Andreas Svensson

Machine learning with state-space models, Gaussian processes and Monte Carlo methods

Unofficial[1] *but reader-friendly version of my PhD thesis*

Please cite this in bibtex/biblatex as

```
@phdthesis{Svensson2018,
author = {Svensson, Andreas},
title = {Machine learning with state-space models,
           {Gaussian} processes and {Monte} {Carlo} methods},
school = {Uppsala University},
year = {2018},
}
```

[1]The official online version contains only the introductory background chapters, and the reader is supposed to find the actual papers on her own.

UPPSALA
UNIVERSITET

Abstract
Numbers are present everywhere, and when they are collected and recorded we refer to them as data. Machine learning is the science of learning mathematical models from data. Such models, once learned from data, can be used to draw conclusions, understand behavior, predict future evolution, and make decisions. This thesis is mainly concerned with two particular statistical models for this purpose: the state-space model and the Gaussian process model, as well as a combination thereof. To learn these models from data, Monte Carlo methods are used, and in particular sequential Monte Carlo (SMC) or particle filters.

The thesis starts with an introductory background on state-space models, Gaussian processes and Monte Carlo methods. The main contribution lies in seven scientific papers. Several contributions are made on the topic of learning nonlinear state-space models with the use of SMC. An existing SMC method is tailored for learning in state-space models with little or no measurement noise. The SMC-based method particle Gibbs with ancestor sampling (PGAS) is used for learning an approximation of the Gaussian process state-space model. PGAS is also combined with stochastic approximation expectation maximization (EM). This method, which we refer to as particle stochastic approximation EM, is a general method for learning parameters in nonlinear state-space models. It is later applied to the particular problem of maximum likelihood estimation in jump Markov linear models. An alternative and non-standard approach for how to use SMC to estimate parameters in nonlinear state-space models is also presented.

There are also two contributions not related to learning state-space models. One is how SMC can be used also for learning hyperparameters in Gaussian process regression models. The second is a method for assessing consistency between model and data. By using the model to simulate new data, and compare how similar that data is to the observed one, a general criterion is obtained which follows directly from the model specification. All methods are implemented and illustrated, and several are also applied to various real-world examples.

To everybody who contributes to a happier and more sustainable world

”Saken är den, eller snarare, är det väl mer adekvat, att så att säga säga att saken är biff (trots att det inte handlar om mat). Vad jag vill säga, mera konkret – och nu ska jag gå rakt på sak och vara rak! – är väl helt enkelt att saken är klar. Ja, det vill säga, i sak.”

Claes Eriksson

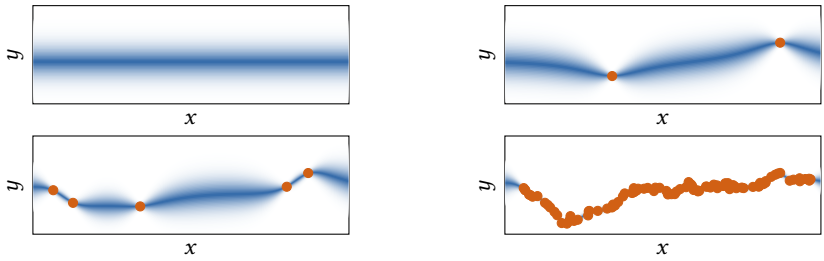
Populärvetenskaplig sammanfattning

Maskininlärning, eller ofta bara ”machine learning”, handlar om att *automatiskt ta fram och använda matematiska modeller från insamlad statistik*. Statistiken kan vara lite vad som helst, till exempel temperaturer, trafikintensitet, opinionsundersökningar, bilder från övervakningskameror, internettrafik, aktiepriser, röstinspelningar, elektricitetsanvändning eller hur snabbt ett virus sprids. Statistiken, eller datan, kan även ha i stort sett vilket format som helst så länge den bara går att spara i en dator. De matematiska modellerna, som maskininlärningen plockar fram ur datan, kan sedan användas för att säga något om ny data som samlas in (”visar den nya bilden från övervakningskameran någon person vi har på bild sedan tidigare?”), förutsäga hur utvecklingen kommer att fortsätta (”kommer viruset att spridas ännu snabbare, eller dö ut?”), eller dra andra slutsatser (”vad skulle hända om trafiken gick längs den här vägen istället?”).

Att ta fram matematiska modeller från den insamlade datan utgör kärnan inom maskininlärning. Det första steget för att göra det, är att vi som användare väljer en klass av modeller som vi vill använda. Det finns många olika modellklasser, och olika modellklasser passar olika bra för olika situationer. Ett populärt exempel, som har fått ganska mycket uppmärksamhet även utanför forskningsvärlden, är faltade neurala nätverk som har visat sig fungera bra för att matematiskt beskriva bilder.

I den här avhandlingen tänker vi oss att den insamlade datan består av siffror som är uppmätta och förändrar sig över tid. Det kan till exempel vara mätningar av koldioxidhalter och börsindex, eller dynamiken i hur ett flygplan reagerar på en förändrad rodervinkel eller hur snabbt en bassäng fylls med vatten när strömmen till en pump slås på. För sådan typ av dynamisk data finns det modelklasser som fungerar särskilt bra, och den här avhandlingen handlar om två av dem: tillståndsmodeller och Gaussprocesser.

Idén med tillståndsmodeller är att matematiskt beskriva det väsentliga i en förändring som sker över tid. Det förklaras nog bäst med ett exempel: Om du får ett foto på en bil, kan du då tala om var bilen skulle befinna sig en sekund senare? Nej, det är svårt, eftersom du från ett vanligt foto inte kan avgöra hur snabbt bilen åker. Men om du får en sekvens av bilder tagna med en sekunds mellanrum så kan du jämföra dem med varandra och få en uppfattning om bilens hastighet, och därifrån göra en ganska noggrann förutsägelse om var bilen skulle befinna sig en sekund efter det sista fotot. Problemet med ett ensamt foto är alltså att det inte innehåller all relevant information. Men *om* fotot hade innehållit även en radarmätning av bilens hastighet (eller en bild på hastighetsmätaren), så hade detta enda foto varit tillräckligt för att göra en bra förutsägelse om bilens position i nästa sekund. Det är det här som är poängen med tillståndsmodeller: de är konstruerade för att i varje ögonblick innehålla all matematisk information som är relevant för framtiden. En väldigt kraftfull egenskap



Den här figuren visar hur en Gaussprocess (blått skuggat område) kan användas för att lära sig ett samband mellan x (horisontell axel) och y (vertikal axel). Målet är att kunna leka leken "om du säger x, så kan jag säga vad y kommer att bli" (utöver att vara en fånig lek, är det också ett vanligt problem i maskininlärning) och till vår hjälp kommer vi att få orangea datapunkter som talar om sanningen y för vissa värden på x.

I rutan uppe till vänster finns det ingen data, och Gaussprocessen säger att vi vet väldigt lite, eftersom det blåa skuggade området är förhållandevis brett. I rutan uppe till höger, däremot, finns det två orangea datapunkter och Gaussprocessen har där anpassat sig till dem: i närheten av punkterna är bredden på det blåa skuggade området, det vill säga osäkerheten om vad y är, liten. Skulle vi däremot be Gaussprocessen om en förutsägelse om vad som händer mitt emellan de två datapunkterna, så är osäkerheten fortfarande ganska stor där.

I den nedre vänstra rutan finns det fem datapunkter, och osäkerheten i Gaussprocessen har minskat ganska mycket på sina håll. En förutsägelse i området mellan de två punkterna längst till vänster innehåller inte mycket osäkerhet alls: eftersom de två punkterna ligger så nära varandra, anser sig Gaussprocessen veta ganska väl vad som händer även i det lilla intervallet emellan dem. Slutligen, i rutan längst ned till höger, har vi fått 100 datapunkter, och Gaussprocessen har anpassat sig till dem alla.

Själva Gaussprocessen är definierad betydligt mer matematiskt än vad som presenteras här, men poängen är att det är en modell som är kapabel att resonera kring osäkerheter om vad y är, vilket är något som efterfrågas i många moderna maskininlärningstillämpningar.

hos tillståndsmodeller är därför att förutsägelser om framtiden inte förbättras ens om det visar sig finnas mer historisk data att tillgå; de har redan sorterat ut allt som är "värt att veta". De här modellerna har visat sig vara användbara och kraftfulla för att beskriva tidsserier och dynamik för många olika tillämpningar, och används flitigt inom så skilda områden som ekonometri och reglerteknik. I den här avhandlingen förekommer tillståndsmodeller många gånger, och vi tittar på och vidareutvecklar metoder att lära sig tillståndsmodeller från insamlade data.

Gaussprocesser är en annan klass av modeller, vars styrka ligger i att kunna interpolera bra mellan olika datapunkter, som illustreras i figuren högst upp på sidan. Med Gaussprocesser beskrivs sannolikheter för olika matematiska funktioner, och på så sätt blir det en modell som kan hantera osäkerheter (att inte veta, en osäkerhet, kan matematiskt beskrivas som att de möjliga alternativen alla har en viss sannolikhet). Gaussprocesser är en relativt ny klass av modeller, och har än så länge kanske haft sin största plats inom maskininlärningsforskningen, men börjar allt mer hitta tillämpningar inom vitt skilda områden där det finns ett behov av att göra förutsägelser om hur stor osäkerhet som finns i ett problem. I den här avhandlingen tittar vi dels på hur Gaussprocesser kan läras från insamlade data, och dels hur de kan kombineras med tillståndsmodeller på ett användbart sätt.

I maskininlärningen, när datan kombineras med en modellklass, används alltså datan för att *lära sig* okända storheter, parametrar, i modellklassen. Att till exempel anpassa en rät linje till några punkter är att lära sig en modell (alltså att lära sig hur mycket linjen ska luta). Principen är densamma även för tillståndsmodeller och Gaussprocesser – att hitta parametrar så att modellen stämmer bra överens med den insamlade datan – men beräkningarna som behöver utföras är ofta mer invecklade.

För att automatiskt lära sig modeller från data, alltså att hitta lämpliga parametervärden, har (kanske lite oväntat?) metoder som bygger på slumpen visat sig vara användbara. Det är nämligen så att de beräkningar som behöver göras för många modellklasser, däribland tillståndsmodeller och Gaussprocesser, är såpass invecklade att de sällan kan göras exakt, och då finns det olika tekniker för att räkna ungefärligt.

Om vi (av någon oklar anledning) har fått i uppgift att slå 3,5 på en vanlig tärning, kan vi välja mellan att leta upp sidan med fyra prickar och säga ”tyvärr, det här var det närmaste jag kunde komma”, eller att kasta tärningen många gånger och säga ”tyvärr, det finns ingen sida som är 3,5, men vi får titta på genomsnittet av många tärningskast istället” (vilket kommer att vara ungefär 3,5). Den första metoden skulle kunna sägas vara en klassisk metod, medan den senare metoden är en metod som bygger på slumpen (att kasta tärningen) och har egenskapen att den ”är i genomsnitt exakt” (vi gör det många gånger); även om tärningen aldrig kommer att visa tre och en halv prick i något enskilt kast, så är det genomsnittet vi tittar på. (Det här exemplet är ju förstås lite fånigt, men liknande situationer kan faktiskt uppstå när man försöker lära sig parametervärden från data.)

Metoder som systematiskt utnyttjar slumpen kallas för Monte Carlo-metoder (efter kasinot i staden med samma namn). Många av Monte Carlo-metoderna[2] har utvecklats just för att lära sig parametrar i matematiska modeller. De är ofta mer beräkningstunga än de klassiska metoderna, men kan ha egenskaper (till exempel att ”göra i genomsnitt rätt”) som gör det värt den extra beräkningskraften, särskilt nuförtiden när det finns snabba datorer som kan användas.

Bidragen i den här avhandlingen handlar till stor del om att utveckla, anpassa och använda sådana här Monte Carlo-metoder för att lära sig parametrar i olika varianter av tillståndsmodeller och Gaussprocesser. I avhandlingen finns det dessutom även ett bidrag som handlar om att utvärdera hur väl en modellklass passar till den insamlade datan.

[2] De olika Monte Carlo-metoderna har mer eller mindre lustiga namn, som till exempel avslagsdragning, viktighetsdragning, Markovkedje-Monte Carlo, partikelfilter, partikel-Markovkejde-Monte Carlo, partikel-Gibbs med förfädersdragning, sekventiell Monte Carlo, sekventiell Monte Carlo upphöjd till två, studsig partikeldragare, och så vidare…

"Danke shön!"
Angela Merkel

Acknowledgments

First of all, I would like to thank professor Thomas Schön. It has been a great pleasure to pursue my doctoral studies under your supervision. I am particularly grateful for your positive attitude, support and coaching whenever I have taken on new (more or less research-related) challenges. It has also been a pleasant journey from starting as your second student here in Uppsala, to now graduating from a vivid and growing machine learning group with a two-digit number of members.

I am also very happy that dr Fredrik Lindsten has served as my co-supervisor. With your deep technical knowledge and willingness to share it, it has been a joy working with you, I have learned a lot!

Thanks also to all my co-authors Dave, Petre, Johan D, Arno, Simo, Manon, Lawrence, Niklas, Dennis and Mahmoud for fruitful (and more or less intense) collaborations. Thanks also to Anna and Calle J who helped with the proofreading of the introductory chapters of this thesis.

This thesis had not been written, had not the Swedish Foundation for Strategic Research (SSF, via the project ENSEMBLE, nr RIT15-0012) and Uppsala University generously funded my position. I hope you find it was worth it. Thanks!

As a part of a Swedish PhD education, I have been undertaking some courses. Some were really good. Thank you Erik Broman, Nicolas Chopin, Omiros Papaspiliopoulos and Henrik Hult for your teaching efforts. A special thanks also goes to the organizers of the Machine Learning Summer School in Tübingen 2015.

The most important part of a workplace is probably the colleagues. Thank you Johan W for many interesting discussions (and sorry for interrupting you all the time you're in your office). Thank you Anna, Niklas, Thomas and Fredrik L for fun times teaching together. In addition to some of you already mentioned, thank you Calle A, Fredrik, Koen and David for nice running sessions. A big thanks also to Calle J, Diana, Guo, Helena, Jack, Johan A, Juozas, Lawrence, Marcus, Maria, Niklas, Pelle, Rubén, Tatiana and Viktor for making it worth to walk all the way to the coffee room three times a day (even though I don't even drink coffee myself). Thanks also to Katarina, Dick and Marina for helping out with all administrative and practical issues.

I would also like to thank some people from my life outside the thesis, Jonathan & Elisabet, Julia & Robert, Lina & Bernhard, David, Diana and Victor, for your nice and more or less regular company during these years!

A big thanks of course also goes to my parents, Bosse & Christina, for your encouragement and support throughout my life. Without you, literally, neither me nor this thesis would exist. My sisters, Brita and Anna, also played an inevitable role in fostering me to the one I am today. Whether that is a good or bad thing, I leave to other to decide, but thank you anyway!

And, finally, thank you Sanna, for your existence, patience and love!

Andreas
Uppsala, August 2018

"Learn—*that's a trendy word.*"
Andrew Gelman

Table of contents

"I'm being quoted to introduce something, but I have no idea what it is and certainly don't endorse it."
Randall Munroe

1

Introduction

Data, in the format of recorded numbers, is literally present everywhere. Examples range from temperature measurements, traffic intensity, polls, internet traffic and stock market prices, to speech recordings, electricity usage and epidemiological data. To process such data in computer systems, mathematical models can be helpful.

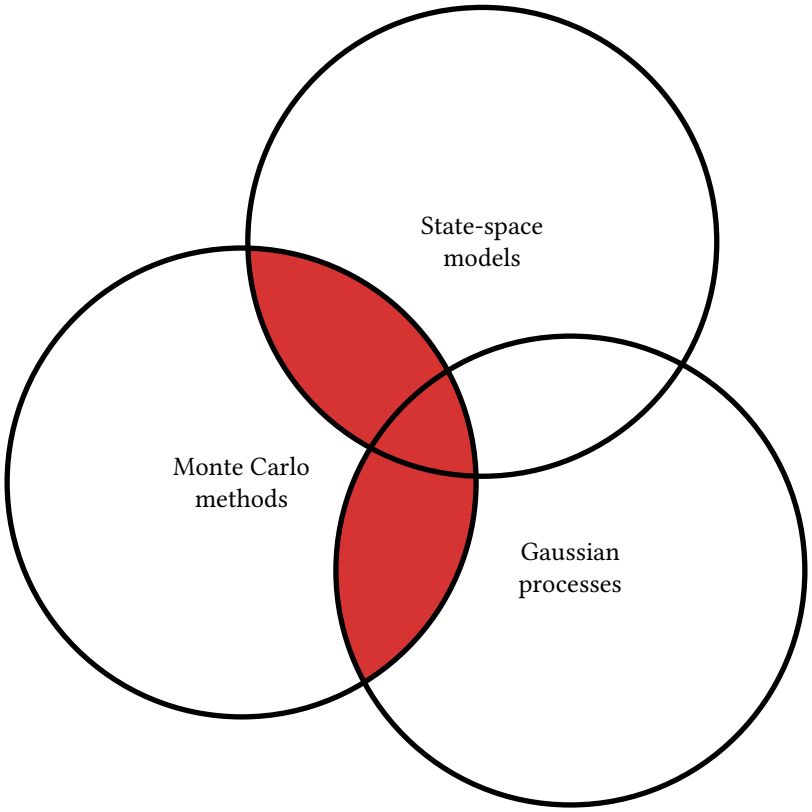
A mathematical model is a set of equations. Those equations can—if carefully chosen—be a compact and powerful tool to describe links and causalities of the data. Thus, by *learning* a mathematical model from recorded data, the learned model can—sometimes—help in explaining the mechanisms behind the data, answer 'what if'-questions and make predictions for the future evolution. For this reason, *mathematical models are an essential tool for making sense of data.* This thesis has a focus on a subset of mathematical models which are motivated by probability theory and statistics. Such models are referred to as *statistical models.* Statistical models learned from recorded data can, for example, be used to predict future weather, advise on how to drive to avoid congestion, make scenario analysis to decide on future network designs, price assets automatically, translate speech to text, solve energy aggregation problems and predict disease spread.

Different words for the same thing

In Swedish there is a saying 'kärt barn har många namn', meaning 'we have many names for the things we love'. That is definitely true for this topic. Depending on context, names such as *machine learning*, *statistical learning*, *system identification*, *parameter estimation*, *cybernetics* and many more, are used to describe the science of learning mathematical models from data. Somewhat (at least historically) incorrect is also the term *artificial intelligence* used nowadays. Also the term *learning* has many synonyms itself, including *training*, *calibration*, *inference* and *estimation*.

1.1 Focus of the thesis

The field of machine learning is vast and ever increasing. Of course, this thesis cannot cover everything, and not even close to. This thesis is primarily about *learning statistical models from data* when *the data has a sequential nature* (such as time series and input-output relationships of dynamical systems). In a technical lingo, the models concerned in this thesis are primarily state-space models (hidden Markov models) and Gaussian processes. The learning, which is done by a computer, is nothing but a set of mathematical computations. There are different methods to perform these computations, and this thesis focuses on a set of methods which makes clever use of randomness, namely Monte Carlo methods. A pictorial view could perhaps look like this, where the shaded red regions are the focus of the thesis:



This thesis, and also the research behind it, is focused on methods rather than applications. That does not mean there are no applications, and examples of applications are given in several of the papers.

1.2 Outline of the introductory chapters

The first part of the thesis contains five introductory chapters (including this chapter) and one concluding chapter. The purpose of these introductory chapters is to summarize the background and put the papers into a broader perspective. The concluding chapter, number 6, is meant to be read after the papers.

After this first chapter, we start in Chapter 2 by dissect machine learning into its main pieces *data*, *models* and *how to learn models from data*. Two statistical models are then introduced in detail, the state-space model in Chapter 3 and the Gaussian-process model in Chapter 4. We thereafter devote Chapter 5 to Monte Carlo methods, which are used to make the computations required for the learning. Appendix A contains a central result for sequential Monte Carlo (Chapter 5), and Appendix B contains a derivation of the conjugate prior for Gaussian linear regression; important expressions for Paper I.

1.3 Main contributions

The main scientific contributions in this thesis are the following:

- A numerically feasible approximative implementation of the Gaussian process state-space model (Paper I).
- A novel criterion for assessing consistency between data and model (Paper II).
- An introduction to and theoretical analysis of the particle stochastic approximation EM algorithm, as well as its formulation for jump Markov linear models and the empirical Bayes problem (Paper III and IV).
- A novel alternative to SMC^2 for state-space models with highly informative observations (Paper V).
- A novel approach to parameter estimation in non-linear state-space models using particle filters (Paper VI).
- The use of an SMC sampler to learn hyperparameters for the Gaussian process model (Paper VII).

1.4 Articles included in the thesis

The second part of this thesis contains scientific articles. The articles are listed below, together with a brief summary and statement of my (the thesis author's) contributions.

Paper I

> Andreas Svensson and Thomas B. Schön (2017). "A flexible state-space model for learning nonlinear dynamical systems". In: *Automatica* 80, pp. 189–199.

A conceptually interesting combination of the state-space model and the Gaussian process model is provided by the Gaussian process state-space model (Section 4.6). This article presents a numerically feasible approximation of that combination, from a system identification perspective. My contributions to this paper are the mathematical derivations, implementation and experiments. I have written the major part, but also Thomas B. Schön, to whom the original idea should be attributed, has contributed to the writing.

Paper II

> Andreas Svensson, Dave Zachariah, Petre Stoica, and Thomas B. Schön (2018). "Data consistency approach to model validation". Submitted for publication.

This paper presents a criterion to assess if a given data set and a model class are consistent, in the sense that the given data should be 'similar' to data artificially generated from the model. The writing, as well as the coining of the technical idea, was done jointly with Dave Zachariah and Petre Stoica. The implementations are mine.

Paper III

> Andreas Svensson and Fredrik Lindsten (2018). "Learning dynamical systems with particle stochastic approximation EM". Submitted for publication.

The particle stochastic approximation EM (PSAEM) is a method for learning state-space models, based on particle filters and the Expectation-Maximization (EM) algorithm. I have written the major part of the paper and performed the experiments. The idea was originally coined by Fredrik Lindsten, who has also done most of the theoretical analysis.

Paper IV

> Andreas Svensson, Thomas B. Schön, and Fredrik Lindsten (2014). "Identification of jump Markov linear models using particle filters". In: *Proceedings of the 53rd IEEE Conference on Decision and Control (CDC)*. Los Angeles, CA, USA, pp. 6504–6509.

The PSAEM method, which is the topic of Paper III, is here adapted to jump Markov linear models, a special class of state-space models. The original idea is due to Thomas B. Schön and Fredrik Lindsten, whereas I have done the major part of the writing and all implementations.

Paper V

Andreas Svensson, Thomas B. Schön, and Fredrik Lindsten (2018). "Learning of state-space models with highly informative observations: a tempered Sequential Monte Carlo solution". In: *Mechanical Systems and Signal Processing* 104, pp. 915–928.

State-space models with very little or no measurement noise turn out, perhaps surprisingly, to be very hard to learn with methods based on the particle filter. To this end, a scheme is proposed where artificial measurement noise is introduced and gradually decreased, in a consistent way. The original idea, analysis and implementation are all my work, and I have also done the major part of the writing.

Paper VI

Andreas Svensson, Fredrik Lindsten, and Thomas B. Schön (2018). "Learning nonlinear state-space models using smooth particle-filter-based likelihood approximations". In: *Proceedings of the 18th IFAC symposium on system identification (SYSID)*. Stockholm, Sweden, pp. 652–657.

This paper is a novel approach to maximum likelihood estimation of unknown parameters in non-linear state-space models. By scrutinizing the particle-filter algorithm, a slightly different interpretation of it can be found, which can be used to formulate a maximization problem to which conventional optimization methods can be applied. The original idea and implementation are all my work, and I have also written the major part of the paper. The analysis was done jointly with Fredrik Lindsten and Thomas B. Schön.

Paper VII

Andreas Svensson, Johan Dahlin, and Thomas B. Schön (2015). "Marginalizing Gaussian process hyperparameters using sequential Monte Carlo". In: *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. Cancún, Mexico, pp. 489–492.

The hyperparameters in the Gaussian process model can, if unknown, either be estimated or marginalized. This paper suggests the use of the sequential Monte Carlo sampler for the latter. The original idea should be attributed to Johan Dahlin, whereas I have implemented the examples and written the major part of the paper.

1.5 Related but not included work

Beyond the articles included in the thesis, the following research (to which I have contributed) is also relevant to this thesis:

[A] Andreas Svensson, Dave Zachariah, and Thomas B. Schön (2018). "How consistent is my model with the data? Information-theoretic model check". In: *Proceedings of the 18th IFAC symposium on system identification (SYSID)*. Stockholm, Sweden, pp. 407–412.

[B] Thomas B. Schön, Andreas Svensson, Lawrence M. Murray, and Fredrik Lindsten (2018). "Probabilistic learning of nonlinear dynamical systems using sequential Monte Carlo". In: *Mechanical Systems and Signal Processing* 104, pp. 866–883.

[C] Dennis W. van der Meer, Mahmoud Shepero, Andreas Svensson, Joakim Widén, and Joakim Munkhammar (2018). "Probabilistic forecasting of electricity consumption, photovoltaic power generation and net demand of an individual building using Gaussian Processes". In: *Applied Energy* 213, pp. 195–207.

[D] Andreas Svensson, Arno Solin, Simo Särkkä, and Thomas B. Schön (2016). "Computationally efficient Bayesian learning of Gaussian process state space models". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cádiz, Spain, pp. 213–221.

[E] Andreas Svensson, Thomas B. Schön, Arno Solin, and Simo Särkkä (2015). "Nonlinear state space model identification using a regularized basis function expansion". In: *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. Cancún, Mexico, pp. 493–496.

[F] Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wågberg, Christian A. Naesseth, Andreas Svensson, and Liang Dai (2015). "Sequential Monte Carlo methods for system identification". In: *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*. Beijing, China, pp. 775–786.

[G] Andreas Svensson and Thomas B. Schön (2016). *Comparing two recent particle filter implementations of Bayesian system identification.* Tech. rep. 2016-008. (Presented at Reglermöte 2016, Gothenburg, Sweden). Department of Information Technology, Uppsala University.

[H] Andreas Svensson, Thomas B. Schön, and Manon Kok (2015). "Nonlinear state space smoothing using the conditional particle filter". In: *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*. Beijing, China, pp. 975–980.

[I] Andreas Svensson (2016). "Learning probabilistic models of dynamical phenomena using particle filters". Licentiate thesis. Department of Information Technology, Uppsala University.

In addition, I have also contributed to some pedagogic work also of relevance to this thesis:

[J] Fredrik Lindsten, Andreas Svensson, Niklas Wahlström, and Thomas B. Schön (2018). *Statistical Machine Learning. Lecture notes on linear regression, logistic regression, deep learning & boosting*. Department of Information Technology, Uppsala University.

[K] Fredrik Lindsten, Thomas B. Schön, Andreas Svensson, and Niklas Wahlström (2017). *Probabilistic modeling—linear regression & Gaussian processes*. Department of Information Technology, Uppsala University.

[L] Andreas Svensson (2013). *Particle Filter Explained without Equations*. URL: https://www.youtube.com/watch?v=aUkBa1zMKv4.

1.6 A word on notation

The first part of the thesis (Chapter 1–5) are meant to have a consistent notation (a complete list can be found on page 65). The notation in the included articles is, however, slightly different, and introduced separately for each article.

In general we use a probabilistic language and notation, and use the word 'model' mainly to refer to some (possibly complicated) probability distribution. We work in the first place with probability distributions in terms of their densities (or mass) $p(\cdot)$, and thereby we implicitly assume its existence. The generalization to the case when the density does not exist is often possible, but not covered. Also the existence of a σ-algebra and dominating measure is implicit. Different densities are distinguished by their arguments, and $p(\cdot \mid \cdot)$ denotes a conditional density.

All random variables are written with lowercase letters x, θ, etc., and no distinction between random variables and their realizations is made in the notation. Integrals without any explicit limits are over the entire domain of the integration variable.

"All models are wrong but some are useful."
George E. P. Box

2

Learning models from data

MACHINE learning is a a multifaceted term, but in this thesis we will understand it as the *processing of observed data into a statistical model.* The key elements in this process are

(i) the recorded *data* y (Section 2.1),

(ii) the statistical *model* $p(y \mid \theta)$, which has some degrees of freedom expressed via a parameter θ (Section 2.2),

(iii) the *learning* which links the model to the data by drawing conclusions about θ from y (Sections 2.3–2.5).

Loosely speaking, one could say that learning is to extract the essence of the data y, and put that information into the parameters θ. In this chapter, we think in the first place of θ as being finite-dimensional and real-valued. In Chapter 4, we consider a model where θ is infinite-dimensional.

The purpose of this chapter is to give an introduction to the way we think about data and models, and also to cover some background on the learning side. We use a statistical perspective, and sometimes use the more technical term *statistical inference* for learning. We cover two different paradigms for how to perform the inference, namely the point estimation approach and the Bayesian approach.

We remain on a rather general level in this chapter. Particular examples of data and models will be introduced first in Chapter 3 and 4, as well as in some of the papers. We also leave integrals and optimization problems hanging in midair without attempting to compute them for now. Methods for performing these computations are be introduced in Chapter 5 and in some of the papers.

2.1 Data y

The first and foremost thing in machine learning is the data y. The data could in principle be anything that can be recorded, but we limit ourselves to numbers, typically (but not necessarily) recorded sequentially during some period of time, as[1] $y = \{y_1, \ldots, y_T\}$. The data could be artificially generated by a computer, but in most (if not all) cases of interest for the broader society, the data is recorded from some real phenomenon which is not yet completely understood. Examples of typical data could be logs of outdoor temperatures, measured forces in a mechanical system, or stock prices. We make no assumptions on the data, other than that it exists and has a certain format (such as $y \in \mathbb{R}^{n_y \times T}$ or similar). Throughout this thesis, we will be in the position that the data is already recorded and available to us, meaning that questions on how to record the data or how to design experiments to 'reveal as much information as possible' falls outside of the scope of this thesis (see, e.g., Chaloner and Verdinelli 1995; Hjalmarsson 2009; Pukelsheim 1993).

Throughout the first part of this chapter, we use a toy example to illustrate the concepts. Let us therefore say that we have data which consists of two observations $y_1 = 1.54$ and $y_2 = 3.72$ ($n_y = 1, T = 2$).

2.2 Models $p(y \mid \theta)$

Next, we introduce a model for the data y, which we denote by $p(y \mid \theta)$. The notation $p(y \mid \theta)$ encodes a probability distribution which is assumed to be able to describe y in some respect. Crucially, the model might depend on some unknown parameter θ. This unknown parameter will later be learned, and for that result to be as relevant as possible, it is good if the present knowledge (and ignorance) about the problem is included in the model: if the data exhibits strong saturation effects, the learned parameters in a linear model might carry very little insights.

The model can be derived from first principles (such as Newton's laws of motions, Leavitt's law or the ideal gas law) where the unknown parameters have some physical interpretation. The model can also be of a more generic flexible type, where the parameters carry no direct physical interpretation. The latter is perhaps more of a typical machine-learning case.

Let us demystify the abstract notion of a model by using the toy example. Say that we decide to model the data points as independent draws from the same Gaussian distribution. The unknown parameters are then the mean μ and the variance σ^2 of the Gaussian distribution, i.e., $\theta \triangleq \{\mu, \sigma^2\}$, and we write the model as

$$p(y \mid \theta) = \mathcal{N}\left(y_1; \mu, \sigma^2\right) \cdot \mathcal{N}\left(y_2; \mu, \sigma^2\right). \tag{2.1}$$

Note that we have not limited ourselves to data actually generated by $p(y \mid \theta)$: the model is only an *assumption* within the learning procedure. We are, in fact, free to make arbitrary model assumptions, perhaps in the interest of feasible computations! This means that inference results should *always be read with the model assumptions in mind.* To validate a model assumption $p(y \mid \theta)$ for some data y, we present a new method in Paper II.

[1] We later use the notation $y_{1:t} = \{y_1, \ldots, y_t\}$ when there is a need to emphasize exactly which data points are under consideration. For now, we settle with only y to denote all the available data.

2.3 Two paradigms for deducing unknown parameters

In our notation, we prepared for the learning by including the possible dependence on a parameter θ in the model $p(y \mid \theta)$. The big question throughout the rest of this chapter is the following: if an unknown parameter θ is present in the model, how should the data y and the model $p(y \mid \theta)$ be used for drawing conclusions about θ?

This question is at the core of statistical inference, and some textbook references are Casella and R. L. Berger (2002), Gelman et al. (2014), and Schervish (1995). The field has traditionally been divided into several paradigms, ultimately differing perhaps in their interpretation of probabilities. We will pursue two alternative ways of handling the unknown parameters θ throughout the thesis. The two following questions are meant to reflect the underlying alternative reasoning about how to learn θ,

(i) which estimate $\widehat{\theta}$ fits the data y the best?

(ii) after digesting the information brought to us by the data y, what degree of belief $p(\theta \mid y)$ do we have in different values of θ?

We will refer to these alternatives as (i) the *point estimation* and (ii) the *Bayesian* approach, respectively. The distinction between the different paradigms is not always entirely clear in the literature, but below we give an explanation of the way in which we use the terms. Some texts on the major paradigms in statistical inference, in addition to the discussion below, are Efron (1986), Efron (2013), Efron and Hastie (2016), and Lindley (1990).

2.3.1 Finding a point estimate for θ: $\widehat{\theta}$

The first learning approach we review, is that of finding a *point estimate* $\widehat{\theta}$ that fits the observed data as well as possible. Which particular point estimate $\widehat{\theta}$ to choose, i.e., the meaning of 'fit' in the rhetoric question above, might however vary. One choice (common in this thesis) is to maximize the likelihood function

$$\mathcal{L}(\theta) \triangleq p(y \mid \theta), \tag{2.2}$$

an alternative that we will refer to as *maximum likelihood*. Note that even though the likelihood function $\mathcal{L}(\theta)$ is a function of θ, the object $p(y \mid \theta)$ describes how 'likely' y (not θ) is. Also note that in (2.2), the data y is fixed[2], in contrast to when we talk about the model $p(y \mid \theta)$. Two alternatives to maximum likelihood are to either optimize the predictive capabilities of the model, or to maximize the likelihood function subject to some additional constraints, such as keeping the numerical values close to zero or promote sparsity in $\widehat{\theta}$. The latter alternatives are often referred to as *regularization*, a theme we will return to in Section 2.5.2. The choice of which point estimate to use can be formalized mathematically using decision theory, a topic not considered in this thesis (see, e.g., Schervish 1995, Chapter 3).

Computing point estimates $\widehat{\theta}$ is often at the heart of the classical/frequentist/Neyman-Pearson-Wald school in the literature, whereas inference based on the likelihood function historically is separated into the Fisherian tradition. In the statistical

[2] In the traditional notation with uppercase letter for random variables, and lowercase for their realizations, we could write (2.2) as $\mathcal{L}(\theta) \triangleq p(Y = y \mid \theta)$, whereas the term 'model' refers to $p(Y \mid \theta)$.

literature, it is also common to introduce confidence regions expressing uncertainty about $\widehat{\theta}$ (not θ). In this thesis, we group these schools together as the point estimation approach. We refrain from putting focus on confidence regions, because of the tradition and available computational tools for the models to be presented later on.

In the toy example from above, a maximum likelihood point estimate $\widehat{\theta} \triangleq \{\widehat{\mu}, \widehat{\sigma}^2\}$ is found by solving the problem

$$\widehat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\mu,\ \sigma^2} \mathcal{N}\left(1.54; \mu, \sigma^2\right) \cdot \mathcal{N}\left(3.72; \mu, \sigma^2\right). \tag{2.3}$$

The solution turns out to be $\widehat{\mu} = 2.63$ and $\widehat{\sigma}^2 = 1.09$, i.e., some numbers $\widehat{\theta}$ that can be used to analyze the data, making subsequent predictions, etc.

2.3.2 Finding the posterior distribution for θ: $p(\theta \mid y)$

The second approach relates to the interpretation of probabilities as degrees of belief, and makes use of Bayes' theorem (named after Thomas Bayes 1763)

$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{p(y)} \tag{2.4}$$

to update the *prior* belief $p(\theta)$ into the *posterior* belief $p(\theta \mid y)$. Going from the prior to the posterior can be understood as *conditioning the belief on data.* The right hand side of (2.4) contains, apart from the prior $p(\theta)$, the *likelihood* (or *data density*) $p(y \mid \theta)$ and[3] $p(y)$.

Bayesian inference is all about computing the posterior $p(\theta \mid y)$. The central role of Bayes' theorem is the obvious reason behind the name of the paradigm. However, also the name of Pierre-Simon de Laplace (1820) (Stigler 1986) and Bruno de Finetti (1992) occurs in the literature.

There is nothing conceptually different between the prior $p(\theta)$ and the posterior $p(\theta \mid y)$: they both reflect a degree of belief about θ, before and after observing the data y, respectively. If more data is observed subsequently, Bayes' theorem may be applied repeatedly to incorporate the new observations into the belief. However, Bayes' theorem only provides a mechanism for *updating* beliefs, not creating beliefs from nothing. Therefore, the prior $p(\theta)$ has to be *chosen*[4]. To obtain a useful result, the choice of prior should preferably reflect present ignorance and knowledge, in the very same way as the model $p(y \mid \theta)$ should be chosen carefully.

[3]Note that $p(y)$, the denominator, can be written as an integral over the numerator with respect to θ, $p(y) = \int p(y \mid \theta)p(\theta)d\theta$. $p(y)$ can thus be thought of as a normalization to ensure that $\int p(\theta \mid y) = 1$, and can be ignored if it is sufficient to compute (2.4) up to proportionality.

