\}du_{kl} \tag{A.17}$$

After, employing the point (i) of Lemma A.5 in (A.17), we can see that $J_1 = O_{nm}$, where O_{nm} is the zero matrix of dimension $n \times m$. This result, and the fact that J_2 is simply $\mathsf{E}(1) = 1$, allows us to state that $\mathsf{E}(X) = \hat{X}$ and thus to conclude the proof of part b).

c) The expected value of $XHX^\top$, taken over (A.9b), is defined as

$$\mathsf{E}(XHX^\top) = \mathcal{I}^{-1}\int XHX^\top \times \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(Y^{-1}(X-\hat{X})^\top Z^{-1}(X-\hat{X})\Big)\Big\}dX,$$

which, after utilizing (A.11) and (A.13), leads to

$$\mathsf{E}(XHX^\top) = (Z^{\frac{1}{2}})^\top J_1 Z^{\frac{1}{2}} + (Z^{\frac{1}{2}})^\top J_2 Y^{\frac{1}{2}} H\hat{X}^\top + \hat{X}H(Y^{\frac{1}{2}})^\top J_2^\top Z^{\frac{1}{2}} + \hat{X}H\hat{X}^\top J_3,$$

where J_2 and J_3 are respectively equivalent to J_1 and J_2 of part b). Therefore, we will be concerned with computing only the first integral, which is given by

$$J_1 = \mathcal{I}^{-1}\int UAU^\top \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(U^\top U\Big)\Big\}|Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}}d\mathrm{vec}(U), \tag{A.18}$$

where $A = Y^{\frac{1}{2}}H(Y^{\frac{1}{2}})^\top$. To calculate (A.18), we resort to the entry-wise approach as before. For this reason, we write the entries of the product $UAU^\top$ according to

$$(UAU^\top)_{ij} = \sum_{k=1}^{n}\sum_{l=1}^{n} u_{il}a_{lk}u_{jk}. \tag{A.19}$$

Now, we need to distinguish between diagonal and non-diagonal entries. For $i \neq j$, the integral can be expressed as

$$\begin{aligned} J_{1,ij} = (2\pi)^{-\frac{nm}{2}} \sum_{k=1}^{n}\sum_{l=1}^{n} a_{lk} \int u_{il} \exp\Big\{-\tfrac{1}{2}u_{il}^2\Big\}du_{il} \\ \times \int u_{jk} \exp\Big\{-\tfrac{1}{2}u_{jk}^2\Big\}du_{jk} \\ \times \prod_{\substack{p=1,p\neq i\\ p\neq j}}^{n} \prod_{\substack{q=1,q\neq l\\ q\neq k}}^{m} \int \exp\Big\{-\tfrac{1}{2}u_{pq}^2\Big\}du_{pq}, \end{aligned}$$

where we recognize the integral given by the point (i) of Lemma A.5, with $a = \frac{1}{2}$, and we therefore obtain $J_{1,ij} = 0$. For $i = j$, the integral can be decomposed according to

$$\begin{aligned} J_{1,jj} = (2\pi)^{-\frac{nm}{2}} \sum_{k=1}^{n}\sum_{l=1,l\neq k}^{n} a_{lk} \int u_{jl} \exp\Big\{-\tfrac{1}{2}u_{jl}^2\Big\}du_{jl} \\ \times \int u_{jk} \exp\Big\{-\tfrac{1}{2}u_{jk}^2\Big\}du_{jk} \\ \times \prod_{p=1,p\neq j}^{n} \prod_{\substack{q=1,q\neq l\\ q\neq k}}^{m} \int \exp\Big\{-\tfrac{1}{2}u_{pq}^2\Big\}du_{pq} \\ + (2\pi)^{-\frac{nm}{2}} \sum_{r=1}^{n} a_{rr} \int u_{jr}^2 \exp\Big\{-\tfrac{1}{2}u_{jr}^2\Big\}du_{jr} \\ \times \prod_{p=1,p\neq j}^{n} \prod_{q=1,q\neq r}^{m} \int \exp\Big\{-\tfrac{1}{2}u_{pq}^2\Big\}du_{pq}, \end{aligned} \tag{A.20}$$

The first term of (A.20) leads, again, to the use of the point (i) of Lemma A.5 and is therefore equal to zero. However, the second term of (A.20) relies on the point (ii) of Lemma A.5. As both the integrals in the second term are equal to $(2\pi)^{\frac{1}{2}}$, we obtain, after canceling $(2\pi)^{-\frac{nm}{2}}$,

$$J_{1,jj} = \sum_{r=1}^{n} a_{rr}.$$

Now, recalling that $A = Y^{\frac{1}{2}}H(Y^{\frac{1}{2}})^\top$, we write

$$J_1 = \mathrm{tr}(YH)I_n$$

which is the resulting value of (A.18). Finally, gathering up the integrals leads to

$$\mathsf{E}(XHX^\top) = \operatorname{tr}(YH)Z + \hat{X}H\hat{X}^\top$$

and the conclusion of the proof for part c).

d) We begin by writing the expected value of the quadratic form $X^\top HX$ w.r.t. (A.9b) as

$$\begin{aligned}\mathsf{E}(X^\top HX) = \mathcal{I}^{-1}\int & X^\top HX \\ & \times \exp\Big\{-\tfrac{1}{2}\operatorname{tr}\Big(Y^{-1}(X-\hat{X})^\top Z^{-1}(X-\hat{X})\Big)\Big\}dX. \quad \text{(A.21)}\end{aligned}$$

Then, we proceed by using (A.11) and (A.13) in (A.21) to obtain

$$\begin{aligned}\mathsf{E}(X^\top HX) = (Y^{\frac{1}{2}})^\top J_1 Y^{\frac{1}{2}} + (Y^{\frac{1}{2}})^\top J_2^\top Z^{\frac{1}{2}} H\hat{X} \\ + \hat{X}^\top H(Z^{\frac{1}{2}})^\top J_2 Y^{\frac{1}{2}} + \hat{X}^\top H\hat{X} J_3,\end{aligned}$$

where J_2 and J_3 are the same as J_1 and J_2 of part b), respectively. The remaining integral is given by

$$J_1 = \mathcal{I}^{-1}\int U^\top AU \exp\Big\{-\tfrac{1}{2}\operatorname{tr}\Big(U^\top U\Big)\Big\}|Y|^{\frac{n}{2}}|Z|^{\frac{m}{2}} d\operatorname{vec}(U), \quad \text{(A.22)}$$

with $A = Z^{\frac{1}{2}}H(Z^{\frac{1}{2}})^\top$. The calculations are performed entry-wise as in the previous parts. A small difference consists in that we now have

$$(U^\top AU)_{ij} = \sum_{k=1}^{m}\sum_{l=1}^{m} u_{il}a_{lk}u_{jk},$$

instead of (A.19). The calculation of (A.22) is then carried out in the same way as in part c), which provides us with

$$\mathsf{E}(X^\top HX) = \operatorname{tr}(ZH)Y + \hat{X}^\top H\hat{X}$$

and concludes the proof.