[4]The (inevitably subjective) prior choice is, according to the authors personal experience, one of the main criticisms that is brought up towards the Bayesian paradigm. It is, however, unclear to the author why the prior choice should be subject to criticism, whereas the equally subjective model choice (present in both paradigms) mostly is kept out of the discussion. J. O. Berger (2006) comments to this concern as follows: '[The model choice] *will typically have a much greater effect on the answer than will such things as choice of prior distributions for model parameters. Model-building is not typically part of the objective/subjective debate, however—in part because of the historical success of using models, in part because all the major philosophical approaches to statistics use models and, in part, because models are viewed as "testable," and hence subject to objective scrutiny. It is quite debatable whether these arguments are sufficient to remove model choice from the objective/subjective debate, but I will simply follow statistical (and scientific) tradition and do so.*'

It should, however, be remembered that all degree of belief in θ is still conditional on the choice of model $p(y \mid \theta)$. A popular sales pitch for the Bayesian approach is that the posterior distribution provides *uncertainty* about the parameters. That is indeed true, but note that the use of Bayes' theorem does not imply a question whether $p(y \mid \theta)$ is relevant for modeling y; if no choice of the unknown θ gives a reasonable model for y, the posterior belief $p(\theta \mid y)$ will only concentrate around the 'least bad' parameter value.

It is also possible to extract point estimates of θ from the posterior $p(\theta \mid y)$. Popular estimates of that kind are the posterior mean and the posterior mode, where the latter usually is referred to as maximum a posteriori (MAP) estimation. However, since a point estimate does not represent a degree of belief (which is a core component in the Bayesian approach), we do not consider it to be a Bayesian method here. It does, however, bear some resemblances to the regularized maximum likelihood approach, as we will see in Section 2.5.2.

Let us have a look at the little toy example again, now from the Bayesian point of view. First, we have to append our assumption $p(y \mid \theta)$ also with assumptions about μ and σ^2. Let us assume a normal-inverse-gamma prior distribution (Appendix B), $p(\mu, \sigma^2) = \mathcal{NIG}\left(\mu, \sigma^2; 0, 1, 1, 1\right)$. By inserting all expressions into Bayes' theorem (2.4) and performing some algebraic manipulations, we find the posterior $p(\mu, \sigma^2 \mid y) = \mathcal{NIG}\left(\mu, \sigma^2; 1.75, 3, 2, 4.49\right)$. This is a distribution, which we may use subsequently to analyze the data, do predictions, etc.

The particular choice of prior in the toy example was a so-called *conjugate prior* since the prior, a $\mathcal{NIG}$ distribution, together with the likelihood model (2.1), a Gaussian distribution with unknown mean and variance, yields another $\mathcal{NIG}$ distribution as the posterior. For some priors, the posterior may not admit a closed form, and conjugate priors only exist for a limited set of models.

2.4 Posterior distributions vs. point estimates

The most striking difference between the point estimates and the Bayesian paradigm for a user, is perhaps not the different underlying philosophies about the meaning of probabilities, nor the presence or absence of priors. Instead, the major difference for a user is that point estimates $\widehat{\theta}$ and distributions $p(\theta \mid y)$ are very different objects: A point estimate $\widehat{\theta}$ is a number, whereas $p(\theta \mid y)$ is, well, a distribution. If the user interest is, for example, to predict a future observation $y^\star$, the point estimation approach is to put $\widehat{\theta}$ into the model and take the mean

$$\widehat{y^\star} = \mathbb{E}\left[p(y^\star \mid \widehat{\theta})\right] \tag{2.5}$$

as the (point) prediction $y^\star$. For the Bayesian case on the contrary[5], the prediction of $y^\star$ is the predictive distribution

$$p(y^\star \mid y) = \int p(y^\star \mid \theta) p(\theta \mid y) d\theta. \tag{2.6}$$

[5]Indeed, $p(y^\star \mid \widehat{\theta})$ is also a distribution. However, as it bears no meaning akin to (2.6), and the point estimation approach is more concerned with point estimates, the entire distribution $p(y^\star \mid \widehat{\theta})$ is typically not considered, but only its mean (2.5), or similar.

In many cases, the predictive distribution (and often also the posterior) admits no closed form expression. Instead, those distributions have to be approximated. Two such approximative alternatives are provided by the variational approach (e.g., Blei et al. 2016) and the Monte Carlo approach (Chapter 5).

Whether to take the point estimate or the Bayesian approach, may depend on several aspects. Often, but not always, point estimation can be less computationally intensive compared to the Bayesian approach, an argument for preferring the former. However, if the computational aspect allows a choice, one may consider questions such as

- What is the intended use of the obtained results: does a posterior distribution $p(\theta \mid y)$ provide valuable information in the solution, which is not preserved by a single point estimate $\widehat{\theta}$?
- Is it sensible, or even crucial, to include prior beliefs about θ into the solution? (See Section 2.5.2)

Personal preferences may of course also influence the choice: point estimates have, for example, traditionally dominated the system identification community (an interesting uphill struggling paper arguing for the Bayesian approach is Peterka (1981)).

If the data is highly informative about the parameters θ, the differences between the two paradigms may be small. Consider a toy example with T observations $\{y_t\}_{t=1}^T$ of a one-dimensional parameter $\theta \triangleq \mu$. We model the observations to be exchangeable (see, e.g., Section 1.2 in Schervish 1995) and all have a Gaussian distribution with mean μ and variance 1, and we assume a prior $p(\mu) = \mathcal{N}(\mu; 0, 1)$. This yields the posterior

$$p(\mu \mid y) \propto \underbrace{\mathcal{N}(\theta; 0, 1)}_{p(\mu)} \underbrace{\prod_{t=1}^{T} \mathcal{N}(\mu; y_t, 1)}_{p(y \mid \mu),} \tag{2.7a}$$

which after some algebraic manipulations can be written

$$p(\mu \mid y) = \mathcal{N}\left(\mu; \tfrac{\sum_{t=1}^{T} y_t}{T+1}, \tfrac{1}{T+1}\right). \tag{2.7b}$$

That is, the posterior variance tends to 0 as the number of observations $T \to \infty$. *Thus, with a large number of observations T, it may (from a practical point of view) suffice to represent the (Bayesian) posterior* (2.7b) *with a single point estimate!*

By this argument, one may catch a sight of a bridge between the two paradigms. The argument is often relevant when $T \to \infty$, not only for the toy case (2.7). It is, however, not completely generally applicable, for instance not if

(i) the number of parameters is large, so that the 'information per parameter' is still low despite a large number of observations T,

(ii) the data cannot determine the parameters uniquely, e.g., $\theta = \{\alpha, \beta\}$, but only information about the product $\alpha \cdot \beta$ is observed (a problem sometimes referred to as non-identifiability),

(iii) the variance in the example model would have been proportional to T instead of 1, which would yield a posterior variance that does not decrease with T.

2.5 Priors and regularization

Let us now consider the role of the prior. The prior has a central role in the Bayesian approach, and is not present at all when computing maximum likelihood point estimates. Its presence may therefore appear as a major difference between the two approaches. The role of the prior is, however, not always crucial when it comes to the practical aspects, as we will discuss in this section.

2.5.1 When the prior does not matter

From the previous section, we have the example of T exchangeable observations of μ with Gaussian noise, where we also could write (cf. (2.7b))

$$p(\mu \mid y) = \mathcal{N}\left(\mu; \tfrac{\sum_{t=1}^{T} y_t}{T+1}, \tfrac{1}{T+1}\right) \approx \mathcal{N}\left(\mu; \tfrac{\sum_{t=1}^{T} y_t}{T}, \tfrac{1}{T}\right) = p(y \mid \mu) = \mathcal{L}(\theta), \tag{2.8}$$

i.e., the posterior and the likelihood function are approximately equal, and the mode of the posterior is approximately the same as the maximum likelihood solution when there is a large amount of data available (T large). One may say that 'the prior is swamped by the data' or refer to the situation as 'stable estimation' (J. O. Berger 1985, Section 4.7.8; Vaart 1998, Section 10.2). It is, however, possible to construct counterexamples, such as pathological cases with Dirac priors etc.

2.5.2 When the prior does matter

The point estimation, and in particular the maximum likelihood approach, might seem intuitively appealing: 'finding the parameter θ for which the data y is as likely as possible' sounds very reasonable. It is, however, important to realize that this is *not* equivalent to 'finding the most likely parameter θ given the data y'. The latter statement is related to the posterior $p(\theta \mid y)$, whereas the former is related to the likelihood function $\mathcal{L}$. Failing to distinguish between these is sometimes referred to as 'the fallacy of the transposed conditional'. We illustrate this by the toy example in Figure 2.1: Consider 8 data points on the form (x, y). We make the decision to model the data using an nth order polynomial and Gaussian measurement noise as

$$p(y \mid \theta) = \mathcal{N}\left(y; c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n, \sigma_n^2\right). \tag{2.9}$$

We let the polynomial order be undecided, meaning that $\theta = \{n, c_0, \ldots, c_n, \sigma_n^2\}$. This is arguably a very flexible model, which is able to take many different shapes: a feature that might be desired by the user who wishes not to make too many restrictions beforehand. The maximum likelihood solution is $n = 7$ (i.e., as many degrees of freedoms as data points), $\sigma_n^2 = 0$ (i.e., no noise) and $c_0, \ldots, c_7$ chosen to fit the data perfectly. This is illustrated by the solid blue line in Figure 2.1. Two suboptimal solutions, *not* maximizing the likelihood function for this flexible model of polynomials with arbitrary orders, are $n = 2$ (green) and $n = 1$ (orange), also shown in Figure 2.1.

Studying Figure 2.1, we may ask ourselves if the 7th order polynomial, the maximum likelihood solution, actually is able to capture and generalize the data well? Indeed all data points are exactly on the blue line, but the behavior in between the data points is not very appealing to our intuition—instead the 2nd or perhaps even

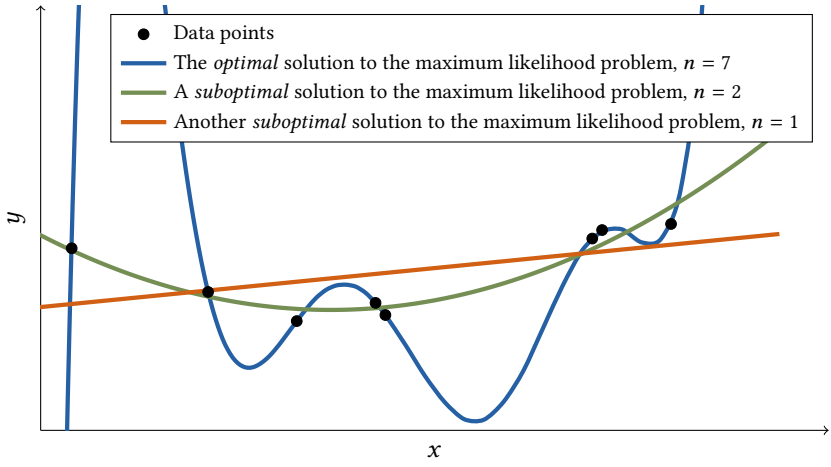


Figure 2.1: *Eight data points marked with black dots, modeled using nth order polynomials and Gaussian noise, where the polynomial order n is undecided. The optimal maximum likelihood solution is* $n = 7$*, with the* 8 *polynomial coefficients chosen such that it (blue curve) fits the* 8 *data points perfectly. Two suboptimal solutions are* $n = 2$ *(green curve) and* $n = 1$ *(orange curve), which—despite their suboptimality in a maximum likelihood sense—both might appear to be more sensible models, in terms of inter- and extrapolating the behavior seen in the data. The key aspect here is that the maximum likelihood solution is explaining the data the best exactly as it is seen; indeed, the blue curve fits the data perfectly. There is, however,* no *claim that the blue curve is the 'most likely solution' (cf. the Bayesian approach). The green and orange curves could, however, have been obtained as regularized maximum likelihood estimates, if a regularization term penalizing large values of n had been added to the objective function* (2.2).

the 1st order polynomial would be more reasonable, even though none of them fit the data exactly. The problem with the blue line, the maximum likelihood solution, is often referred to as *overfitting*. Overfitting occurs when the parameter estimate is adapted to some behavior in the data which we do not believe should be considered as useful information, but rather as stochastic noise.

There are several solutions proposed for how to avoid overfitting, such as aborting the optimization procedure prematurely (early stopping: e.g., Duvenaud, Maclaurin, et al. 2016; Sjöberg and Ljung 1995), some 'information criteria' (e.g., the Akaike information criterion, AIC: Akaike 1974, or the Bayesian information criterion, BIC: Schwarz 1978) or the use of cross-validation (Hastie et al. 2009, Section 7.10). We will, however, try to understand the overfit problem as an unfortunate ignorance during the modeling process: From Figure 2.1, we realize that we may actually have a preference for a lower order polynomial, and our mistake is that we have considered the maximum likelihood approach when we actually had different prior beliefs in different parameter values: we prefer the predictable behavior of a low order polynomial to avoid the strange behavior of a higher order polynomial.[6]

[6]The related philosophical question whether simpler models (in this case, a 1st or 2nd order polynomial) should be preferred over more advanced models (the 7th order polynomial) is often referred to as Occam's razor or the principle of parsimony, a discussion we leave aside.

In the Bayesian framework, on the other hand, the prior $p(\theta)$ is also taken into consideration using Bayes' theorem (2.4). Via Bayes' theorem, it is (on the contrary to maximum likelihood) possible to reason about likely parameters. A sensibly chosen prior would in the example describe a preference for low order polynomials, and the posterior would then dismiss the 7th order polynomial solution (unless it had fitted the data significantly better than a low order polynomial). Hence, there is no Bayesian counterpart to the overfit problem, an advantage that comes at the price of instead having to choose a prior and working with probability distributions rather than point estimates.[7]

Either inspired by the Bayesian approach or heuristically motivated, a popular modification of the maximum likelihood approach is *regularized* maximum likelihood, which appends the likelihood function with a regularization term $R(\cdot)$. The regularization plays a role akin to that of the prior, by 'favoring' solutions of, e.g., low orders. There are a few popular choices of $R(\cdot)$ with a variety of names, such as the $\|\cdot\|_1$ norm (Lasso or L_1 regularization: Tibshirani 1996), the $\|\cdot\|_2$ norm (L_2 or Tikhonov regularization, ridge regression: Hoerl and Kennard 1970; Phillips 1962), or a combination thereof (elastic net regularization: Zou and Hastie 2005).

The connection between regularization and the Bayesian approach can be detailed as follows: If having a scalar θ with prior $\mathcal{N}\left(\theta; 0, \sigma^2\right)$, the logarithm of the posterior becomes

$$\log p(\theta \mid y) = \log p(y \mid \theta) + \log p(\theta) - \log p(y) = C + \log p(y \mid \theta) - |\theta|^2, \tag{2.10}$$

which apart from the constant C is equivalent to the regularized (log) likelihood function

$$\mathcal{L}^r(\theta) = \log p(y \mid \theta) - R(\theta), \tag{2.11}$$

if $R(\cdot) = \|\cdot\|_2$, i.e., L_2 regularization. The same equivalence can be shown for L_1 and the use of a Laplace prior. Thus, regularization gives another connection between the point estimation and the Bayesian approach.

In 1960, Bertil Matérn wrote in his thesis on stochastic models that *'needless to say, a model must often be almost grotesquely oversimplified in comparison with the actual phenomenon studied'* (Matérn 1960, p. 28). As long as the statement by Matérn holds true and the model is rigid and much less complicated than the behavior of the data (which perhaps was the case for most computationally feasible models in 1960), regularization is probably of limited interest. However, if the model class under consideration is more complex and contains a huge number of parameters, overfitting may be an actual problem. In such cases, additional information encoded in priors or regularization has in several areas proven to be of great importance, such as compressed sensing (Eldar and Kutyniok 2012) with applications in, e.g., MRI (Lustig et al. 2007) and face recognition (Wright et al. 2009), machine learning (Hastie et al. 2009, Chapter 5) and system identification (T. Chen et al. 2012, Paper I). The increased access to cheap computational power during the last decades might therefore explain the massive recent interest in regularization.

[7] There are two different perspective one can take when understanding the absence of overfitting in the Bayesian paradigm: Pragmatically seen, any sensible prior will (as argued in the text) have a regularizing effect. From a more philosophical point of view, there is no overfitting since the posterior *by definition* represents our (subjective) beliefs about the situation, and therefore contains nothing but useful information (and hence no overfitting to non-informative noise).

2.5.3 Circumventing the prior assumptions?

Sometimes the user of the Bayesian approach might feel uncomfortable making prior assumptions, perhaps in the interest of avoiding another subjective choice (in addition to the model choice $p(y \mid \theta)$). Several alternatives for avoiding, or at least minimizing the influence of the prior choice, have therefore been investigated.

'Noninformative' priors

Attempts to formulate 'noninformative' priors containing 'no' prior knowledge have been made. In the toy example above, a 'noninformative' prior for σ^2 would intuitively perhaps be a flat prior $p(\sigma^2) \propto 1$ for $\sigma^2 > 0$, since it puts equal mass on all feasible values for σ^2. Apart from the obvious fact that such a density would not integrate to 1, there is also a more subtle and disturbing issue: why should the variance σ^2, and not the standard deviation σ, have a flat prior? In fact, if the prior for the variance σ^2 is $p(\sigma^2) \propto 1$ for $\sigma^2 > 0$, it implies that the prior for the standard deviation σ is $p(\sigma) \propto \sigma$ for $\sigma > 0$, which does not appear very 'noninformative'.

To avoid this undesired effect, a prior that is invariant under re-parametrizations has been proposed, the so-called Jeffreys prior. Jeffreys prior is, however, not always 'noninformative' in the sense that a flat prior intuitively is: Efron (2013) provides a simple example where the Jeffreys prior has a clear and perhaps unwanted influence on the posterior. On this topic, Peterka (1981) writes *'However, it turns out that it is impossible to give a satisfactory definition of "knowing nothing" and that a model of an "absolute ignorant", in fact, does not exist. (Perhaps, for the reason that an ignorant has no problems to solve.)'*.

J. O. Berger (2006) argues, on the other hand, that the process of translating expert knowledge into prior assumptions are typically costly (and not always very crucial to the final result), and 'standard' priors (such as Jeffreys) should for this reason be considered by the practitioner: it is still far more useful than abandoning the Bayesian approach entirely.

Hyperparameters and empirical Bayes

Another alternative is to chose a prior $p(\theta \mid \eta)$ with some undecided *hyper*parameters η, and choose a point estimate $\widehat{\eta}$ which fits the data. This is commonly referred to as *empirical Bayes* or maximum likelihood type II. This combination of point estimation and Bayesian inference is perhaps more pragmatic than faithful to any of the paradigms, but can be seen as a promising combination of them, indeed proven to work well in many situations (see, e.g., Paper III; Bishop 2006; Efron 2013 and references therein).

Since empirical Bayes involves point estimation, overfitting may occur, in that the prior becomes overly adapted to the data. In many situations, this only has minor practical implications (typically not as severe as the situation in Figure 2.1), but the user should be aware of the risk.

Hyperpriors

A third option on the topic of circumventing the explicit formulation of prior assumptions, is to take a Bayesian (rather than a point estimation) approach to hyperpa-

rameters, and formulate *hyper*priors on the hyperparameters η. Then, the inference amounts to inferring

$$p(\eta \mid y) = \int \frac{p(y \mid \theta)p(\theta \mid \eta)p(\eta)}{p(y)}\, d\theta \tag{2.12}$$

rather than $p(\theta \mid y)$. For a subsequent prediction, the prediction $p(y^\star \mid y)$ would instead of (2.6) be

$$p(y^\star \mid y) = \iint p(y^\star \mid \theta)p(\theta \mid \eta)p(\eta \mid y)\, d\theta\, d\eta. \tag{2.13}$$

Obviously such a nested construction does not avoid the choice of a prior, but only defers it to the level of $p(\eta)$ instead of $p(\theta)$, and also adds to the computational complexity of the sometimes already involved computations needed. However, in cases shown to be computationally feasible, interesting and promising results have been obtained, e.g., for the Gaussian process (Chapter 4) model, even with relatively simple choices of hyperpriors: Heinonen et al. (2016), Shah et al. (2014) and Paper VII. An insight from these developments is perhaps that the introduction of a hyperprior $p(\eta)$ may in some models open up for a significantly more flexible modeling process compared to directly choosing a prior $p(\theta)$.

2.6 Summary of the chapter

We have discussed three cornerstones of machine learning from a statistical perspective: data y, models $p(y \mid \theta)$ and two approaches for learning (or, equivalently, inference). The data is given, whereas the model is chosen by us. The choice of model is important in that it heavily influences which results we obtain. With data and model in place, there are still different options for how to learn the parameters θ from the data, either the point estimation or the Bayesian approach, or possibly something in between (regularization etc). We have, however, only discussed different high-level approaches for learning, and we return to some methods for performing the actual computations later on in Chapter 5.

"Models are to be used, not believed."
Henri Theil

3 State-space models

State-space models, or hidden Markov models, is a popular family of models. In this chapter, we introduce the general state-space model, and thereafter also introduce two important special cases, namely the linear and the jump-Markov linear state-space models. (In the next chapter, also a third special case is introduced, namely the Gaussian process state-space model.) We also devote a section to discuss learning for state-space models.

3.1 The general state-space model

At the core of the state-space model we have a Markov process $\ldots, x_{t-1}, x_t, x_{t+1} \ldots$, which evolves as $p(x_{t+1} \mid x_t) = f(x_{t+1} \mid x_t)$, where $f(\cdot \mid \cdot)$ is called the state transition density. We refer to $x_t \in \mathbb{R}^{n_x}$ as the *state*, and $t = 0, \ldots, T$ is an index typically representing time in time-series data, but other interpretations are also possible.

The state x_t may represent the physical state of an object under study, such as the position, speed, heading and acceleration of a vehicle, but it can also be an abstract representation without any clear physical interpretation. The Markov property means that once x_t is known, the previous states $\ldots, x_{t-1}$ do not add any information about the later states $x_{t+1}, \ldots$, i.e.,

$$p(x_{t+1} \mid \ldots, x_{t-1}, x_t) = p(x_{t+1} \mid x_t). \tag{3.1}$$

This Markov assumption is the key for efficiency when learning state-space models.

The state-space model also includes the observation density $g(\cdot \mid \cdot)$, which models the relation between the state x_t and the observation, or output, $y_t \in \mathbb{R}^{n_y}$, as $p(y_t \mid x_t) = g(y_t \mid x_t)$. Note that the Markov property (3.1) does *not* necessarily hold for the observations $\ldots, y_{t-1}, y_t, y_{t+1}, \ldots$!

To summarize the state-space model, we write

$$p(x_{t+1} \mid x_t) = f(x_{t+1} \mid x_t), \tag{3.2a}$$
$$p(y_t \mid x_t) = g(y_t \mid x_t). \tag{3.2b}$$

For completeness, the model also has to include a density $p(x_0)$ for the initial state x_0.

An alternative naming of (3.2) is a hidden Markov model, where 'hidden Markov' refers to the unobserved states x_t that obey the Markov assumption (3.1). The term is, however, also (and perhaps more often) used for models where x_t lives in a discrete space rather than in $\mathbb{R}^{n_x}$.

In the automatic control literature, state-space models are often used with the addition of an exogenous (and known) input signal $u_t \in \mathbb{R}^{n_u}$. Another flavor of (3.2) is the time-varying state-space model, where f and g (and possibly also n_x) explicitly depends on t.

State-space models are typically used to model time-series data $\{y_1, \ldots, y_T\}$ which exhibits some dynamical behavior, i.e., there is a non-trivial correlation between different data points. In a common user case the data $\{y_1, \ldots, y_T\}$ is far from obeying the Markov assumption, but thanks to the state-space model a state sequence $\{x_1, \ldots, x_T\}$ can be (re)constructed. If the model describes the data well, the state sequence will follow the Markov assumption. The reasons for using a state-space model may, at least, be twofold:

- The states bear a physical meaning (e.g., the position and speed of a vehicle) which is of interest.
- In the interest of making predictions, the states x_t provide a compact summary of all relevant history: rather than storing and processing all data $y_1, \ldots, y_t$, it suffices to consider x_t for predicting the future observations $y_{t+1}, \ldots$, provided that the Markov assumption for the states x_t holds.

A relevant question is whether a state-space model, which accurately describes any data set recorded from the same process, always exists? The answer is no; several practically relevant counterexamples exist (e.g., Ljung and Glad 2004, Chapter 7) where the state-space model is insufficient. Nevertheless, the state-space model has proven a practically useful model for many cases.

3.2 Linear Gaussian state-space models

The perhaps most well-studied version of the state-space model is the *linear* state-space model with additive Gaussian noise,

$$x_{t+1} = Ax_t + Bu_t + w_t, \qquad w_t \sim \mathcal{N}(0, Q), \tag{3.3a}$$

$$y_t = Cx_t + Du_t + e_t, \qquad e_t \sim \mathcal{N}(0, R). \tag{3.3b}$$

Here, A, B, C, D, Q and R are matrices of appropriate sizes, and w_t and e_t are stochastic noise, i.i.d. with respect to time. In (3.3) we have deviated from the probabilistic notation, and also included an exogenous input signal u_t, in order to conform with the standard notation in the system identification literature.

Entire books (e.g., Kailath 1980; Rugh 1993) have been written on models of the type (3.3) and its almost equivalent alternative formulation as a transfer function. We make no attempt on covering that literature here.

The linear Gaussian state-space model (3.3) has the advantage that many learning problems can be carried out relatively easy, if not in closed form at least with relatively

efficient algorithms. The downside, however, is its limited expressiveness (even though it has turned out to be very useful, judging from its widespread use) compared to the much more general model (3.2).

A compromise between the expressiveness of the nonlinear state-space model and the analytical tractability of the linear Gaussian state-space model is to keep the linear state transition (i.e., $x_{t+1} = Ax_t + Bu_t + w_t$), but also append (3.3) with some nonlinear feature. Two such examples are the jump-Markov linear state-space models (which we discuss in the next section) and the Wiener and Hammerstein models.

3.3 Jump-Markov linear state-space models

To obtain an expressiveness beyond the linear state-space model (3.3), the *jump-Markov* linear state-space model augments (3.3) with another Markov process (in addition to x_t), namely the mode sequence $\ldots, s_{t-1}, s_t, s_{t+1}, \ldots$. The sequence takes values on the finite discrete space $\{1, 2, \ldots, K\}$, and is defined via its transitions probabilities

$$p(s_{t+1} \mid s_t) = \pi_{s_t, s_{t+1}}. \tag{3.4a}$$

One linear state-space model belongs to each mode (all with the same state dimensions n_x), whose corresponding matrices we denote by a subscript. Conditioned on the mode sequence, the states evolve as (cf. (3.3))

$$x_{t+1} = A_{s_t} x_t + B_{s_t} u_t + w_t, \qquad w_t \sim \mathcal{N}\left(0, Q_{s_t}\right), \tag{3.4b}$$

$$y_t = C_{s_t} x_t + D_{s_t} u_t + e_t, \qquad e_t \sim \mathcal{N}\left(0, R_{s_t}\right). \tag{3.4c}$$

Clearly, (3.4) is a more general model than (3.3) (if $k > 1$), but it is still just a special case of the general state-space model (3.2). Paper IV develops a particular inference algorithm tailored for models on the form (3.4).

3.4 Learning state-space models

Due to the Markov structure of the state-space model (3.2), most inference problems in state-space models take a particular form. We give an introduction here, and Paper I, IV, III, V and VI are all concerned with particular aspects of learning state-space models.

3.4.1 Quantities to learn: states and model parameters

When we discussed learning in Chapter 2, we talked about parameters θ, referring to some unknown numerical quantities in the model that remains to be determined using observed data $\{y_1, \ldots, y_T\}$ (and also inputs $\{u_1, \ldots, u_T\}$ if applicable). It has, however, not yet been said what θ correspond to in the state-space model: By construction, the states x_t are not observed and might be of interest to learn, but there might also be unknown quantities in the model itself, i.e., $f(\cdot \mid \cdot)$ and $g(\cdot \mid \cdot)$ might be parameterized by some unknown *model parameters* ϑ as $f_\vartheta(\cdot \mid \cdot)$ and $g_\vartheta(\cdot \mid \cdot)$.

There is no inherent difference between the states x_t and the model parameters ϑ from a learning perspective: they are both unknown quantities in the state-space model. However, depending on the user's case, different settings are of interest:

(i) The state-space model (i.e., $f(\cdot \mid \cdot)$ and $g(\cdot \mid \cdot)$ in (3.2)) is completely known, and only the state sequence $\{x_1, \ldots, x_T\}$ remains to be determined. We refer to this problem as *state inference*, a problem typically appearing if the model is derived from first principles, implying that the states bear a physical meaning (e.g., the position and velocity of a vehicle).

(ii) Only limited knowledge about the state-space model is present, and we have to infer a set of unknown model parameters ϑ (the states are not available either[1]). We refer to this case as *model parameter learning*, typically occurring if the physical insight about the real process (from which the data is recorded) is limited.

It should be noted that while the model parameters ϑ typically are of a rather low dimension (say[2], 1-20), the entire state sequence $\{x_1, \ldots, x_T\}$ is of dimension $T \cdot n_x$, where $T > 100\,000$ is not unrealistic. For this reason, the state inference and the model parameter learning algorithms have to be designed differently, in order to gain computationally feasible solutions.

The model parameter learning problem contains a spectrum of settings, ranging from learning a single parameter value to determining the entire functional forms of $f(\cdot \mid \cdot)$ or $g(\cdot \mid \cdot)$. In this thesis, Paper IV, Paper III and Paper VI represent the former problem (in particular, inference of the numerical values in (3.4)), whereas paper I deals with the latter case where no parametric form of $f(\cdot \mid \cdot)$ nor $g(\cdot \mid \cdot)$ is known a priori. A very well studied case is inference of the matrices A, B, C, D, Q, R in (3.3), referred to as *linear system identification* (Ljung 1999; Söderström and Stoica 1989).

We assume the state dimension n_x is known. Learning n_x is another problem, not considered in this thesis.

3.4.2 A Bayesian approach or point estimates?

Given the two learning problems in the state-space model, state and model parameter inference respectively, we now turn to the next question: what inference paradigm to use, the Bayesian or the point estimation approach?

The learning approach for the model parameter may vary with the amount of data, properties of the model, intended use, etc., as discussed in Section 2.4. The point estimation approach has historically been favored (e.g., Ljung 1999; Söderström and Stoica 1989), but a discussion in favor of the Bayesian approach is given by Peterka (1981). If the dimension of ϑ is low, a large amount of data is available (T is large), and ϑ is identifiable (Söderström and Stoica 1989, Section 6.4), the maximum likelihood and the Bayesian solution can often be expected to provide similar results in practice (cf. Section 2.4). Also other point estimates than maximum likelihood are popular in the literature, such as the one minimizing the simulation error of the model.

[1] For this reason, the case (i) can be seen as a subproblem of (ii).

[2] We explore much larger cases in Paper I.

For the state inference problem (i.e., finding $x_{1:T}$ when given $y_{1:T}$ and ϑ), we may once again refer back to the discussion in Section 2.4, and note that the problem is of the peculiar form that with more data (i.e., larger T), the dimension of the state sequence $\{x_1, \ldots, x_T\}$ also grows. Thus, the argument from Section 2.4 about concentration of the posterior towards a point as the data record grows is not applicable[3], and we should for this reason be cautious about applying a point estimation approach: we may ignore important uncertainty information if we do so. Perhaps for this reason, the state inference problem is almost exclusively approached by the Bayesian paradigm in the literature, which we will review now.

Bayesian filtering

To alleviate the notation, we use the shorthand symbol $x_{1:t} \triangleq \{x_1, \ldots, x_t\}$, and similar for $y_{1:t}$. The state inference in the Bayesian paradigm (2.4) can be written as

$$p(x_{1:T} \mid y_{1:T}) = \frac{p(y_{1:T} \mid x_{1:T})p(x_{1:T})}{p(y_{1:T})}. \tag{3.5}$$

We may interpret this as (3.2a) providing the prior for the states $p(x_{1:T}) = \prod_{t=1}^{T-1} f(x_{t+1} \mid x_t)$, and (3.2b) giving the model[4] for the data as $p(y_{1:T} \mid x_{1:T}) = \prod_{t=1}^{T} g(y_t \mid x_t)$. In a computational perspective, however, (3.5) is of very limited use. Instead the recursion (see, e.g., Särkkä 2013)

$$p(x_t \mid y_{1:t}) = \frac{1}{p(y_t \mid y_{1:t-1})} g(y_t \mid x_t) \int f(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1} \tag{3.6}$$

has proven useful for computing the (marginal) posterior distributions $p(x_t \mid y_{1:t})$. The denominator in (3.6) only serves the purpose of normalization (and may in some computational schemes be omitted), and the remaining quantities are known. The Kalman filter (below) as well as the particle filter (Chapter 5) are direct implementations of (3.6). We refer to (3.6) as the *Bayesian filtering recursion*, a name commonly used[5]. The term *filtering* refers to the distributions $p(x_1 \mid y_1), p(x_2 \mid y_{1:2}), \ldots, p(x_T \mid y_{1:T})$, as opposed to the (marginal) *smoothing* distributions $p(x_1 \mid y_{1:T}), p(x_2 \mid y_{1:T}), \ldots, p(x_T \mid y_{1:T})$ (note the different conditioning). For computing the smoothing distributions, there is a variety of popular recursions used, for which we refer to the literature, e.g., Lindsten and Schön 2013; Särkkä 2013; Svensson, Schön, et al. 2015.

The Kalman filter

Without doubt, the most popular implementation of the Bayesian filtering recursion is the Kalman filter, named after Rudolf Kálmán (1960). The Kalman filter is nothing but (3.6) written down for the special case of the linear Gaussian state-space

[3] From a time-series perspective, we may use the argument that a data point y_t does not necessarily provide more information about the state x_τ if $t \gg \tau$ or $t \ll \tau$.

[4] For consistency, we should thus refer to (3.2b) as the model and (3.2a) as the prior. Maximum likelihood estimation of some unknown parameters ϑ in $f(\cdot \mid \cdot)$ should then be termed empirical Bayes. Such a terminology would perhaps provide some additional insight, but would probably cause more confusion than clarity in the end.

[5] The Bayesian filtering recursion is commonly also named 'optimal' filtering, where 'optimal' only reflects that it is the Bayesian solution.

model[6] (3.3). We refer to, e.g., Peterka (1981) and Schön and Lindsten (2011) for the derivation and the final equations.

The Kalman filter is often applied also to more general state-space models not exactly on the linear Gaussian form (3.3), due to its relative simplicity. Often modifications are made to approximately handle more general formulations than (3.3), e.g., the extended Kalman filter, the unscented Kalman filter, etc. (Särkkä 2013).

The likelihood for the state space model

We also introduce the probability density for $y_{1:T}$ given ϑ, i.e., $p(y_{1:T} \mid \vartheta)$. When we in Chapter 5 discuss numerical methods for model parameter learning, this expression is at the center of attention:

$$p(y_{1:T} \mid \vartheta) = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \vartheta) = \prod_{t=1}^{T} \int p(y_t \mid x_{t-1}, \vartheta) p(x_{t-1} \mid y_{1:t-1}, \vartheta) dx_{t-1}, \quad (3.7)$$

where we have factorized the expression in such a way that we can see that finding $p(x_t \mid y_{1:t})$ might help in computing $p(y_{1:T} \mid \vartheta)$. Thus, solving the state inference problem in the Bayesian paradigm, i.e., finding $p(x_t \mid y_{1:t})$, may help also when a maximum likelihood point estimate of ϑ is sought!

3.5 Summary of the chapter

This chapter has introduced the general state-space model, as well as some special cases of it; the linear state-space model and the jump-Markov linear state-space model. We have also introduced and discussed two different learning problems for state-space models, the state inference problem and the model parameter learning problem.

[6] The Kalman filter can alternatively also be derived as the optimal (in mean-square-error sense) linear estimator for a more wide class than (3.3).

"I think it is much more interesting to live with uncertainty than to live with answers that might be wrong."
Richard Feynman

4 Gaussian processes

The Gaussian process (GP) defines a probability distribution over functions f, and is commonly used as a model for functional relationships between variables. The GP is tightly connected with the Bayesian paradigm, and conditioning on data y, i.e., updating the prior $p(f)$ into the posterior $p(f \mid y)$, will be our most common usage of the GP model.

The GP is a so-called *nonparametric* model, in that it does not rely on a finite set of parameters θ. A parametric model for f (e.g., a polynomial of finite order) involves a set of parameters θ acting as a 'mid-layer' between the data and the posterior over f. In a parametric model, finding the posterior $p(f \mid y)$ amounts to first find $p(\theta \mid y)$ and then compute $p(f \mid y) = \int p(f \mid \theta)p(\theta \mid y)d\theta$ (cf. (2.6)). In a nonparametric model, however, the distribution $p(f \mid y)$ is computed directly without (explicitly) involving any parameters θ. One may alternatively understand this as the data (in a nonparamteric model) takes the role of the parameters (in a parametric model). The main advantage of a nonparametric model is perhaps that there is no upper limit on 'how much information the model can contain', in contrast to a parametric model.

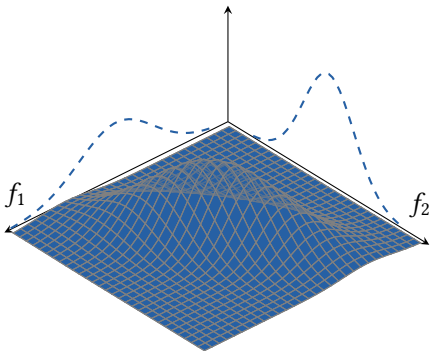
4.1 Introducing the Gaussian process

The nonparametric GP can be understood as a limit of the k-dimensional multivariate Gaussian distribution as k tends to infinity. We will try to follow the intuition behind this limit, in order to develop an understanding for the connections between the Gaussian distribution and the GP. All technical details can be found in the literature (MacKay 1998; Rasmussen and Williams 2006).

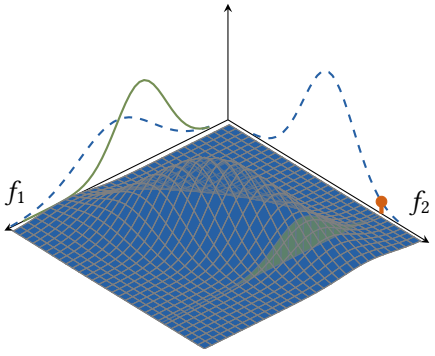
The density for the k-dimensional multivariate Gaussian distribution is

$$\mathcal{N}\left(\overline{f}; \mu, \Sigma\right) = (2\pi)^{-\frac{k}{2}} \det(\Sigma)^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(\overline{f} - \mu)^{\mathsf{T}}\Sigma^{-1}(\overline{f} - \mu)\right), \tag{4.1}$$

where $\overline{f} = [f_1 \ \cdots \ f_k]^{\mathsf{T}}$ is a k-dimensional vector with random scalar elements $f_1, \cdots, f_k$, $\mu \in \mathbb{R}^k$ is the mean, and $\Sigma \in \mathbb{R}^{k \times k}$ is the (positive semidefinite) covariance



(a) *A two-dimensional Gaussian distribution for the random variables f_1 and f_2, with a blue surface plot for the density, and the marginal distribution for each component sketched using dashed blue lines along each axis. Note that the marginal distributions do* not *contain all information about the distribution of f_1 and f_2, since the covariance information is lacking in that representation.*



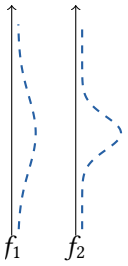
(b) *The conditional distribution of f_1 (green line), when f_2 is observed (orange dot). The conditional distribution of f_1 is given by* (4.3)*, which (apart from a normalizing constant) in this graphical representation also is the green 'slice' of the joint distribution (blue surface). The marginals of the joint distribution from Figure 4.1a are kept for reference (blue dashed lines).*

Figure 4.1: *A two-dimensional multivariate Gaussian distribution for f_1 and f_2 in (a), and the conditional distribution for f_1, when a particular value of f_2 is observed, in (b).*

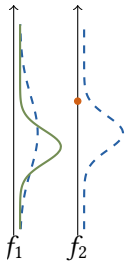
matrix, which means that it has $k + \frac{k(k+1)}{2}$ parameters. In the limit $k \to \infty$, the number of parameters tends to infinity, which can be intuitively understood as the transition from the parametric Gaussian distribution to the nonparametric GP. (The technical challenge in this limit, which we will not fully address here, is the generalization from countable to measurable infinity.)

Considering the Gaussian distribution (4.1), we can partition $\overline{f}$ into $\begin{bmatrix}\overline{f}_1^\mathsf{T} & \overline{f}_2^\mathsf{T}\end{bmatrix}^\mathsf{T}$, and μ and Σ similarly, and then write

$$p\left(\begin{bmatrix}\overline{f}_1\\ \overline{f}_2\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\overline{f}_1\\ \overline{f}_2\end{bmatrix}; \begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right). \tag{4.2}$$

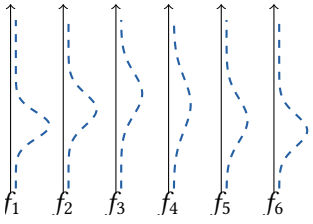


(a) *The marginal distributions for f_1 and f_2 from Figure 4.1a.*

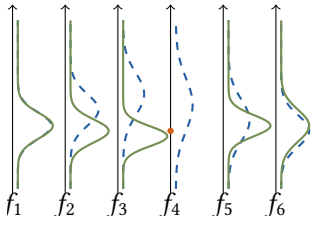


(b) *The distribution for f_1 (green line) when f_2 is observed (orange dot), as in Figure 4.1b.*

Figure 4.2: *The marginals of the distributions in Figure 4.1, here plotted slightly differently. Note that this more compact plot comes with the cost of missing the information about the covariance between f_1 and f_2.*



(a) *A 6-dimensional Gaussian distribution, plotted in the same way as Figure 4.2a, i.e., only its marginals are illustrated.*



(b) *The conditional distribution f_1, f_2, f_3, f_5 and f_6 when f_4 is observed (orange dot), illustrated by its marginals (green lines), cf Figure 4.2b.*

Figure 4.3: *A 6-dimensional Gaussian distribution, illustrated in the same fashion as Figure 4.2.*

If some elements of $\overline{f}$, let us say the ones in $\overline{f}_2$, are observed, the conditional distribution for $\overline{f}_1$ given the observation of $\overline{f}_2$ is

$$p\left(\overline{f}_1 \mid \overline{f}_2\right) = \mathcal{N}\left(\overline{f}_1; \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\overline{f}_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right). \tag{4.3}$$

The conditional distribution is nothing but another Gaussian distribution with closed-form expressions for the mean and covariance. This is particularly useful.

Figure 4.1 shows a 2-dimensional example where a multivariate Gaussian distribution is conditioned on data. In Figure 4.2, we have plotted the marginal distributions from Figure 4.1, to prepare for the generalization to GP. It is also straightforward to plot a 6-dimensional multivariate Gaussian distribution by its margins, akin to Figure 4.2, as we do in Figure 4.3. Bear in mind that to fully illustrate the joint distribution for $f_1, \ldots, f_6$, a 6-dimensional surface plot would be needed, whereas Figure 4.3a only contains the marginal distributions for each component. As earlier, we may also condition the 6-dimensional distribution underlying Figure 4.3a on an observation of, e.g., f_4. Once again, the conditional distribution is another Gaussian distribution, and the marginals of the 5-dimensional distribution are plotted in Figure 4.3b.

In Figure 4.2 and 4.3, we had a distribution over a finite set of discrete points. If we were to study a phenomenon taking values on a finite set of discrete points, like $\{1, 2, 3, 4, 5, 6\}$ in Figure 4.3, we could use this as a probabilistic model. However, our aim is the GP, a probabilistic model for functions on a *continuous* space.

The extension of the Gaussian distribution (defined on a finite set) to the GP (defined on a continuous space) is achieved by replacing the index set $\{1, 2, 3, 4, 5, 6\}$ in Figure 4.3 by a variable x taking values on the continuous real line. In the Gaussian distribution, μ is a vector with k components (e.g., $\mu \in \mathbb{R}^2$ in Figure 4.2, and $\mu \in \mathbb{R}^6$ in Figure 4.3), and similarly for the covariance matrices. In the GP, we replace μ by a mean *function* $\mu(x)$ parameterized by x, and the covariance matrix Σ by a covariance *function* $\kappa(x, x')$ parameterized by x and x'. The GP is then defined as:

Definition (the Gaussian process)**.** *Let $\{x_1, \ldots, x_n\}$ be any finite set of points for which $\mu(x_i)$ and $\kappa(x_i, x_j)$ are defined. Then,*

$$p\left(\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}; \begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} \kappa(x_1, x_1) & \cdots & \kappa(x_1, x_n) \\ \vdots & & \vdots \\ \kappa(x_n, x_1) & \cdots & \kappa(x_n, x_n) \end{bmatrix}\right). \tag{4.4}$$

That is, for any choice of $\{x_1, \ldots, x_n\}$, we have a multivariate Gaussian distribution, just like the one in Figure 4.3. Since $\{x_1, \ldots, x_n\}$ can be chosen arbitrarily on the continuous line, this implicitly defines a distribution for *all* points on that line. Of course, for this definition to make sense, $\kappa(\cdot, \cdot)$ has to be such that a positive semidefinite covariance matrix is obtained for any choice of $\{x_1, \ldots, x_n\}$.

We will use the notation

$$f \sim \mathcal{GP}\left(\mu(\cdot), \kappa(\cdot, \cdot)\right) \tag{4.5}$$

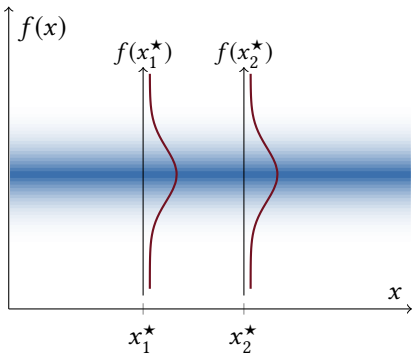
to express that the function f is distributed according to a GP with mean function $\mu(\cdot)$ and covariance function $\kappa(\cdot, \cdot)$. If we want to plot the GP, which we do in Figure 4.4, we may choose $\{x_1, \ldots, x_n\}$ to correspond to the pixels on the screen or the printer dots on the paper, so that it appears as a continuous line to the eye (despite that we actually can access the distribution only in a finite, however arbitrary, set of points).

The perhaps most interesting procedure is the calculation of the conditional distribution given some observations $\{f(x_1^d), \ldots, f(x_m^d)\}$, the GP counterpart to Figure 4.1b, 4.2b and 4.3b. We start by introducing the following more compact notation,

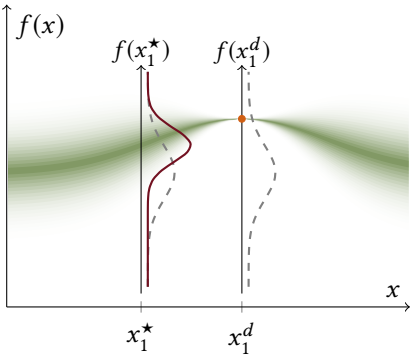
$$x^\star \triangleq \begin{bmatrix} x_1^\star \\ \vdots \\ x_n^\star \end{bmatrix}, \quad K^{\star\star} \triangleq \begin{bmatrix} \kappa(x_1^\star, x_1^\star) & \cdots & \kappa(x_1^\star, x_n^\star) \\ \vdots & & \vdots \\ \kappa(x_n^\star, x_1^\star) & \cdots & \kappa(x_n^\star, x_n^\star) \end{bmatrix}, \tag{4.6a}$$

$$x^d \triangleq \begin{bmatrix} x_1^d \\ \vdots \\ x_m^d \end{bmatrix}, \quad K^{dd} \triangleq \begin{bmatrix} \kappa(x_1^d, x_1^d) & \cdots & \kappa(x_1^d, x_m^d) \\ \vdots & & \vdots \\ \kappa(x_m^d, x_1^d) & \cdots & \kappa(x_m^d, x_m^d) \end{bmatrix}, \tag{4.6b}$$

$$K^{\star d} \triangleq \begin{bmatrix} \kappa(x_1^\star, x_1^d) & \cdots & \kappa(x_1^\star, x_m^d) \\ \vdots & & \vdots \\ \kappa(x_n^\star, x_1^d) & \cdots & \kappa(x_n^\star, x_m^d) \end{bmatrix} = (K^{d\star})^\mathsf{T}. \tag{4.6c}$$



(a) *A GP defined on the real line parameterized by x, not conditioned on any observations. The intensity of the blue color is proportional to the (marginal) density, and the marginal distributions for some $x_1^\star$ and $x_2^\star$ are pictured in red. Akin to Figure 4.3, we only plot the marginal distribution for each $x^\star$, but the GP defines a full joint distribution for all points on the x-axis, even though it is hard to illustrate.*



(b) *The conditional GP distribution given the observation of $f(x_1^d)$ in the point x_1^d corresponding to $x_2^\star$ in (a). The prior distribution from Figure (a) is dashed gray. Note how the conditional distribution adjusts to the observation, both in terms of mean (closer to the observation) and (marginal) variance (smaller in the proximity of the observation, but it remains unchanged in areas distant from it).*

Figure 4.4: *A GP. Figure (a) shows the prior distribution (shaded blue), whereas (b) shows the posterior distribution (shaded green) after conditioning on one observation (orange dot).*

We can use this notation and the definition to write the joint distribution between the values $f(x^d)$ in the points x^d, and the value $f(x^\star)$ in some other points $x^\star$ as

$$p\left(\begin{bmatrix} f(x^\star) \\ f(x^d) \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} f(x^\star) \\ f(x^d) \end{bmatrix}; \begin{bmatrix} \mu(x^\star) \\ \mu(x^d) \end{bmatrix}, \begin{bmatrix} K^{\star\star} & K^{\star d} \\ K^{d\star} & K^{dd} \end{bmatrix}\right). \tag{4.7}$$

Now, as we have observed $f(x^d)$, we can express the posterior distribution for $f(x^\star)$ conditional on the observations as

$$\begin{aligned} &p\left(f(x^\star) \mid f(x^d)\right) = \\ &\mathcal{N}\left(f(x^\star); \mu(x^\star) + K^{\star d}(K^{dd})^{-1}\left(f(x^d) - \mu(x^d)\right), K^{\star\star} - K^{\star d}(K^{dd})^{-1}K^{d\star}\right), \end{aligned} \tag{4.8}$$

i.e., nothing but another multivariate Gaussian distribution for any finite set $x^\star$. We illustrate this by Figure 4.4.

The GP, and in particular (4.8), now provides a way to probabilistically inter- and extrapolate observations under the assumption that the observations are drawn from a Gaussian process. In most practical cases this assumption is most likely not true, but it has nevertheless proven to be a useful model. The typical use of the GP as a modeling tool is illustrated in Figure 4.5.

The Gaussian process can alternatively also be introduced as a nonlinear and nonparametric generalization of linear regression, which perhaps is more standard in the literature (cf. Bishop 2006, Section 6.4; MacKay 1998; Rasmussen and Williams 2006).

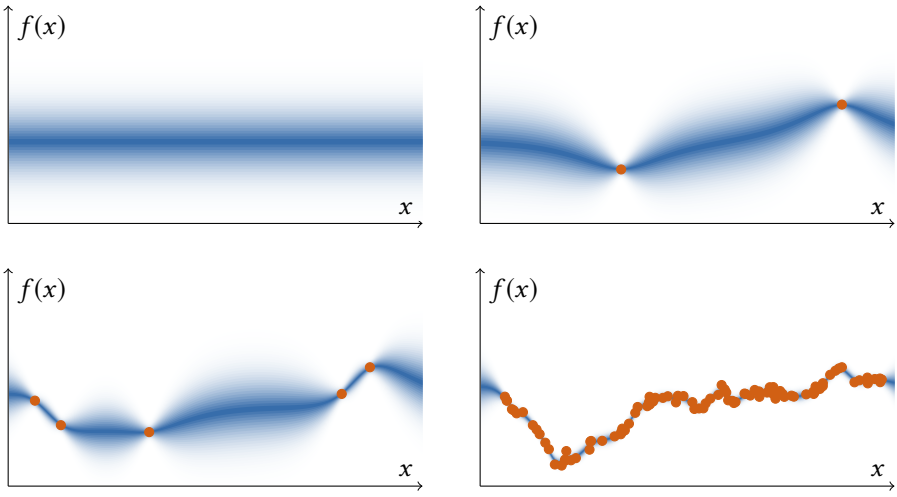


Figure 4.5: *The GP as a modeling tool: the conditional distribution (shaded blue) for $f(x)$ after 0, 2, 5 and 100 observations (orange dots) of $y = f(x) + \text{noise}$. (We have now left our earlier convention of plotting the posterior distribution after conditioning on data in green, since the prior–posterior notion becomes entangled when we sequentially condition on more and more data.)*

4.2 Noise density, mean and covariance functions

We have in the previous section assumed the existence of a mean $\mu(\cdot)$ and covariance function[1] $\kappa(\cdot, \cdot)$. When using the GP to model observed data, these functions somehow have to be chosen by the user. If there is detailed domain knowledge present, it can be incorporated into the covariance function: one such example is Solin, Kok, et al. (2018), where the magnetic field is modeled using a GP covariance function tailored to obey Maxwell's equations (see also Jidling et al. 2018 for general constructions along these lines). In many situations, however, such detailed knowledge is not present, and one has to make a less informed choice of covariance function. Two common choices are the exponentiated quadratic and the Matérn class[2] of covariance functions; their expressions are found in Table 4.1, and their properties have been widely discussed in the literature (e.g., Rasmussen and Williams 2006, Section 4.2) and will not be repeated here. There are also ways to combine different covariance functions into new ones, creating, e.g., periodic covariance functions (Rasmussen and Williams 2006, Section 4.2.4; Duvenaud, Lloyd, et al. 2013). A standard terminology is that if $\kappa(x, x')$ is a function of only $x - x'$, it is referred to as *stationary*, and if it is only a function of $\|x - x'\|$, it is called *isotropic*.

A common choice for the mean function is $\mu(x) = 0$, which at a first glance may seem very restrictive. However, already by inspection of (4.8) or Figure 4.4b, it is clear that the posterior mean (i.e., after conditioning on observations) may be non-zero even though the prior is 0. In fact, $\mu(x) = 0$ appears to work well in many situations.

[1] The covariance function is often referred to as a *kernel* in the literature. We refrain from that terminology here to avoid confusion with the MCMC kernel in the next chapter.

[2] Named after the Swedish statistician Bertil Matérn (1960).

Function	Meaning	Limitations	Examples
Mean $\mu(x)$	Prior assumption about mean	-	C (constant) $a \cdot x$ (linear)
Covariance $\kappa(x, x')$	Assumption on how correlated two x-values are	Must be positive semidefinite	$\exp\left(-\frac{\|x-x'\|^2}{2\ell^2}\right)$ (exponentiated quadratic) $\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|x-x'\|}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}\|x-x'\|}{\ell}\right)$ (Matérn class)
Observation noise	Assumption about noise level in observed data	Analytically tractable only if Gaussian distribution	$\varepsilon = 0$ (noiseless) $\mathcal{N}(\varepsilon; 0, \sigma_n^2)$ (Gaussian distribution)

Table 4.1: *A summary and some examples of functions involved in the GP model.*

In addition to a mean and covariance function, also a third function can be introduced: if $f(x^d)$ is not observed directly, but corrupted by some additive noise ε, as $y^d = f(x^d) + \varepsilon$, the distribution for ε also has to be modeled. In effect, the noise model determines how much the observed data should be 'trusted'. If the noise model is chosen as a Gaussian distribution, it can be incorporated into the covariance function, and (4.8) is still valid. Other alternatives are possible, but gives no closed-form expressions à la (4.8). All functions discussed here, including some typical examples, are summarized in Table 4.1.

4.3 Hyperparameter learning

Most mean functions, covariance functions, and noise distributions contain some parameters, such as the length scale parameter ℓ in the exponentiated quadratic covariance function, or the noise variance σ_n^2 in the Gaussian noise model. We will refer to these as hyperparameters, denoted by η. The hyperparameters are often interpretable (such as length scale or noise level, Rasmussen and Williams 2006, Section 2.3), but due to ignorance (such as limited physical insight) when using the GP as a model, the hyperparameters might effectively be unknown.

As discussed in the learning chapter in Section 2.5.3, there are two common alternatives for how to learn unknown hyperparameters: empirical Bayes or hyperpriors.

4.3.1 Empirical Bayes: Finding a point estimate $\widehat{\eta}$

The empirical Bayes approach can be applied to find a point estimate $\widehat{\eta}$ of η, by maximizing the marginal likelihood[3]

$$p(y^d \mid \eta) = \mathcal{N}\left(y^d; \mu_\eta(x^d), K_\eta^{dd}\right), \tag{4.9}$$

[3]The convention to name (4.9) marginal likelihood is because of the following: In a parametric model, the likelihood is $p(y \mid \theta)$, whereas $p(y \mid \eta)$ is its marginal with respect to θ: $p(y \mid \eta) = \int p(y \mid \eta)p(\theta \mid \eta)d\theta$.

where we have added the subscript η to stress the dependence on the hyperparameters. Note that (4.9) is nothing but a multivariate Gaussian distribution. Due to the way η enters into the problem, this is typically a highly non-convex problem, calling for numerical optimization tools. The major benefit with the point estimate is indeed that a single numerical value $\widehat{\eta}$ is obtained, which is easy to use in, e.g., prediction

$$p(y^\star \mid y^d, \widehat{\eta}) = \mathcal{N}\left(y^\star; \mu_{\widehat{\eta}}(x^\star) + K_{\widehat{\eta}}^{\star d}\left(K_{\widehat{\eta}}^{dd}\right)^{-1}\left(y^d - \mu_{\widehat{\eta}}(x^d)\right), K_{\widehat{\eta}}^{\star\star} - K_{\widehat{\eta}}^{\star d}(K_{\widehat{\eta}}^{dd})^{-1}K_{\widehat{\eta}}^{d\star}\right), \tag{4.10}$$

a lengthy but computationally tractable expression. That a single numerical value for the hyperparameters is chosen is however also a major drawback of the approach. In many cases the 'landscape' of (4.9) is widespread and multimodal, which makes the optimization very hard, and the global optimum sensitive to small changes in data y^d as well as initialization of the optimization procedure. A further discussion with examples is found in Paper VII.

4.3.2 Hyperpriors: Marginalizing out η

The alternative approach to a point estimate is the Bayesian approach with hyperpriors. This approach amounts to inferring the posterior distribution $p(\eta \mid y^d) \propto p(y^d \mid \eta)p(\eta)$, and use this posterior distribution rather than a point estimate in subsequent tasks, such as prediction

$$p(y^\star \mid y^d) = \int \mathcal{N}\left(y^\star; \mu_\eta(x^\star) + K_\eta^{\star d}\left(K_\eta^{dd}\right)^{-1}\left(y^d - \mu_\eta(x^d)\right), K_\eta^{\star\star} - K_\eta^{\star d}(K_\eta^{dd})^{-1}K_\eta^{d\star}\right) p(\eta \mid y^d) d\eta. \tag{4.11}$$

Because of the integral over η in (4.11), we will also refer to this approach as *marginalization*: the integrand of (4.11) is $p(y^\star, \eta \mid y^d)$, and the integral computes the marginal distribution $p(y^\star \mid y^d)$. However, the marginalization (4.11) is in general not analytically tractable, in contrast to the prediction with a point estimate (4.10). Typically not even the posterior distribution $p(\eta \mid y^d)$ itself is tractable, which perhaps is the major drawback of this approach.

A numerical solution for this is to draw Monte Carlo samples from $p(\eta \mid y^d)$, and then approximate the integral in (4.11) with a sum over these samples. One method for acquiring such samples is presented in Paper VII. Other alternatives include variational inference, see, e.g., Titsias and Lázaro-Gredilla (2014).

4.4 Computational aspects

The computational load of (4.8), the main workhorse of the GP model, is dominated by the inversion of the matrix K^{dd}, an operation essentially of complexity $O(m^3)$. Thus, the computational complexity of the GP grows with data in a rather unfavorable way, which may prohibit its use in many applications. A rich literature on approximations is therefore available, e.g., Rasmussen and Williams (2006, Chapter 8); Snelson (2007) and Chalupka et al. (2013). In particular, we will make use of an approximation proposed by Solin and Särkkä (2014) in Paper I.

Essentially, most approximative methods amount to creating a lower-dimensional representation of the data. The lower-dimensional representation resembles a parameter θ. A naïve but illustrative such approximation method is the 'subset of data' method, where only a subset of all data y (chosen either randomly or in a more systematic way) is considered. If the subset is of size $p < m$, the computational load of the GP reduces from $O(m^3)$ to $O(p^3)$.

4.5 Two remarks

The GP provides a widely used and perhaps intuitively appealing model for nonlinear functions. In this section, we will make two remarks that are important to keep in mind when working with GPs for modeling.

4.5.1 A posterior variance independent of observed function values?

The Gaussian process is a flexible model, as seen in, e.g., Figure 4.5. However, in its use with fixed hyperparameters η_0, it has the peculiarity that its variance is independent of the actually observed function values $f(x^d)$, for instance in the predictive variance[4] in (4.8). This is nothing but a direct consequence of the prior assumption that the data was generated by a GP with the specified mean and covariance function with hyperparameters η_0. However, if the hyperparameters are not considered fixed, but inferred from data (either using empirical Bayes or by assuming hyperpriors and marginalizing them), the predictive variance depends on the observed function values, somewhat indirectly via the hyperparameter learning procedure.

4.5.2 What is a typical sample of a GP?

The mean of the GP can be used to characterize the distribution. It is, however, important to remember that the mean is a rather atypical sample of the GP, just as 0 is a very particular sample of a $\mathcal{N}(0, 1)$ distribution. Furthermore, the GP also encodes a smoothness assumption, which is not very clear in the plotting style of Figure 4.5: in Figure 4.6, 5 samples are drawn from these distributions, where it is clear that also the correlation (along the x-axis) contains important information in the GP distribution as well. This is essentially the same point as we discussed when we considered the 2-dimensional Gaussian distribution in Figure 4.1, and only plotted its marginal distributions in 4.2 even though a full joint distribution is specified by the model.

[4] This point has its counterpart in the Kalman filter, Section 3.4.2, where also the predictive covariance is independent of the observed measurements. It is essentially the very same phenomenon, since the Kalman filter can be interpreted as a Gaussian process (Solin and Särkkä 2014).

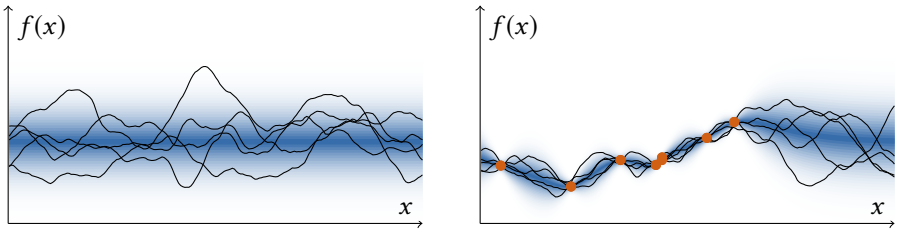


Figure 4.6: *Five samples of the GP from Figure 4.5 (with a Matérn $\nu = 3/2$ covariance function). Note, in particular, that the samples are more wiggly than the mean function: a reminder that the blue shades do not contain all information, but is only the marginal distribution for each x (cf. Figure 4.1 and 4.2).*

4.6 Gaussian-process state-space models

A combination of the state-space model and the GP is the relatively recent GP state-space model,

$$p(x_{t+1} \mid x_t) = \mathcal{N}\left(x_{t+1}; f(x_t), Q\right), \qquad f \sim \mathcal{GP}\left(\mu_f(\,\cdot\,), \kappa_f(\,\cdot\,,\,\cdot\,)\right), \tag{4.12a}$$

$$p(y_t \mid x_t) = \mathcal{N}\left(y_t; g(x_t), R\right), \qquad g \sim \mathcal{GP}\left(\mu_g(\,\cdot\,), \kappa_g(\,\cdot\,,\,\cdot\,)\right). \tag{4.12b}$$

The somewhat cumbersome notation should simply be read as 'x_{t+1} equals a GP of x_t plus Gaussian noise', and similar for y_t. The promising feature of the model is that it combines the nonparametric flexibility of a GP with the dynamical nature of the state-space model, allowing for complex and highly nonlinear dynamical phenomena to be described. Currently the best overview of the GP state-space model is probably found in the thesis by Frigola-Alcade (2015).

Due to the somewhat entangled use of the GP in (4.12a), where the output of the GP, x_{t+1}, is the input at the next time step, the inference problem becomes relatively hard. Frigola, Lindsten, et al. (2013) proposed a conceptually interesting but computationally brutal solution, and the subsequent Frigola, Y. Chen, et al. (2014) and Paper I (and in particular, its predecessor Svensson, Solin, et al. 2016) present further developments in different directions; For a numerical example which took Frigola, Lindsten, et al. (2013) about 10 hours of compuational time, Svensson, Solin, et al. 2016 and Paper Frigola, Y. Chen, et al. (2014) only requires a few minutes.

4.7 Summary of the chapter

This chapter has introduced the GP as a generalization of the multivariate Gaussian distribution. A crucial aspect is that some important expressions are available in closed form, such as (4.10). The use of the GP in machine learning is as a model for (nonlinear) functions f, of which we only have observed the values in a few points (cf. Figure 4.5). It can also be combined with the state-space model into the GP state-space model (4.12).

"Expose yourself to as much randomness as possible."
Ben Casnocha

5 Monte Carlo methods for machine learning

Monte Carlo methods are a class of numerical methods named after the casino in the capital of Monaco (Figure 5.1). They originated in physics research with disputable purposes during the first half of the 20th century. An accessible introduction from that era, still well worth reading, is 'The Monte Carlo method' by Metropolis and Ulam (1949). Today, Monte Carlo methods are established tools within many different scientific fields, in particular in machine learning and some related areas.

Monte Carlo methods are useful when the mathematical computations are not analytically tractable, meaning, e.g., that an integral lacks a closed-form solution. There are also other alternatives, such as the variational approach (see Blei et al. 2016 for an overview). The idea in the variational approach is to impose additional assumptions until the modified problem becomes tractable. This thesis, however, focus on the Monte Carlo approach.

We give in this chapter an overview and introduction to sequential Monte Carlo (SMC) and Markov chain Monte Carlo (MCMC) in general, as well as their application for learning in state-space models.

5.1 The Monte Carlo idea

Consider a probability density $\pi(\cdot)$ over the space of a parameter θ, that is defined in such a way that the analysis of interest (e.g., computing the variance of θ) is not analytically tractable. The Monte Carlo idea is to approximately represent π by random samples (an empirical measure). Those random samples should be generated such that their properties resemble the properties of the distribution π. The samples are nothing but numerical values stored in a computer, and it is (hopefully) easier to analyze those samples than analyzing π directly.

Figure 5.1: *Casino de Monte-Carlo in Monaco. A place of gambling and broken dreams, and moreover the source of the name 'Monte Carlo method'. Photo: Andreas Svensson.*

Formally, we introduce the notation of N weighted[1] samples $\{\theta^{(i)}, w^{(i)}\}_{i=1}^{N}$. This collection of weighted samples is a *Monte Carlo* (or *particle*) *approximation* of the density π if it holds that the empirical measure is 'close' to π, by which we mean

$$\frac{1}{\sum_{j=1}^{N} w^{(j)}} \sum_{i=1}^{N} w^{(i)} \mathbb{I}_A(\theta_i) \approx \int_A \pi(\theta) d\theta \tag{5.1}$$

for every measurable set A, with equality almost surely in the limit as $N \to \infty$. This is illustrated in Figure 5.2. If it is possible to draw samples from π directly, one may simply draw N such samples and set all weights to 1. If samples cannot be drawn from π directly, there are alternatives, of which we will review some.

For some methods, (5.1) does not only hold in the limit as $N \to \infty$, but also when taking the expectation over different realization of the Monte Carlo method itself as $\mathbb{E}\left[\frac{1}{\sum_{j=1}^{N} w^{(j)}} \sum_{i=1}^{N} w^{(i)} \mathbb{I}_A(\theta_i)\right] = \int_A \pi(\theta) d\theta$ for a fixed N. That is a stronger property, which holds for, e.g., rejection sampling but not $p(x_t \mid y_{1:t}, \vartheta)$ in a particle filter.

[1]Note that we use non-normalized weights throughout this chapter.

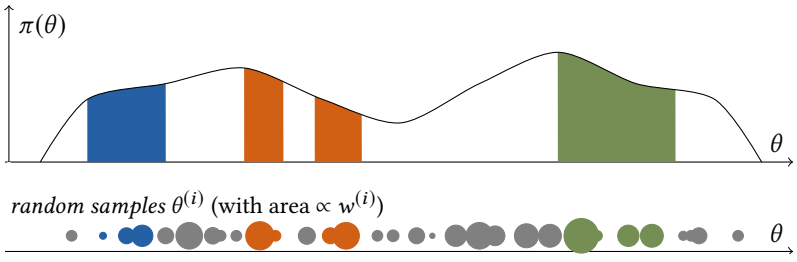


Figure 5.2: *The Monte Carlo idea: A probability density $\pi(\theta)$ at the top, and weighted random samples of that distribution below (the area of each sample is proportional to its weight). Each color is a choice of A in (5.1), so we expect each colored area in the upper part of the figure (i.e., $\int_A \pi(\theta) d\theta$) to be roughly proportional to the area of its corresponding samples (i.e., $\sum_{i=1}^{N} w^{(i)} \mathbb{I}_A(\theta_i)$).*

Algorithm 1: Bootstrap particle filter

Input: State space model $f(\cdot \mid \cdot)$, $g(\cdot \mid \cdot)$, $p(x_0)$, and data $y_{1:T}$.

Output: Weighted samples $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N_x}$ from $p(x_t \mid y_{1:t}, \vartheta)$ for $t = 1, \ldots, T$.

1 Draw $x_0^{(i)} \sim p(x_0)$ and set $w_0^{(i)} = 1$

2 **for** $t = 1$ **to** T **do**

3 Draw $a_t^{(i)}$ with $\mathbb{P}\left(a_t^{(i)} = j\right) \propto w_{t-1}^{(j)}$ *resampling,* $\{x_{t-1}^{a_t^{(i)}}, 1\} \approx p(x_{t-1} \mid y_{1:t-1}, \vartheta)$

4 Draw $x_t^{(i)}$ from $f(x_t | x_{t-1}^{a_t^{(i)}})$ *propagation,* $\{x_t^{(i)}, 1\} \approx p(x_t \mid y_{1:t-1}, \vartheta)$

5 Set $w_t^{(i)} = g(y_t | x_t^{(i)})$ *weighting,* $\{x_t^{(i)}, w_t^{(i)}\} \approx p(x_t \mid y_{1:t}, \vartheta)$

6 **end**

All statements with (i) *are for* $i = 1, \ldots, N_x$. *The notation* $\approx$ *means that the weighted samples on the left hand side are approximately (in the meaning of* (5.1)*) the density on the right hand side.*

5.2 The bootstrap particle filter

As a popular example of a non-trivial Monte Carlo algorithm, we start by introducing the particle filter. The origin of the particle filter is to be found in Gordon et al. (1993) and Stewart and McCarty (1992). It is a Monte Carlo implementation of the Bayesian filtering recursion (3.6) solving the *state inference* problem, i.e., computing the filtering distributions $p(x_1 \mid y_1, \vartheta), \ldots, p(x_T \mid y_{1:T}, \vartheta)$ (cf. the generic π in the previous section) in the state-space model, when the model parameters ϑ are known. An animated beginner's introduction to the particle filter is found in Svensson (2013), and there is a myriad of written introductions, e.g, Arulampalam et al. (2002), Gustafsson et al. (2002), Haykin and Freitas (2004), and Särkkä (2013). A good overview (but perhaps not a first introduction) is provided by Doucet and Johansen (2011).

The key idea of the particle filter is to propagate a set of N_x weighted particles $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N_x}$ (samples of the state) along the time dimension t, by propagating them from time $t-1$ to the next time step t by drawing samples from $f(\cdot \mid x_{t-1}^{(i)})$ (3.2a), and adapt them to the measurements according to $g(y_t \mid x_t^{(i)})$ (3.2b). An important step in the implementation is also the resampling step, where (loosely speaking) particles with small weights are discarded and particles with large weights are duplicated. This is summarized in Algorithm 1, the so-called bootstrap particle filter.[2]

5.2.1 Resampling

The resampling step ensures that computational resources are spent in the most interesting parts of the state-space, and that a situation where all but one particle eventually have zero weights is avoided. This can be seen as deciding a genealogy of the particles, i.e., how many descendants a certain particle will have, and which of

[2] The connection between Algorithm 1 and the straps aimed for helping when putting on a pair of leather boots may seem rather weak. The history involves the saying 'pull oneself up by one's bootstraps' (often, but probably falsely, attributed to the fictional character Baron Munchausen by Raspe 1786), which is the background for the naming of the statistical idea 'bootstrap' (Efron 1979), which has a close connection to the resampling.

the particle branches that will become extinct. (The genealogy analogue can be particularly helpful when considering the inference problem of the entire sequence $x_{1:T}$; clearly, x_t is correlated with x_{t-1}). To obtain a consistent algorithm, the resampling scheme has to be constructed such that

$$\mathbb{E}\left[\#\text{ of descendants to } x_{t-1}^{(i)}\right] = \sum_{j=1}^{N_x} \mathbb{P}\left(a_t^{(j)} = i\right) \propto w_{t-1}^{(i)}. \tag{5.2}$$

There are alternatives when it comes to designing a resampling algorithm that fulfills (5.2), see, e.g., Douc and Cappé (2005) and Murray, Lee, et al. (2015) for overviews. It is also possible to design resampling schemes where the duplicated particles are not assigned unit weights (as implicitly done in Algorithm 1) see, Paige et al. (2014) for an example. This is also the underlying key observation for the novel method proposed in Paper VI.

In all non-trivial cases the resampling step is a stochastic procedure, which unfortunately also adds to the variance of the final estimates obtained from the particle filter. It is therefore common to perform the resampling only when needed, which is usually determined by monitoring the so-called effective sample size (ESS, Kong et al. 1994) $\left(\sum_{i=1}^{N_x}(w^{(i)}/\sum_{j=1}^{N_x} w^{(j)})^2\right)^{-1}$, taking values between 1 and N_x, and perform resampling only when the ESS falls below a certain threshold, e.g, $N_x/2$. If an adaptive resampling scheme is used, a slight modification of the weight update in Algorithm 1 is needed.

5.2.2 Positive and unbiased estimates of $p(y_{1:T} \mid \vartheta)$

The particle filter was first used as a tool for solving the filtering problem in nonlinear state-space models, but it can also be used to estimate the likelihood $p(y_{1:T} \mid \vartheta)$ (3.7). The estimate is created from the weights $w_t^{(i)}$ in Algorithm 1 as

$$\widehat{p}_{N_x}(y_{1:T} \mid \vartheta) = \prod_{t=1}^{T}\left(\frac{1}{N_x}\sum_{i=1}^{N_x} w_t^{(i)}\right), \tag{5.3}$$

where we emphasize in the notation that it is a Monte Carlo-based estimate based on N_x particles. It can be shown (see, e.g., Appendix A) that (5.3) is an unbiased estimate of the likelihood, i.e.,

$$\mathbb{E}\left[\widehat{p}_{N_x}(y_{1:T} \mid \vartheta)\right] = p(y_{1:T} \mid \vartheta), \tag{5.4}$$

This claim is not asymptotic in N_x, but holds for any finite number $N_x \geq 1$ of particles. The expectation in (5.4) is over realizations of Algorithm 1 itself, i.e., the randomness involved in the propagation and resampling step. It further holds (as can be seen by inspection of (5.3)) that $\widehat{p}(y_{1:T} \mid \vartheta) \geq 0$. This can, as we will see, be used in algorithms for learning the model parameters ϑ. We will also mention a few more theoretical properties about Algorithm 1 later in Section 5.4.4.