□

Proposition A.2 (The inverse-Wishart probability density and its properties). *A symmetric, positive definite, matrix random variable $Z \in \mathbb{R}^{n\times n}$ is distributed according to inverse-Wishart density if it follows a probability density function in the form*

$$Z \sim i\mathcal{W}(\nu, \Lambda) = \mathcal{I}^{-1}|Z|^{-\frac{1}{2}(\nu+n+1)}\exp\Big\{-\tfrac{1}{2}\operatorname{tr}\Big(Z^{-1}\Lambda\Big)\Big\}, \quad \text{(A.23)}$$

where $\nu > n-1$ is the number of degrees of freedom, and $\Lambda \in \mathbb{R}^{n\times n}$ is the (symmetric positive definite) scale matrix. We assume the properties given as follows:

a) the proportionality constant is

$$\mathcal{I} = 2^{\frac{n\nu}{2}} |\Lambda|^{-\frac{\nu}{2}} \pi^{n\frac{n-1}{4}} \prod_{i=1}^{n} \Gamma\Big(\frac{\nu+1-i}{2}\Big),$$

b) the first non-central moment of Z^{-1} is $\mathsf{E}(Z^{-1}) = \nu\Lambda^{-1}$,

c) the first non-central moment of Z is $\mathsf{E}(Z) = \frac{\Lambda}{\nu-n-1}$

d) the first non-central moment of $\ln|Z|$ is

$$\mathsf{E}(\ln|Z|) = \ln|\Lambda| - n\ln 2 - \sum_{i=1}^{n} \Psi\Big(\frac{\nu+1-i}{2}\Big),$$

where $\psi(a) = \frac{d\Gamma(a)}{da}$ is the digamma function [1].

Proof. Let us begin by preparing tools that will be needed prove the above statements. Performing integration directly with (A.23) is inconvenient. We therefore need to find a transformation $Z = F(U)$ in order to ease the involved calculus. Let us consider $\Lambda = (\Lambda^{\frac{1}{2}})^\top \Lambda^{\frac{1}{2}}$; then, choosing the substitution given by

$$2U = (\Lambda^{\frac{1}{2}})^\top Z^{-1} \Lambda^{\frac{1}{2}},$$

leads to the required transformation

$$Z = (2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}) U^{-1} (2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}})^\top. \tag{A.24}$$

To obtain the Jacobian determinant, we differentiate both sides of (A.24), that is,

$$dZ = -(2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}U^{-1}) dU (2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}U^{-1})^\top. \tag{A.25}$$

We continue by applying the vectorization operator to (A.25),

$$\mathrm{vec}(dZ) = -(2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}U^{-1}) \otimes (2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}U^{-1}) \mathrm{vec}(dU). \tag{A.26}$$

However, since Z is a symmetric matrix, we need to take into account the repeating terms. Thus, we use $v(A)$ operator that extracts the unique terms of $\mathrm{vec}(A)$ by eliminating all supradiagonal elements of A, with $A \in \mathbb{R}^{a\times a}$ being a symmetric matrix [143]. The vectors $v(A)$ and $\mathrm{vec}(A)$ are related by the duplication matrix D_n according to $D_n v(A) = \mathrm{vec}(A)$, which, after applying to the both sides of (A.26), gives

$$dv(Z) = -D_n^+ (2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}U^{-1}) \otimes (2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}U^{-1}) D_n dv(U), \tag{A.27}$$

where D_n^+ is the Moore-Penrose inverse of D_n. The Jacobian determinant is then calculated by utilizing $|A \otimes B| = |A|^b |B|^a$, where $A \in \mathbb{R}^{a\times a}$ and $B \in \mathbb{R}^{b\times b}$, in (A.27), which yields

$$\left|\frac{\partial v(Z)}{\partial v(U)^\top}\right| = (-1)^{\frac{1}{2}n(n+1)} |2^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}U^{-1}|^{n+1}.$$

The absolute value of this determinant—needed to carry out the change of variables—is given by

$$\left\| \frac{\partial v(Z)}{\partial v(U)^\top} \right\| = 2^{-\frac{1}{2}n(n+1)} |\Lambda|^{\frac{1}{2}(n+1)} |U|^{-(n+1)}. \tag{A.28}$$

In the remaining parts of the proof, there will be the need to reshape the entries of U according to

$$U^{ii} = P_{\sigma_i} U P_{\sigma_i}^\top = \begin{bmatrix} u_{ii} & u_{\text{-}ii} \\ u_{\text{-}ii}^\top & U_{\text{-}ii} \end{bmatrix}, \tag{A.29}$$

where $P_{\sigma_i} = (e_{\sigma_i(1)}^\top, \dots, e_{\sigma_i(n)}^\top)^\top$ is the permutation matrix with a particular choice of the permutation $\sigma_i : \{1, 2, \dots, i, \dots, n\} \to \{i, 2, \dots, 1, \dots, n\}$, and e_j is the vector with one at the jth entry and zeros otherwise. Thus, by writing $\sigma_i(j)$, we select jth entry of the vector σ_i after permuting the first and ith entry. The entries of (A.29) are described as follows: u_{ii} is the diagonal entry we intend to have at the first position, $u_{\text{-}ii}$ is the row vector with u_{ii} being excluded, and $U_{\text{-}ii}$ is the matrix containing the rest of the entries after the exchange of the ith row and the ith column. The determinant of U^{ii} is invariant under the above permutation.

Similarly to (A.6), the determinant and the trace of (A.29), are calculated with using (A.1a) of Lemma A.1 as

$$|U^{ii}| = u_{ii} |\bar{U}_{\text{-}ii}|, \tag{A.30a}$$

$$\operatorname{tr}(U^{ii}) = u_{ii} + \operatorname{tr}(\bar{U}_{\text{-}ii}) + \operatorname{tr}(u_{\text{-}ii}^\top u_{ii}^{-1} u_{\text{-}ii}), \tag{A.30b}$$

where $\bar{U}_{\text{-}ii} = U_{\text{-}ii} - u_{\text{-}ii}^\top u_{ii}^{-1} u_{\text{-}ii}$.

a) We use the fact that integrating (A.23) leads to $\mathsf{E}(1) = 1$, which allows us to write

$$\mathcal{I} = \int_{Z>0} |Z|^{-\frac{1}{2}(\nu+n+1)} \exp\Big\{ -\tfrac{1}{2}\operatorname{tr}\Big(Z^{-1}\Lambda\Big)\Big\} dv(Z). \tag{A.31}$$