Algorithm 2: Markov chain Monte Carlo sampler

Input: A transition kernel $\mathcal{K}$ with stationary distribution π.
Output: Unweighted samples $\{\theta^{(k)}\}_{k=0}^{K}$ from (in the limit $K \to \infty$) π.

1 Draw $\theta^{(0)}$ arbitrarily
2 **for** $k = 1$ **to** K **do**
3 | Draw $\theta^{(k)}$ from $\mathcal{K}\left(\theta \mid \theta^{(k-1)}\right)$
4 **end**

5.3 The Markov chain Monte Carlo sampler

Let us now leave the particle filter and the state-space model aside, and return to the general problem we formulated in Section 5.1. That is, we are interested in drawing conclusions about some analytically intractable distribution $\pi(\theta)$, typically a posterior $p(\theta \mid y)$. If we can not draw samples from π directly, but instead evaluate π point wise (i.e., query the value of $\pi(\theta)$ for any θ, at least up to proportionality), we can use the Markov chain Monte Carlo (MCMC) methodology to generate samples from π. The MCMC sampler is an algorithm that stochastically explores the θ-space, and thereby defines a stochastic process (a Markov chain) in that space. We denote the realization of the stochastic process, i.e., the outcome of one run of the algorithm, as $\{\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(K)}\}$. An MCMC sampler is designed such that $\{\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(K)}\}$ becomes an (unweighted) particle approximation of π in the limit[3] as $K \to \infty$.

We briefly review the essential ideas of how to construct an MCMC sampler. A more complete treatment of the topic is found in, e.g., Tierney (1994), Andrieu, Freitas, et al. (2003), Robert and Casella (2004, Chapter 6) and Liang et al. (2010). The key ingredient in an MCMC algorithm is a transition kernel $\mathcal{K}(\cdot \mid \cdot)$ with a certain stationary distribution. A transition kernel is any function $\mathcal{K}(\cdot \mid \cdot)$ (where both arguments live in θ-space) such that $\mathcal{K}(\cdot \mid \theta')$ is a probability density for every θ'. A stationary distribution π of $\mathcal{K}$ is such that $\mathcal{K}(\cdot \mid \pi) = \pi(\cdot)$, where we use the shorthand notation $\mathcal{K}(\cdot \mid \pi) \triangleq \int \mathcal{K}(\cdot \mid \theta')\,\pi(\theta')d\theta'$. If $\mathcal{K}$ fulfills certain technical conditions, it can be applied in Algorithm 2 to produce samples from π in the limit as $K \to \infty$. The conditions are essentially that $\mathcal{K}$ should not admit periodic cycles and that for any θ and θ', there should exist an n such that $\mathcal{K}^n(\theta \mid \theta') > 0$ (where $\mathcal{K}^n$ denotes an n-fold iterative application of $\mathcal{K}$),

The transition kernel $\mathcal{K}$ in MCMC is often defined by an algorithm itself, rather than a closed form expression. Many different methods for designing MCMC kernels exist, such as slice sampling (Neal 2003), Hamiltonian Monte Carlo (Duane et al. 1987; Neal 2011), the bouncy particle sampler (Bouchard-Côté et al. 2017; Peters and With 2012), etc. We will now introduce the perhaps two most basic and standard algorithms for designing $\mathcal{K}$, Metropolis-Hastings and Gibbs sampling.

[3]The asymptotic behavior as $K \to \infty$ is (if the sampler fulfills certain conditions) independent of the initialization $\theta^{(0)}$, but in practice a so-called burn-in period of some length K_b typically has to be considered, and the corresponding first K_b samples are discarded. For the performance in practice, it can be crucial to consider and analyze this transient behavior of the MCMC sampler. We will, however, not reflect any more on this, but refer to, e.g., Chapter 12 of Robert and Casella (2004).

Algorithm 3: Metropolis-Hastings transition kernel $\mathcal{K}$

Input: $\theta^{(k-1)}$
Output: $\theta^{(k)}$
1 Draw θ' from $q(\theta \mid \theta^{(k-1)})$ — *A candidate for $\theta^{(k)}$*
2 Compute $\alpha = \min\left(\frac{\gamma(\theta')}{\gamma(\theta^{(k-1)})}\frac{q(\theta^{(k-1)} \mid \theta')}{q(\theta' \mid \theta^{(k-1)})}\right)$ — *The acceptance probability*
3 Set $\theta^{(k)} = \begin{cases} \theta' & \text{with probability } \alpha \\ \theta^{(k-1)} & \text{with probability } 1-\alpha \end{cases}$ — *Decide if candidate is accepted or not*

Algorithm 4: Gibbs transition kernel $\mathcal{K}$

Input: $\theta^{(k-1)}$
Output: $\theta^{(k)}$
1 Draw $\theta_1^{(k)}$ from $p(\theta_1 \mid \theta_2^{(k-1)})$
2 Draw $\theta_2^{(k)}$ from $p(\theta_2 \mid \theta_1^{(k)})$

5.3.1 The Metropolis-Hastings kernel

The Metropolis-Hastings algorithm (named[4] after Nicholas Metropolis, Rosenbluth, et al. 1953 and Wilfred K. Hastings 1970) is a popular plug-in kernel, only requiring that π can be evaluated point wise up to proportionality as $\pi(\theta) = \gamma(\theta)/Z$. A proposal density $q(\cdot \mid \theta^{(k-1)})$ is also needed, from which samples of θ can be drawn, and is either symmetric ($q(\theta \mid \theta') = q(\theta' \mid \theta)$) or can be evaluated point wise. The Metropolis-Hastings algorithm is outlined by Algorithm 3. The idea is to sample a candidate θ' from the proposal, and always (with an adjustment to account for bias caused by the proposal) accept the candidate as $\theta^{(k)}$ if $\pi(\theta') \geq \pi(\theta^{(k-1)})$. However, also if $\pi(\theta') < \pi(\theta^{(k-1)})$, the candidate may be accepted with a certain acceptance probability, designed in a way to create the correct stationary distribution. If the support of the proposal $q(\cdot \mid \theta^{(k-1)})$ covers the support of π, it can be proved (e.g., Robert and Casella 2004, Theorem 7.2) that π is the stationary distribution of Algorithm 3, and it can be used in the MCMC sampler (Algorithm 2) to generate samples from π.

5.3.2 The Gibbs kernel

The Metropolis-Hastings algorithm has an element of rejection sampling, effectively a trial and error approach where a large fraction of the computational resources may be spent on computing $\gamma(\theta')$ for proposals that are never accepted. The Gibbs algorithm (named after Josiah Willard Gibbs, coined by S. Geman and D. Geman 1984) is an alternative kernel that does not suffer from this drawback, but produces samples that are always accepted (but may on the other hand suffer from a high autocorrelation). The Gibbs kernel requires that θ can be partitioned as $\theta = \{\theta_1, \theta_2, \ldots, \theta_M\}$ (preferably with low cross-dependence between the partitions) so that it is possible to draw samples from $p(\theta_m \mid \theta \setminus \theta_m) = \frac{\pi(\theta)}{\int \pi(\theta) d\theta_m}$ for every partition m. Then, this sampling is

[4] It should, however, be remembered that the original article has 5 authors, and Metropolis happened to be the first one in the alphabetical ordering.

iterated over all m, as summarized by Algorithm 4 for the case $M = 2$. The analysis for the Gibbs sampler is, however, rather intricate (see, e.g., Robert and Casella 2004, Chapter 9 and 10 and references therein), but the resulting Markov chain can under certain conditions be proven to fulfill the necessary conditions for producing samples of π when used in the MCMC sampler (Algorithm 2) as $K \to \infty$.

It is also possible to construct combinations of the Metropolis-Hastings and Gibbs algorithm (Liang et al. 2010, Section 3.4; Müller 1991 and Robert and Casella 2004, Section 10.3), although care must be taken in order not to change the stationary distribution (Dyk and Jiao 2014).

5.3.3 Convergence

The convergence of Algorithm 2 in the asymptotic case $K \to \infty$ follows, under some additional assumptions on $\mathcal{K}$, a central limit theorem. For a measurable test function $h(\theta)$, the difference between the true (and after-sought) expectation $\mathbb{E}[h(\theta)]$ and the sample-based estimate of it $h_K(\{\theta^{(k)}\}_{k=1}^K) = \frac{1}{K}\sum_{k=1}^K h(\theta^{(k)})$ is

$$\sqrt{K}\left(h_K(\{\theta^{(k)}\}_{k=1}^K) - \mathbb{E}[h(\theta)]\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2_{\text{MCMC}}(h, \pi)\right) \tag{5.5}$$

where $\sigma^2_{\text{MCMC}}(h)$ is a bounded function of h and π (Tierney 1994, Theorem 4 and 5; Robert and Casella 2004, Theorem 6.65 and 6.67).

5.4 The Sequential Monte Carlo sampler

As discussed in Section 3.4.1, the state inference in a state-space model is a particular learning problem. Similarly, the particle filter can be seen as a particular instance of the more general sequential Monte Carlo (SMC) method. SMC can also be formulated for other types of models, such as graphical models (Naesseth, Lindsten, and Schön 2014).

The most generic formulation of SMC can be found in the Feynman-Kac formalism (Del Moral 2004; Del Moral and Doucet 2014). Yet another instance of SMC is the SMC sampler (Del Moral, Doucet, and Jasra 2006), here presented as Algorithm 5. The SMC sampler is formulated for the same problem as the MCMC sampler, namely to sample from a static density π which only can be evaluated point wise up to proportionality.

The particle filter targets the filtering distributions (3.6) sequentially[5]. For the SMC sampler, only a static distribution π is typically of user interest, but a sequence of probability distributions $\{\pi_0, \pi_1, \ldots, \pi_P\}$ is introduced as an intermediate tool, and the particles are then propagated along this sequence. It is assumed that all π_p can be evaluated up to proportionality, i.e., $\pi_p(\theta) = \gamma_p(\theta)/Z_p$, where $\gamma_p(\theta)$ can be computed for any θ.

[5]Hence the name *sequential* Monte Carlo.

Algorithm 5: Sequential Monte Carlo sampler

Input: Sequence of densities $\{\pi_0, \pi_1, \ldots, \pi_P\}$ on the form $\pi_p(\theta) = \gamma_p(\theta)/Z_p$, with $\gamma_p(\theta)$ possible to evaluate point wise.

Output: Weighted samples $\{\theta_p^{(i)}, w_p^{(i)}\}_{i=1}^{N_\theta}$ from $\pi_p(\theta)$, for each $p = 0, \ldots, P$.

1 Draw $\theta_0^{(i)} \sim \gamma_0(\theta_0)$ and set $w_0^{(i)} = 1$

2 **for** $p = 1$ **to** P **do**

3 Draw $a_p^{(i)}$ with $\mathbb{P}\left(a_p^{(i)} = j\right) \propto w_{p-1}^{(j)}$ *resampling,* $\{\theta_{p-1}^{a_p^{(i)}}, 1\} \approx \pi_{p-1}$

4 Draw $\theta_p^{(i)}$ from $\mathcal{K}_p(\theta_p \mid \theta_{p-1}^{a_p^{(i)}})$ *propagation,* $\{\theta_p^{(i)}, 1\} \approx \mathcal{K}_p(\cdot \mid \pi_{p-1})$

5 Set $w_p^{(i)} = W_p(\theta_p^{(i)}, \theta_{p-1}^{a_p^{(i)}})$ *weighting,* $\{\theta_p^{(i)}, w_p^{(i)}\} \approx \pi_p$

6 **end**

All statements with (i) are for $i = 1, \ldots, N_\theta$, and $\mathcal{K}_p$ can be taken as Algorithm 3.

5.4.1 Connection to particle filters

We can retrieve the bootstrap particle filter (Algorithm 1) from the SMC sampler (Algorithm 5) by letting $\theta = x$, $P = T$, $\pi_p(\theta_p) = p(x_t \mid y_{1:t})$, $W_p(\theta_p, \theta_{p-1}) = g(y_t \mid x_t)$ and $K_p(\theta_p, \theta_{p-1}) = f(x_t \mid x_{t-1})$. More advanced versions of the particle filter are also possible to formulate, where $f(x_t \mid x_{t-1})$ is replaced by a more general proposal density, and the weighting is adjusted accordingly (see, e.g., Doucet and Johansen 2011 for an overview). The aim of such a construction is typically to decrease the variance of the particle weights and the final estimates.

5.4.2 Constructing a sequence $\{\pi_p\}_{p=0}^P$

The particle filter sequentially targets the densities $p(x_t \mid y_{1:t}, \vartheta)$. The SMC sampler, on the other hand, targets a static density π. Therefore, we have to construct an artificial sequence of distributions $\{\pi_p\}_{p=0}^P$ (with π_0 easy to sample from and $\pi_P = \pi$) along which the particles can be propagated. Preferably, the distance between any consecutive π_{p-1} and π_p should be 'small' in order to guide the particles well towards $\pi_P = \pi$. This idea resembles simulated annealing (also introduced by Metropolis, Rosenbluth, et al. 1953) and continuation methods (Richter and DeCarlo 1983).

If $\pi(\theta)$ is a posterior, i.e., $\propto p(\theta)p(y \mid \theta)$, one option is to construct $\{\pi_p\}_{p=0}^P$ as the likelihood-tempered sequence

$$\pi_p \propto p(\theta)p(y \mid \theta)^{p/P}. \tag{5.6}$$

Another alternative is the data-tempered sequence

$$\pi_p \propto p(\theta \mid y_{B_{0:p}}), \tag{5.7}$$

where $\{B_p\}_{p=0}^P$ is a sequence with batches of the data y, such that B_0 is empty and $B_{0:P}$ contains all data y. A third option is proposed in Paper V.

5.4.3 Propagating the particles

For the SMC sampler, there is no underlying state-space model as for the particle filter that can be used to propagate or weight the particles. Therefore, W_p and $\mathcal{K}_p$ has to be chosen by the user. Different alternatives are possible (Del Moral, Doucet, and Jasra 2006, Section 3.3), but one choice is

$$\mathcal{K}_p(\cdot \mid \cdot) \text{ Metropolis-Hastings kernel with stationary distribution } \pi_{p-1}, \tag{5.8a}$$

$$W_p(\theta_p, \theta_{p-1}) = \frac{\pi_p(\theta_{p-1})}{\pi_{p-1}(\theta_{p-1})}, \tag{5.8b}$$

which can be shown to yield a consistent algorithm. The SMC sampler with the choices (5.6-5.8) is a rather general scheme, which can be applied to a broad range of problems. One example is found in Paper VII and another in Del Moral, Doucet, and Jasra (2012a). We will later also review how it can be applied to the parameters ϑ in the state-space model, resulting in the SMC^2 algorithm (Chopin, Jacob, et al. 2013).

5.4.4 Convergence

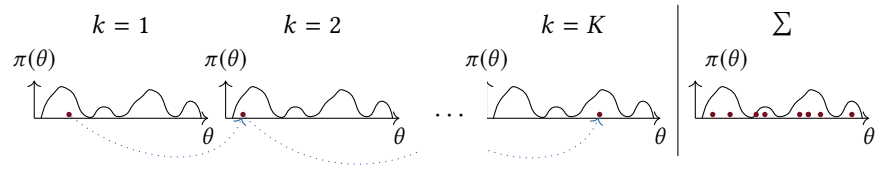
We have already discussed an important property of the particle filter (Algorithm 1), namely that $\widehat{p}_{N_x}(y_{1:T} \mid \vartheta)$ is unbiased for any finite $N_x \geq 1$. In a similar manner, it is possible to construct an unbiased estimator also for the normalizing constants Z_p in the SMC sampler. Results concerning the long-term stability of SMC, and in particular particle filters, also exist (Douc, Moulines, et al. 2014; Whiteley 2013).

Akin to the MCMC case, results are available also for the asymptotic case $N_\theta \to \infty$. As for MCMC (Section 5.4.4), we can for every measurable test function $h(\theta)$ establish (under some technical assumptions) the central limit theorem for Algorithm 5

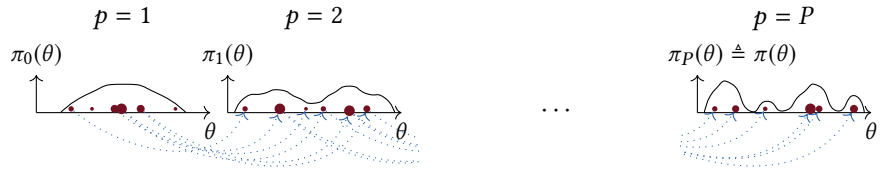
$$\sqrt{N_\theta}\left(h_{N_\theta}(\{\theta_p^{(i)}, w_p^{(i)}\}_{i=1}^{N_\theta}) - \mathbb{E}\left[h(\theta)\right]\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2_{\text{SMC}}(h, \pi)\right), \tag{5.9}$$

when $N_\theta \to \infty$, where $\sigma^2_{\text{SMC}}(h)$ is a bounded function of h and π (Del Moral, Doucet, and Jasra 2006, Proposition 2). This result is applicable to any SMC algorithm (Chopin 2004; Del Moral 2004), and in particular also for the particle filter in Algorithm 1. The case when resampling is performed only adaptively (as discussed in Section 5.2.1; also applicable to Algorithm 5) is more intricate to analyze, but similar results have been presented by Del Moral, Doucet, and Jasra (2012b).

To summarize, the bottom line is that the SMC sampler has a central limit theorem on the same form as MCMC.



(a) *The MCMC idea: propagate a single sample (red dot) through the landscape of π, such that its random trace (summarized in the rightmost plot) eventually becomes samples of the distribution of interest π. That is, the chain has to 'visit' areas where π is large more often than areas where π is small. It will most likely have visited every mode of π as $K \to \infty$, but not necessarily within a reasonable finite time (i.e., before the user's computational budget is consumed).*



(b) *The SMC idea: propagate a set of N_x ($N_x = 6$ in this illustration) particles (samples, red dots) through a sequence of P distributions $\pi_0, \ldots \pi_P$, to eventually end up with samples from the distribution of interest $\pi(\cdot) \triangleq \pi_P(\cdot)$. By making a 'smooth' transition from the easy-to-sample distribution π_0 to the distribution of interest π the hope is that the samples represent π more efficiently than in the MCMC setting (by exploring different modes in parallel, etc.).*

Figure 5.3: *The key concept of the MCMC (a) and SMC samplers (b). The idea of MCMC is to make a (more or less informed) stochastic walk with a single particle in θ-space such that the walk will be proportional to the density π. The SMC idea is to propagate a whole bunch of particles through an evolving landscape (cf. how the particle filter solves the state inference problem), which after a pre-defined number of iterations P ends up in π.*

5.5 Markov Chain or Sequential Monte Carlo?

MCMC has been around since the 1950's, whereas SMC is younger than the author of this thesis[6]. With that perspective, it is perhaps not surprising that the MCMC sampler can essentially be seen as the special case of the SMC sampler with $\pi_p = \pi$ and $N_\theta = 1$. For this reason, we may also expect (as confirmed by Svensson and Schön 2016 for a particular case) that the SMC sampler requires more user effort, in terms of implementation time. It is also worth highlighting that the number of iterations K in the MCMC sampler (Algorithm 2) does not have to be specified beforehand, but the algorithm can be run until the computational budget is consumed, a so-called anytime algorithm. For the SMC sampler (Algorithm 5), both N_θ and P have to be specified before beforehand, and is thereby not an anytime algorithm.

The different underlying ideas on how the samples are drawn are illustrated in Figure 5.3. Any attempt to claim superiority of one approach in general is probably fruitless. However, a rudimentary knowledge about both alternatives can probably help in making wise choices: the historical timeline might have given MCMC an advantage.

[6]Who would like to claim that he is rather young.

5.6 Monte Carlo for state-space model parameters ϑ

The particle filter (Section 5.2) with its various extensions and generalizations provides an often unbeaten Monte Carlo solution for inferring the states x_t in the state-space model (3.2). (MCMC may, however, be beneficial for some particular problem settings, such as the case in Svensson, Schön, et al. 2015.) For the problem of finding model parameters ϑ, on the other hand, the particle filter cannot provide a solution on its own[7]. However, the particle filter can be a very useful building block of an MCMC or SMC sampler to construct well-performing and theoretically consistent algorithms for inferring the posterior $p(\vartheta \mid y_{1:T})$, as well as the maximum likelihood estimate $\widehat{\vartheta}$.

5.6.1 MCMC for nonlinear state-space models: PMCMC

For inferring ϑ in linear state-space models (3.3), MCMC can be used essentially out of the box. The use of a Metropolis-Hastings sampler is shown in Ninness and Henriksen (2010) (although formulated for transfer functions; a state-space model formulation is found in Schön, Lindsten, et al. 2015, Example 4), and the Gibbs sampler in Wills et al. 2012. In both cases the Kalman filter (and some extensions of it) provides the required expressions for $p(\vartheta \mid y_{1:T})$ (for the Metropolis-Hastings solution) and $p(x_{1:T} \mid \vartheta, y_{1:T})$ (for the Gibbs solution). In the Gibbs solution we also need an expression for $p(\vartheta \mid x_{1:T}, y_{1:T})$, which is (if the conjugate prior is used) provided by the matrix normal inverse Wishart distribution (Appendix B).

The two cases in the previous paragraph are special cases in that the required expressions are available analytically. For the general nonlinear state-space model (3.2), neither the likelihood $p(y_{1:T} \mid \vartheta)$ nor the conditional distribution $p(x_{1:T} \mid \vartheta, y_{1:T})$ are available in closed form, nor can they be computed exactly. It turns out that the particle filter provides a good approach for approximating these distributions, in the combined particle-filter-within-MCMC framework[8], PMCMC (Andrieu, Doucet, et al. 2010).

Pseudo-marginal Metropolis-Hastings

What happens to the Metropolis-Hastings sampler (Algorithm 3) if $\pi(\theta)$ cannot be evaluated exactly, but only stochastically estimated $\widehat{\pi}(\theta)$? A naïve approach would perhaps be to pretend that $\widehat{\pi}(\theta)$ is exact (i.e., contains no stochastic element) and apply Algorithm 3. (Another attempt could be to average over a few realizations of $\widehat{\pi}(\theta)$ for every θ, and use that average when computing the acceptance probability α.) It turns out (Andrieu and Roberts 2009), somewhat surprisingly, that if $\widehat{\pi}(\theta)$ is positive and unbiased, i.e., $\mathbb{E}\left[\widehat{\pi}(\theta)\right] = \pi(\theta)$ and $\widehat{\pi}(\theta) > 0$, using $\widehat{\pi}(\theta)$ as if it were exact (the approach suggested above) creates a consistent algorithm, in the sense that the stationary distribution of Algorithm 3 remains unchanged!

[7] If the unknown parameters have a low dimensionality, they can possibly be considered as part of the state x_t (and modeled to be slowly time-varying), and the problem is thereby transferred to a vanilla state inference setting. This solution highlights the (mild) arbitrariness of splitting unknown parameters θ in the state-space model into model parameters ϑ and states x_t.

[8] The original meaning of PMCMC is simply 'particle Markov chain Monte Carlo', but 'particle-filter-within-MCMC' is a more explanatory interpretation.

This quite remarkable fact can be proven by handling the randomness of $\widehat{\pi}(\theta)$ explicitly by introducing another random variable v, and considering $\widehat{\pi}(\theta)$ to be deterministic when conditioned on v. Then, it is possible to show that the Metropolis-Hastings sampler targets an extended distribution $p(\theta, v)$, and that $\pi(\theta)$ can be obtained by integrating v out. Thus the name of the approach, pseudo-marginal.

Particle marginal Metropolis-Hastings

Following the pseudo-marginal Metropolis-Hastings approach with $\widehat{\pi}(\theta) \propto \widehat{p}_{N_x}(y_{1:T} \mid \vartheta)p(\vartheta)$ from (5.3), the particle marginal Metropolis-Hastings approach is obtained (Andrieu, Doucet, et al. 2010, Section 2.4.2). Although not affecting the asymptotical properties, the choice of the number of particles $N_x \geq 1$ and the proposal density $q(\cdot \mid \cdot)$ are crucial for its practical performance. Some discussion on how to choose N_x can be found in Andrieu, Doucet, et al. (2010), and some design methods for q can be found in Dahlin, Lindsten, et al. (2015). Two beginner's introduction to particle Metropolis-Hastings are provided by Dahlin and Schön (2016) and Schön, Svensson, et al. (2018).

Particle Gibbs

It is also possible to construct a Gibbs sampler, Algorithm 4, for state-space model parameters ϑ. Such a construction is possible by taking θ in Algorithm 4 as $\{x_{1:T}, \vartheta\}$, i.e., iteratively sample $x_{1:T}^{(k)}$ conditional on the model parameters $\vartheta^{(k-1)}$, and the model parameters $\vartheta^{(k)}$ conditional on the state $x_{1:T}^{(k)}$. Thus, we need to draw samples from $p(x_{1:T} \mid \vartheta^{(k)})$ as well as $p(\vartheta \mid x_{1:T}^{(k)})$.

For certain state-space model structures (e.g., the linear model in Wills et al. 2012, the models in Section 7 in Lindsten, Jordan, et al. 2014 and the model in Paper I), $p(\vartheta \mid x_{1:T}^{(k)})$ is available in closed form and possible to sample from. If that is not the case, other sampling strategies can be used, see, e.g., Example 8 of Schön, Lindsten, et al. (2015).

To sample approximately from $p(x_{1:T} \mid \vartheta^{(k)})$, a particle filter can be used: the approximation is due to the finite number of particles N_x in the particle filter. However, with a slightly more involved Gibbs sampling scheme it is possible to draw MCMC samples of $x_{1:T}$ with a kernel (constructed using the so-called conditional particle filter) with exactly $p(x_{1:T} \mid \vartheta^{(k)})$ as its stationary distribution. A particularly well-performing conditional particle filter construction has proven to be the one introduced by Lindsten, Jordan, et al. (2014), the conditional particle filter with ancestor sampling. We will not detail this construction any further here, but we refer to Andrieu, Doucet, et al. (2010) and Lindsten, Jordan, et al. (2014) for all technical details on this so-called particle Gibbs construction.

5.6.2 Particle Gibbs for maximum likelihood estimation

If the maximum likelihood estimate $\widehat{\vartheta}$ (rather than the posterior $p(\vartheta \mid y_{1:T})$) is of interest, Lindsten (2013) and Paper III presents a combination of particle Gibbs and a stochastic approximation (Robbins and Monro 1951) version of the expectation maximization (EM) algorithm (Dempster et al. 1977). The construction makes use of particle

Gibbs only for the state inference problem, and uses the stochastic approximation EM framework (Delyon et al. 1999; Kuhn and Lavielle 2004) for the maximum likelihood estimation of ϑ. The use of EM for maximum likelihood estimation of ϑ in nonlinear state-space models has been around since at least Ghahramani and Roweis (1998), and the combination of SMC and EM for this purpose has been proposed by Cappé et al. (2005), Olsson et al. (2008), and Schön, Wills, et al. (2011). The combination of particle Gibbs and stochastic approximation EM, as proposed by Lindsten (2013), improves the convergence properties and reduces the computational load compared to previous algorithms. A more detailed introduction is given in Paper III, and Papers I and IV both use the method for two particular model structures.

5.6.3 SMC for state-space model parameters: SMC^2

In the same spirit as the MCMC methodology can be used for sampling the posterior $p(\vartheta \mid y_{1:T})$ of the state-space model parameters, so can the SMC sampler. The SMC sampler can be applied directly to a linear state-space model, akin to the MCMC sampler case, since $p(y_{1:T} \mid \vartheta)$ is explicitly available from the Kalman filter. A natural way to construct a sequence of densities is the data-tempered alternative $P = T$, $\pi_0(\vartheta) = p(\vartheta)$, $\pi_1(\vartheta) = p(\vartheta \mid y_1)$, $\ldots$, $\pi_T(\vartheta) = p(\vartheta \mid y_{1:T})$. An alternative construction, for the special case when the state-space model has very little measurement noise, is proposed in Paper V. For the general case with a nonlinear state-space model, the particle filter is required to approximately evaluate $p(y_{1:t} \mid \vartheta)$ as $\widehat{p}_{N_x}(y_{1:t} \mid \vartheta)$, yielding the SMC^2 algorithm[9] (Chopin, Jacob, et al. 2013; Fulop and Li 2013). For propagating the particles in step 4 in Algorithm 5, the particle Metropolis-Hastings kernel (Algorithm 3) can be used. Once again the unbiasedness $\mathbb{E}\left[\widehat{p}_{N_x}(y_{1:t} \mid \vartheta)\right] = p(y_{1:t} \mid \vartheta)$ is key to obtaining a consistent algorithm; the details are found in Section 3.1 in Chopin, Jacob, et al. (2013).

This somewhat involved construction leaves the user with several design choices, for instance the trade-off between the number of particles N_x in the particle filters and the number of particles N_ϑ at the SMC sampler level. Chopin, Ridgway, et al. (2015) have suggested how to automatically adapt these numbers.

SMC^2 is not to be confused with nested SMC (Naesseth, Lindsten, and Schön 2015), which is a general framework for using SMC to construct proposal densities within an SMC algorithm.

5.7 Summary of the chapter

This chapter has introduced some Monte Carlo ideas useful for machine learning, and we have in particular considered the particle filter (for state inference in the state-space model) as well as the MCMC and SMC samplers (for general problems). We have also introduced the combinations PMCMC and SMC^2, both primarily aimed for learning model parameters ϑ in state-space models.

[9] The naming should be read as 'SMC square', i.e., SMC to the power of two; a particle filter (an SMC algorithm) is used within an SMC sampler (another SMC algorithm).

"Wait, what if these quote marks are inside out, so everything in the rest of the document is the quotation and this *part isn't?* Duuuuude."
Randall Munroe

6

Conclusions and future work

This chapter contains some overall conclusions of the research presented in Paper I–VII. In addition to what is written in each paper, this chapter also has an outlook into further possible research directions.

6.1 Conclusions

The contributions of this thesis include applications of state-of-the art learning methods to non-trivial models, new versions of learning methods themselves, as well as a contribution to model validation methodology. At the heart of all methods lies the use of Monte Carlo approximations to handle otherwise intractable integrals, and often the sequential Monte Carlo method in particular. The results are often promising, but the existence of this thesis suggests that:

- The work is most likely not done yet, but there are probably more problems where (sequential) Monte Carlo can make a difference than the ones included in this thesis. With increasing access to computational power, together with a raised interest for the Bayesian approach, there are probably plenty of opportunities.
- Applications and tweaking of Monte Carlo, and sequential Monte Carlo in particular, is apparently complicated enough to be topic for a doctoral thesis. There is probably work to be done in packaging and providing sequential Monte Carlo as an off-the-shelf method to practitioners not having a PhD degree on the topic.

6.2 Future work

As suggested above, sequential Monte Carlo is potentially useful for a wider range of situations than those where it is used today. Its use is, however, probably limited by the relative complexity of implementing it. A natural research direction would therefore be to provide a 'user interface' to sequential Monte Carlo, accessible for an application domain expert not knowledgeable within Monte Carlo methods. Indeed, such initiatives already exist (I have partly been funded by such a project: ASSEMBLE, Murray and Schön 2018), but the work is nowhere close to be finished.

Parts of this thesis are focused around Bayesian learning in somewhat complicated state-space models. However, to the best of my knowledge, little research has been done on how to interpret and efficiently transform the posterior $p(\vartheta \mid y)$ into a 'posterior' for the dynamical behavior (such as the input-output relationship). Such results would be of particular interest when the parameters ϑ themselves do not bear a physical meaning (such as in Paper I).

A common argument for the Bayesian approach is the quantification of the present uncertainty provided by the posterior distribution. It is, however, important to bear in mind that the posterior distribution, and thereby also the uncertainty, is *conditional* on the choice of model. (This conditioning is not only a formality, but critical for the results obtained.) This reflection was part of the inspiration behind Paper II, but several (fundamental) questions still remain: Are there more aspects of a model that should (and can) be validated than the one proposed in Paper II? Is it possible to judge the severity of model misspecifications from a posterior distribution? As a tough and applied question, consider decision making in self-driving cars based on posterior uncertainties from a 'Bayesian neural network' (whatever that would mean): what aspects of the data/reality have to be present in the model (the neural network), in order to guarantee that the posterior uncertainty represents something meaningful?

"Policy should always be rooted in unbiased science."

Christine Todd Whitman

A

The unbiased estimator $\widehat{p}_{N_x}(y_{1:T})$

This chapter contains a proof of the fact that (5.3),

$$\widehat{p}_{N_x}(y_{1:T} \mid \vartheta) = \prod_{t=1}^{T} \left(\frac{1}{N_x} \sum_{i=1}^{N_x} w_t^{(i)} \right), \tag{A.1}$$

with $w_t^{(i)}$ generated by the bootstrap particle filter Algorithm 1 in Chapter 5, is an unbiased estimator of the likelihood $p(y_{1:T} \mid \vartheta)$ (3.7) of a state-space model with model parameters ϑ, for any finite $N_x \geq 1$. With unbiasedness, we mean $\mathbb{E}\left[\widehat{p}_{N_x}(y_{1:T} \mid \vartheta)\right] = p(y_{1:T} \mid \vartheta)$, where the expectation is over the randomness in the particle filter algorithm itself. This result was first presented by Del Moral (2004, Section 7.4.2) and is important to many parameter learning strategies, such as particle marginal Metropolis-Hastings (Section 5.6.1) and Paper VI. The proof here follows closely that of Pitt et al. (2012), which is written for the more general case of the auxiliary particle filter.

In the following, ϑ will be suppressed in the notation, since all expressions are conditioned on ϑ. We start by introducing the estimator[1]

$$\widehat{p}_{N_x}(y_t \mid y_{1:t-1}) = \frac{1}{N_x} \sum_{i=1}^{N_x} w_t^{(i)}, \tag{A.2}$$

which has the natural property that $\prod_{t=1}^{T} \widehat{p}_{N_x}(y_t \mid y_{1:t-1}) = \widehat{p}_{N_x}(y_{1:T})$. We also define $\widehat{p}_{N_x}(y_{t-h:t} \mid y_{1:t-h-1})$ naturally as $\prod_{t'=t-h}^{t} \widehat{p}_{N_x}(y_{t'} \mid y_{1:t'-1})$ for $h \geq 0$.

The structure of the proof is as follows: First, in Lemma 1, it will be proved that

$$\mathbb{E}\left[\widehat{p}_{N_x}(y_t \mid y_{1:t-1}) \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] = \sum_{i=1}^{N_x} \frac{w_{t-1}^{(i)}}{\sum_{j=1}^{N_x} w_{t-1}^{(j)}} p(y_t \mid x_{t-1}^{(i)}), \tag{A.3}$$

[1] Note the somewhat subtle notation: p denotes probability densities, whereas $\widehat{p}_{N_x}$ denotes deterministic functions (which we distinguish by their different arguments) of quantities stochastically generated by the particle filter. The point with the proof is to show that the $\widehat{p}_{N_x}$-function (A.1) is an unbiased estimator of the corresponding p.

i.e., $\widehat{p}_{N_x}(y_t \mid y_{1:t-1})$ (the contribution to (A.1) from iteration t of the particle filter) is unbiased, if conditioned on a realization of particles from the previous iteration at time $t-1$. Then, in Lemma 2, we prove that it also holds for $h \geq 1$ sequential iterations of the particle filter, once again conditioned on a realization of particles at time $t-h-1$. Finally, by letting $h = T$, we conclude in Theorem 1 that if $\{x_0^{(i)}\}_{i=1}^{N_x}$ are unbiased samples from $p(x_0)$, then must $\widehat{p}_{N_x}(y_{1:T})$ (A.1) also be unbiased.

Lemma 1. *With the definition of $\widehat{p}_{N_x}(y_t \mid y_{1:t-1})$ in* (A.2)*, it holds that*

$$\mathbb{E}\left[\widehat{p}_{N_x}(y_t \mid y_{1:t-1}) \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] = \sum_{i=1}^{N_x} \frac{w_{t-1}^{(i)}}{\sum_{j=1}^{N_x} w_{t-1}^{(j)}} p(y_t \mid x_{t-1}^{(i)}). \tag{A.4}$$

Proof.

$$\begin{aligned}
&\mathbb{E}\left[w_t^{(j)} \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] = \\
&\qquad = \mathbb{E}\left[\mathbb{E}\left[w_t^{(j)} \mid a_t^{(j)}, \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] = \\
&= \mathbb{E}_{a_t^{(j)}}\left[\mathbb{E}_{x_t^{(j)} \sim f(x_t^{(j)} \mid x_{t-1}^{a_t^{(j)}})}\left[g(y_t \mid x_t^{(j)}) \mid a_t^{(j)}, \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] = \\
&= \mathbb{E}_{a_t^{(j)}}\left[p(y_t \mid x_{t-1}^{a_t^{(j)}}) \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] = \sum_{k=1}^{N_x} p(a_t^{(j)} = k \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}) p(y_t \mid x_{t-1}^{(k)}).
\end{aligned} \tag{A.5}$$

Then,

$$\begin{aligned}
\mathbb{E}\left[\sum_{j=1}^{N_x} w_t^{(j)} \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] &= \sum_{j=1}^{N_x} \mathbb{E}\left[w_t^{(j)} \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}\right] = /(\text{A.5})/ = \\
&= \sum_{k=1}^{N_x} \left(\sum_{j=1}^{N_x} p(a_t^{(j)} = k \mid \{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x})\right) p(y_t \mid x_{t-1}^{(k)}) = \\
&= /(5.2)/ = N_x \sum_{k=1}^{N_x} \frac{w_{t-1}^{(k)}}{\sum_{i=1}^{N_x} w_{t-1}^{(i)}} p(y_t \mid x_{t-1}^{(k)}),
\end{aligned} \tag{A.6}$$

and the lemma follows. □

We have now proved that given a realization of weighted particles $\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^{N_x}$ representing $p(x_{t-1} \mid y_{t-1})$, the estimator $\widehat{p}_{N_x}(y_t \mid y_{1:t-1})$ (A.2), i.e., the contribution to (A.1) from one single iteration of the particle filter for the following time t, is unbiased. We now present the next lemma, concerning the corresponding unbiasedness of $\widehat{p}_{N_x}(y_{t-h:t} \mid y_{1:t-h-1})$.