To obtain a more convenient form for the integration, we apply the change of variables with previously prepared transformation (A.24) and the absolute value of the associated Jacobian determinant (A.28), which gives

$$\mathcal{I} = 2^{\frac{n\nu}{2}} |\Lambda|^{-\frac{\nu}{2}} \int_{Z>0} |U|^{\frac{1}{2}(\nu-n-1)} \exp\Big\{ -\operatorname{tr} U\Big)\Big\} dv(U),$$

where we recognize the integral that is equivalent to the multivariate Gamma function, and we therefore simply use Lemma A.6 to obtain the result

$$\mathcal{I} = 2^{\frac{n\nu}{2}} |\Lambda|^{-\frac{\nu}{2}} \Gamma_n(0.5\nu).$$

The proof of part a) is here concluded.

b) The expected value of Z^{-1} taken w.r.t. (A.23) yields

$$\mathsf{E}(Z^{-1}) = \mathcal{I}^{-1} \int_{Z>0} Z^{-1}|Z|^{-\frac{1}{2}(\nu+n+1)} \exp\Big\{ -\tfrac{1}{2}\mathrm{tr}\Big(Z^{-1}\Lambda\Big)\Big\} dv(Z),$$

which, similarly as in part a), after using the change of variables with (A.24) and (A.28), provides us with

$$\mathsf{E}(Z^{-1}) = 2\Gamma_n(0.5\nu)^{-1}(\Lambda^{-\frac{1}{2}})^\top J(\Lambda^{-\frac{1}{2}}),$$

where

$$J = \int_{U>0} U|U|^{\frac{1}{2}(\nu-n-1)} \exp\{-\mathrm{tr}U\} dv(U). \tag{A.32}$$

We approach the calculation of (A.32) in the entry-wise manner. Thus, for the diagonal entries u_{ii}, the integral (A.32) can be decomposed as

$$\begin{aligned} J_{ii} = \int_0^\infty u_{ii}^{\frac{1}{2}(\nu-n+1)} \exp\{-u_{ii}\} \prod_{j=1}^{n-1} \int \exp\{-u_{ii}^{-1}u_{\text{-}ii,j}^2\} du_{\text{-}ii,j} du_{ii} \\ \times \int_{\bar{U}_{\text{-}ii}>0} |\bar{U}_{\text{-}ii}|^{\frac{1}{2}(\nu-n-1)} \exp\{-\mathrm{tr}(\bar{U}_{\text{-}ii})\} dv(\bar{U}_{\text{-}ii}), \end{aligned} \tag{A.33}$$

where we use (A.30), with $u_{\text{-}ii,j}$ denoting the jth entry of the vector $u_{\text{-}ii}$. We continue by applying the point (iii) of Lemma A.5 to compute the inner integral in (A.33), $\int \exp\{-u_{ii}^{-1}u_{\text{-}ii,j}^2\}dx_{\text{-}ii,j} = (u_{ii}\pi)^{\frac{1}{2}}$, and Lemma A.6 to obtain the last integral in (A.33), which results in

$$\begin{aligned} J_{ii} &= \pi^{\frac{n-1}{2}} \int_0^\infty u_{ii}^{\frac{\nu}{2}} \exp\{-u_{ii}\} du_{ii} \Gamma_{n-1}\Big(\tfrac{\nu-1}{2}\Big) \\ &= \pi^{\frac{n-1}{2}} \Gamma\Big(\tfrac{\nu}{2}+1\Big)\Gamma_{n-1}\Big(\tfrac{\nu-1}{2}\Big) \\ &= \tfrac{\nu}{2}\Gamma_n\Big(\tfrac{\nu}{2}\Big), \end{aligned}$$

where Lemma A.5 is applied once more in the first line. For the non-diagonal entries $u_{\text{-}ii,j}$, we have

$$\begin{aligned} J_{\text{-}ii,j} = \int_0^\infty u_{ii}^{\frac{1}{2}(\nu-n-1)} \exp\{-u_{ii}\} \int u_{\text{-}ii,j} \exp\{-u_{ii}^{-1}u_{\text{-}ii,j}^2\} du_{\text{-}ii,j} \\ \times \prod_{k=1,k\neq j}^{n-1} \int \exp\{-u_{ii}^{-1}u_{\text{-}ii,k}^2\} du_{\text{-}ii,k} du_{ii} \\ \times \int_{\bar{U}_{\text{-}ii}>0} |\bar{U}_{\text{-}ii}|^{\frac{1}{2}(\nu-n-1)} \exp\{-\mathrm{tr}(\bar{U}_{\text{-}ii})\} dv(\bar{U}_{\text{-}ii}) = 0, \end{aligned} \tag{A.34}$$

where according to the point (iii) Lemma A.5, the integral in the second line of (A.34), $\int u_{\text{-}ii,j} \exp\{-u_{ii}^{-1}u_{\text{-}ii,j}^2\}du_{\text{-}ii,j}$, is equal to zero, making the non-diagonal entries zero. Gathering the results of the particular entries, we obtain

$$J = 0.5\nu\Gamma_n(0.5\nu)I_n,$$

which concludes the proof of part b).

c) The proof is left as an exercise to the reader.

d) The expected value of $\ln|Z|$ w.r.t. (A.23) is

$$\mathsf{E}(\ln|Z|) = \mathcal{I}^{-1}\int_{Z>0} \ln|Z||Z|^{-\frac{1}{2}(\nu+n+1)} \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(Z^{-1}\Lambda\Big)\Big\}dv(Z). \quad (A.35)$$

Here, we do not have to proceed by a complete integration as in the previous cases. We only need to realize that $a^x \ln a = \frac{da^x}{dx}$, which yields, after using in (A.35),

$$\mathsf{E}(\ln|Z|) = \mathcal{I}^{-1}\int_{Z>0} -2\frac{d}{d\nu}|Z|^{-\frac{1}{2}(\nu+n+1)} \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big(Z^{-1}\Lambda\Big)\Big\}dv(Z).$$

By utilizing (A.31), and simply interchanging the order of the integration and derivation, we can write

$$\mathsf{E}(\ln|Z|) = -2\mathcal{I}^{-1}\frac{d}{d\nu}\mathcal{I},$$

which yields, when using another simple identity $\frac{1}{f(x)}\frac{df(x)}{dx} = \frac{d\ln f(x)}{dx}$,

$$\mathsf{E}(\ln|Z|) = -2\frac{d}{d\nu}\ln\mathcal{I}.$$

Subsequently, from part a), we have

$$\begin{aligned}\frac{d}{d\nu}\ln\mathcal{I} &= \frac{1}{2}x\ln 2 - \frac{1}{2}\ln|\Lambda| + \sum_{i=1}^{x}\frac{d\Big(\frac{\nu+1-i}{2}\Big)}{d\nu}\frac{d}{d\Big(\frac{\nu+1-i}{2}\Big)}\Gamma\Big(\frac{\nu+1-i}{2}\Big)\\ &= \frac{1}{2}x\ln 2 - \frac{1}{2}\ln|\Lambda| + \frac{1}{2}\sum_{i=1}^{x}\Psi\Big(\frac{\nu+1-i}{2}\Big),\end{aligned}$$

which concludes the proof.

□

Lemma A.7 (The Gauss-inverse-Wishart and exponential family densities)**.** *The Gauss-inverse-Wishart density*

$$\begin{aligned}p(\mu,\Sigma) = \pi^{-\frac{n}{2}}2^{-\frac{n(\nu_\alpha-n-1)}{2}}|\Lambda_\alpha|^{\frac{\nu_\alpha-n-2}{2}}\Gamma_n\Big(\frac{\nu_\alpha-n-2}{2}\Big)^{-1}\Sigma_\alpha^{-\frac{n}{2}}|\Sigma|^{-\frac{\nu_\alpha}{2}}\\ \times\exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big[\Sigma^{-1}\Big((\mu-\mu_\alpha)\Sigma_\alpha^{-1}(\mu-\mu_\alpha)^\top+\Lambda_\alpha\Big)\Big]\Big\}\end{aligned}$$

is related to the exponential family (prior) density

$$p(\theta|\nu_\alpha, V_\alpha) = \exp\{\langle\eta(\theta), V_\alpha\rangle - \nu_\alpha\zeta(\theta) - \log\mathcal{I}(\nu_\alpha, V_\alpha)\}$$

through

$$\nu_\alpha = \nu_\alpha, \qquad \zeta(\theta) = \frac{1}{2}\log|\Sigma|,$$

$$V_\alpha = \begin{bmatrix}\Lambda_\alpha+\mu_\alpha\Sigma_\alpha^{-1}\mu_\alpha^\top & \mu_\alpha\Sigma_\alpha^{-1}\\ \Sigma_\alpha^{-1}\mu_\alpha^\top & \Sigma_\alpha^{-1}\end{bmatrix}, \qquad \eta(\theta) = \begin{bmatrix}\Sigma^{-1} & \Sigma^{-1}\mu\\ \mu^\top\Sigma^{-1} & \mu^\top\Sigma^{-1}\mu\end{bmatrix},$$

$$\mathcal{I}(\nu_\alpha, V_\alpha) = \pi^{\frac{n}{2}}2^{\frac{n(\nu_\alpha-n-1)}{2}}|\Lambda_\alpha|^{-\frac{\nu_\alpha-n-2}{2}}\Gamma_n\Big(\frac{\nu_\alpha-n-2}{2}\Big)\Sigma_\alpha^{\frac{n}{2}}.$$

Proof. The result follows from isolating a simple identity in the exponent of the Gauss-inverse-Wishart density according to

$$\begin{aligned}&(\mu-\mu_\alpha)\Sigma_\alpha^{-1}(\mu-\mu_\alpha)^\top+\Lambda_\alpha\\&\qquad=\begin{bmatrix}I_n\\-\mu^\top\end{bmatrix}^\top\begin{bmatrix}\mu_\alpha\Sigma_\alpha^{-1}\mu_\alpha^\top & \mu_\alpha\Sigma_\alpha^{-1}\\ \Sigma_\alpha^{-1}\mu_\alpha^\top & \Sigma_\alpha^{-1}\end{bmatrix}\begin{bmatrix}I_n\\-\mu^\top\end{bmatrix}+\begin{bmatrix}I_n\\-\mu^\top\end{bmatrix}^\top\begin{bmatrix}\Lambda_\alpha & \mathbf{0}\\ \mathbf{0}^\top & 0\end{bmatrix}\begin{bmatrix}I_n\\-\mu^\top\end{bmatrix}.\end{aligned}$$

□

A.3 Joint, Conditional, and Marginal Densities

Lemma A.8 (Gaussian conditional and marginal densities). *Let us consider the Gaussian probability density function $\mathcal{N}(x;\mu,\Sigma)$, where $x\in\mathbb{R}^n$, and μ and Σ denote the mean vector and covariance matrix, respectively. If $\mathcal{N}(x;\mu,\Sigma)$ is the joint density that can be partitioned according to*

$$p(x_a,x_b)=\mathcal{N}\left(\begin{bmatrix}x_a\\x_b\end{bmatrix};\begin{bmatrix}\mu_a\\\mu_b\end{bmatrix},\begin{bmatrix}\Sigma_{aa}&\Sigma_{ab}\\\Sigma_{ba}&\Sigma_{bb}\end{bmatrix}\right),\tag{A.36}$$

then the conditional and marginal densities are

$$p(x_a|x_b)=\mathcal{N}(x_a;\mu_{a|b},\Sigma_{a|b}),\tag{A.37}$$

$$p(x_b)=\mathcal{N}(x_b;\mu_b,\Sigma_b),\tag{A.38}$$

where

$$\mu_{a|b}=\mu_a+\Sigma_{ab}\Sigma_{bb}^{-1}(x_b-\mu_b),\tag{A.39a}$$

$$\Sigma_{a|b}=\Sigma_{aa}-\Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}.\tag{A.39b}$$

Proof. The density (A.36) reads

$$\begin{aligned}p(x_a,x_b)&=(2\pi)^{-\frac{n}{2}}\begin{vmatrix}\Sigma_{aa}&\Sigma_{ab}\\\Sigma_{ba}&\Sigma_{bb}\end{vmatrix}^{-\frac{1}{2}}\\&\qquad\times\exp\left\{-\frac{1}{2}\begin{bmatrix}x_a-\mu_a\\x_b-\mu_b\end{bmatrix}^\top\begin{bmatrix}\Sigma_{aa}&\Sigma_{ab}\\\Sigma_{ba}&\Sigma_{bb}\end{bmatrix}^{-1}\begin{bmatrix}x_a-\mu_a\\x_b-\mu_b\end{bmatrix}\right\}.\end{aligned}\tag{A.40}$$