Lemma 2. *With the definitions of $\widehat{p}_{N_x}(y_t \mid y_{1:t-1})$ and $\widehat{p}_{N_x}(y_{t-h:t} \mid y_{1:t-h-1})$ from above, it holds that*

$$\mathbb{E}\left[\widehat{p}_{N_x}(y_{t-h:t} \mid y_{1:t-h-1}) \mid \{x_{t-h-1}^{(i)}, w_{t-h-1}^{(i)}\}_{i=1}^{N_x}\right] = \sum_{k=1}^{N_x} \frac{w_{t-h-1}^{(k)}}{\sum_{i=1}^{N_x} w_{t-h-1}^{(i)}} p(y_{t-h:t} \mid x_{t-h-1}^{(k)}). \tag{A.7}$$

Proof. The proof is by induction. For $h = 0$, (A.7) is true by Lemma 1. We now assume that (A.7) holds also for an arbitrary h, and show that it implies that (A.7) also holds for $h + 1$. For $h + 1$, the left hand side of (A.7) is

$$
\begin{aligned}
&\mathbb{E}\left[\widehat{p}_{N_x}(y_{t-h-1:t} \mid y_{1:t-h-2}) \mid \{x_{t-h-2}^{(i)}, w_{t-h-2}^{(i)}\}_{i=1}^{N_x}\right] = \\
&\quad = \mathbb{E}\left[\widehat{p}_{N_x}(y_{t-h:t} \mid y_{1:t-h-1})\widehat{p}_{N_x}(y_{t-h-1} \mid y_{1:t-h-2}) \mid \{x_{t-h-2}^{(i)}, w_{t-h-2}^{(i)}\}_{i=1}^{N_x}\right] = \\
&\quad\quad = \mathbb{E}\Big[\mathbb{E}\left[\widehat{p}_{N_x}(y_{t-h:t} \mid y_{1:t-h-1}) \mid \{x_{t-h-1}^{(i)}, w_{t-h-1}^{(i)}\}_{i=1}^{N_x}\right] \times \\
&\quad\quad\quad \widehat{p}_{N_x}(y_{t-h-1} \mid y_{1:t-h-2}) \mid \{x_{t-h-2}^{(i)}, w_{t-h-2}^{(i)}\}_{i=1}^{N_x}\Big] = \\
&\quad\quad\quad = /\text{Induction assumption and (A.2)}/ = \\
&= \mathbb{E}\left[\sum_{j=1}^{N_x} \frac{w_{t-h-1}^{(j)}}{\sum_{i=1}^{N_x} w_{t-h-1}^{(i)}} p(y_{t-h:t} \mid x_{t-h-1}^{(j)}) \frac{1}{N_x} \sum_{i=1}^{N_x} w_{t-h-1}^{(i)} \mid \{x_{t-h-2}^{(i)}, w_{t-h-2}^{(i)}\}_{i=1}^{N_x}\right] = \\
&\quad = \mathbb{E}\left[\sum_{j=1}^{N_x} w_{t-h-1}^{(j)} p(y_{t-h:t} \mid x_{t-h-1}^{(j)}) \frac{1}{N_x} \mid \{x_{t-h-2}^{(i)}, w_{t-h-2}^{(i)}\}_{i=1}^{N_x}\right] = \\
&\quad = /\text{akin to (A.5)} : \mathbb{E}\left[w_{t-h-1}^{(j)} p(y_{t-h:t} \mid x_{t-h-1}^{(j)}) \mid \{x_{t-h-2}^{(i)}, w_{t-h-2}^{(i)}\}_{i=1}^{N_x}\right] = \\
&\quad\quad = \sum_{k=1}^{N_x} p(a_{t-h-1}^{(j)} = k \mid \{x_{t-h-2}^{(i)}, w_{t-h-2}^{(i)}\}_{i=1}^{N_x}) p(y_{t-h-1:t} \mid x_{t-h-2}^{(k)})/ = \\
&\quad = \frac{1}{N_x} \sum_{k=1}^{N_x} \left(\sum_{j=1}^{N_x} p(a_{t-h-1}^{(j)} = k \mid \{x_{t-h-2}^{(i)}, w_{t-h-2}^{(i)}\}_{i=1}^{N_x})\right) p(y_{t-h-1:t} \mid x_{t-h-2}^{(k)}) = \\
&\quad\quad = \sum_{k=1}^{N_x} \frac{w_{t-1}^{(k)}}{\sum_{i=1}^{N_x} w_{t-1}^{(i)}} p(y_{t-h-1:t} \mid x_{t-h-2}^{(k)}), \qquad \text{(A.8)}
\end{aligned}
$$

and the lemma follows. □

We have proved that the result from Lemma 1 also holds for $h \geq 1$ iterations of the particle filter. From Lemma 2, we now have that (with $t = T$ and $h = t - 1$)

$$
\mathbb{E}\left[\widehat{p}_{N_x}(y_{1:T}) \mid \{x_0^{(i)}, w_0^{(i)}\}_{i=1}^{N_x}\right] = \sum_{k=1}^{N_x} \frac{w_0^{(k)}}{\sum_{i=1}^{N_x} w_0^{(i)}} p(y_{1:T} \mid x_0^{(k)}) = \sum_{k=1}^{N_x} p(y_{1:T} \mid x_0^{(k)}) \frac{1}{N_x}. \qquad \text{(A.9)}
$$

If $x_0^{(k)} \sim p(x_0)$, we can conclude that

$$
\mathbb{E}\left[\frac{1}{N_x} \sum_{k=1}^{N_x} p(y_{1:T} \mid x_0^{(k)})\right] = \int p(y_{1:T} \mid x_0) p(x_0) dx_0 = p(y_{1:T}). \qquad \text{(A.10)}
$$

We can now formulate the following theorem (where we have re-introduced ϑ to the notation).

Theorem 1. *For the estimator $\widehat{p}_{N_x}(y_{1:T} \mid \vartheta)$, as defined by* (A.1) *and Algorithm 1 in Chapter 5, it holds that*

$$
\mathbb{E}\left[\widehat{p}_{N_x}(y_{1:T} \mid \vartheta)\right] = p(y_{1:T} \mid \vartheta), \qquad \text{(A.11)}
$$

for any finite $N_x \geq 1$.

"The most important questions of life (...) are indeed for the most part only problems of probability."

Pierre-Simon Laplace

B

The matrix normal inverse Wishart distribution in linear regression

This appendix gives an introduction to the matrix normal inverse Wishart distribution (and its scalar case normal inverse gamma). The normal inverse gamma and some of its generalizations is often in the literature highlighted as the conjugate prior for a data likelihood model on the form $p(y \mid \mu, \sigma^2) = \mathcal{N}\left(y; \mu, \sigma^2\right)$, where both μ and σ^2 are unknown. In this appendix, we will derive the expressions for the slightly more involved case of a linear regression model, i.e., $p(y \mid a, \sigma^2) = \mathcal{N}\left(y; ax, \sigma^2\right)$, with x known and a and σ^2 unknown, and also its multivariable extension. Similar expressions can also be found in Quintana (1987).

B.1 The matrix normal and inverse Wishart distributions

In this section, we introduce the matrix normal inverse Wishart distribution, by first considering the scalar case, and thereafter its multivariable generalization. Introductions can also be found in Dawid (1981) and Press (1982). We will assume a basic familiarity with the Gaussian and the gamma distributions.

B.1.1 The scalar case: $\mathcal{NIG}$

The Gaussian distribution,

$$\mathcal{N}\left(y;\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right), \tag{B.1}$$

is a probability with support on the entire real line, however with a clear preference for values around its mean $\mu \pm$ a few standard deviations σ. Because of these easy-to-grasp properties, in combination with its frequent appearance as a limiting distribution (cf. the central limit theorem) and its analytically tractable form, it is ubiquitous in statistical modeling.

A simple problem is that of inferring $\theta = \mu$ when we observe data $y_{1:T}$ as exchangeable observations $p(y_t \mid \theta) = \mathcal{N}\left(y_t;\mu,\sigma^2\right)$. If we decide to follow the Bayesian way of reasoning, we formulate a prior $p(\theta)$. A natural choice for the prior might be $p(\mu) = \mathcal{N}\left(\mu;m,\varsigma^2\right)$, and the posterior then becomes (after some algebra) $p(\theta \mid y_{1:T}) = \mathcal{N}\left(\mu;\left(\frac{m}{\varsigma^2}+\frac{\sum_t y_t}{\sigma^2}\right)\left(\frac{1}{\varsigma^2}+\frac{T}{\sigma^2}\right)^{-1},\left(\frac{1}{\varsigma^2}+\frac{T}{\sigma^2}\right)^{-1}\right)$, i.e., another Gaussian distribution. Thus, the Gaussian distribution is the conjugate prior for a Gaussian likelihood model with unknown mean.

The above example is, however, somewhat unrealistic, since the mean is unknown whereas the variance is assumed to be known! A less artificial situation would be the problem of inferring $\theta = \{\mu,\sigma^2\}$ jointly. However, the Gaussian distribution is clearly not a good prior for σ^2, since the Gaussian distribution has support on the entire real line, whereas a negative variance bears no meaning in our model. A way of constructing a distribution with support only on the positive real line, is Proceedings of 26th to consider the square of a standard Gaussian random variable z, or more generally, the sum of ℓ such squared Gaussian random variables z_j,

$$q = \sum_{j=1}^{\ell} z_j^2, \qquad p(z_j) \sim \mathcal{N}\left(z_j;0,1\right). \tag{B.2}$$

The density for q can be written as

$$p(q) = \frac{1}{2^{\ell/2}\Gamma\left(\frac{\ell}{2}\right)}(q)^{\ell/2-1}\exp\left(-\frac{q}{2}\right) \triangleq \mathcal{G}\left(q;1,\ell\right), \tag{B.3}$$

where we use $\mathcal{G}$ to be the notation for the so-called *gamma* distribution. By its construction (B.2), we may realize that the mean of $\mathcal{G}\left(q;1,\ell\right)$ is ℓ, and its variance increases with ℓ. The gamma distribution can be generalized to non-integer $\ell > 1$, and also a scale parameter $\lambda > 0$ can be introduced, as

$$\mathcal{G}\left(q;\lambda,\ell\right) = \frac{\lambda^{\ell/2}}{2^{\ell/2}\Gamma\left(\frac{\ell}{2}\right)}(q)^{\ell/2-1}\exp\left(-\frac{q\lambda}{2}\right). \tag{B.4}$$

Now, this distribution could be used as a prior for σ^2. However, to retain conjugacy properties, we have to work with the inverse of q: if q is gamma distributed, then is

its inverse $\sigma^2 \triangleq 1/q$, distributed as

$$\mathcal{IG}\left(\sigma^2; \lambda, \ell\right) = \frac{\lambda^{\ell/2}}{2^{\ell/2}\Gamma\left(\frac{\ell}{2}\right)}(\sigma^2)^{-\ell/2-1}\exp\left(-\frac{\lambda}{2\sigma^2}\right), \tag{B.5}$$

the so-called *inverse gamma* ($\mathcal{IG}$) distribution[1], with support on $(0, \infty)$, mean $\frac{2\lambda}{\ell-1}$ and variance increasing with λ and decreasing with ℓ.

The inverse gamma distribution can now be combined with the Gaussian distribution into the normal inverse gamma distribution ($\mathcal{NIG}$) in the following way:

$$\begin{aligned}\mathcal{NIG}\left(\mu, \sigma^2; m, v, \lambda, \ell\right) \triangleq \mathcal{N}\left(\mu; m, v\sigma^2\right)\mathcal{IG}\left(\sigma^2; \lambda, \ell\right) \propto \\ \propto (\sigma^2)^{-\ell/2-3/2}\exp\left(-\frac{\frac{1}{v}(\mu-m)^2+\lambda}{2\sigma^2}\right)\end{aligned} \tag{B.6}$$

Note that this is a hierarchical construction on the form $p(\mu, \sigma^2) = p(\mu \mid \sigma^2)p(\sigma^2)$, and *not* the independent form $p(\mu, \sigma^2) = p(\mu)p(\sigma^2)$. If we again assume the observations $y_{1:T}$ are exchangeable and observed as $p(y_t \mid \theta) = \mathcal{N}\left(y_t; \mu, \sigma^2\right)$, now with both mean and variance unknown, the posterior becomes $p(\theta \mid y) = \mathcal{NIG}\left(\mu, \sigma^2; \frac{m/v+\sum_t y_t}{1/v+T}, \frac{1}{1/v+T}, \lambda + \sum_t y_t^2 + m^2/v - \frac{(m/v+\sum_t y_t)^2}{1/v+T}, \ell + T\right)$. That is, the posterior is just another normal inverse gamma distribution, which indeed is the conjugate prior to $\mathcal{N}\left(y_t; \mu, \sigma^2\right)$ with unknown mean and variance.

B.1.2 Generalizing to the matrix case: $\mathcal{MNIW}$

The generalization of the univariate Gaussian distribution to the multivariate Gaussian distribution is well established. The generalization to the matrix case is, however, perhaps less so. Following Dawid (1981), we introduce the matrix normal ($\mathcal{MN}$) distribution as follows: If the random $k \times p$ matrix Z has independent standard Gaussian entries, we write $p(Z) = \mathcal{MN}\left(Z; 0, I_k, I_p\right)$. If, more generally, the rows of Z are independent, and each column has a multivariate Gaussian $\mathcal{N}(0, V)$ distribution (V is $p \times p$), we write $p(Z) = \mathcal{MN}(Z; 0, I_k, V)$. Similarly, we write $p(Z) = \mathcal{MN}\left(Z; 0, U, I_p\right)$ if each column of Z is independent, and each row has a multivariate Gaussian $\mathcal{N}(0, U)$ distribution (U is $k \times k$).

In the most general form, we may say that if all elements $z_{i,j}$ of the $k \times p$ random matrix Z have a jointly Gaussian distribution, element $z_{i,j}$ has the marginal distribution $p(z_{i,j}) = \mathcal{N}\left(z_{i,j}; m_{i,j}, u_{i,i} \cdot v_{j,j}\right)$, and the covariance between $z_{i,j}$ and $z_{m,T}$ is $\operatorname{cov}\left[z_{i,j}, z_{m,T}\right] = u_{i,m} \cdot v_{j,T}$, then the distribution of Z is $p(Z) = \mathcal{MN}(Z; M, U, V)$. We may write its density as

$$\mathcal{MN}(Z; M, U, V) = (2\pi v)^{-kp/2}|U|^{-p/2}|V|^{-k/2}\exp\left(-\tfrac{1}{2}\operatorname{tr}\left((A-M)^{\mathsf{T}}U^{-1}(A-M)V^{-1}\right)\right). \tag{B.7}$$

[1] Note that this is not the most common parameterization of the inverse gamma distribution.

Analogously to the gamma $\mathcal{G}(1, \ell)$ distribution, we can construct the Wishart distribution $\mathcal{W}(I_k, \ell)$ (named after John Wishart 1928) as follows: Let Z be distributed as $p(Z) = \mathcal{MN}(Z; 0, I_k, I_\ell)$. Then ZZ^T is distributed as $\mathcal{W}(I_k, \ell)$. As in the scalar case, we can generalize to non-integer ℓ, introduce a scale parameter (in the matrix case, a $k \times k$ symmetric positive definite matrix Λ) and consider the inverse $(ZZ^\mathsf{T})^{-1}$ (which exists with probability 1 if $\ell > k - 1$), yielding the inverse Wishart distribution with density

$$\mathcal{IW}(\Sigma; \Lambda, \ell) = \frac{|\Lambda|^{\ell/2}}{2^{\ell/2}\Gamma_k\left(\frac{\ell}{2}\right)} |\Sigma|^{-\frac{\ell+k+1}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\Lambda\Sigma^{-1}\right)\right) \tag{B.8}$$

if Σ is symmetric positive definite, and $\Gamma_k(\cdot)$ is the multivariate gamma function. $\mathcal{IW}(\Sigma; \Lambda, \ell)$ has a mean $\Lambda/(\ell - k - 1)$ (for $\ell > k - 1$) and a variance increasing (element-wise) with Λ and decreasing with ℓ (e.g., Rosen 1988). The diagonal elements of Σ are distributed as inverse gamma (e.g., Theorem 5.2.1 in Press 1982).

Following the scalar case, we construct the $\mathcal{MNIW}$ distribution as

$$\begin{aligned}\mathcal{MNIW}(A, \Sigma; M, V, \Lambda, \ell) &\triangleq \mathcal{MN}(A; M, \Sigma, V)\,\mathcal{IW}(\Sigma; \Lambda, \ell) \propto \\ &\propto |\Sigma|^{-(\ell+p)/2-1} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\left((A - M)V^{-1}(A - M)^\mathsf{T} + \Lambda\right)\right)\right).\end{aligned} \tag{B.9}$$

The special case $p = 1$, when $\mathcal{MN}(M, \Sigma, 1) = \mathcal{N}(M, \Sigma)$ is often referred to as the normal inverse Wishart distribution, the conjugate prior[2] for the case when observing vector-valued data $p(y_t \mid \theta) = \mathcal{N}(y_t; \mu, \Sigma)$ (e.g., Gelman et al. 2014, Section 3.6).

[2] The inverse Wishart is indeed the conjugate prior, but whether it is a sensible choice of prior is subject to debate, e.g, Alvarez et al. (2014) and Yang and J. O. Berger (1994) and references therein.

B.2 Scalar linear regression: $y_t = ax_t + e_t$

We now consider the problem of scalar linear regression with T exchangeable observations:, i.e., $y_t = ax_t + e_t$, $e_t \sim \mathcal{N}\left(0, \sigma^2\right)$ and x_t is known. That is, we have the model $p(y_{1:T} \mid a, \sigma^2) = \prod_{t=1}^{T} \mathcal{N}\left(y_t; ax_t, \sigma^2\right)$. We want to infer $a \in \mathbb{R}$ and $\sigma^2 \in \mathbb{R}^+$ with the Bayesian approach, and assume a normal inverse gamma (B.6) prior $\mathcal{NIG}\left(a, \sigma^2; m, v, \lambda, \ell\right)$. This yields the posterior

$$
\begin{aligned}
p(a, \sigma^2) &\propto \mathcal{NIG}\left(a, \sigma^2; m, v, \lambda, \ell\right) \cdot \prod_{t=1}^{T} \mathcal{N}\left(y_t; ax_t, \sigma^2\right) \propto \\
&\propto \left(\sigma^2\right)^{-l/2-3/2-T/2} \exp\left(-\frac{\frac{1}{v}(a-m)^2 + \lambda + \sum_{t=1}^{T}(y_t - ax_t)^2}{2\sigma^2}\right) = \\
&\Bigg/ \tfrac{1}{v}(a-m)^2 + \lambda + \textstyle\sum_t (y_t - ax_t)^2 = \tfrac{1}{v}(a^2 - 2am + m^2) + \lambda + \sum_t y_t^2 - 2a\sum_t y_t x_t + a^2 \sum_t x_t^2 = \\
&\qquad \left(\tfrac{1}{v} + \textstyle\sum_t x_t^2\right)\left(a - \frac{m/v + \sum_t y_t x_t}{1/v + \sum_t x_t^2}\right)^2 + \lambda + \sum_t y_t^2 + \tfrac{m^2}{v} - \frac{\left(m/v + \sum_t x_t y_t\right)^2}{\sum_t x_t^2 + 1/v} \Bigg/ \\
&= (\sigma^2)^{-(\ell+T)/2-3/2} \exp\left(-\frac{\left(\frac{1}{v} + \sum_t x_t^2\right)\left(a - \frac{m/v + \sum_t y_t x_t}{1/v + \sum_t x_t^2}\right)^2 + \lambda + \sum_t y_t^2 + \frac{m^2}{v} - \frac{(m/v + \sum_t x_t y_t)^2}{\sum_t x_t^2 + 1/v}}{2\sigma^2}\right) \propto \\
&\propto \big/ \text{cf. (B.6)} \big/ \propto \mathcal{NIG}\left(a, \sigma^2; \overline{m}, \overline{v}, \overline{\lambda}, \overline{\ell}\right)
\end{aligned}
\tag{B.10}
$$

with

$$\overline{m} = \frac{m/v + \sum_t y_t x_t}{1/v + \sum_t x_t^2}, \tag{B.11a}$$

$$\tfrac{1}{\overline{v}} = \tfrac{1}{v} + \textstyle\sum_t x_t^2, \tag{B.11b}$$

$$\overline{\lambda} = \lambda + \sum_t y_t^2 + \frac{m^2}{v} - \frac{(m/v + \sum_t x_t y_t)^2}{1/v + \sum_t x_t^2}, \tag{B.11c}$$

$$\overline{\ell} = \ell + T. \tag{B.11d}$$

B.3 Multivariable linear regression: $y_t = Ax_t + e_t$

We now consider the matrix case, where we observe T exchangeable observations

$$\underbrace{\begin{bmatrix} \\ y_t \\ \\ \end{bmatrix}}_{k\times 1} = \underbrace{\begin{bmatrix} \\ A \\ \\ \end{bmatrix}}_{k\times p} \underbrace{\begin{bmatrix} \\ x_t \\ \\ \end{bmatrix}}_{p\times 1} + \underbrace{\begin{bmatrix} \\ e_t \\ \\ \end{bmatrix}}_{k\times 1}, \qquad e_t \sim \mathcal{N}(0, \Sigma), \tag{B.12}$$

with known x_t. The data likelihood is given by

$$\begin{aligned} p(y_{1:T} \mid A, \Sigma) &= \textstyle\prod_{t=1}^{T} \mathcal{N}(y_t; Ax_t, \Sigma) = \\ &= \textstyle\prod_{t=1}^{T} |\Sigma|^{-k/2} \exp\left(-\frac{1}{2}(y_t - Ax_t)^\intercal \Sigma^{-1}(y_t - Ax_t)\right) \end{aligned} \tag{B.13}$$

We want to infer $A \in \mathbb{R}^{k\times p}$ and the $k \times k$ a covariance matrix Σ, in a Bayesian fashion. As a prior, we assume $\mathcal{MNIW}(A, \Sigma; M, V, \Lambda, \ell)$ (B.9). This gives the posterior

$$p(A, \Sigma \mid y_{1:T}) \propto \mathcal{MNIW}(A, \Sigma; M, V, \Lambda, \ell) \cdot \textstyle\prod_{t=1}^{T} \mathcal{N}(y_t; Ax_t, \Sigma) \propto$$

$$\propto |\Sigma|^{-(\ell+p+kn)/2-1} \exp\left(-\frac{1}{2}\operatorname{tr}\left(\Sigma^{-1}\left((A-M)V^{-1}(A-M)^\intercal + \Lambda + \textstyle\sum_{t=1}^{T}(y_t - Ax_t)(y_t - Ax_t)^\intercal\right)\right)\right) =$$

$$= \Big| (A-M)V^{-1}(A-M)^\intercal + \Lambda + \textstyle\sum_{t=1}^{T}(y_t - Ax_t)(y_t - Ax_t)^\intercal =$$

$$= \underbrace{\left[A - \left(MV^{-1} + \textstyle\sum_t y_t x_t^\intercal\right)\left(V^{-1} + \textstyle\sum_t x_t x_t^\intercal\right)^{-1}\right]\left(V^{-1} + \textstyle\sum_t x_t x_t^\intercal\right)\left[A - \left(MV^{-1} + \textstyle\sum_t y_t x_t^\intercal\right)\left(V^{-1} + \textstyle\sum_t x_t x_t^\intercal\right)^{-1}\right]^\intercal}_{(\star)} +$$

$$+ \underbrace{\Lambda + \textstyle\sum_t y_t y_t^\intercal + MV^{-1}M^\intercal - \left(MV^{-1} + \textstyle\sum_t y_t x_t^\intercal\right)\left(V^{-1} + \textstyle\sum_t x_t x_t^\intercal\right)^{-1}\left(MV^{-1} + \textstyle\sum_t y_t x_t^\intercal\right)^\intercal}_{(\star\star)} \Big| =$$

$$= |\Sigma|^{-(\ell+p+kn)/2-1} \exp\left(-\tfrac{1}{2}\operatorname{tr}\left((\star) + (\star\star)\right)\right) \propto \big/ \text{cf. (B.9)} \big/ \propto$$

$$\propto \mathcal{NIW}\left(a, \Sigma; \overline{M}, \overline{V}, \overline{\Lambda}, \overline{\ell}\right) \tag{B.14}$$

with

$$\overline{M} = \left(MV^{-1} + \textstyle\sum_t y_t x_t^\intercal\right)\left(V^{-1} + \textstyle\sum_t x_t x_t^\intercal\right)^{-1}, \tag{B.15a}$$

$$\overline{V}^{-1} = V^{-1} + \textstyle\sum_t x_t x_t^\intercal, \tag{B.15b}$$

$$\overline{\Lambda} = (\star\star), \tag{B.15c}$$

$$\overline{\ell} = \ell + kT. \tag{B.15d}$$

"We could, of course, use any notation we want."
Richard Feynman

Notation list

The notation used in the introductory chapters is summarized below. The notation used in the papers is introduced separately in each paper.

Symbol	Meaning
General	
a	A scalar or vector
A	A matrix or a set
$\mathbb{I}_A(\theta)$	Indicator function: 1 if $\theta \in A$, 0 otherwise
$\setminus$	Relative complement
$\mathbb{R}$	The set of real numbers
$\|\cdot\|$	The Euclidean distance
$\Gamma(\cdot)$	Gamma function
$K_\nu(\cdot)$	Modified Bessel function (Rasmussen and Williams 2006, p. 84)
p	Probability density or mass
$\mathbb{P}$	Probability
$\mathbb{E}[\cdot]$	The expected value of the argument
$\mathcal{N}(\cdot\,; \mu, \sigma^2)$	The density for a univariate Gaussian distribution with mean μ and variance σ^2.
$\mathcal{N}(\cdot\,; \mu, \Sigma)$	The density for a multivariate Gaussian distribution with mean μ and covariance matrix Σ.
$\xrightarrow{d}$	Convergence in distribution
Data, models and learning	
y	Data
y_t	The data sample with index t
T	The number of data samples
$y_{1:T}$	$\{y_t\}_{t=1}^T$
n_y	The dimension of one data sample
θ	Parameters in a model
η	Hyperparameters in a model
$p(\theta)$	Prior distribution for θ
$p(\theta \mid y)$	Posterior distribution for θ
$p(y \mid \theta)$	Density for y given θ
$\mathcal{L}(\theta)$	Likelihood function for θ (2.2)
$\widehat{\theta}$	Point estimate of θ

State-space models

x_t	The state (at time t) in a state-space model
n_x	The dimension of the state in a state-space model
n_u	The dimension of the input to a state-space model
$f(\cdot \mid \cdot)$	The state transition function in a state-space model
$g(\cdot \mid \cdot)$	The observation function in a state-space model
ϑ	The parameters in a state-space model

Gaussian processes

$x^\star$	The points where the value of the Gaussian process is predicted
x^d	The points where the Gaussian process has been observed
$\mu(\cdot)$	The mean function
$\kappa(\cdot, \cdot)$	The covariance function
ε	Observation noise
$K^{\star\star}, K^{\star d}, K^{d\star}, K^{dd}$	Shorthand notation for κ evaluated in certain points; see definitions on page 30

Monte Carlo

N	The number of particles in a general particle approximation
$w^{(i)}$	The weight of particle i in a weighted particle approximation
N_x	The number of particles in the particle filter, Algorithm 1 in Chapter 5
K	The number of iterations of the MCMC sampler, Algorithm 2 in Chapter 5
$\mathcal{K}(\cdot \mid \cdot)$	The transition kernel in the MCMC sampler, Algorithm 2 in Chapter 5
N_θ	The number of particles in the SMC sampler, Algorithm 5 in Chapter 5
P	The number of iterations of the SMC sampler, Algorithm 5 in Chapter 5

"Citing an author whose ideas or information you used is paying a debt."

Umberto Eco

References

Hirotugu Akaike (1974). "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6, pp. 716–723.

Ignacio Alvarez, Jarad Niemi, and Matt Simpson (2014). "Bayesian inference for a covariance matrix". In: *Proceedings of the 26th Annual Conference on Applied Statistics in Agriculture*. Manhattan, KS, USA, pp. 71–82.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan (2003). "An introduction to MCMC for machine learning". In: *Machine Learning* 50.1, pp. 5–43.

Christophe Andrieu and Gareth O. Roberts (2009). "The pseudo-marginal approach for efficient Monte Carlo computations". In: *Annals of Statistics* 37.2, pp. 967–725.

M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp (2002). "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". In: *IEEE Transactions on Signal Processing* 50.2, pp. 174–188.

Thomas Bayes (1763). "An essay towards solving a problem in the doctrine of chances". In: *Philosophical Transactions (1683-1775)* 53, pp. 370–418.

James O. Berger (1985). *Statistical decision theory and Bayesian analysis*. 2nd ed. New York, NY, USA: Springer.

James O. Berger (2006). "The case for objective Bayesian analysis". In: *Bayesian Analysis* 1.3, pp. 385–402.

Christopher M. Bishop (2006). *Pattern recognition and machine learning*. New York, NY, USA: Springer.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe (2016). "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112 (518), pp. 859–877.

Alexandre Bouchard-Côté, Sebastian J. Vollmer, and Arnaud Doucet (2017). "The Bouncy Particle Sampler: A non-reversible rejection-free Markov chain Monte Carlo method". In: *arXiv:1510.02451*.

Olivier Cappé, Éric Moulines, and Tobias Rydén (2005). *Inference in hidden Markov models*. Springer Series in Statistics. New York, NY, USA: Springer.

George Casella and Roger L. Berger (2002). *Statistical inference*. 2nd ed. Pacific Grove, CA, USA: Duxbury.

Kathryn Chaloner and Isabella Verdinelli (1995). "Bayesian experimental design: a review". In: *Statistical Science* 10.3, pp. 273–304.

Krzysztof Chalupka, Christopher K. I. Williams, and Iain Murray (2013). "A framework for evaluating approximation methods for Gaussian process regression". In: *The Journal of Machine Learning Research (JMLR)* 14.2, pp. 333–350.

Tianshi Chen, Henrik Ohlsson, and Lennart Ljung (2012). "On the estimation of transfer functions, regularizations and Gaussian processes—Revisited". In: *Automatica* 48.8, pp. 1525–1535.

Nicolas Chopin (2004). "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference". In: *Annals of Statistics* 36.6, pp. 2385–2411.

Nicolas Chopin, Pierre E. Jacob, and Omiros Papaspiliopoulos (2013). "SMC^2: an efficient algorithm for sequential analysis of state space models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75.3, pp. 397–426.

Nicolas Chopin, James Ridgway, Mathieu Gerber, and Omiros Papaspiliopoulos (2015). "Towards automatic calibration of the number of state particles within the SMC^2 algorithm". In: *arXiv:1506.00570*.

Johan Dahlin, Fredrik Lindsten, and Thomas B. Schön (2015). "Particle Metropolis-Hastings using gradient and Hessian information". In: *Statistics and Computing* 25.1, pp. 81–92.

Johan Dahlin and Thomas B. Schön (2016). "Getting started with particle Metropolis-Hastings for inference in nonlinear models". In: *arXiv:1511:01707*.

A. Philip Dawid (1981). "Some matrix-variate distribution theory: notational considerations and a Bayesian application". In: *Biometrika* 68.1, pp. 265–274.

Bruno de Finetti (1992). "Foresight: its logical laws, its subjective sources". In: *Breakthroughs in Statistics: Foundations and Basic Theory*. Ed. by Samuel Kotz and L. Norman Johnson. Trans. by Henry E. Kyberg. Vol. 1. (Originally published in 1937 as "La prévision: ses lois logiques, ses sources subjectives" in *Annales de l'Institut Henri Poincaré* 7, pp. 1–68.) New York, NY, USA: Springer, pp. 134–174.

Pierre Del Moral (2004). *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. New York, NY, US: Springer.

Pierre Del Moral and Arnaud Doucet (2014). "Particle methods: an introduction with applications". In: *ESAIM: Proceedings* 44.1, pp. 1–46.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra (2006). "Sequential Monte Carlo samplers". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3, pp. 411–436.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra (2012a). "An adaptive sequential Monte Carlo method for approximate Bayesian computation". In: *Statistics and Computing* 22.5, pp. 1009–1020.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra (2012b). "On adaptive resampling strategies for sequential Monte Carlo methods". In: *Bernoulli* 18.1, pp. 252–278.

Bernard Delyon, Marc Lavielle, and Éric Moulines (1999). "Convergence of a stochastic approximation version of the EM algorithm". In: *Annals of Statistics* 27.1, pp. 94–128.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–38.

Randal Douc and Olivier Cappé (2005). "Comparison of resampling schemes for particle filtering". In: *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA)*. Zagreb, Croatia, pp. 64–69.

Randal Douc, Éric Moulines, and Jimmy Olsson (2014). "Long-term stability of sequential Monte Carlo methods under verifiable conditions". In: *Annals of Applied Probability* 24.5, pp. 1767–1802.

Arnaud Doucet and Adam M. Johansen (2011). "A tutorial on particle filtering and smoothing: fifteen years later". In: *Nonlinear Filtering Handbook*. Ed. by D. Crisan and B. Rozovsky. Oxford, UK: Oxford University Press, pp. 656–704.

Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth (1987). "Hybrid Monte Carlo". In: *Physics Letter B* 195.2, pp. 216–222.

David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Zoubin Ghahramani (2013). "Structure discovery in nonparametric regression through compositional kernel search". In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. Atlanta, GA, USA, pp. 1166–1174.

David Duvenaud, Dougal Maclaurin, and Ryan Adams (2016). "Early stopping as nonparametric variational inference". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cádiz, Spain, pp. 1070–1077.

David A. van Dyk and Xiyun Jiao (2014). "Metropolis-Hastings within partially collapsed Gibbs samplers". In: *Journal of Computational and Graphical Statistics* 24.2, pp. 301–327.

Bradley Efron (1979). "Bootstrap methods: another look at the jackknife". In: *Annals of Statistics* 7.1, pp. 1–26.

Bradley Efron (1986). "Why isn't everyone a Bayesian?" In: *The American Statistician* 40.1. Including discussion by H. Chernoff, D. V. Lindley, C.N. Morris, S. J. Press and A. F. M. Smith, pp. 1–5.

Bradley Efron (2013). "A 250-year argument: belief, behavior, and the bootstrap". In: *Bulletin of the American Mathematical Society* 50.1, pp. 129–146.

Bradley Efron and Trevor Hastie (2016). *Computer age statistical inference*. Cambridge, UK: Cambridge University Press.

Yonina C. Eldar and Gitta Kutyniok, eds. (2012). *Compressed sensing: theory and applications*. Cambridge, UK: Cambridge University Press.

Roger Frigola-Alcade (2015). "Bayesian time series learning with Gaussian processes". PhD thesis. UK: University of Cambridge.

Roger Frigola, Yutian Chen, and Carl Rasmussen (2014). "Variational Gaussian process state-space models". In: *Advances in Neural Information Processing Systems 27 (NIPS)*. Montréal, QC, Canada, pp. 3680–3688.

Roger Frigola, Fredrik Lindsten, Thomas B. Schön, and Carl Rasmussen (2013). "Bayesian inference and learning in Gaussian process state-space models with particle MCMC". In: *Advances in Neural Information Processing Systems 26 (NIPS)*. Lake Tahoe, NV, USA, pp. 3156–3164.

Andras Fulop and Junye Li (2013). "Efficient learning via simulation: a marginalized resample-move approach". In: *Journal of Econometrics* 176.2, pp. 146–161.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin (2014). *Bayesian data analysis*. 3rd ed. Boca Raton, FL, USA: Chapman & Hall/ CRC Press.

Stuart Geman and Donald Geman (1984). "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6.6, pp. 721–741.

Zoubin Ghahramani and Sam T. Roweis (1998). "Learning nonlinear dynamical systems using an EM algorithm". In: *Advances in Neural Information Processing Systems (NIPS) 11*. Denver, CO, USA, pp. 431–437.

Neil J. Gordon, David J. Salmond, and Adrian F.M. Smith (1993). "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". In: *IEE Proceedings F - Radar and Signal Processing*, pp. 107–113.

Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and Per-Johan Nordlund (2002). "Particle filters for positioning, navigation, and tracking". In: *IEEE Transactions on Signal Processing* 50.2, pp. 425–437.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. New York, NY, USA: Springer.

Wilfred K. Hastings (1970). "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1.

"Sequential state estimation: From Kalman filters to particle filters" (2004). In: *Proceedings of the IEEE* 92.3. Ed. by Simon Haykin and Nando de Freitas. Special issue.

Markus Heinonen, Henrik Mannerström, Juho Rousu, Samuel Kaski, and Harri Lähdesmäki (2016). "Non-stationary Gaussian process regression with Hamiltonian Monte Carlo". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cádiz, Spain, pp. 737–740.

Håkan Hjalmarsson (2009). "System identification of complex and structured systems". In: *European Journal of Control* 15.3–4, pp. 275–310.

Arthur E. Hoerl and Robert W. Kennard (1970). "Ridge regression: biased estimation for nonorthogonal problems". In: *Technometrics* 42.1, pp. 80–86.

Carl Jidling, Niklas Wahlström, Adrian Wills, and Thomas B. Schön (2018). "Linearly constrained Gaussian processes". In: *Advances in Neural Information Processing Systems 30 (NIPS)*. Long Beach, CA, USA, pp. 217–224.

Thomas Kailath (1980). *Linear systems*. Englewood Cliffs, NJ, USA: Prentice Hall.

Rudolf E. Kálmán (1960). "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering* 82.1, pp. 35–45.

Augustine Kong, Jun S. Liu, and Wing Hung Wong (1994). "Sequential imputations and Bayesian missing data problems". In: *Journal of the American Statistical Association* 89.425, pp. 278–288.

Estelle Kuhn and Marc Lavielle (2004). "Coupling a stochastic approximation version of EM with an MCMC procedure". In: *ESAIM: Probability and Statistics* 8, pp. 115–131.