After utilizing (A.3c) and $n=n_a+n_b$ in (A.40), we obtain

$$\begin{aligned}p(x_a,x_b)&=(2\pi)^{-\frac{n_a}{2}}|\Sigma_{aa}-\Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}|^{-\frac{1}{2}}\\&\quad\times\exp\{-\tfrac{1}{2}(x_a-(\mu_a+\Sigma_{ab}\Sigma_{bb}^{-1}(x_b-\mu_b)))^\top(\Sigma_{aa}-\Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1}\\&\qquad\times(x_a-(\mu_a+\Sigma_{ab}\Sigma_{bb}^{-1}(x_b-\mu_b)))\}\\&\qquad\qquad\times(2\pi)^{-\frac{n_b}{2}}|\Sigma_{bb}|^{-\frac{1}{2}}\exp\{-\tfrac{1}{2}(x_b-\mu_b)^\top\Sigma_{bb}^{-1}(x_b-\mu_b)\},\end{aligned}$$

which yields

$$\begin{aligned}p(x_a, x_b) = (2\pi)^{-\frac{n_a}{2}} |\Sigma_{a|b}|^{-\frac{1}{2}} \exp\{-\tfrac{1}{2}(x_a - \mu_{a|b})^\top \Sigma_{a|b}^{-1}(x_a - \mu_{a|b})\} \\ \times (2\pi)^{-\frac{n_b}{2}} |\Sigma_{bb}|^{-\frac{1}{2}} \exp\{-\tfrac{1}{2}(x_b - \mu_b)^\top \Sigma_{bb}^{-1}(x_b - \mu_b)\} \\ = \mathcal{N}(x_a; \mu_{a|b}, \Sigma_{a|b})\mathcal{N}(x_b; \mu_b, \Sigma_b),\end{aligned}$$

where we recognize the desired conditional (A.37) and marginal (A.38) densities. □

Lemma A.9 (Student's t conditional and marginal densities)**.** *Let us consider the Student's t probability density function* $\mathrm{St}(x; \mu, \Sigma, \nu)$*, where* $x \in \mathbb{R}^n$*, and* μ*,* Σ*, and* ν *denote the mean vector, scale matrix, and the number of degrees of freedom, respectively. If* $\mathrm{St}(x; \mu, \Sigma, \nu)$ *is the joint density that can be partitioned according to*

$$p(x_a, x_b) = \mathrm{St}\left(\begin{bmatrix} x_a \\ x_b \end{bmatrix}; \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}, \nu\right), \tag{A.41}$$

then the conditional and marginal densities are given by

$$p(x_a|x_b) = \mathrm{St}(x_a; \mu_{a|b}, \Sigma_{a|b}, \nu_{a|b}), \tag{A.42}$$

$$p(x_b) = \mathrm{St}(x_b; \mu_b, \Sigma_b, \nu), \tag{A.43}$$

where

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b), \tag{A.44a}$$

$$\Sigma_{a|b} = \frac{\nu + (x_b - \mu_b)^\top \Sigma_{bb}^{-1}(x_b - \mu_b)}{\nu + n_b}(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}), \tag{A.44b}$$

$$\nu_{a|b} = \nu + n_b. \tag{A.44c}$$

Proof. The density (A.41) is written as

$$\begin{aligned}p(x_a, x_b) = \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}(\nu\pi)^{-\frac{n}{2}} \begin{vmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{vmatrix}^{-\frac{1}{2}} \\ \times \left(1 + \frac{1}{\nu}\begin{bmatrix} x_a - \mu_a \\ x_b - \mu_b \end{bmatrix}^\top \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}^{-1} \begin{bmatrix} x_a - \mu_a \\ x_b - \mu_b \end{bmatrix}\right)^{-\frac{\nu+n}{2}}.\end{aligned} \tag{A.45}$$

After applying (A.3c) in (A.45), we get

$$\begin{aligned}p(x_a, x_b) = \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}(\nu\pi)^{-\frac{n}{2}} |\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}|^{-\frac{1}{2}} |\Sigma_{bb}|^{-\frac{1}{2}} \\ \times \Bigg(1 + \frac{1}{\nu}(x_a - (\mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b)))^\top (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1} \\ \times (x_a - (\mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b))) \\ + \frac{1}{\nu}(x_b - \mu_b)^\top \Sigma_{bb}^{-1}(x_b - \mu_b)\Bigg)^{-\frac{\nu+n}{2}},\end{aligned} \tag{A.46}$$

which directly reveals (A.44a). To make the subsequent algebra more compact, we rewrite (A.46) according to

$$p(x_a, x_b) = \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} (\nu\pi)^{-\frac{n}{2}} |\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}|^{-\frac{1}{2}} |\Sigma_{bb}|^{-\frac{1}{2}} \left(1 + \frac{a}{\nu} + \frac{b}{\nu}\right)^{-\frac{\nu+n}{2}}, \quad \text{(A.47)}$$

and we introduce the intermediate quantities

$$a = (x_a - \mu_{a|b})^\top (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1} (x_a - \mu_{a|b}), \quad \text{(A.48a)}$$

$$b = (x_b - \mu_b)^\top \Sigma_{bb}^{-1} (x_b - \mu_b). \quad \text{(A.48b)}$$

Let us consider

$$\left(1 + \frac{a}{\nu} + \frac{b}{\nu}\right) = \left(1 + \frac{a(\nu+b)}{\nu(\nu+b)} + \frac{b}{\nu}\right) = \left(1 + \frac{a}{\nu+b}\right)\left(1 + \frac{b}{\nu}\right),$$

which, after substituting in (A.47) and applying $n = n_a + n_b$, leads to

$$p(x_a, x_b) = \frac{\Gamma\left(\frac{\nu+n_a+n_b}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} (\nu\pi)^{-\frac{n_a}{2}} \left(1 + \frac{b}{\nu}\right)^{-\frac{n_a}{2}} |\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}|^{-\frac{1}{2}}$$
$$\times \left(1 + \frac{a}{\nu+b}\right)^{-\frac{\nu+n_a+n_b}{2}} (\nu\pi)^{-\frac{n_b}{2}} |\Sigma_{bb}|^{-\frac{1}{2}} \left(1 + \frac{b}{\nu}\right)^{-\frac{\nu+n_b}{2}}. \quad \text{(A.49)}$$

Furthermore, we extend (A.49) as

$$p(x_a, x_b) = \frac{\Gamma\left(\frac{\nu+n_a+n_b}{2}\right)}{\Gamma\left(\frac{\nu+n_b}{2}\right)} (\nu\pi)^{-\frac{n_a}{2}} \frac{(\nu+n_b)^{-\frac{n_a}{2}}}{(\nu+n_b)^{-\frac{n_a}{2}}} \left(\frac{\nu}{\nu} + \frac{b}{\nu}\right)^{-\frac{n_a}{2}}$$
$$\times |\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}|^{-\frac{1}{2}} \left(1 + \frac{a}{\frac{(\nu+n_b)}{(\nu+n_b)}(\nu+b)}\right)^{-\frac{\nu+n_a+n_b}{2}}$$
$$\times \frac{\Gamma\left(\frac{\nu+n_b}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} (\nu\pi)^{-\frac{n_b}{2}} |\Sigma_{bb}|^{-\frac{1}{2}} \left(1 + \frac{b}{\nu}\right)^{-\frac{\nu+n_b}{2}}. \quad \text{(A.50)}$$

When substituting (A.48) in (A.50) and rearranging the red terms, we can write

$$p(x_a, x_b) = \frac{\Gamma\left(\frac{\nu+n_a+n_b}{2}\right)}{\Gamma\left(\frac{\nu+n_b}{2}\right)} \left((\nu+n_b)\pi\right)^{-\frac{n_a}{2}} \left|\frac{\nu+(x_b-\mu_b)^\top\Sigma_{bb}^{-1}(x_b-\mu_b)}{\nu+n_b}(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})\right|^{-\frac{1}{2}}$$
$$\times \left(1 + \frac{1}{\nu+n_b}(x_a - \mu_{a|b})^\top\right.$$
$$\left.\times \left(\frac{\nu+(x_b-\mu_b)^\top\Sigma_{bb}^{-1}(x_b-\mu_b)}{\nu+n_b}(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})\right)^{-1}(x_a - \mu_{a|b})\right)^{-\frac{\nu+n_a+n_b}{2}}$$
$$\times \frac{\Gamma\left(\frac{\nu+n_b}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} (\nu\pi)^{-\frac{n_b}{2}} |\Sigma_{bb}|^{-\frac{1}{2}} \left(1 + \frac{(x_b-\mu_b)^\top\Sigma_{bb}^{-1}(x_b-\mu_b)}{\nu}\right)^{-\frac{\nu+n_b}{2}},$$

where we use $c^{n_a}|A| = |cA|$ for $A \in \mathbb{R}^{n_a \times n_a}$ and $c \in \mathbb{R}$. Identifying the remaining statistics (A.44b) and (A.44c) provides us with

$$\begin{aligned}
p(x_a, x_b) &= \frac{\Gamma\left(\frac{\nu_{a|b}+n_a}{2}\right)}{\Gamma\left(\frac{\nu_{a|b}}{2}\right)} \left(\nu_{a|b}\pi\right)^{-\frac{n_a}{2}} |\Sigma_{a|b}|^{-\frac{1}{2}} \left(1 + \frac{\left(x_a - \mu_{a|b}\right)^\top \Sigma_{a|b}^{-1}(x_a - \mu_{a|b})}{\nu_{a|b}}\right)^{-\frac{\nu_{a|b}+n_a}{2}} \\
&\quad \times \frac{\Gamma\left(\frac{\nu+n_b}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} (\nu\pi)^{-\frac{n_b}{2}} |\Sigma_{bb}|^{-\frac{1}{2}} \left(1 + \frac{\left(x_b - \mu_b\right)^\top \Sigma_{bb}^{-1}(x_b - \mu_b)}{\nu}\right)^{-\frac{\nu+n_b}{2}} \\
&= \mathrm{St}(x_a; \mu_{a|b}, \Sigma_{a|b}, \nu_{a|b}) \mathrm{St}(x_b; \mu_b, \Sigma_b, \nu),
\end{aligned}$$

which shows the desired conditional (A.42) and marginal (A.43) densities and concludes the proof. □

Corollary A.1 (Uncorrelated Student's t random variables)**.** *Let us assume that Σ_{ab} and Σ_{ba} are zero matrices in (A.41), then the random variables x_a and x_b are uncorrelated and distributed according to*

$$p(x_a|x_b) = \mathrm{St}(x_a; \mu_{a|b}, \Sigma_{a|b}, \nu_{a|b}), \tag{A.51}$$

$$p(x_b) = \mathrm{St}(x_b; \mu_b, \Sigma_b, \nu), \tag{A.52}$$

where

$$\begin{aligned}
\mu_{a|b} &= \mu_a, \\
\Sigma_{a|b} &= \frac{\nu + \left(x_b - \mu_b\right)^\top \Sigma_{bb}^{-1}(x_b - \mu_b)}{\nu + n_b} \Sigma_{aa}, \\
\nu_{a|b} &= \nu + n_b.
\end{aligned}$$

Remark A.1. *In Corollary A.1, we see that having uncorrelated Student's t random variables does not imply their independence (as in the case of Gaussian random variables). The Student's t random variables can only become independent if $\nu \to \infty$.*

Lemma A.10 (The Gauss-inverse-Wishart density and its conditional and marginal densities)**.** *Let us consider the Gaussian probability density function $\mathcal{N}(x; \mu, \Sigma)$, where $x \in \mathbb{R}^n$, with μ and Σ denoting the mean vector and covariance matrix, respectively. Furthermore, let $\mathcal{N}i\mathcal{W}(\mu, \Sigma; \mu_\beta, \Sigma_\beta, \nu_\beta, \Lambda_\beta)$ be the Gauss-inverse-Wishart density, where μ_β is the least-square estimate of μ, Σ_β is the variance of the least-square estimate, ν_β is the number of degrees of freedom, and Λ_β is the least-square reminder. Then, the joint density*

$$p(\mu, \Sigma, x) = \mathcal{N}(x; \mu, \Sigma)\mathcal{N}i\mathcal{W}(\mu, \Sigma; \mu_\beta, \Sigma_\beta, \nu_\beta, \Lambda_\beta) \tag{A.53}$$

admits the conditional density of the parameters (μ, Σ) and the marginal density of the data x given by

$$p(\mu, \Sigma|x) = \mathcal{N}i\mathcal{W}(\mu, \Sigma; \mu_\alpha, \Sigma_\alpha, \nu_\alpha, \Lambda_\alpha), \tag{A.54}$$

$$p(x) = \mathrm{St}(x; \mu_\beta, \bar{\Sigma}, \bar{\nu}), \tag{A.55}$$

where

$$\mu_\alpha = \mu_\beta + \Sigma_\alpha(x - \mu_\beta), \tag{A.56a}$$
$$\Sigma_\alpha = \frac{\Sigma_\beta}{1+\Sigma_\beta}, \tag{A.56b}$$
$$\nu_\alpha = \nu_\beta + 1, \tag{A.56c}$$
$$\Lambda_\alpha = \Lambda_\beta + \frac{1}{1+\Sigma_\beta}(x-\mu_\beta)(x-\mu_\beta)^\top, \tag{A.56d}$$

and

$$\bar{\Sigma} = \frac{1+\Sigma_\beta}{\nu_\beta - n + 1}\Lambda_\beta, \tag{A.57a}$$
$$\bar{\nu} = \nu_\beta - n + 1. \tag{A.57b}$$

Proof. Let us start the proof by expressing the densities in (A.53) as

$$p(x|\mu,\Sigma) = (2\pi)^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}}\exp\Big\{-\tfrac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\Big\} \tag{A.58}$$

and

$$p(\mu,\Sigma) = \pi^{-\frac{n}{2}}2^{-\frac{n(\nu_\beta+1)}{2}}|\Lambda_\beta|^{\frac{\nu_\beta}{2}}\Gamma_n(0.5\nu_\beta)^{-1}\Sigma_\beta^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}(\nu_\beta+n+2)} \times \exp\Big\{-\tfrac{1}{2}\mathrm{tr}\Big[\Sigma^{-1}\Big((\mu-\mu_\beta)\Sigma_\beta^{-1}(\mu-\mu_\beta)^\top + \Lambda_\beta\Big)\Big]\Big\}. \tag{A.59}$$