Pierre Simon de Laplace (1820). *Théorie analytique des probabilités*. 3rd ed. Paris, France: Mme Ve Courcier, imprimeur-libraire pour les mathématiques.

Faming Liang, Chuanhai Liu, and Raymond Carroll (2010). *Advanced Markov chain Monte Carlo methods: learning from past samples*. West Sussex, United Kingdom: John Wiley & Sons.

Dennis V. Lindley (1990). "The 1988 Wald memorial lectures: the present position in Bayesian statistics". In: *Statistical Science* 6.1, pp. 44–65.

Fredrik Lindsten (2013). "An efficient stochastic approximation EM algorithm using conditional particle filters". In: *Proceedings of the 38th International Conference on*

Acoustics, Speech, and Signal Processing (ICASSP). Vancouver, BC, Canada, pp. 6274–6278.

Fredrik Lindsten, Michael I. Jordan, and Thomas B. Schön (2014). "Particle Gibbs with ancestor sampling". In: *The Journal of Machine Learning Research (JMLR)* 15.1, pp. 2145–2184.

Fredrik Lindsten and Thomas B. Schön (2013). "Backward simulation methods for Monte Carlo statistical inference". In: *Foundations and Trends in Machine Learning* 6.1, pp. 1–143.

Lennart Ljung (1999). *System identification: theory for the user*. 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall.

Lennart Ljung and Torkel Glad (2004). *Modellbygge och simulering*. 2nd ed. Lund, Sweden: Studentlitteratur.

Michael Lustig, David Donoho, and John M. Pauly (2007). "Sparse MRI: the application of compressed sensing for rapid MR imaging". In: *Magnetic resonance in medicine* 58.6, pp. 1182–1195.

David J. C. MacKay (1998). "Introduction to Gaussian processes". In: *Neural Networks and Machine Learning*. Ed. by C. M. Bishop. Vol. 168. NATO ASI Series F: Computational and Systems Sciences. Berlin, Germany: Springer-Verlag, pp. 133–165.

Bertil Matérn (1960). "Spatial Variation". PhD thesis. Sweden: Statens skogsforskningsinstitut.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller (1953). "Equation of state calculations by fast computing machines". In: *Journal of Chemical Physics* 21.6, pp. 1087–1092.

Nicholas Metropolis and Stanisław Ulam (1949). "The Monte Carlo method". In: *Journal of the American Statistical Association* 44.247, pp. 335–341.

Peter Müller (1991). *A generic approach to posterior intergration and Gibbs sampling*. Tech. rep. West Lafayette, IN, USA: Department of Statistics, Purdue University.

Lawrence M. Murray, Anthony Lee, and Pierre E. Jacob (2015). "Parallel resampling in the particle filter". In: *Journal of Computational and Graphical Statistics* 25.3, pp. 789–805.

Lawrence M. Murray and Thomas B. Schön (2018). "Automated learning with a probabilistic programming language: Birch". To be submitted.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön (2014). "Sequential Monte Carlo for graphical models". In: *Advances in Neural Information Processing Systems 27 (NIPS)*. Montréal, QC, Canada, pp. 1862–1870.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön (2015). "Nested sequential Monte Carlo methods". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Lille, France, pp. 1292–1301.

Radford M. Neal (2003). "Slice sampling". In: *Annals of Statistics* 31.3, pp. 705–767.

Radford M. Neal (2011). "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo*. Ed. by Steve Brooks, Andrew Gelman, Galin jones, and Xiao-Li Meng. Chapman & Hall/CRC Press.

Brett Ninness and Soren Henriksen (2010). "Bayesian system identification via Markov chain Monte Carlo techniques". In: *Automatica* 46.1, pp. 40–51.

Jimmy Olsson, Olivier Cappé, Randal Douc, and Éric Moulines (2008). "Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state-space models". In: *Bernoulli* 14.1, pp. 155–179.

Brooks Paige, Frank Wood, Arnaud Doucet, and Yee Whye Teh (2014). "Asynchronous anytime sequential Monte Carlo". In: *Advances in Neural Information Processing Systems 27 (NIPS)*. Montréal, QC, Canada, pp. 1–9.

Václav Peterka (1981). "Bayesian system identification". In: *Automatica* 17.1, pp. 41–53.

E. A. J. Frank Peters and Gijsbertus de With (2012). "Rejection-free Monte Carlo sampling for general potentials". In: *Physical Review E* 85.2, pp. 1–5.

David L. Phillips (1962). "A technique for the numerical solution of certain integral equations of the first kind". In: *Journal of the ACM* 9.1, pp. 84–97.

Michael K. Pitt, Ralph dos Santos Silva, Paolo Giordani, and Robert Kohn (2012). "On some properties of Markov chain Monte Carlo simulation methods based on the particle filter". In: *Journal of Econometrics* 171.2, pp. 134–151.

S. James Press (1982). *Applied multivariate analysis: using Bayesian and frequentist methods of inference*. Malabar, FL, USA: Robert E. Krieger Publishing Company.

Friedrich Pukelsheim (1993). *Optimal design of experiments*. New York, NY, USA: Wiley.

José Mario Quintana (1987). "Multivariate Bayesian forecasting models". PhD thesis. UK: University of Warwick.

Carl E. Rasmussen and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. Cambridge, MA, USA: MIT Press.

Rudolf Erich Raspe (1786). *Baron Munchausen's narrative of his marvellous travels and campaigns in Russia*. Oxford, UK: Smith.

Stephen L. Richter and Raymond A. DeCarlo (1983). "Continuation methods: theory and applications". In: *IEEE Transactions on Circuits and Systems* 30.6, pp. 347–352.

Herbert Robbins and Sutton Monro (1951). "A stochastic approximation method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407.

Christian P. Robert and George Casella (2004). *Monte Carlo statistical methods*. 2nd ed. New York, NY, USA: Springer.

Dietrich von Rosen (1988). "Moments for the inverted Wishart distribution". In: *Scandinavian Journal of Statistics* 15.2, pp. 91–109.

Wilson J. Rugh (1993). *Linear system theory*. Englewood Cliffs, NJ, USA: Prentice Hall.

Simo Särkkä (2013). *Bayesian filtering and smoothing*. Cambridge, UK: Cambridge University Press.

Mark J. Schervish (1995). *Theory of statistics*. New York, NY, USA: Springer.

Thomas B. Schön and Fredrik Lindsten (2011). *Manipulating the multivariate Gaussian density*. Tech. rep. Linköping, Sweden: Division of Automatic Control, Linköping University.

Thomas B. Schön, Andreas Svensson, Lawrence M. Murray, and Fredrik Lindsten (2018). "Probabilistic learning of nonlinear dynamical systems using sequential Monte Carlo". In: *Mechanical Systems and Signal Processing* 104, pp. 866–883.

Thomas B. Schön, Adrian Wills, and Brett Ninness (2011). "System identification of nonlinear state-space models". In: *Automatica* 47.1, pp. 39–49.

Gideon Schwarz (1978). "Estimating the dimension of a model". In: *Annals of Statistics* 6.2, pp. 461–464.

Amar Shah, Andrew Gordon Wilson, and Zoubin Ghahramani (2014). "Student-t processes as alternatives to Gaussian processes". In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Reykjavik, Iceland, pp. 877–885.

Jonas Sjöberg and Lennart Ljung (1995). "Overtraining, regularization and searching for a minimum, with application to neural networks". In: *International Journal of Control* 62.6, pp. 1391–1407.

Edward Snelson (2007). "Flexible and efficient Gaussian process models for machine learning". PhD thesis. UK: University College London.

Torsten Söderström and Petre Stoica (1989). *System identification*. Hemel Hempstead, UK: Prentice-Hall, Inc.

Arno Solin, Manon Kok, Niklas Wahlström, Thomas B. Schön, and Simo Särkkä (2018). "Modeling and interpolation of the ambient magnetic field by Gaussian processes". In: *IEEE Transations on Robotics*. Accepted for publication.

Arno Solin and Simo Särkkä (2014). "Explicit link between periodic covariance functions and state space models". In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Reykjavik, Iceland, pp. 904–912.

Arno Solin and Simo Särkkä (2014). "Hilbert space methods for reduced-rank Gaussian process regression". In: *arXiv:1401.5508*.

Leland Stewart and Perry Jr. McCarty (1992). "Use of Bayesian belief networks to fuse continuous and discrete information for target recognition, tracking, and situation assessment". In: *Proceedings of SPIE 1699, Signal Processing, Sensor Fusion, and Target Recognition*. Orlando, FL, USA, pp. 177–185.

Stephen M. Stigler (1986). "Laplace's 1774 memoir on inverse probability". In: *Statistical Science* 1.3, pp. 359–378.

Andreas Svensson (2013). *Particle Filter Explained without Equations*. URL: https://www.youtube.com/watch?v=aUkBa1zMKv4.

Andreas Svensson and Thomas B. Schön (2016). *Comparing two recent particle filter implementations of Bayesian system identification*. Tech. rep. 2016-008. (Presented at Reglermöte 2016, Gothenburg, Sweden). Department of Information Technology, Uppsala University.

Andreas Svensson, Thomas B. Schön, and Manon Kok (2015). "Nonlinear state space smoothing using the conditional particle filter". In: *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*. Beijing, China, pp. 975–980.

Andreas Svensson, Arno Solin, Simo Särkkä, and Thomas B. Schön (2016). "Computationally efficient Bayesian learning of Gaussian process state space models". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cádiz, Spain, pp. 213–221.

Robert Tibshirani (1996). "Regression shrinkage and selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 58.1, pp. 267–288.

Luke Tierney (1994). "Markov chains for exploring posterior distributions". In: *Annals of Statistics* 22.4, pp. 1701–1728.

Michail K. Titsias and Lázaro-Gredilla (2014). "Doubly Stochastic Variational Bayes for non-Conjugate Inference". In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. Beijing, China, pp. 1971–1979.

Aad W. van der Vaart (1998). *Asymptotic Statistics*. Cambridge, UK: Cambridge University Press.

Nick Whiteley (2013). "Stability properties of some particle filters". In: *The Annals of Applied Probability* 23.6, pp. 2500–2537.

Adrian Wills, Thomas B. Schön, Fredrik Lindsten, and Brett Ninness (2012). "Estimation of linear systems using a Gibbs sampler". In: *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*. Brussels, Belgium, pp. 203–208.

John Wishart (1928). "The generalised product moment distribution in samples from a normal multivariate population". In: *Biometrika* 20A.1/2, pp. 32–52.

John Wright, Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma (2009). "Robust face recognition via sparse representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.2, pp. 210–227.

Ruoyong Yang and James O. Berger (1994). "Estimation of a covariance matrix using the reference prior". In: *Annals of Statistics* 22.3, pp. 1195–1211.

Hui Zou and Trevor Hastie (2005). "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 67.2, pp. 301–320.

Paper I

Title

A flexible state-space model for learning nonlinear dynamical systems

Authors

Andreas Svensson and Thomas B. Schön

Edited version of

Andreas Svensson and Thomas B. Schön (2017). "A flexible state-space model for learning nonlinear dynamical systems". In: *Automatica* 80, pp. 189–199.

Digital identity

doi:10.1016/j.automatica.2017.02.030

Parts of the content in this paper have previously been presented in

Andreas Svensson, Arno Solin, Simo Särkkä, and Thomas B. Schön (2016). "Computationally efficient Bayesian learning of Gaussian process state space models". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cádiz, Spain, pp. 213–221

and

Andreas Svensson, Thomas B. Schön, Arno Solin, and Simo Särkkä (2015). "Nonlinear state space model identification using a regularized basis function expansion". In: *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. Cancún, Mexico, pp. 493–496.

Financial support

The Swedish Research Council (VR) via the project *Probabilistic modeling of dynamical systems* (contract number: 621-2013-5524) and the Swedish Foundation for Strategic Research (SSF) via the project *ASSEMBLE*.

Thanks to

Dave Zachariah, Per Mattsson and the anonymous reviewers for useful comments on the manuscript.

A flexible state-space model for learning nonlinear dynamical systems

Abstract

We consider a nonlinear state-space model with the state transition and observation functions expressed as basis function expansions. The coefficients in the basis function expansions are learned from data. Using a connection to Gaussian processes we also develop priors on the coefficients, for tuning the model flexibility and to prevent overfitting to data, akin to a Gaussian process state-space model. The priors can alternatively be seen as a regularization, and helps the model in generalizing the data without sacrificing the richness offered by the basis function expansion. To learn the coefficients and other unknown parameters efficiently, we tailor an algorithm using state-of-the-art sequential Monte Carlo methods, which comes with theoretical guarantees on the learning. Our approach indicates promising results when evaluated on a classical benchmark as well as real data.

1 Introduction

Nonlinear system identification (Ljung 1999; Ljung 2010; Sjöberg et al. 1995) aims to learn nonlinear mathematical models from data generated by a dynamical system. We will tackle the problem of learning nonlinear state-space models with only weak assumptions on the nonlinear functions, and make use of the Bayesian framework (Peterka 1981) to encode prior knowledge and assumptions to guide the otherwise too flexible model.

Consider the (time invariant) state-space model

$$x_{t+1} = f(x_t, u_t) + v_t, \qquad v_t \sim \mathcal{N}(0, Q), \tag{1a}$$

$$y_t = g(x_t, u_t) + e_t, \qquad e_t \sim \mathcal{N}(0, R). \tag{1b}$$

The variables are denoted as the state $x_t \in \mathbb{R}^{n_x}$, which is not observed explicitly, the input $u_t \in \mathbb{R}^{n_u}$, and the output $y_t \in \mathbb{R}^{n_y}$. We will learn the state transition function $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_x}$ and the observation function $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_y}$ as well as Q and R from a set of training data of input-output signals $\{u_{1:T}, y_{1:T}\}$.

Consider a situation when a finite-dimensional linear, or other sparsely parameterized model, is too rigid to describe the behavior of interest, but only a limited data record is available so that any too flexible model would overfit (and be of no help in generalizing to events not exactly seen in the training data). In such a situation, a *systematic way to encode prior assumptions and thereby tuning the flexibility of the*

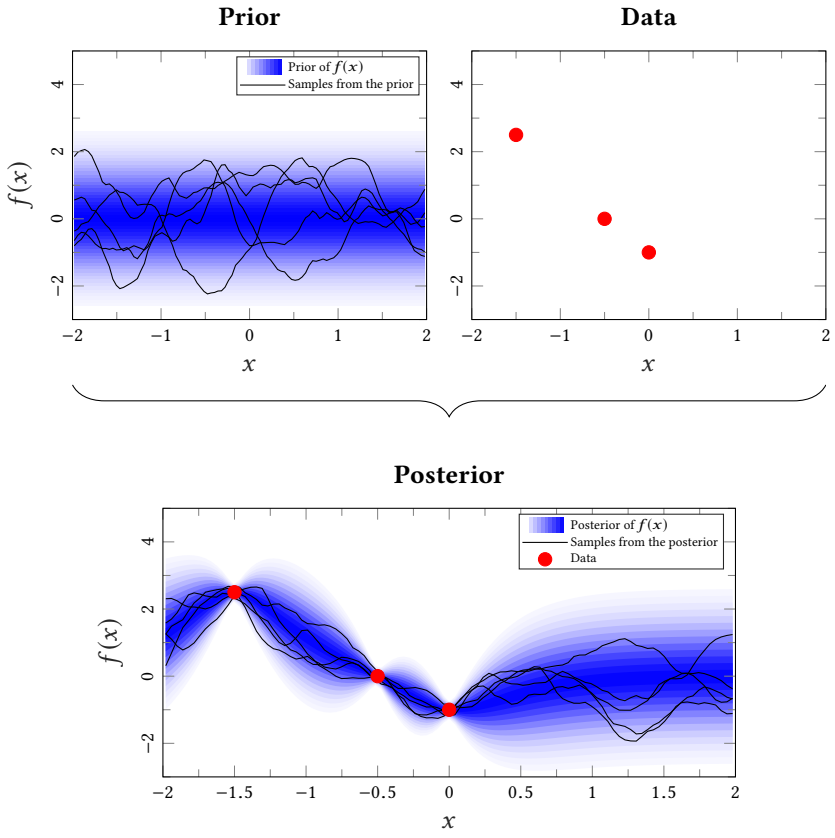


Figure 1: *The Gaussian process as a modeling tool for an one-dimensional function $f : \mathbb{R} \mapsto \mathbb{R}$. The prior distribution (upper left plot) is represented by the shaded blue color (the more intense color, the higher density), as well as 5 samples drawn from it. By combining the prior and the data (upper right plot), the posterior (lower plot) is obtained. The posterior mean basically interpolates between the data points, and adheres to the prior in regions where the data is not providing any information. This is clearly a desirable property when it comes to generalizing from the training data—consider the thought experiment of using a 2nd order polynomial instead. Further, the posterior also provides a quantification of the uncertainty present, high in data-scarce regions and low where the data provides knowledge about $f(\cdot)$.*

model can be useful. For this purpose, we will take inspiration from Gaussian processes (GPs, Rasmussen and Williams 2006) as a way to encode prior assumptions on $f(\cdot)$ and $g(\cdot)$. As illustrated by Figure 1, the GP is a distribution over functions which gives a probabilistic model for inter- and extrapolating from observed data. GPs have successfully been used in system identification for, e.g., response estimation, nonlinear ARX models and GP state-space models (Frigola-Alcade 2015; Juš Kocijan 2016; Pillonetto and De Nicolao 2010).

To parameterize $f(\cdot)$, we expand it using basis functions

$$f(x) = \sum_{j=0}^{m} w^{(j)} \phi^{(j)}(x), \tag{2}$$

and similarly for $g(\cdot)$. The set of basis functions is denoted by $\{\phi^{(j)}(\cdot)\}_{j=0}^{m}$, whose coefficients $\{w^{(j)}\}_{j=0}^{m}$ will be learned from data. By introducing certain *priors* $p(w^{(j)})$ *on the basis function coefficients* the connection to GPs will be made, based on a Karhunen-Loève expansion (Solin and Särkkä 2014). We will thus be able to understand our model in terms of the well-established and intuitively appealing GP model, but still benefit from the computational advantages of the linear-in-parameter structure of (2). Intuitively, the idea of the priors $p(w^{(j)})$ is to keep $w^{(j)}$ 'small unless data convinces otherwise', or equivalently, introduce a regularization of $w^{(j)}$.

To learn the model (1), i.e., determine the basis function coefficients $w^{(j)}$, we tailor a learning algorithm using recent sequential Monte Carlo/particle filter methods (Kantas et al. 2015; Schön, Lindsten, et al. 2015). The learning algorithm infers the posterior distribution of the unknown parameters from data, and come with theoretical guarantees. We will pay extra attention to the problem of finding the maximum mode of the posterior, or equivalent, regularized maximum likelihood estimation.

Our contribution is the development of a flexible nonlinear state-space model with a tailored learning algorithm, which together constitutes a new nonlinear system identification tool. The model can either be understood as a GP state-space model (generalized allowing for discontinuities, Section 3.2), or as a nonlinear state-space model with a regularized basis function expansion.

2 Related work

Important work using the GP in system identification includes impulse response estimation (T. Chen et al. 2012; Pillonetto, Chiuso, et al. 2011; Pillonetto and De Nicolao 2010), nonlinear ARX models (Bijl et al. 2016; Juš Kocijan et al. 2005), Bayesian learning of ODEs (Calderhead et al. 2008; Macdonald et al. 2015; Wang and Barber 2014) and the latent force model (Alvarez et al. 2013). In the GP state-space model (Frigola-Alcade 2015) the transition function $f(\cdot)$ in a state-space model is learned with a GP prior, particularly relevant to this paper. A conceptually interesting contribution to the GP state-space model was made by Frigola, Lindsten, et al. (2013), using a Monte Carlo approach (similar to this paper) for learning. The practical use of Frigola, Lindsten, et al. (2013) is however very limited, due to its extreme computational burden. This calls for approximations, and a promising approach is presented by Frigola, Y. Chen, et al. (2014) (and somewhat generalized by Mattos et al. (2016)), using inducing points

and a variational inference scheme. Another competitive approach is Svensson, Solin, et al. (2016), where we applied the GP approximation proposed by Solin and Särkkä (2014) and used a Monte Carlo approach for learning (Frigola-Alcade (2015) covers the variational learning using the same GP approximation). In this paper, we extend this work by considering basis function expansions in general (not necessarily with a GP interpretation), introduce an approach to model discontinuities in $f(\cdot)$, as well as including both a Bayesian and a maximum likelihood estimation approach to learning.

To the best of our knowledge, the first extensive paper on the use of a basis function expansion inside a state-space model was written by Ghahramani and Roweis (1998), who also wrote a longer unpublished version (Roweis and Ghahramani 2000). The recent work by Tobar et al. (2015) resembles that of Ghahramani and Roweis (1998) on the modeling side, as they both use basis functions with locally concentrated mass spread in the state space. On the learning side, Ghahramani and Roweis (1998) use an expectation maximization (EM, Dempster et al. 1977) procedure with extended Kalman filtering, whilst Tobar et al. (2015) use particle Metropolis-Hastings (Andrieu et al. 2010). There are basically three major differences between Tobar et al. (2015) and our work. We will (i) use another (related) learning method, particle Gibbs, allowing us to take advantage of the linear-in-parameter structure of the model to increase the efficiency. Further, we will (ii) mainly focus on a different set of basis functions (although our learning procedure will be applicable also to the model used by Tobar et al. (2015)), and – perhaps most important – (iii) we will pursue a systematic encoding of prior assumptions further than Tobar et al. (2015), who instead assume $g(\cdot)$ to be known and use 'standard sparsification criteria from kernel adaptive filtering' as a heuristic approach to regularization.

There are also connections to Paduart et al. (2010), who use a polynomial basis inside a state-space model. In contrast to our work, however, Paduart et al. (2010) prevent the model from overfitting to the training data not by regularization, but by manually choosing a low enough polynomial order and terminating the learning procedure prematurely (early stopping). Paduart et al. are, in contrast to us, focused on the frequency properties of the model and rely on optimization tools. An interesting contribution by Paduart et al. is to first use classical methods to find a linear model, which is then used to initialize the linear term in the polynomial expansion. We suggest to also use this idea, either to initialize the learning algorithm, or use the nonlinear model only to describe deviations from an initial linear state-space model.

Furthermore, there are also connections to our previous work (Svensson, Schön, et al. 2015), a short paper only outlining the idea of learning a regularized basis function expansion inside a state-space model. Compared to Svensson, Schön, et al. (2015), this work contains several extensions and new results. Another recent work using a regularized basis function expansion for nonlinear system identification is that of Delgado et al. (2015), however not in the state-space model framework. Delgado et al. (2015) use rank constrained optimization, resembling an L^0-regularization. To achieve a good performance with such a regularization, the system which generated the data has to be well described by only a few number of the basis functions being 'active', i.e., have non-zero coefficients, which makes the choice of basis functions important and problem-dependent. The recent work by Mattsson et al. (2016) is also covering learning of a regularized basis function expansion, however for input-output type of models.

3 Constructing the model

We want the model, whose parameters will be learned from data, to be able to describe a broad class of nonlinear dynamical behaviors without overfitting to training data. To achieve this, important building blocks will be the basis function expansion (2) and a GP-inspired prior. The order n_x of the state-space model (1) is assumed known or set by the user, and we have to learn the transition and observation functions $f(\cdot)$ and $g(\cdot)$ from data, as well as the noise covariance matrices Q and R. For brevity, we focus on $f(\cdot)$ and Q, but the reasoning extends analogously to $g(\cdot)$ and R.

3.1 Basis function expansion

The common approaches in the literature on black-box modeling of functions inside state-space models can broadly be divided into three groups: neural networks (Bishop 2006; Narendra and Li 1996; Nørgård et al. 2000), basis function expansions (Ghahramani and Roweis 1998; Paduart et al. 2010; Sjöberg et al. 1995; Tobar et al. 2015) and GPs (Frigola-Alcade 2015; Rasmussen and Williams 2006). We will make use of a basis function expansion inspired by the GP. There are several reasons for this: Firstly, a basis function expansion provides an expression which is linear in its parameters, leading to a computational advantage: neural networks do not exhibit this property, and the naïve use of the nonparametric GP is computationally very expensive. Secondly, GPs and some choices of basis functions allow for a straightforward way of including prior assumptions on $f(\cdot)$ and help generalization from the training data, also in contrast to the neural network.

We write the combination of the state-space model (1) and the basis function expansion (2) as

$$x_{t+1} = \underbrace{\begin{bmatrix} w_1^{(1)} & \cdots & w_1^{(m)} \\ \vdots & & \vdots \\ w_{n_x}^{(1)} & \cdots & w_{n_x}^{(m)} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \phi^{(1)}(x_t, u_t) \\ \vdots \\ \phi^{(m)}(x_t, u_t) \end{bmatrix}}_{\bar{\varphi}(x_t, u_t)} + v_t, \tag{3a}$$

$$y_t = \underbrace{\begin{bmatrix} w_{g,1}^{(1)} & \cdots & w_{g,1}^{(m)} \\ \vdots & & \vdots \\ w_{g,n_y}^{(1)} & \cdots & w_{g,n_y}^{(m)} \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} \phi_g^{(1)}(x_t, u_t) \\ \vdots \\ \phi_g^{(m)}(x_t, u_t) \end{bmatrix}}_{\bar{\varphi}_g(x_t, u_t)} + e_t. \tag{3b}$$

There are several alternatives for the basis functions, e.g., polynomials (Paduart et al. 2010), the Fourier basis (Svensson, Schön, et al. 2015), wavelets (Sjöberg et al. 1995), Gaussian kernels (Ghahramani and Roweis 1998; Tobar et al. 2015) and piecewise constant functions. For the one-dimensional case (e.g., $n_x = 1$, $n_u = 0$) on the interval $[-L, L] \in \mathbb{R}$, we will choose the basis functions as

$$\phi^{(j)}(x) = \frac{1}{\sqrt{L}} \sin\left(\frac{\pi j (x + L)}{2L}\right). \tag{4}$$

This choice, which is the eigenfunctions to the Laplace operator, enables a particularly convenient connection to the GP framework (Solin and Särkkä 2014) in the priors we will introduce in Section 3.2. This choice is, however, important only for the interpretability[1] of the model. The learning algorithm will be applicable to any choice of basis functions.

Higher state-space dimensions

The generalization to models with a state space and input dimension such that $n_x + n_u > 1$ offers no conceptual challenges, but potentially computational ones. The counterpart to the basis function (4) for the space $[-L_1, L_1] \times \cdots \times [-L_{n_x+n_u}, L_{n_x+n_u}] \in \mathbb{R}^{n_x+n_u}$ is

$$\phi^{(j_1,\ldots,j_{n_x+n_u})}(x) = \prod_{k=1}^{n_x+n_u} \frac{1}{\sqrt{L_k}} \sin\left(\frac{\pi j_k (x^k + L_k)}{2L_k}\right), \tag{5}$$

(where x^k is the kth component of x), implying that the number of terms m grows exponentially with $n_x + n_u$. This problem is inherent in most choices of basis function expansions. For $n_x > 1$, the problem of learning $f : \mathbb{R}^{n_x+n_u} \mapsto \mathbb{R}^{n_x}$ can be understood as learning n_x number of functions $f_i : \mathbb{R}^{n_x+n_u} \mapsto \mathbb{R}$, cf. (3).

There are some options available to overcome the exponential growth with $n_x + n_u$, at the cost of a limited capability of the model. *Alternative 1* is to assume $f(\cdot)$ to be 'separable' between some dimensions, e.g., $f(x_t, u_t) = f^x(x_t) + f^u(u_t)$. If this assumption is made for all dimensions, the total number of parameters present grows quadratically (instead of exponentially) with $n_x + n_u$. *Alternative 2* is to use a radial basis function expansion (Sjöberg et al. 1995), i.e., letting $f(\cdot)$ only be a function of some norm $\|\cdot\|$ of (x_t, u_t), as $f(x_t, u_t) = f(\|(x_t, u_t)\|)$. The radial basis functions give a total number of parameters growing linearly with $n_x + n_u$. Both alternatives will indeed limit the space of functions possible to describe with the basis function expansion. However, as a pragmatic solution to the otherwise exponential growth in the number of parameters it might still be worth considering, depending on the particular problem at hand.

Manual and data-driven truncation

To implement the model in practice, the number of basis functions m has to be fixed to a finite value, i.e., truncated. However, fixing m also imposes a harsh restriction on which functions $f(\cdot)$ that can be described. Such a restriction can prevent overfitting to training data, an argument used by Paduart et al. (2010) for using polynomials only up to 3rd order. We suggest, on the contrary, to use priors on $w^{(j)}$ to prevent overfitting, and we argue that the interpretation as a GP is a preferred way to tune the model flexibility, rather than manually and carefully tuning the truncation. We therefore suggest to choose m as big as the computational resources allows, and let the prior and data decide which $w^{(j)}$ to be nonzero, a *data-driven truncation*.

Related to this is the choice of L in (4): if L is chosen too small, the state space becomes limited and thereby also limits the expressiveness of the model. On the

[1]Other choices of basis functions are also interpretable as GPs. The choice (4) is, however, preferred since it is independent of the choice of which GP covariance function to use.

other hand, if L is too big, an unnecessarily large m might also be needed, wasting computational power. To chose L to have about the same size as the maximum of u_t or y_t seems to be a good guideline.

3.2 Encoding prior assumptions—regularization

The basis function expansion (3) provides a very flexible model. A prior might therefore be needed to generalize from, instead of overfit to, training data. From a user perspective, the prior assumptions should ultimately be formulated in terms of the input-output behavior, such as gains, rise times, oscillations, equilibria, limit cycles, stability etc. As of today, tools for encoding such priors are (to the best of the authors' knowledge) not available. As a resort, we therefore use the GP state-space model approach, where we instead encode prior assumptions on $f(\cdot)$ as a GP. Formulating prior assumptions on $f(\cdot)$ is relevant in a model where the state space bears (partial) physical meaning, and it is natural to make assumptions whether the state x_t is likely to rapidly change (non-smooth $f(\cdot)$), or state equilibria are known, etc. However, also the truly black-box case offers some interpretations: a very smooth $f(\cdot)$ corresponds to a locally close-to-linear model, and vice versa for a more curvy $f(\cdot)$, and a zero-mean low variance prior on $f(\cdot)$ will steer the model towards a bounded output (if $g(\cdot)$ is bounded).

To make a connection between the GP and the basis function expansion, a Karhunen-Loève expansion is explored by Solin and Särkkä (2014). We use this to formulate Gaussian priors on the basis function expansion coefficients $w^{(j)}$, and learning of the model will amount to infer the posterior $p(w^{(j)}|y_{1:T}) \propto p(y_{1:T}|w^{(j)})p(w^{(j)})$, where $p(w^{(j)})$ is the prior and $p(y_{1:T}|w^{(j)})$ the likelihood. To use a prior $w^{(j)} \sim \mathcal{N}(0, \alpha^{-1})$ and inferring the maximum mode of the posterior can equivalently be interpreted as regularized maximum likelihood estimation

$$\arg\min_{w^{(j)}} \; -\log p(y_{1:T}|w^{(j)}) + \alpha|w^{(j)}|^2. \tag{6}$$

Smooth GP-priors for the functions

The Gaussian process provides a framework for formulating prior assumptions on functions, resulting in a non-parametric approach for regression. In many situations the GP allows for an intuitive generalization of the training data, as illustrated by Figure 1. We use the notation

$$f(x) \sim \mathcal{GP}(m(x), \kappa(x, x')) \tag{7}$$

to denote a GP prior on $f(\cdot)$, where $m(x)$ is the mean function and $\kappa(x, x')$ the covariance function. The work by Solin and Särkkä (2014) provides an explicit link between basis function expansions and GPs based on the Karhunen-Loève expansion, in the case of isotropic[2] covariance functions, i.e., $\kappa(x, x') = \kappa(|x - x'|)$. In particular, if the basis functions are chosen as (4), then

$$f(x) \sim \mathcal{GP}(0, \kappa(x, x')) \Leftrightarrow f(x) \approx \sum_{j=0}^{m} w^{(j)} \phi^{(j)}(x), \tag{8a}$$

[2]Note, this concerns only $f(\cdot)$, which resides *inside* the state-space model. This does *not* restrict the input-output behavior, from $u(t)$ to $y(t)$, to have an isotropic covariance.

with[3]

$$w^{(j)} \sim \mathcal{N}(0, S(\lambda^{(j)})), \tag{8b}$$

where S is the spectral density of κ, and $\lambda^{(j)}$ is the eigenvalue of $\phi^{(j)}$. Thus, this gives a systematic guidance on how to choose basis functions and priors on $w^{(i)}$. In particular, the eigenvalues of the basis function (4) are

$$\lambda^{(j)} = \left(\frac{\pi j}{2L}\right)^2, \text{ and } \lambda^{(j_{1:n_x+n_u})} = \sum_{k=1}^{n_x+n_u} \left(\frac{\pi j_k}{2L_k}\right)^2 \tag{9}$$

for (5). Two common types of covariance functions are the exponentiated quadratic κ_{eq} and Matérn κ_{M} class (Rasmussen and Williams 2006),

$$\kappa_{\text{eq}}(r) = s_f \exp\left(-\frac{r^2}{2l^2}\right), \tag{10a}$$

$$\kappa_{\text{M}}(r) = s_f \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}r}{l}\right), \tag{10b}$$

where $r \triangleq x - x'$, K_ν is a modified Bessel function, and ℓ, s_f and ν are hyperparameters to be set by the user or to be marginalized out, see Svensson, Solin, et al. (2016) for details. Their spectral densities are

$$S_{\text{eq}}(s) = s_f \sqrt{2\pi l^2} \exp\left(-\frac{\pi^2 l^2 s^2}{2}\right), \tag{11a}$$

$$S_{\text{M}}(s) = s_f \frac{2\pi^{\frac{1}{2}}\Gamma(\nu+\frac{1}{2})(2\nu)^\nu}{\Gamma(\nu) l^{2\nu}} \left(\frac{2\nu}{l^2} + s^2\right)^{-(\nu+\frac{1}{2})}. \tag{11b}$$

Altogether, by choosing the priors for $w^{(j)}$ as (8b), it is possible to approximately interpret $f(\cdot)$, parameterized by the basis function expansion (2), as a GP. For most covariance functions, the spectral density $S(\lambda^{(j)})$ tends towards 0 when $\lambda^{(j)} \to \infty$, meaning that the prior for large j tends towards a Dirac mass at 0. Returning to the discussion on truncation (Section 3.1), we realize that truncation of the basis function expansion with a reasonably large m therefore has no major impact to the model, but the GP interpretation is still relevant.

As discussed, finding the posterior mode under a Gaussian prior is equivalent to L^2-regularized maximum likelihood estimation. There is no fundamental limitation prohibiting other priors, for example Laplacian (corresponding to L^1-regularization: Tibshirani 1996). We use the Gaussian prior because of the connection to a GP prior on $f(\cdot)$, and it will also allow for closed form expressions in the learning algorithm.

For book-keeping, we express the prior on $w^{(j)}$ as a Matrix normal ($\mathcal{MN}$, Dawid 1981) distribution over A. The $\mathcal{MN}$ distribution is parameterized by a mean matrix $M \in \mathbb{R}^{n_x \times m}$, a right covariance $U \in \mathbb{R}^{n_x \times n_x}$ and a left covariance $V \in \mathbb{R}^{m \times m}$. The $\mathcal{MN}$ distribution can be defined by the property that $A \sim \mathcal{MN}(M, U, V)$ if and only if $\text{vec}(A) \sim \mathcal{N}(\text{vec}(M), V \otimes U)$, where $\otimes$ is the Kronecker product. Its density can be written as

$$\mathcal{MN}(A \,|\, M, U, V) = \frac{\exp\left(-\frac{1}{2}\text{tr}\left\{(A-M)^{\mathsf{T}} U^{-1}(A-M)V^{-1}\right\}\right)}{(2\pi)^{n_x m}|V|^{n_x/2}|U|^{m/2}}. \tag{12}$$

[3]The approximate equality in (8a) is exact if $m \to \infty$ and $L \to \infty$, refer to Solin and Särkkä (2014) for details.

By letting $M = 0$ and V be a diagonal matrix with entries $S(\lambda^{(j)})$, the priors (8b) are incorporated into this parametrization. We will let $U = Q$ for conjugacy properties, to be detailed later. Indeed, the marginal variance of the elements in A is then not scaled only by V, but also Q. That scaling however is constant along the rows, and so is the scaling by the hyperparameter s_f (10). We therefore suggest to simply use s_f as tuning for the overall influence of the priors; letting $s_f \to \infty$ gives a flat prior, or, a non-regularized basis function expansion.

Prior for noise covariances

Apart from $f(\cdot)$, the $n_x \times n_x$ noise covariance matrix Q might also be unknown. We formulate the prior over Q as an inverse Wishart ($\mathcal{IW}$, Dawid 1981) distribution. The $\mathcal{IW}$ distribution is a distribution over real-valued positive definite matrices, which puts prior mass on all positive definite matrices and is parametrized by its number of degrees of freedom $\ell > n_x - 1$ and an $n_x \times n_x$ positive definite scale matrix Λ. The density is defined as

$$\mathcal{IW}(Q \mid \ell, \Lambda) = \frac{|\Lambda|^{\ell/2}|Q|^{-(n_x+\ell+1)/2}}{2^{\ell n_x/2}\Gamma_{n_x}(\ell/2)} \exp\left(-\frac{1}{2}\mathrm{tr}\left(Q^{-1}\Lambda\right)\right), \tag{13}$$

where $\Gamma_{n_x}(\cdot)$ is the multivariate gamma function. The mode of the $\mathcal{IW}$ distribution is $\frac{\Lambda}{\ell+n_x+1}$. It is a common choice as a prior for covariance matrices due to its properties (e.g., Shah et al. 2014; Wills et al. 2012). When the $\mathcal{MN}$ distribution (12) is combined with the $\mathcal{IW}$ distribution (13) we obtain the $\mathcal{MNIW}$ distribution, with the following hierarchical structure

$$\mathcal{MNIW}(A, Q \mid M, V, \Lambda, \ell) = \mathcal{MN}(A \mid M, Q, V)\mathcal{IW}(Q \mid \ell, \Lambda). \tag{14}$$

The $\mathcal{MNIW}$ distribution provides a joint prior for the A and Q matrices, compactly parameterizing the prior scheme we have discussed, and is also the conjugate prior for our model, which will facilitate learning.

Discontinuous functions: Sparse singularities

The proposed choice of basis functions and priors is encoding a smoothness assumption of $f(\cdot)$. However, as discussed by Juditsky et al. (1995) and motivated by Example 5.3, there are situations where it is relevant to assume that $f(\cdot)$ is smooth *except at a few points*. Instead of assuming an (approximate) GP prior for $f(\cdot)$ on the entire interval $[-L, L]$ we therefore suggest to divide $[-L, L]$ into a number n_p of segments, and then assume an individual GP prior for each segment $[p_i, p_{i+1}]$, independent of all other segments, as illustrated in Figure 2. The number of segments and the discontinuity points dividing them need to be learned from data, and an important prior is how the discontinuity points are distributed, i.e., the number n_p (e.g., geometrically distributed) and their locations $\{p_i\}_{i=1}^{n_p}$ (e.g., uniformly distributed).

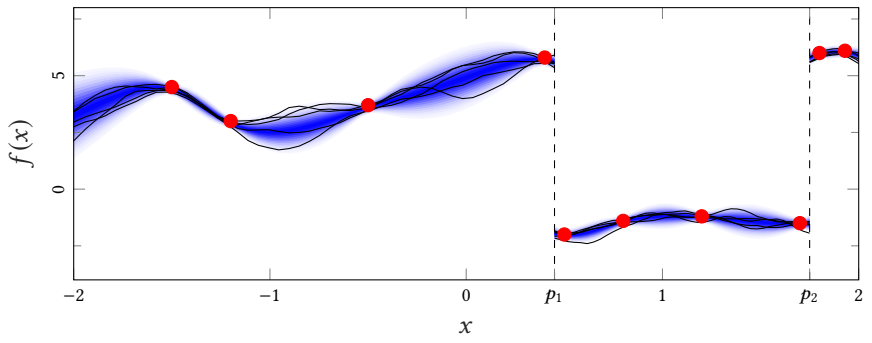


Figure 2: *The idea of a piecewise GP: the interval* $[-2, -2]$ *is divided by* $n_p = 2$ *discontinuity points* p_1 *and* p_2*, and a GP is used to model a function on each of these segments, independently of the other segments. For practical use, the learning algorithm have to be able to also infer the discontinuity points from data.*

3.3 Model summary

We will now summarize the proposed model. To avoid notational clutter, we omit u_t as well as the observation function (1b):

$$x_{t+1} = \sum_{i=0}^{n_p} A_i \bar{\varphi}(x_t) \mathbb{I}_{p_i \leq x_t < p_{i+1}} + v_t, \tag{15a}$$

$$v_t \sim \mathcal{N}(0, Q), \tag{15b}$$

with priors

$$[A_i, Q_i] \sim \mathcal{MNIW}(0, V, \ell, \Lambda), \; i = 0, \ldots, n_p, \tag{15c}$$

$$n_p, \{p_i\}_{i=1}^{n_p} \sim \text{arbitrary prior}, \tag{15d}$$

where $\mathbb{I}$ is the indicator function parameterizing the piecewise GP, and $\bar{\varphi}(x_t)$ was defined in (3). If the dynamical behavior of the data is close-to-linear, and a fairly accurate linear model is already available, this can be incorporated by adding the known linear function to the right hand side of (15a).

A good user practice is to sample parameters from the priors and simulate the model with those parameters, as a sanity check before entering the learning phase. Such a habit can also be fruitful for understanding what the prior assumptions mean in terms of dynamical behavior. There are standard routines for sampling from the $\mathcal{MN}$ as well as the $\mathcal{IW}$ distribution.

The suggested model can also be tailored if more prior knowledge is present, such as a physical relationship between two certain state variables. The suggested model can then be used to learn only the unknown part, as briefly illustrated by Svensson, Schön, et al. (2015, Example IV.B).

4 Learning

We now have a state-space model with a (potentially large) number of unknown parameters

$$\theta \triangleq \left\{ \{A_i, Q_i\}_{i=0}^{n_p}, n_p, \{p_i\}_{i=1}^{n_p} \right\}, \tag{16}$$

all with priors. ($g(\cdot)$ is still assumed to be known, but the extension follows analogously.) Learning the parameters is a quite general problem, and several learning strategies proposed in the literature are (partially) applicable, including optimization (Paduart et al. 2010), EM with extended Kalman filtering (Ghahramani and Roweis 1998) or sigma point filters (Kokkala et al. 2016), and particle Metropolis-Hastings (Tobar et al. 2015). We use another sequential Monte Carlo-based learning strategy, namely particle Gibbs with ancestor sampling (PGAS, Lindsten, Jordan, et al. 2014). PGAS allows us to take advantage of the fact that our proposed model (3) is linear in A (given x_t), at the same time as it has desirable theoretical properties.

4.1 Sequential Monte Carlo for system identification

Sequential Monte Carlo (SMC) methods have emerged as a tool for learning parameters in state-space models (Kantas et al. 2015; Schön, Lindsten, et al. 2015). At the very core when using SMC for system identification is the particle filter (Doucet and Johansen 2011), which provides a numerical solution to the state filtering problem, i.e., finding $p(x_t \mid y_{1:t})$. The particle filter propagates a set of weighted samples, particles, $\{x_t^i, \omega_t^i\}_{i=1}^N$ in the state-space model, approximating the filtering density by the empirical distribution $\widehat{p}(x_t \mid y_{1:t}) = \sum_{i=1}^N \omega_t^i \delta_{x_t^i}(x_t)$ for each t. Algorithmically, it amounts to iteratively weighting the particles with respect to the measurement y_t, resample among them, and thereafter propagate the resampled particles to the next time step $t+1$. The convergence properties of this scheme have been studied extensively (see references in Doucet and Johansen (2011)).

When using SMC methods for learning parameters, a key idea is to repeatedly infer the unknown states $x_{1:T}$ with a particle filter, and interleave this iteration with inference of the unknown parameters θ, as follows:

$$\begin{aligned} &\text{I. Use SMC to infer the states } x_{1:T} \text{ for given parameters } \theta. \\ &\text{II. Update the parameters } \theta \text{ to fit the states } x_{1:T} \text{ from the previous step.} \end{aligned} \tag{17}$$

There are several details left to specify in this iteration, and we will pursue two approaches for updating θ: one sample-based for exploring the full posterior $p(\theta|y_{1:T})$, and one EM-based for finding the maximum mode of the posterior, or equivalently, a regularized maximum likelihood estimate. Both alternatives will utilize the linear-in-parameter structure of the model (15), and use the Markov kernel PGAS (Lindsten, Jordan, et al. 2014) to handle the states in Step I of (17).

The PGAS Markov kernel resembles a standard particle filter, but has one of its state-space trajectories fixed. It is outlined by Algorithm 1, and is a procedure to asymptotically produce samples from $p(x_{1:T} \mid y_{1:T}, \theta)$, if repeated iteratively in a Markov chain Monte Carlo (MCMC, Robert and Casella 2004) fashion.

Algorithm 1: PGAS Markov kernel.

Input: Trajectory $x_{1:T}[k]$, number of particles N, known state space model (f, g, Q, R).
Output: Trajectory $x_{1:T}[k+1]$.

Sample $x_1^i \sim p(x_1)$, for $i = 1, \ldots, N-1$.
Set $x_1^N = x_1[k]$.
for $t = 1$ **to** T **do**
  Set $\omega_t^i = \mathcal{N}\left(y_t \mid g(x_t^i), R\right)$, for $i = 1, \ldots, N$.
  Sample a_t^i with $\mathbb{P}\left(a_t^i = j\right) \propto \omega_t^j$, for $i = 1, \ldots, N-1$.
  Sample $x_{t+1}^i \sim \mathcal{N}\left(f(x_t^{a_t^i}), Q\right)$, for $i = 1, \ldots, N-1$.
  Set $x_{t+1}^N = x_{t+1}[k]$.
  Sample a_t^N with $\mathbb{P}\left(a_t^N = j\right) \propto \omega_t^j \mathcal{N}\left(x_{t+1}^N \mid f(x_t^j), Q\right)$.
  Set $x_{1:t+1}^i = \{x_{1:t}^{a_t^i}, x_{t+1}^i\}$, for $i = 1, \ldots, N$.
end
Sample J with $\mathbb{P}(J = i) \propto \omega_T^i$ and set $x_{1:T}[k+1] = x_{1:T}^J$.

4.2 Parameter posterior

The learning problem will be split into the iterative procedure (17). In this section, the focus is on a key to Step II of (17), namely the conditional distribution of θ given states $x_{1:T}$ and measurements $y_{1:T}$. By utilizing the Markovian structure of the state-space model, the density $p(x_{1:T}, y_{1:T} \mid \theta)$ can be written as the product

$$p(x_{1:T}, y_{1:T} \mid \theta) = p(x_1) \prod_{t=1}^{T-1} p(x_{t+1} \mid x_t, \theta) p(y_t \mid x_t) = \\ = \underbrace{p(x_1) \prod_{t=1}^{T-1} p(x_{t+1} \mid x_t, \theta)}_{p(x_{1:T} \mid \theta)} \underbrace{\prod_{t=1}^{T} p(y_t \mid x_t)}_{p(y_{1:T} \mid x_{1:T})}. \quad (18)$$

Since we assume that the observation function (1b) is known, $p(y_t \mid x_t)$ is independent of θ, which in turn means that (18) is proportional to $p(x_{1:T} \mid \theta)$. Further, we assume for now that $p(x_1)$ is also known, and therefore omit it. Let us consider the case without discontinuity points, $n_p = 0$. Since v_t is assumed to be Gaussian, $p(x_{t+1} \mid x_t, u_t, \theta) = \mathcal{N}(x_{t+1} \mid A\bar{\varphi}(x_t, u_t), Q)$, we can with some algebraic manipulations (Gibson and Ninness 2005) write

$$\log p(x_{1:T} \mid A, Q) = \\ = -\tfrac{T n_x}{2} \log(2\pi) - \tfrac{T}{2} \log \det(Q) - \tfrac{1}{2} \operatorname{tr}\left\{Q^{-1}\left(\Phi - A\Psi^{\mathsf{T}} - \Psi A^{\mathsf{T}} + A\Sigma A^{\mathsf{T}}\right)\right\}, \quad (19)$$

with the (sufficient) statistics

$$\Phi = \sum_{t=1}^{T} x_{t+1} x_{t+1}^{\mathsf{T}}, \quad \Psi = \sum_{t=1}^{T} x_{t+1} \bar{\varphi}(x_t, u_t)^{\mathsf{T}}, \quad \Sigma = \sum_{t=1}^{T} \bar{\varphi}(x_t, u_t) \bar{\varphi}(x_t, u_t)^{\mathsf{T}}. \quad (20a)$$

The density (19) gives via Bayes' rule and the $\mathcal{MNIW}$ prior distribution for A, Q from Section 3

$$\begin{aligned}\log p(A,Q) &= \log p(A \mid Q) + \log p(Q) \propto \\ &\propto -\tfrac{1}{2}(n_x + \ell + m + 1)\log\det(Q) - \tfrac{1}{2}\mathrm{tr}\left\{Q^{-1}\left(\Lambda + AV^{-1}A^{\mathsf{T}}\right)\right\}, \quad (21)\end{aligned}$$

the posterior

$$\begin{aligned}\log p(A,Q \mid x_{1:t}) &\propto \log p(x_{1:t} \mid A,Q) + \log p(A,Q) \propto \\ &\propto -\tfrac{1}{2}(n_x + T + \ell + m + 1)\log\det Q - \tfrac{1}{2}\mathrm{tr}\big\{Q^{-1}\big(\Lambda + \Phi - \Psi(\Sigma + V^{-1})^{-1}\Psi^{\mathsf{T}} + \\ &\qquad + (A - \Psi(\Sigma + V^{-1})^{-1})Q^{-1}(A - \Psi(\Sigma + V^{-1})^{-1})^{\mathsf{T}}\big)\big\}. \quad (22)\end{aligned}$$

This expression will be key for learning: For the fully Bayesian case, we will recognize (22) as another $\mathcal{MNIW}$ distribution and sample from it, whereas we will maximize it when seeking a point estimate.

Remarks: The expressions needed for an unknown observation function $g(\cdot)$ are completely analogous. The case with discontinuity points becomes essentially the same, but with individual A_i, Q_i and statistics for each segment. If the right hand side of (15a) also contains a known function $h(x_t)$, e.g., if the proposed model is used only to describe deviations from a known linear model, this can easily be taken care of by noting that now $p(x_{t+1} \mid x_t, u_t, \theta) = \mathcal{N}(x_{t+1} - h(x_t) \mid A\bar{\varphi}(x_t, u_t), Q)$, and thus compute the statistics (20) for $(x_{t+1} - h(x_t))$ instead of x_{t+1}.

4.3 Inferring the posterior—Bayesian learning

There is no closed form expression for $p(\theta \mid y_{1:T})$, the distribution to infer in the Bayesian learning. We thus resort to a numerical approximation by drawing samples from $p(\theta, x_{1:T} \mid y_{1:T})$ using MCMC. (Alternative, variational methods could be used, akin to Frigola, Y. Chen, et al. (2014)). MCMC amounts to constructing a procedure for 'walking around' in θ-space in such a way that the steps $\ldots, \theta[k], \theta[k+1], \ldots$ eventually, for k large enough, become samples from the distribution of interest.

Let us start in the case without discontinuity points, i.e., $n_p \equiv 0$. Since (21) is $\mathcal{MNIW}$, and (19) is a product of (multivariate) Gaussian distributions, (22) is also an $\mathcal{MNIW}$ distribution (Dawid 1981; Wills et al. 2012). By identifying components in (22), we conclude that

$$\begin{aligned}&p(\theta \mid x_{1:T}, y_{1:T}) = \quad (23)\\ =&\mathcal{MNIW}\big(A, Q \mid \Psi(\Sigma + V^{-1})^{-1}, (\Sigma + V^{-1})^{-1}, \Lambda + \Phi - \Psi(\Sigma + V^{-1})^{-1}\Psi^{\mathsf{T}}, \ell + Tn_x\big)\end{aligned}$$

We now have (23) for sampling θ given the states $x_{1:T}$ (cf. (17), step II), and Algorithm 1 for sampling the states $x_{1:T}$ given the model θ (cf. (17), step I). This makes a particle Gibbs sampler (Andrieu et al. 2010), cf. (17).

If there are discontinuity points to learn, i.e., n_p is to be learned, we can do that by acknowledging the hierarchical structure of the model. For brevity, we denote $\{n_p, \{p_i\}_{i=1}^{n_p}\}$ by ξ, and $\{A_i, Q_i\}_{i=1}^{n_p}$ simply by A, Q. We suggest to first sample ξ from $p(\xi \mid x_{1:T})$, and next sample A, Q from $p(A, Q \mid x_{1:T}, \xi)$. The distribution for sampling

A, Q is the $\mathcal{MNIW}$ distribution (23), but conditional on data only in the relevant segment. The other distribution, $p(\xi \mid x_{1:T})$, is trickier to sample from. We suggest to use a Metropolis-within-Gibbs step (Müller 1991), which means that we first sample ξ^* from a proposal $q(\xi^* \mid \xi[k])$ (e.g., a random walk), and then accept it as $\xi[k{+}1]$ with probability $\min\left(1, \frac{p(\xi^* \mid x_{1:T})}{p(\xi[k] \mid x_{1:T})} \frac{q(\xi[k] \mid \xi[k])}{q(\xi^* \mid \xi[k])}\right)$, and otherwise just set $\xi[k{+}1] = \xi[k]$. Thus we need to evaluate $p(\xi^* \mid x_{1:T}) \propto p(x_{1:T} \mid \xi^*)p(\xi^*)$. The prior $p(\xi^*)$ is chosen by the user. The density $p(x_{1:T} \mid \xi)$ can be evaluated using the expression (see Appendix A.1)

$$p(x_{1:T} \mid \xi) = \\ = \prod_{i=0}^{n_p} \frac{2^{n_x T_i/2}}{(2\pi)^{T_i/2}} \frac{\Gamma_{n_x}(\frac{l+N}{2})}{\Gamma_{n_x}(\frac{l}{2})} \frac{|V^{-1}|^{n_x/2}}{|\Sigma_i + V^{-1}|^{n_x/2}} \times \frac{|\Lambda|^{l/2}}{|\Lambda + \Phi_i + \Psi_i(\Sigma_i + V^{-1})^{-1}\Psi_i^{\mathsf{T}}|^{\frac{l+N}{2}}} \tag{24}$$

where Φ_i etc. denotes the statistics (20) restricted to the corresponding segment, and T_i is the number of data points in segment i ($\sum_i T_i = T$). The suggested Bayesian learning procedure is summarized in Algorithm 2.

Algorithm 2: Bayesian learning of (15).

Input: Data $y_{1:T}$, priors on A, Q and ξ.
Output: K MCMC-samples with $p(x_{1:T}, A, Q, \xi \mid y_{1:T})$ as invariant distribution.

1 Initialize $A[0], Q[0], \xi[0]$.
2 **for** $k = 0$ **to** K **do**
3 Sample $x_{1:T}[k{+}1] \mid A[k], Q[k], \xi[k]$ — *Algorithm 1*
4 Sample $\xi[k{+}1] \mid x_{1:T}[k{+}1]$ — *Section 4.3*
5 Sample $Q[k{+}1] \mid \xi[k{+}1], x_{1:T}[k{+}1]$ — *by (23)*
6 Sample $A[k{+}1] \mid Q[k{+}1], \xi[k{+}1], x_{1:T}[k{+}1]$ — *by (23)*
7 **end**

Our proposed algorithm can be seen as a combination of a collapsed Gibbs sampler and Metropolis-within-Gibbs, a combination which requires some attention to be correct (Dyk and Jiao 2014), see Appendix A.2 for details in our case. If the hyperparameters parameterizing V and/or the initial states are unknown, it can be included by extending Algorithm 2 with extra Metropolis-within-Gibbs steps (see Svensson, Solin, et al. (2016) for details).

4.4 Regularized maximum likelihood

A widely used alternative to Bayesian learning is to find a *point estimate* of θ maximizing the likelihood of the training data $p(y_{1:T} \mid \theta)$, i.e., *maximum likelihood*. However, if a very flexible model is used, some kind of mechanism is needed to prevent the model from overfit to training data. We will therefore use the priors from Section 3 as regularization for the maximum likelihood estimation, which can also be understood as seeking the maximum mode of the posterior. We will only treat the case with no discontinuity points, as the case with discontinuity points does not allow for closed form maximization, but requires numerical optimization tools, and we therefore suggest Bayesian learning for that case instead.

The learning will build on the particle stochastic approximation EM (PSAEM) method proposed by Lindsten (2013), which uses a stochastic approximation of the EM scheme (Delyon et al. 1999; Dempster et al. 1977; Kuhn and Lavielle 2004). EM addresses maximum likelihood estimation in problems with latent variables. For system identification, EM can be applied by taking the states $x_{1:T}$ as the latent variables, (Ghahramani and Roweis (1998); another alternative would be to take the noise sequence $v_{1:T}$ as the latent variables, Umenberger et al. (2015)). The EM algorithm then amounts to iteratively (cf. (17)) computing the expectation (E-step)

$$Q(\theta, \theta[k]) = \mathbb{E}_{x_{1:T}} \left[\log p(\theta \mid x_{1:T}, y_{1:T}) \mid y_{1:T}, \theta[k] \right], \tag{25a}$$

and updating θ in the maximization (M-step) by solving

$$\theta[k+1] = \arg\max_{\theta} Q(\theta, \theta[k]), \tag{25b}$$

In the standard formulation, Q is usually computed with respect to the joint likelihood density for $x_{1:T}$ and $y_{1:T}$. To incorporate the prior (our regularization), we may consider the prior as an additional observation of θ, and we have thus replaced (19) by (22) in Q. Following Gibson and Ninness (2005), the solution in the M-step is found as follows: Since Q^{-1} is positive definite, the quadratic form in (22) is maximized by

$$A = \Phi(\Sigma + V^{-1}). \tag{26a}$$

Next, substituting this into (22), the maximizing Q is

$$Q = \tfrac{1}{n_x + T n_x + \ell + m + 1} \left(\Lambda + \Phi - \Psi(\Sigma + V^{-1})^{-1} \Psi \right). \tag{26b}$$

We thus have solved the M-step exactly. To compute the expectation in the E-step, approximations are needed. For this, a particle smoother (Lindsten and Schön 2013) could be used, which would give a learning strategy in the flavor of Schön, Wills, et al. (2011). The computational load of a particle smoother is, however, unfavorable, and PSAEM uses Algorithm 1 instead.

PSAEM also replaces and replace the Q-function (25a) with a Robbins-Monro stochastic approximation of Q,

$$\mathbb{Q}_k(\theta) = (1 - \gamma_k)\mathbb{Q}_{k-1}(\theta) + \gamma_k \log p(\theta \mid x_{1:T}[k], y_{1:T}), \tag{27}$$

where $\{\gamma_k\}_{k \geq 1}$ is a decreasing sequence of positive step sizes, with $\gamma_1 = 1$, $\sum_k \gamma_k = \infty$ and $\sum_k \gamma_k^2 < \infty$. I.e., γ_k should be chosen such that $k^{-1} \leq \gamma_k < k^{-0.5}$ holds up to proportionality, and the choice $\gamma_k = k^{-2/3}$ has been suggested in the literature (Delyon et al. 1999, Section 5.1). Here, $x_{1:T}[k]$ is a sample from an ergodic Markov kernel with $p(x_{1:T} \mid y_{1:T}, \theta)$ as its invariant distribution, i.e., Algorithm 1. At a first glance, the complexity of $\mathbb{Q}_k(\theta)$ appears to grow with k because of its iterative definition. However, since $p(x_{1:T}, y_{1:T} \mid \theta)$ belongs to the exponential family, we can write

$$p(x_{1:T}[k], y_{1:T} \mid \theta) = h(x_{1:T}[k], y_{1:T}) c(\theta) \exp\left(\eta^{\mathsf{T}}(\theta) t[k] \right), \tag{28}$$

where $t[k]$ is the statistics (20) of $\{x_{1:T}[k], y_{1:T}\}$. The stochastic approximation $\mathbb{Q}_k(\theta)$ (27) thus becomes

$$\mathbb{Q}_k(\theta) \propto \log p(\theta) + \log c(\theta) + \eta^{\mathsf{T}}(\theta) \left(\gamma_k t[k] + (1 - \gamma_k)\gamma_{k-1} t[k-1] + \dots \right). \tag{29}$$

Algorithm 3: Regularized maximum likelihood learning of (15).

1 Initialize $\theta[1]$.
2 **for** $k > 0$ **do**
3 Sample $x_{1:T}[k]$ with parameters $\theta[k]$. *Algorithm 1*
4 Compute and update the statistics of $x_{1:T}[k]$ *by (20, 30)*
5 Compute $\theta[k+1] = \arg\max_\theta \mathbb{Q}(\theta)$ *by (26)*
6 **end**

Now, we note that if keeping track of the statistics $\gamma_k t[k] + \gamma_{k\text{-}1} t[k\text{-}1] + \ldots$, the complexity of $\mathbb{Q}$ does not grow with k. We therefore introduce the following iterative update of the statistics

$$\Phi_k = (1-\gamma_k)\Phi_{k-1} + \gamma_k \Phi(x_{1:T}[k]), \tag{30a}$$
$$\Psi_k = (1-\gamma_k)\Psi_{k-1} + \gamma_k \Psi(x_{1:T}[k]), \tag{30b}$$
$$\Sigma_k = (1-\gamma_k)\Sigma_{k-1} + \gamma_k \Sigma(x_{1:T}[k]), \tag{30c}$$

where $\Phi(x_{1:T}[k])$ refers to (20), etc. With this parametrization, we obtain $\arg\max_\theta \mathbb{Q}_k(\theta)$ as the solutions for the vanilla EM case by just replacing Φ by Φ_k, etc., in (26). Algorithm 3 summarizes.

4.5 Convergence and consistency

We have proposed two algorithms for learning the model introduced in Section 3. The Bayesian learning, Algorithm 2, will by construction (as detailed in Appendix A.2) asymptotically provide samples from the true posterior density $p(\theta \mid y_{1:T})$ (Andrieu et al. 2010). However, no guarantees regarding the length of the burn-in period can be given, which is the case for all MCMC methods, but the numerical comparisons in Svensson, Solin, et al. (2016) and in Section 5.1 suggest that the proposed Gibbs scheme is efficient compared to its state-of-the-art alternatives. The regularized maximum likelihood learning, Algorithm 3, can be shown to converge under additional assumptions (Kuhn and Lavielle 2004; Lindsten 2013) to a stationary point of $p(\theta|y_{1:T})$, however not necessarily a global maximum. The literature on PSAEM is not (yet) very rich, and the technical details regarding the additional assumptions remains to be settled, but we have not experienced any problems of non-convergence in practice.

4.6 Initialization

The convergence of Algorithm 2 is not relying on the initialization, but the burn-in period can nevertheless be reduced. One useful idea by Paduart et al. (2010) is thus to start with a linear model, which can be obtained using classical methods. To avoid Algorithm 3 from converging to a poor local minimum, Algorithm 2 can first be run to explore the 'landscape' and from that, a promising point for initialization of Algorithm 3 can be chosen.

For convenience, we assumed the distribution of the initial states, $p(x_1)$, to be known. This is perhaps not realistic, but its influence is minor in many cases. If needed,

they can be included in Algorithm 2 by an additional Metropolis-within-Gibbs step, and in Algorithm 3 by including them in (22) and use numerical optimization tools.

5 Experiments

We will give three numerical examples: a toy example, a classic benchmark, and thereafter a real data set from two cascaded water tanks. Matlab code for all examples is available via the first authors homepage.

5.1 A first toy example

Consider the following example from Tobar et al. (2015),

$$x_{t+1} = 10\text{sinc}\left(\frac{x_t}{7}\right) + v_t, \qquad v_t \sim \mathcal{N}(0, 4), \tag{31a}$$

$$y_t = x_t + e_t, \qquad e_t \sim \mathcal{N}(0, 4). \tag{31b}$$

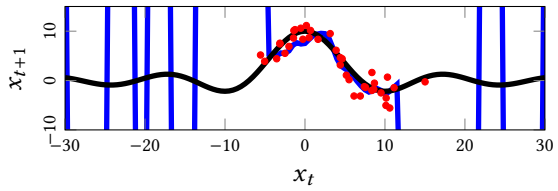
We generate $T = 40$ observations, and the challenge is to learn $f(\cdot)$, when $g(\cdot)$ and the noise variances are known. Note that even though $g(\cdot)$ is known, y is still corrupted by a non-negligible amount of noise.

In Figure 3 (a) we illustrate the performance of our proposed model using $m = 40$ basis functions on the form (4) when Algorithm 3 is used *without* regularization. This gives a nonsense result that is overfitted to data, since $m = 40$ offers too much flexibility for this example. When a GP-inspired prior from an exponentiated quadratic covariance function (10a) with length scale $\ell = 3$ and $s_f = 50$ is considered, we obtain (b), that is far more useful and follows the true function rather well in regions were data is present. We conclude that we do *not* need to choose m carefully, but can rely on the priors for regularization. In (c), we use the same prior and explore the full posterior by Algorithm 2, obtaining information about uncertainty as a part of the learned model (illustrated by the a posteriori credibility interval), in particular in regions where no data is present.

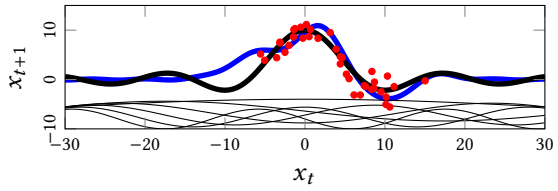
In the next figure, (d), we replace the set of $m = 40$ basis functions on the form (4) with 8 Gaussian kernels to reconstruct the model proposed by Tobar et al. (2015). As clarified by Tobar (2016), the prior on the coefficients is a Gaussian distribution inspired by a GP, which makes a close connection to out work. We use Algorithm 2 for learning also in (d) (which is possible thanks to the Gaussian prior). In (e), on the contrary, the learning algorithm from Tobar et al. (2015), Metropolis-Hastings, is used, requiring more computation time. Tobar et al. (2015) spend a considerable effort to pre-process the data and carefully distribute the Gaussian kernels in the state space, see the bottom of (d).

5.2 Narendra-Li benchmark

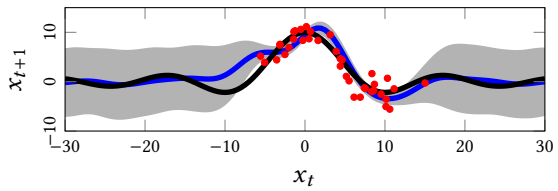
The example introduced by Narendra and Li (1996) has become a benchmark for nonlinear system identification, e.g., Pan et al. 2009; Roll et al. 2005; Stenman 1999; The MathWorks, Inc. 2015; Wen et al. 2007; Xu et al. 2009. The benchmark is defined



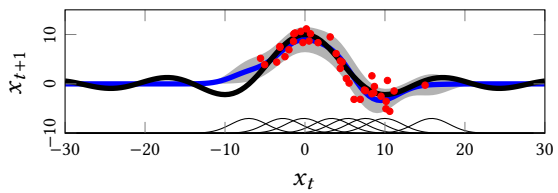
(a) *Maximum likelihood estimation of our proposed model, without regularization; a useless model.*



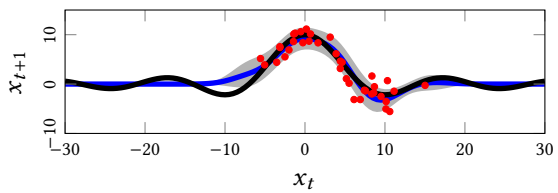
(b) *Maximum likelihood estimation of our proposed model, with regularization. A subset of the $m = 40$ basis functions used are sketched at the bottom. Computation time: 12 s.*



(c) *Bayesian learning of our proposed model, i.e., the entire posterior is explored. Computation time: 12 s.*



(d) *Posterior distribution for the basis functions (sketched at the bottom) used by Tobar et al. (2015), but Algorithm 2 for learning. Computation time: 9 s.*



(e) *The method presented by Tobar et al. (2015), using Metropolis-Hastings for learning. Computation time: 32 s.*

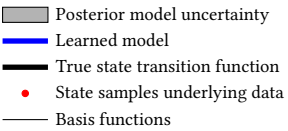


Figure 3: *True function (black), states underlying the data (red) and learned model (blue, gray) for the example in Section 5.1.*

Reference	RMSE	T
This paper	**0.06***	2 000
Roll et al. (2005)	0.43	50 000
Stenman (1999)	0.46	50 000
Xu et al. (2009) (AHH)	0.31	2 000
Xu et al. (2009) (MARS)	0.49	2 000

**The number is averaged over 10 data realizations.*

Table 7.1: *Results of the Narendra-Li Benchmark (T is the number of data samples in training data).*

by the model

$$x_{t+1}^1 = \left(\frac{x_t^1}{1+(x_t^1)^2} + 1\right)\sin(x_t^2), \tag{32a}$$

$$x_{t+1}^2 = x_t^2\cos(x_t^2) + x_t^1\exp\left(-\frac{(x_t^1)^2+(x_t^2)^2}{8}\right) + \frac{(u_t)^3}{1+(u_t)^2+0.5\cos(x_t^1+x_t^2)}, \tag{32b}$$

$$y_t = \frac{x_t^1}{1+0.5\sin(x_t^2)} + \frac{x_t^2}{1+0.5\sin(x_t^1)}, \tag{32c}$$

where $x_t = [x_t^1\ x_t^2]^\mathsf{T}$. The training data (only input-output data) is obtained with an input sequence sampled uniformly and iid from the interval $[-2.5, 2.5]$. The input data for the test data is $u_t = \sin(2\pi t/10) + \sin(2\pi t/25)$.

According to Narendra and Li (1996, p. 369), it 'does not correspond to any real physical system and is deliberately chosen to be complex and distinctly nonlinear'. The original formulation is somewhat extreme, with no noise and $T = 500\,000$ data samples for learning. In the work by Stenman (1999), a white Gaussian measurement noise with variance 0.1 is added to the training data, and less data is used for learning. We apply Algorithm 2 with a second order state-space model, $n_p = 0$, and a known, linear $g(\cdot)$. (Even though the data is generated with a nonlinear $g(\cdot)$, it turn out this will give a satisfactory performance.) We use 7 basis functions per dimension (i.e., 686 coefficients $w^{(j)}$ to learn in total) on the form (5), with prior from the covariance function (10a) with length scale $\ell = 1$.

For the original case without any noise, but using only $T = 500$ data points, a root mean square error (RMSE) for the simulation of 0.039 is obtained. Our result is in contrast to the significantly bigger simulation errors by Narendra and Li (1996), although they use 1 000 times as many data points. For the more interesting case *with* measurement noise in the training data, we achieve a result almost the same as for the noise-free data. We compare to some previous results reported in the literature in Table 7.1. It is clear that the proposed model is capable enough to well describe the behavior of the system (32).

5.3 Water tank data

We consider the data sets provided by M. Schoukens et al. (2015), collected from a physical system consisting of two cascaded water tanks, where the outlet of the first tank goes into the second one. A training and a test data set is provided, both with 1024 data samples. The input u (voltage) governs the inflow to the first tank,

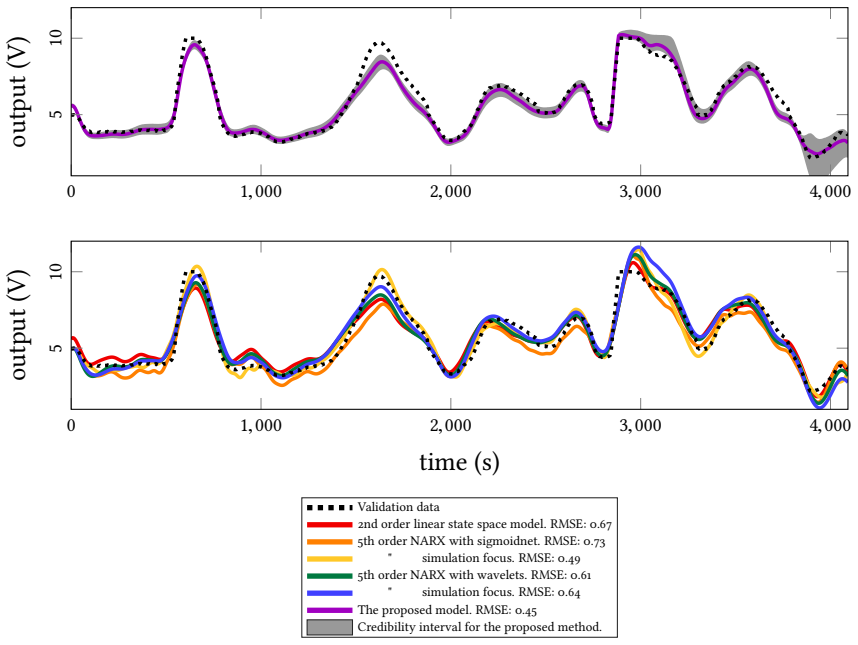


Figure 4: *The simulated and true output for the test data in the water tank experiment (Section 5.3). The order of the NARX models refers to the number of regressors in u and y.*

and the output y (voltage) is the measured water level in the second tank. This is a well-studied system (e.g., Wigren and J. Schoukens 2013), but a peculiarity in this data set is the presence of overflow, both in the first and the second tank. When the first tank overflows, it goes only partly into the second tank.

We apply our proposed model, with a two dimensional state space. The following structure is used:

$$x^1_{t+1} = f^1(x^1_t, u_t) + v^1_t, \tag{33a}$$
$$x^2_{t+1} = f^2(x^1_t, x^2_t, u_t) + v^2_t, \tag{33b}$$
$$y_t = x^2_t + e_t. \tag{33c}$$

It is surprisingly hard to perform better than linear models in this problem, perhaps because of the close-to-linear dynamics in most regimes, in combination with the non-smooth overflow events. This calls for discontinuity points to be used. Since we can identify the overflow level in the second tank directly in the data, we fix a discontinuity point at $x^2 = 10$ for $f^2(\cdot)$, and learn the discontinuity points for $f^1(\cdot)$. Our physical intuition about the water tanks is a close-to-linear behavior in most regimes, apart from the overflow events, and we thus use the covariance function (10a) with a rather long length scale $\ell = 3$ as prior. We also limit the number of basis functions to 5 per dimension for computational reasons (in total, there are 150 coefficients $w^{(j)}$ to learn).

Algorithm (2) is used to sample from the model posterior. We use all samples to simulate the test output from the test input for each model to represent a posterior for

the test data output, and compute the RMSE for the difference between the posterior mode and the true test output. A comparison to nonlinear ARX-models (NARX, Ljung 1999) is also made in Figure 4. It is particularly interesting to note how the different models handle the overflow around time 3 000 in the test data. We have tried to select the most favorable NARX configurations, and when finding their parameters by maximizing their likelihood (which is equivalent to minimizing their 1-step-ahead prediction, Ljung 1999), the best NARX model is performing approximately 35% worse (in terms of RMSE) than our proposed model. When instead learning the NARX models with 'simulation focus', i.e., minimizing their simulation error on the training data, their RMSE decreases, and approaches almost the one of our model for one of the models[4]. While the different settings in the NARX models have a large impact on the performance, and therefore a trial-and-error approach is needed for the user to determine satisfactory settings, our approach offers a more systematic way to encode the physical knowledge at hand into the modeling process, and achieves a competitive performance.