We are first concerned with how to combine the exponents of (A.58) and (A.59). The remaining elements of these densities are treated later. The exponent of (A.58) can be rewritten as

$$p(x|\mu,\Sigma) \propto \exp\left\{-\frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\begin{bmatrix} I_n \\ -\mu^\top\end{bmatrix}^\top\begin{bmatrix} x \\ 1\end{bmatrix}\begin{bmatrix} x \\ 1\end{bmatrix}^\top\begin{bmatrix} I_n \\ -\mu^\top\end{bmatrix}\right)\right\}, \tag{A.60}$$

and, similarly as in Lemma A.7, the exponent of (A.59) can be rearranged as

$$p(\mu,\Sigma) \propto \exp\left\{-\frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\begin{bmatrix} I_n \\ -\mu^\top\end{bmatrix}^\top\begin{bmatrix} V_\beta^{xx} & V_\beta^{x1} \\ V_\beta^{1x} & V_\beta^{11}\end{bmatrix}\begin{bmatrix} I_n \\ -\mu^\top\end{bmatrix}\right)\right\}. \tag{A.61}$$

Now, the exponent of the joint density $p(\mu,\Sigma,x)$ is obtained by multiplying (A.60) and (A.61), that is,

$$p(\mu,\Sigma,x) \propto \exp\left\{-\frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\begin{bmatrix} I_n \\ -\mu^\top\end{bmatrix}^\top\begin{bmatrix} V_\alpha^{xx} & V_\alpha^{x1} \\ V_\alpha^{1x} & V_\alpha^{11}\end{bmatrix}\begin{bmatrix} I_n \\ -\mu^\top\end{bmatrix}\right)\right\}, \tag{A.62}$$

where

$$\begin{bmatrix} V_\alpha^{xx} & V_\alpha^{x1} \\ V_\alpha^{1x} & V_\alpha^{11}\end{bmatrix} = \begin{bmatrix} V_\beta^{xx} & V_\beta^{x1} \\ V_\beta^{1x} & V_\beta^{11}\end{bmatrix} + \begin{bmatrix} xx^\top & x \\ x^\top & 1\end{bmatrix}. \tag{A.63}$$

By applying Lemma A.1, the quadratic form in (A.62) yields

$$\begin{bmatrix} I_n \\ -\mu^\top \end{bmatrix}^\top \begin{bmatrix} V_\alpha^{xx} & V_\alpha^{x1} \\ V_\alpha^{1x} & V_\alpha^{11} \end{bmatrix} \begin{bmatrix} I_n \\ -\mu^\top \end{bmatrix} = (\mu - \mu_\alpha)^\top \Sigma_\alpha^{-1} (\mu - \mu_\alpha) + \Lambda_\alpha, \tag{A.64}$$

defining the updated statistics

$$\Sigma_\alpha = (V_\alpha^{11})^{-1}, \tag{A.65a}$$
$$\mu_\alpha = V_\alpha^{x1} (V_\alpha^{11})^{-1}, \tag{A.65b}$$
$$\Lambda_\alpha = V_\alpha^{xx} - V_\alpha^{x1} (V_\alpha^{11})^{-1} V_\alpha^{1x}. \tag{A.65c}$$

Expressing the exponent (A.62) by means of the r.h.s. of (A.64) and rearranging the remaining (non-exponential) elements of (A.58) and (A.59) leads to the joint density in the form

$$\begin{aligned} p(\mu, \Sigma, x) = \pi^{-n} 2^{-\frac{n(\nu_\beta+2)}{2}} |\Lambda_\beta|^{\frac{\nu_\beta}{2}} \Gamma_n(0.5\nu_\beta)^{-1} \Sigma_\beta^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}(\nu_\alpha+n+2)} \\ \times \exp\Big\{ -\tfrac{1}{2} \mathrm{tr}\Big[\Sigma^{-1} \Big((\mu - \mu_\alpha) \Sigma_\alpha^{-1} (\mu - \mu_\alpha)^\top + \Lambda_\alpha \Big) \Big] \Big\}. \end{aligned} \tag{A.66}$$

Computing the updated statistics according to (A.65) is cumbersome. We therefore aim to find recursive relations in the same way as in [171]. From (A.63), we can see that plugging $V_\beta^{11} + 1$ for V_α^{11} in (A.65a) allows us to derive

$$\begin{aligned} \Sigma_\alpha &= (V_\beta^{11} + 1)^{-1} \\ &= (\Sigma_\beta^{-1} + 1)^{-1} \\ &= \frac{\Sigma_\beta}{1 + \Sigma_\beta}. \end{aligned} \tag{A.67}$$

Next, by inserting $V_\beta^{x1} + x$ for V_α^{x1} and Σ_α for $(V_\alpha^{11})^{-1}$ in (A.65b), we find the recursive formula for updating the least-square estimate

$$\begin{aligned} \mu_\alpha &= (V_\beta^{x1} + x)\Sigma_\alpha \\ &= (\mu_\beta \Sigma_\beta^{-1} + x)\Sigma_\alpha \\ &= (\mu_\beta(\Sigma_\alpha^{-1} - 1) + x)\Sigma_\alpha \\ &= \mu_\beta + \Sigma_\alpha (x - \mu_\beta). \end{aligned} \tag{A.68}$$

Finally, substituting $V_\beta^{xx} + xx$ for V_α^{xx}, $V_\beta^{x1} + x$ for V_α^{x1}, and Σ_α for $(V_\alpha^{11})^{-1}$ in (A.65c) leads to

$$\begin{aligned}
\Lambda_\alpha &= V_\beta^{xx} + xx^\top - (V_\beta^{x1} + x)\Sigma_\alpha(V_\beta^{x1} + x)^\top \\
&= V_\beta^{xx} + xx^\top - x\Sigma_\alpha x^\top - x\Sigma_\alpha V_\beta^{1x} - V_\beta^{x1}\Sigma_\alpha x^\top - V_\beta^{x1}\Sigma_\alpha V_\beta^{1x} \\
&= V_\beta^{xx} + x\frac{1+\Sigma_\beta}{1+\Sigma_\beta}x^\top - x\Sigma_\alpha x^\top - x\Sigma_\alpha V_\beta^{1x} - V_\beta^{x1}\Sigma_\alpha x^\top - V_\beta^{x1}\Sigma_\alpha V_\beta^{1x} \\
&\qquad + V_\beta^{x1}\Sigma_\beta V_\beta^{1x} - V_\beta^{x1}\Sigma_\beta V_\beta^{1x} \\
&= \Lambda_\beta + x\frac{1}{1+\Sigma_\beta}x^\top - x\frac{\Sigma_\beta}{1+\Sigma_\beta}V_\beta^{1x} - V_\beta^{x1}\frac{\Sigma_\beta}{1+\Sigma_\beta}x^\top + V_\beta^{x1}\Sigma_\beta\frac{1}{1+\Sigma_\beta}\Sigma_\beta V_\beta^{1x} \\
&= \Lambda_\beta + \frac{1}{1+\Sigma_\beta}(x-\mu_\beta)(x-\mu_\beta)^\top. \qquad \text{(A.69)}
\end{aligned}$$

Now we have all what we need to find the conditional and marginal densities we are looking for. Let us extended (A.66) according to

$$\begin{aligned}
p(\mu, \Sigma, x) = \pi^{-\frac{n}{2}} 2^{-\frac{n(\nu_\alpha+1)}{2}} |\Lambda_\alpha|^{\frac{\nu_\alpha}{2}} \Gamma_n(0.5\nu_\alpha)^{-1} \Sigma_\alpha^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}(\nu_\alpha+n+2)} \\
\times \exp\Big\{ -\tfrac{1}{2}\mathrm{tr}\Big[\Sigma^{-1}\Big((\mu-\mu_\alpha)\Sigma_\alpha^{-1}(\mu-\mu_\alpha)^\top + \Lambda_\alpha\Big)\Big]\Big\} \\
\times \frac{\Gamma_n(0.5\nu_\alpha)}{\Gamma_n(0.5\nu_\beta)} \pi^{-\frac{n}{2}} (1+\Sigma_\beta)^{-\frac{n}{2}} |\Lambda_\beta|^{\frac{\nu_\beta}{2}} |\Lambda_\alpha|^{-\frac{\nu_\alpha}{2}}, \qquad \text{(A.70)}
\end{aligned}$$

where, from (A.67), we apply $\Sigma_\beta = \Sigma_\alpha(1+\Sigma_\beta)$. The first part in (A.70) is equivalent to the sought conditional density (A.54). The second part still needs further rearrangements to result in the desired marginal density (A.55). Thus, after substituting (A.69) in the last line of (A.70), we have

$$\begin{aligned}
p(\mu, \Sigma, x) = \mathcal{N}i\mathcal{W}(\mu, \Sigma; \mu_\alpha, \Sigma_\alpha, \nu_\alpha, \Lambda_\alpha) \\
\times \frac{\Gamma_n\left(\frac{\nu_\beta+1}{2}\right)}{\Gamma_n\left(\frac{\nu_\beta}{2}\right)} \pi^{-\frac{n}{2}} (1+\Sigma_\beta)^{-\frac{n}{2}} |\Lambda_\beta|^{\frac{\nu_\beta}{2}} \\
\times \left| \Lambda_\beta + \frac{1}{1+\Sigma_\beta}(x-\mu_\beta)(x-\mu_\beta)^\top \right|^{-\frac{\nu_\alpha}{2}}, \qquad \text{(A.71)}
\end{aligned}$$

where the ratio of the multivariate gamma functions can be simplified as

$$\begin{aligned}
\frac{\Gamma_n\left(\frac{\nu_\beta+1}{2}\right)}{\Gamma_n\left(\frac{\nu_\beta}{2}\right)} &= \frac{\pi^{n\frac{n-1}{4}} \prod_{i=1}^n \Gamma\left(\frac{\nu_\beta+1}{2} - \frac{i-1}{2}\right)}{\pi^{n\frac{n-1}{4}} \prod_{i=1}^n \Gamma\left(\frac{\nu_\beta}{2} - \frac{i-1}{2}\right)} \\
&= \frac{\Gamma\left(\frac{\nu_\beta-n+2}{2}\right)\Gamma\left(\frac{\nu_\beta-n+3}{2}\right)\Gamma\left(\frac{\nu_\beta-n+4}{2}\right)\ldots\Gamma\left(\frac{\nu_\beta+1}{2}\right)}{\Gamma\left(\frac{\nu_\beta-n+1}{2}\right)\Gamma\left(\frac{\nu_\beta-n+2}{2}\right)\Gamma\left(\frac{\nu_\beta-n+3}{2}\right)\ldots\Gamma\left(\frac{\nu_\beta}{2}\right)} \\
&= \frac{\Gamma\left(\frac{\nu_\beta+1}{2}\right)}{\Gamma\left(\frac{\nu_\beta-n+1}{2}\right)}. \qquad \text{(A.72)}
\end{aligned}$$

The last rearrangement consists in making the extensions with the red terms according to

$$
\begin{aligned}
p(\mu, \Sigma, x) = {} & \mathcal{N}i\mathcal{W}(\mu, \Sigma; \mu_\alpha, \Sigma_\alpha, \nu_\alpha, \Lambda_\alpha) \\
& \times \frac{\Gamma\left(\frac{\nu_\beta - n + 1 + n}{2}\right)}{\Gamma\left(\frac{\nu_\beta - n + 1}{2}\right)} \pi^{-\frac{n}{2}} \left| \frac{\nu_\beta - n + 1}{\nu_\beta - n + 1} (1 + \Sigma_\beta) \Lambda_\beta \right|^{-\frac{1}{2}} \\
& \times \left(1 + (x - \mu_\beta)^\top \left(\frac{\nu_\beta - n + 1}{\nu_\beta - n + 1} (1 + \Sigma_\beta) \Lambda_\beta \right)^{-1} (x - \mu_\beta) \right)^{-\frac{\nu_\beta - n + 1 + n}{2}} \qquad \text{(A.73)}
\end{aligned}
$$

and using the identities $c^n|A| = |cA|$ and $|A + cbb^\top| = |A|(1 + cb^\top A^{-1}b)$, where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. After recognizing the statistics (A.57) in (A.73), we can find the sought marginal density (A.55) as demonstrated below

$$
\begin{aligned}
p(\mu, \Sigma, x) &= \mathcal{N}i\mathcal{W}(\mu, \Sigma; \mu_\alpha, \Sigma_\alpha, \nu_\alpha, \Lambda_\alpha) \\
&\qquad \times \frac{\Gamma\left(\frac{\bar{\nu}+n}{2}\right)}{\Gamma\left(\frac{\bar{\nu}}{2}\right)} (\bar{\nu}\pi)^{-\frac{n}{2}} |\bar{\Sigma}|^{-\frac{1}{2}} \left(1 + \frac{(x - \mu_\beta)^\top \bar{\Sigma}^{-1} (x - \mu_\beta)}{\bar{\nu}} \right)^{-\frac{\bar{\nu}+n}{2}} \\
&= \mathcal{N}i\mathcal{W}(\mu, \Sigma; \mu_\alpha, \Sigma_\alpha, \nu_\alpha, \Lambda_\alpha) \mathrm{St}(x; \mu_\beta, \bar{\Sigma}, \bar{\nu}),
\end{aligned}
$$

which concludes the proof. $\square$