6 Conclusions and further work

During the recent years, there has been a rapid development of powerful parameter estimation tools for state-space models. These methods allows for learning in complex and extremely flexible models, and this paper is a response to the situation when the learning algorithm is able to learn a state-space model more complex than the information contained in the training data (cf. Figure 3a). For this purpose, we have in the spirit of Peterka (1981) chosen to formulate GP-inspired priors for a basis function expansion, in order to 'softly' tune its complexity and flexibility in a way that hopefully resonates with the users intuition. In this sense, our work resembles the recent work in the machine learning community on using GPs for learning dynamical models (see, e.g., Bijl et al. 2016; Frigola-Alcade 2015; Mattos et al. 2016). However, not previously well explored in the context of dynamical systems, is the combination of discontinuities and the smooth GP. We have also tailored efficient learning algorithms for the model, both for inferring the full posterior, and finding a point estimate.

It is a rather hard task to make a sensible comparison between our *model-focused approach*, and approaches which provide a general-purpose black-box learning algorithm with very few user choices. Because of their different nature, we do not see any ground to claim superiority of one approach over another. In the light of the promising experimental results, however, we believe this model-focused perspective can provide additional insight into the nonlinear system identification problem. There is certainly more to be done and understand when it comes to this approach, in particular concerning the formulation of priors.

We have proposed an algorithm for Bayesian learning of our model, which renders K samples of the parameter posterior, representing a *distribution* over models. A relevant question is then how to compactly represent and use these samples to efficiently make predictions. Many control design methods provide performance

[4]Since the corresponding change in learning objective is not available to our model, this comparison might only offer partial insight. It would, however, be an interesting direction for further research to implement learning with 'simulation focus' in the Bayesian framework.

guarantees for a perfectly known model. An interesting topic would hence be to incorporate model *uncertainty* (as provided by the posterior) into control design and provide probabilistic guarantees, such that performance requirements are fulfilled with, e.g., 95% probability.

A Appendix: Technical details

A.1 Derivation of (24)

From Bayes' rule, we have

$$p(x_{1:T} \mid \xi) = \frac{p(A, Q \mid \xi) p(x_{1:T} \mid A, Q, \xi)}{p(A, Q \mid \xi, x_{1:T})}. \tag{34}$$

The expression for each term is found in (12-14), (18) and (23), respectively. All of them have a functional form $\eta(\xi) \cdot |Q|^{\chi(\xi)} \cdot \exp\left(-\frac{1}{2}\mathrm{tr}\left(Q^{-1}\tau(A, x_{1:T}, \xi)\right)\right)$, with different η, χ and τ. Starting with the $|Q|$-part, the sum of the exponents for all such terms in both the numerator and the denominator sums to 0. The same thing happens to the exp-part, which can either be worked out algebraically, or realized since $p(x_{1:T} \mid \xi)$ is independent of Q. What remains is everything stemming from η, which indeed is $p(x_{1:T} \mid \xi)$, (24).

A.2 Invariant distribution of Algorithm 2

where the problematic issue, obstructing the invariant distribution, is the joint conditioning on $a[k+1]$ *and* $b[k]$ (marked in red), since $a[k+1]$ has been sampled without conditioning on $b[k]$. Spelling out the details from Algorithm 2 in Algorithm 4, it is clear this problematic conditioning is not present.

1	Sample $a[k+1] \sim p(a \mid b[k])$	*Gibbs*
2	Sample $b[k+1] \sim \mathcal{MH}(b \mid a[k+1], b[k])$	*MH*

So far, this is a valid sampler. However, if collapsing over b, the sampler becomes

1	Sample $a[k+1] \sim p(a)$	*Partially collapsed Gibbs*
2	Sample $b[k+1] \sim \mathcal{MH}(b \mid a[k+1], b[k])$	*MH*

where the problematic issue, obstructing the invariant distribution, is the joint conditioning on $a[k+1]$ *and* $b[k]$ (marked in red), since $a[k+1]$ has been sampled without conditioning on $b[k]$. Spelling out the details from Algorithm 2 in Algorithm 4, it is clear this problematic conditioning is not present:

Algorithm 4: Details of Algorithm 2.

2	**for** $k = 0$ **to** K **do**	
3	Sample $x_{1:T}[k+1] \mid A[k], Q[k], \xi[k]$	*Algorithm 1*
4	Sample $\xi[k+1] \mid x_{1:T}[k+1]$	*Section 4.3*
5	Sample $Q[k+1] \mid \xi[k+1], x_{1:T}[k+1]$	*by* (23)
6	Sample $A[k+1] \mid Q[k+1], \xi[k+1], x_{1:T}[k+1]$	*by* (23)
7	**end**	

References

Mauricio A. Alvarez, David Luengo, and Neil D. Lawrence (2013). "Linear latent force models using Gaussian processes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.11, pp. 2693–2705.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

Hildo Bijl, Thomas B. Schön, Jan-Willem van Wingerden, and Michel Verhaegen (2016). "Onlise sparse Gaussian process training with input noise". In: *arXiv:1601.08068*.

Christopher M. Bishop (2006). *Pattern recognition and machine learning*. New York, NY, USA: Springer.

Ben Calderhead, Mark Girolami, and Neil D. Lawrence (2008). "Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes". In: *Advances in Neural Information Processing Systems 21 (NIPS)*. Vancouver, BC, Canada, pp. 217–224.

Tianshi Chen, Henrik Ohlsson, and Lennart Ljung (2012). "On the estimation of transfer functions, regularizations and Gaussian processes—Revisited". In: *Automatica* 48.8, pp. 1525–1535.

A. Philip Dawid (1981). "Some matrix-variate distribution theory: notational considerations and a Bayesian application". In: *Biometrika* 68.1, pp. 265–274.

Ramón A. Delgado, Juan C. Agüero, Graham C. Goodwin, and Eduardo M.A.M. Mendes (2015). "Application of rank-constrained optimisation to nonlinear system identification". In: *Proceedings of the 1st IFAC Conference on Modelling, Identification and Control of Nonlinear Systems (MICNON)*. Saint Petersburg, Russia, pp. 814–818.

Bernard Delyon, Marc Lavielle, and Éric Moulines (1999). "Convergence of a stochastic approximation version of the EM algorithm". In: *Annals of Statistics* 27.1, pp. 94–128.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–38.

Arnaud Doucet and Adam M. Johansen (2011). "A tutorial on particle filtering and smoothing: fifteen years later". In: *Nonlinear Filtering Handbook*. Ed. by D. Crisan and B. Rozovsky. Oxford, UK: Oxford University Press, pp. 656–704.

David A. van Dyk and Xiyun Jiao (2014). "Metropolis-Hastings within partially collapsed Gibbs samplers". In: *Journal of Computational and Graphical Statistics* 24.2, pp. 301–327.

Roger Frigola-Alcade (2015). "Bayesian time series learning with Gaussian processes". PhD thesis. UK: University of Cambridge.

Roger Frigola, Yutian Chen, and Carl Rasmussen (2014). "Variational Gaussian process state-space models". In: *Advances in Neural Information Processing Systems 27 (NIPS)*. Montréal, QC, Canada, pp. 3680–3688.

Roger Frigola, Fredrik Lindsten, Thomas B. Schön, and Carl Rasmussen (2013). "Bayesian inference and learning in Gaussian process state-space models with particle MCMC". In: *Advances in Neural Information Processing Systems 26 (NIPS)*. Lake Tahoe, NV, USA, pp. 3156–3164.

Zoubin Ghahramani and Sam T. Roweis (1998). "Learning nonlinear dynamical systems using an EM algorithm". In: *Advances in Neural Information Processing Systems (NIPS) 11*. Denver, CO, USA, pp. 431–437.

Stuart Gibson and Brett Ninness (2005). "Robust maximum-likelihood estimation of multivariable dynamic systems". In: *Automatica* 41.10, pp. 1667–1682.

Anatoli Juditsky, Håkan Hjalmarsson, Albert Benveniste, Bernard Delyon, Lennart Ljung, Jonas Sjöberg, and Qinghua Zhang (1995). "Nonlinear black-box models in system identification: mathematical foundations". In: *Automatica* 31.12, pp. 1725–1750.

Nikolas Kantas, Arnaud Doucet, Sumeetpal S. Singh, Jan M. Maciejowski, and Nicolas Chopin (2015). "On particle methods for parameter estimation in state-space models". In: *Statistical Science* 30.3, pp. 328–351.

Juš Kocijan (2016). *Modelling and control of dynamic systems using Gaussian process models*. Basel, Switzerland: Springer International.

Juš Kocijan, Agathe Girard, Blaž Banko, and Roderick Murray-Smith (2005). "Dynamic systems identification with Gaussian processes". In: *Mathematical and Computer Modelling of Dynamical Systems* 11.4, pp. 411–424.

Juho Kokkala, Arno Solin, and Simo Särkkä (2016). "Sigma-point filtering and smoothing based parameter estimation in nonlinear dynamic systems". In: *Journal of Advances in Information Fusion* 11.1, pp. 15–30.

Estelle Kuhn and Marc Lavielle (2004). "Coupling a stochastic approximation version of EM with an MCMC procedure". In: *ESAIM: Probability and Statistics* 8, pp. 115–131.

Fredrik Lindsten (2013). "An efficient stochastic approximation EM algorithm using conditional particle filters". In: *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vancouver, BC, Canada, pp. 6274–6278.

Fredrik Lindsten, Michael I. Jordan, and Thomas B. Schön (2014). "Particle Gibbs with ancestor sampling". In: *The Journal of Machine Learning Research (JMLR)* 15.1, pp. 2145–2184.

Fredrik Lindsten and Thomas B. Schön (2013). "Backward simulation methods for Monte Carlo statistical inference". In: *Foundations and Trends in Machine Learning* 6.1, pp. 1–143.

Lennart Ljung (1999). *System identification: theory for the user*. 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall.

Lennart Ljung (2010). "Perspectives on system identification". In: *Annual Reviews in Control* 34.1, pp. 1–12.

Benn Macdonald, Catherine Higham, and Dirk Husmeier (2015). "Controversy in mechanistic modelling with Gaussian processes". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Lille, France, pp. 1539–1547.

César L. C. Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A. Barreto, and Neil D. Lawrence (2016). "Recurrent Gaussian processes". In: *4th International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.

Per Mattsson, Dave Zachariah, and Petre Stoica (2016). "Recursive identification of nonlinear systems using latent variables". In: *arXiv:1606.04366*.

Peter Müller (1991). *A generic approach to posterior intergration and Gibbs sampling*. Tech. rep. West Lafayette, IN, USA: Department of Statistics, Purdue University.

Kumpati S. Narendra and Sai-Ming Li (1996). "Neural networks in control systems". In: ed. by Paul Smolensky, Michael C. Mozer, and David E. Rumelhart. Hillsdale, NJ, USA: Lawrence Erlbaum Associates. Chap. 11, pp. 347–394.

Magnus Nørgård, Ole Ravn, Niels Kjølstad Poulsen, and Lars Kai Hansen (2000). *Neural networks for modelling and control of dynamic systems*. London, UK: Springer-Verlag.

Johan Paduart, Lieve Lauwers, Jan Swevers, Kris Smolders, Johan Schoukens, and Rik Pintelon (2010). "Identification of nonlinear systems using polynomial nonlinear state space models". In: *Automatica* 46.4, pp. 647–656.

Tian Hong Pan, Shaoyuan Li, and Ning Li (2009). "Optimal bandwidth design for lazy learning via particle swarm optimization". In: *Intelligent Automation & Soft Computing* 15.1, pp. 1–11.

Václav Peterka (1981). "Bayesian system identification". In: *Automatica* 17.1, pp. 41–53.

Gianluigi Pillonetto, Alessandro Chiuso, and Giuseppe De Nicolao (2011). "Prediction error identification of linear systems: a nonparametric Gaussian regression approach". In: *Automatica* 47.2, pp. 291–305.

Gianluigi Pillonetto and Giuseppe De Nicolao (2010). "A new kernel-based approach for linear system identification". In: *Automatica* 46.1, pp. 81–93.

Carl E. Rasmussen and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. Cambridge, MA, USA: MIT Press.

Christian P. Robert and George Casella (2004). *Monte Carlo statistical methods*. 2nd ed. New York, NY, USA: Springer.

Jacob Roll, Alexander Nazin, and Lennart Ljung (2005). "Nonlinear system identification via direct weight optimization". In: *Automatica* 41.3, pp. 475–490.

Sam T. Roweis and Zoubin Ghahramani (2000). "An EM algorithm for identification of nonlinear dynamical systems". Unpublished, available at http://mlg.eng.cam.ac.uk/zoubin/papers.html.

Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wågberg, Christian A. Naesseth, Andreas Svensson, and Liang Dai (2015). "Sequential Monte Carlo methods for system identification". In: *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*. Beijing, China, pp. 775–786.

Thomas B. Schön, Adrian Wills, and Brett Ninness (2011). "System identification of nonlinear state-space models". In: *Automatica* 47.1, pp. 39–49.

Maarten Schoukens, Per Mattsson, Torbjörn Wigren, and Jean-Philippe Noël (2015). *Cascaded tanks benchmark combining soft and hard nonlinearities*. Available: homepages.vub.ac.be/ mschouke/benchmark2016.html.

Amar Shah, Andrew Gordon Wilson, and Zoubin Ghahramani (2014). "Student-t processes as alternatives to Gaussian processes". In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Reykjavik, Iceland, pp. 877–885.

Jonas Sjöberg, Qinghua Zhang, Lennart Ljung, Albert Benveniste, Bernard Delyon, Pierre-Yves Glorennec, Håkan Hjalmarsson, and Anatoli Juditsky (1995). "Nonlinear black-box modeling in system identification: a unified overview". In: *Automatica* 31.12, pp. 1691–1724.

Arno Solin and Simo Särkkä (2014). "Hilbert space methods for reduced-rank Gaussian process regression". In: *arXiv:1401.5508*.

Anders Stenman (1999). "Model on demand: Algorithms, analysis and applications". PhD thesis. Sweden: Linköping University.

Andreas Svensson, Thomas B. Schön, Arno Solin, and Simo Särkkä (2015). "Nonlinear state space model identification using a regularized basis function expansion". In: *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. Cancún, Mexico, pp. 493–496.

Andreas Svensson, Arno Solin, Simo Särkkä, and Thomas B. Schön (2016). "Computationally efficient Bayesian learning of Gaussian process state space models". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cádiz, Spain, pp. 213–221.

The MathWorks, Inc. (2015). *Narendra-Li benchmark system: nonlinear grey box modeling of a discrete-time system*. Example file provided by Matlab® R2015b System Identification Toolbox™. Available at http://mathworks.com/help/ident/examples/narendra-li-benchmark-system-nonlinear-grey-box-modeling-of-a-discrete-time-system.html.

Robert Tibshirani (1996). "Regression shrinkage and selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 58.1, pp. 267–288.

Felipe Tobar (2016). Personal communication.

Felipe Tobar, Petar M. Djurić, and Danilo P. Mandic (2015). "Unsupervised state-space modeling using reproducing kernels". In: *IEEE Transactions on Signal Processing* 63.19, pp. 5210–5221.

Jack Umenberger, Johan Wågberg, Ian R. Manchester, and Thomas B. Schön (2015). "On identification via EM with latent disturbances and Lagrangian relaxation". In: *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*. Beijing, China, pp. 69–74.

Yali Wang and David Barber (2014). "Gaussian processes for Bayesian estimation in ordinary differential equations". In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*. Beijing, China, pp. 1485–1493.

Chengtao Wen, Shuning Wang, Xuexiang Jin, and Xiaoyan Ma (2007). "Identification of dynamic systems using piecewise-affine basis function models". In: *Automatica* 43.10, pp. 1824–1831.

Torbjörn Wigren and Johan Schoukens (2013). "Three free data sets for development and benchmarking in nonlinear system identification". In: *Proceedings of the 2013 European Control Conference (ECC)*. Zurich, Switzerland, pp. 2933–2938.

Adrian Wills, Thomas B. Schön, Fredrik Lindsten, and Brett Ninness (2012). "Estimation of linear systems using a Gibbs sampler". In: *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*. Brussels, Belgium, pp. 203–208.

Jun Xu, Xiaolin Huang, and Shuning Wang (2009). "Adaptive hinging hyperplanes and its applications in dynamic system identification". In: *Automatica* 45.10, pp. 2325–2332.

Paper II

Title

Data consistency approach to model validation

Authors

Andreas Svensson, Dave Zachariah, Petre Stoica and Thomas B. Schön

Edited version of

Andreas Svensson, Dave Zachariah, Petre Stoica, and Thomas B. Schön (2018). "Data consistency approach to model validation". Submitted for publication.

Digital identity

https://arxiv.org/abs/1808.05889

Financial support

This research was financially supported by the Swedish Foundation for Strategic Research (SSF), via the project *ASSEMBLE* (contract number: RIT15-0012), and by the Swedish Research Council, via the project *NewLEADS - New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079).

Data Consistency Approach to Model Validation

Abstract

In scientific inference problems, the underlying statistical modeling assumptions have a crucial impact on the end results. There exist, however, only a few automatic means for validating these fundamental modelling assumptions. The contribution in this paper is a general criterion to evaluate the consistency of a set of statistical models with respect to observed data. This is achieved by automatically gauging the models' ability to generate data that is similar to the observed data. Importantly, the criterion follows from the model class itself and is therefore directly applicable to a broad range of inference problems with varying data types. The proposed data consistency criterion is illustrated and evaluated using three synthetic and two real data sets.

Introduction

In many scientific applications, statistical models provide a basis for inferences about real world phenomena. These inferences are typically dependent on the model assumptions being correct. However, there are few automatic means of evaluating these assumptions. In this paper, we address the problem of model validation by developing a method that automatically assesses the consistency of a set of models with respect to the observed data.

Let the observed data set be denoted as

$$\underline{\mathbf{y}} = \{\underline{\mathbf{y}}_1, \underline{\mathbf{y}}_2, \ldots, \underline{\mathbf{y}}_n\}, \tag{1}$$

which consists of n data blocks of equal dimension, referred to as data points. We describe the mechanism that gave rise to the data by a probability density or mass function $p_0(\mathbf{y})$, which is unknown to us. In many applications, the objective of statistical inference is to determine certain properties of the unknown function $p_0(\mathbf{y})$. Statistical methods typically specify a family or class of probability distributions that aim to model $p_0(\mathbf{y})$. We denote this *model class* as

$$\mathcal{P}_\Theta \triangleq \big\{\, p(\mathbf{y} \mid \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta \,\big\},$$

where each model $p(\mathbf{y} \mid \boldsymbol{\theta})$ is indexed by the parameter vector $\boldsymbol{\theta}$. Our aim in this paper is to assess whether the models in $\mathcal{P}_\Theta$ that best approximate $p_0(\mathbf{y})$ are consistent with the observed data $\underline{\mathbf{y}}$ or not. The idea behind the proposed data consistency criterion (see below) is that if the best models in $\mathcal{P}_\Theta$ fail to generate data sets $\tilde{\mathbf{y}}$ that are 'similar' to $\underline{\mathbf{y}}$, then $\mathcal{P}_\Theta$ can hardly be a valid modelling choice for $\underline{\mathbf{y}}$. This notion will be made precise in the subsequent sections.

Past research efforts have mostly focused on comparing model classes, let us say $\mathcal{P}'_\Theta$ and $\mathcal{P}''_\Theta$, using tools such as the Akaike or Bayesian information criteria and Bayes factors (Akaike 1974; Schwarz 1978; Stoica and Moses 2004). These criteria typically assume that some model class is well specified, meaning that one of the classes contains the unknown $p_0(\mathbf{y})$. While the consistency criterion proposed in this paper also can be used for model comparison, our focus here is rather on validation of *one* specified model class $\mathcal{P}_\Theta$.

One established approach to validation is to use a residual-based criterion which assesses whether there is "any information left" in the data after fitting a model. In the restricted context of linear dynamical systems models, such validation criteria are capable of rejecting model classes that are inconsistent with the data, cf. G. Ljung and Box (1978) and Söderström and Stoica (1989, Ch. 11). For models with assumption of independent and identical distribution of the data points, classical tests such as Cramér-von Mises, Anderson-Darling and Kolmogornov-Smirnov tests are applicable (Anderson and Darling 1952; Lehmann 1975). Those tests are, however, constructed only for a single model, not an entire model class. In the context of Bayesian modelling, validation of a model class can also be performed using posterior predictive checks which require the user to specify a discrepancy measure, cf. Box (1980), Gelman et al. (2014), and Rubin (1984).

Our proposed data consistency criterion (Dcc) evaluates the ability of a model to generate data similar to the observed one. In contrast to posterior predictive checks, Dcc is automatic and does not require the user to specify any quantities except the model class $\mathcal{P}_\Theta$ itself. Furthermore, it applies directly to a broad range of model classes with various data types, e.g., linear regression models, count models, hidden Markov models, autoregressive models, etc. In general, $p(\mathbf{y} \mid \boldsymbol{\theta})$ can be factored as $p(\mathbf{y} \mid \boldsymbol{\theta}) = \prod_{i=1}^{n} p(\mathbf{y}_i \mid \mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \boldsymbol{\theta})$. Dcc is applicable whenever it is possible to point-wise evaluate $p(\mathbf{y}_i \mid \mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \boldsymbol{\theta})$ for all i, and simulate new data $\tilde{\mathbf{y}}$ from the model, for any given $\boldsymbol{\theta} \in \Theta$.

As an introductory application of our approach, we consider a modeling problem in seismology. Then, we explain the principles behind Dcc for a single model $p(\mathbf{y} \mid \boldsymbol{\theta}_\star)$ and, subsequently, for an entire model class $\mathcal{P}_\Theta$. Dcc is thereafter applied to the seismological problem as an illustration. We also discuss the implementation of the method as well as illustrate how it can be applied to other model classes, including regression, autoregressive and latent variables models. The source code for all experiments is available online.[1]

Motivating example: Earthquake counts

A standard assumption in earthquake analysis is that earthquakes occur independently as described by a Poisson point process. That is, the number of earthquakes in a certain region during a certain time interval is Poisson distributed. However, it is also well-known that earthquakes tend to be clustered (both in time and space), where each cluster typically has several 'foreshocks' and 'aftershocks' and one larger 'mainshock'. By modeling the earthquakes within a cluster as a branching process, the negative binomial distribution has been suggested for earthquake counts (Kagan 2013). We consider both model classes, $\mathcal{P}_\Theta$ = Poisson distribution

[1] https://github.com/saerdna-se/consistency-criterion

Magnitude	2012	2013	2014	2015	2016	2017
≥8	2	2	1	1	0	1
≥7	16	19	12	19	16	7
≥6	133	142	155	146	146	111
≥5	1680	1596	1729	1558	1696	1560

Figure 1: *A snippet of the global earthquake count data, for different magnitudes and years. Each row (magnitude class) is a different data set* $\underline{\mathbf{y}}$*. We would like to assess the consistency between each of these data sets and the two model classes, the Poisson and negative binomial distributions, respectively.*

and $\mathcal{P}_\Theta$ = negative binomial distribution, which have one and two free parameters, respectively. We will use our proposed method to assess whether these model classes are consistent with the data $\underline{\mathbf{y}}$ in the United States Geological Survey earthquake catalog[2] (partly shown in Table 1; for the full data set, see Fig. 8). We will return to this example after developing the Dcc.

Data consistency check for a single model

We begin by considering a model class consisting of only a single model, i.e., $\mathcal{P}_\Theta = \{p(\mathbf{y} \mid \boldsymbol{\theta}_\star)\}$ where $\boldsymbol{\theta}_\star$ is a specified parameter. Let $\tilde{\mathbf{y}} \sim p(\mathbf{y} \mid \boldsymbol{\theta}_\star)$ denote a sample generated from the model and $\mathbb{P}_{\tilde{\mathbf{y}}|\boldsymbol{\theta}_\star}(\cdot)$ the probability of an event under the same model.

Initially, consider the simpler case of models in which the data points $i = 1, \ldots, n$ in (1) are assumed to be independent. Let $\tilde{z}_i \triangleq \ln p(\tilde{\mathbf{y}}_i \mid \boldsymbol{\theta}_\star)$ denote the log-likelihood for the ith generated data point, and let its mean be denoted as $\mathbb{E}[\tilde{z}_i]$. For the ith observed data point, let $\underline{z}_i \triangleq \ln p(\underline{\mathbf{y}}_i \mid \boldsymbol{\theta}_\star)$. The observed and generated log-likelihoods, $\underline{z}_i$ and $\tilde{z}_i$ form the basis of our criterion. Intuitively, if the deviation of $\underline{z}_i$ from $\mathbb{E}[\tilde{z}_i]$ is much larger or much smaller than the deviation of $\tilde{z}_i$ from $\mathbb{E}[\tilde{z}_i]$, we consider the observed data $\underline{\mathbf{y}}$ to be *atypical* for the given model $p(\mathbf{y} \mid \boldsymbol{\theta}_\star)$. More formally, we define the following statistic

$$T(\underline{\mathbf{y}}; \boldsymbol{\theta}_\star) = \frac{1}{n} \sum_{i=1}^{n} \frac{\left(\underline{z}_i - \mathbb{E}[\tilde{z}_i]\right)^2}{\operatorname{var}[\tilde{z}_i]}, \tag{2}$$

where $\operatorname{var}[\tilde{z}_i]$ is variance of $\tilde{z}_i$. Similarly, we define the statistic $T(\tilde{\mathbf{y}}; \boldsymbol{\theta}_\star)$ for generated data by replacing $\underline{\mathbf{y}}$ with $\tilde{\mathbf{y}}$. Let us now define the random event of generating a larger statistic than the observed one:

$$S(\tilde{\mathbf{y}}, \underline{\mathbf{y}}) \ : T(\tilde{\mathbf{y}}; \boldsymbol{\theta}_\star) > T(\underline{\mathbf{y}}; \boldsymbol{\theta}_\star). \tag{3}$$

When the probability of this event $\mathbb{P}_{\tilde{\mathbf{y}}|\boldsymbol{\theta}_\star}\left(S(\tilde{\mathbf{y}}, \underline{\mathbf{y}})\right)$ is close to 0, it is highly improbable that $\underline{\mathbf{y}}$ could have been generated by $p(\mathbf{y} \mid \boldsymbol{\theta}_\star)$ and we deem the model to be inconsistent with the observed data. See Fig. 2a for an illustration. This type of inconsistency is due to under-dispersion of the generated log likelihoods, compared to the observed

[2] https://earthquake.usgs.gov/earthquakes/search/

ones. The probability of the complementary event $\mathbb{P}_{\tilde{\mathbf{y}}|\boldsymbol{\theta}_\star}\left(S^c(\tilde{\mathbf{y}}, \underline{\mathbf{y}})\right) = 1 - \mathbb{P}_{\tilde{\mathbf{y}}|\boldsymbol{\theta}_\star}\left(S(\tilde{\mathbf{y}}, \underline{\mathbf{y}})\right)$ indicates inconsistency as well, see Fig. 2b. Also in this case it is improbable that $\underline{\mathbf{y}}$ could have been generated by $p(\mathbf{y} \mid \boldsymbol{\theta}_\star)$. By contrast, if both aforementioned probabilities are significantly different from 0, we do not reject the model as inconsistent, see Fig. 2c.

The criterion above is readily generalized to models in which the data points in (1) are dependent, by extending the definition of z_i to

$$z_i \triangleq \ln p(\mathbf{y}_i \mid \mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \boldsymbol{\theta}_\star). \tag{4}$$

As above, the same symbols $\mathbb{E}[\tilde{z}_i]$ and $\mathrm{var}[\tilde{z}_i]$ are used to define the mean and variance of $\tilde{z}_i$.

If the model were a match of the unknown data-generating distribution, i.e., $p(\mathbf{y} \mid \boldsymbol{\theta}_\star) = p_0(\mathbf{y})$, then the quantity $\mathbb{P}_{\tilde{\mathbf{y}}|\boldsymbol{\theta}_\star}\left(S(\tilde{\mathbf{y}}, \underline{\mathbf{y}})\right)$ would be uniformly distributed between 0 and 1 with respect to the observed data $\underline{\mathbf{y}}$, see the proof below. In this situation, the probability of *falsely* rejecting the model due to the generated log likelihoods being under-dispersed would be

$$\mathrm{PFA}_u(\boldsymbol{\theta}_\star) \triangleq \mathbb{P}_{\tilde{\mathbf{y}}|\boldsymbol{\theta}_\star}\left(S(\tilde{\mathbf{y}}, \underline{\mathbf{y}})\right). \tag{5}$$

Thus, when $p(\mathbf{y} \mid \boldsymbol{\theta}) = p_0(\mathbf{y})$, the probability of PFA_u to be less than ρ, is equal to ρ. Symmetrically, the false alarm probability due to over-dispersion is $\mathrm{PFA}_o(\boldsymbol{\theta}_\star) = 1 - \mathrm{PFA}_u(\boldsymbol{\theta}_\star)$. If neither $\mathrm{PFA}_u(\boldsymbol{\theta}_\star)$ nor $\mathrm{PFA}_o(\boldsymbol{\theta}_\star)$ are small, we cannot reject the model $p(\mathbf{y} \mid \boldsymbol{\theta}_\star)$ on the ground that the observed data $\underline{\mathbf{y}}$ is atypical, as discussed above.

This criterion, that neither $\mathrm{PFA}_u(\boldsymbol{\theta}_\star)$ nor $\mathrm{PFA}_o(\boldsymbol{\theta}_\star)$ is close to 0, follows automatically from the specified model class and does not require user choices specific to the application scenario. The false alarm probabilities PFA above can be approximated numerically using Monte Carlo methods, as we will detail later.

Proof of uniform distribution of PFA

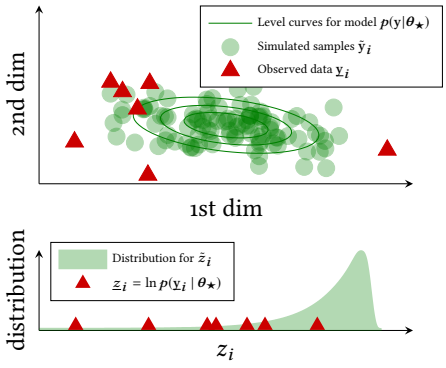
Let $\xi \triangleq T(\tilde{\mathbf{y}}; \boldsymbol{\theta}_\star)$, whose distribution is characterized by a cumulative density function denoted $F_\xi(x)$. Further, let $\underline{\xi} \triangleq T(\underline{\mathbf{y}}; \boldsymbol{\theta}_\star)$, which allows us to write

$$\mathbb{P}_{\tilde{\mathbf{y}}|\boldsymbol{\theta}_\star}\left(S(\tilde{\mathbf{y}}, \underline{\mathbf{y}})\right) = \mathbb{P}_{\tilde{\mathbf{y}}|\boldsymbol{\theta}_\star}\left(\xi > \underline{\xi}\right) = F_\xi(\underline{\xi}). \tag{6}$$

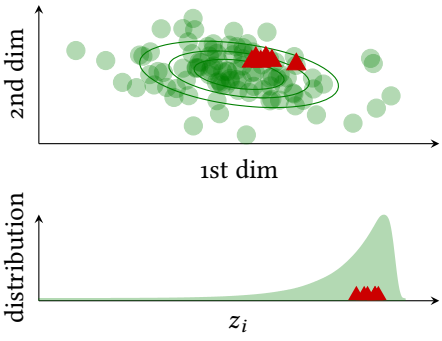
Now, if $\underline{\mathbf{y}} \sim p(\mathbf{y} \mid \boldsymbol{\theta}_\star)$, then also the distribution of $\underline{\xi}$ is characterized by F_ξ, implying that $F_\xi(\underline{\xi}) \sim \mathcal{U}[0, 1]$ according to the probability integral transform (Casella and Berger 2002, Thm. 2.1.10).

Data consistency check for the best models in a class

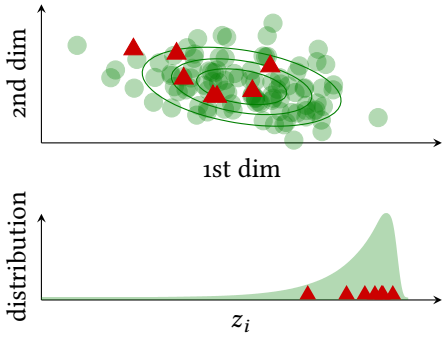
In most applications, $\boldsymbol{\theta}_\star$ is not given. Instead $\mathcal{P}_\Theta$ may consist of a large number of models, possibly an uncountable number when $\boldsymbol{\theta} \in \Theta$ is continuous. In such a case, we will aim to evaluate the false alarm probabilities $\mathrm{PFA}_u(\boldsymbol{\theta})$ and $\mathrm{PFA}_o(\boldsymbol{\theta})$ with respect to the models $p(\mathbf{y} \mid \boldsymbol{\theta})$ that best approximate $p_0(\mathbf{y})$. A natural measure for quantifying



(a) *The observed data points appear as atypical with respect to the model, since they fall into regions of low probability. We obtain* $T(\underline{\mathrm{y}}; \boldsymbol{\theta}_\star) = 14$*, and the probability of generating a higher statistic is* $\mathbb{P}_{\tilde{\mathrm{y}}|\boldsymbol{\theta}_\star}\left(S(\tilde{\mathrm{y}}, \underline{\mathrm{y}})\right) = 0.00$*. Thus, the dispersion of* $\tilde{z}_i$ *is significantly lower than that of* $\underline{z}_i$ *and we reject the model as inconsistent with the observed data.*



(b) *The observed data points appear as atypical with respect to the model, since they are concentrated asymmetrically. We obtain* $T(\underline{\mathrm{y}}; \boldsymbol{\theta}_\star) = 0.24$*, and the probability of generating a lower statistic is* $\mathbb{P}_{\tilde{\mathrm{y}}|\boldsymbol{\theta}_\star}\left(S^c(\tilde{\mathrm{y}}, \underline{\mathrm{y}})\right) = 0.04$*. Thus, the dispersion of* $\tilde{z}_i$ *is significantly higher than that of* $\underline{z}_i$ *and we reject the model as inconsistent with the observed data.*



(c) *The observed data points appear to be typical samples from the model. We obtain* $T(\underline{\mathrm{y}}; \boldsymbol{\theta}_\star) = 0.51$*, and the probability of generating a lower statistic is* $\mathbb{P}_{\tilde{\mathrm{y}}|\boldsymbol{\theta}_\star}\left(S^c(\tilde{\mathrm{y}}, \underline{\mathrm{y}})\right) = 0.39$*. We do therefore not reject the model as inconsistent.*

Figure 2: *Consider a data set* $\underline{\mathrm{y}} = \{\underline{\mathrm{y}}_i\}$ *containing* $n = 7$ *two-dimensional data points (red triangles, upper panels) and assume a Gaussian i.i.d. model* $p(\mathrm{y} \mid \boldsymbol{\theta}_\star) = \prod_i p(\mathrm{y}_i \mid \boldsymbol{\theta}_\star)$ *(green level curves). The log-likelihoods* $\underline{z}_i = \ln p(\underline{\mathrm{y}}_i \mid \boldsymbol{\theta}_\star)$ *for each data point are shown as red triangles in the lower panels. The generated log-likelihoods* $\tilde{z}_i$ *follow the distribution illustrated in green in the same panels. When the deviation of* $\underline{z}_i$ *from* $\mathbb{E}[\tilde{z}_i]$ *is significantly different from that of* $\tilde{z}_i$*, it is unlikely that the model could have generated the observed sample. The deviation is quantified by the statistic* $T(\underline{\mathrm{y}}; \boldsymbol{\theta}_\star)$*. The figure illustrates three cases: two cases of inconsistency (a, b) and a balanced case (c).*

the accuracy of the approximation is the Kullback-Leibler divergence (Kullback 1959). The best models are then defined as those that minimize the divergence:

$$\boldsymbol{\theta}_\star \in \arg\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_0 \left[\ln p_0(\mathbf{y}) - \ln p(\mathbf{y}|\boldsymbol{\theta})\right]}_{\text{model divergence}}, \tag{7}$$

where the expectation $\mathbb{E}_0$ is with respect to $\mathbf{y} \sim p_0(\mathbf{y})$. In particular, if $\boldsymbol{\theta}_\star$ exists such that the model divergence attains the minimum value 0, it follows that $p(\mathbf{y} \mid \boldsymbol{\theta}_\star) = p_0(\mathbf{y})$.

Since $p_0(\mathbf{y})$ is unknown, (7) cannot be used to identify the best models. Consequently we resort to an alternative approach. We assign weights to each model in $\mathcal{P}_\Theta$ so as to average the false alarm probability $\text{PFA}_u(\boldsymbol{\theta})$ across those models that are likely to be the best approximations of $p_0(\mathbf{y})$. The averaged false alarm probability due to under-dispersion is given by

$$\text{PFA}_u^\star = \int_\Theta \text{PFA}_u(\boldsymbol{\theta}) w(\boldsymbol{\theta} \mid \underline{\mathbf{y}})\, d\boldsymbol{\theta} \tag{8}$$

where the weights $w(\boldsymbol{\theta} \mid \underline{\mathbf{y}}) \geq 0$ are high for models in the neighborhood of $\boldsymbol{\theta}_\star$ and integrate to unity. By defining

$$w(\boldsymbol{\theta} \mid \underline{\mathbf{y}}) \triangleq \frac{w_0(\boldsymbol{\theta}) p(\underline{\mathbf{y}} \mid \boldsymbol{\theta})}{\int_\Theta w_0(\boldsymbol{\theta}) p(\underline{\mathbf{y}} \mid \boldsymbol{\theta}) d\boldsymbol{\theta}}, \tag{9}$$

the weights reflect the uncertainty about the location of $\boldsymbol{\theta}_\star$ in the parameter space Θ (Bissiri and Walker 2012; Casella and Berger 2002). The default choice of the initial weights is $w_0(\boldsymbol{\theta}) \equiv 1$. In certain applications, however, we may have prior information about the location of $\boldsymbol{\theta}_\star$ in Θ. Then the initial weights $w_0(\boldsymbol{\theta})$ can be chosen to describe these prior beliefs. Under certain regularity conditions, the weights in (9) concentrate at $\boldsymbol{\theta}_\star$ as $n \to \infty$ if $\boldsymbol{\theta}_\star$ is unique, cf. L. Ljung and Caines (1979) and Berk (1966).

As before, we still have $\text{PFA}_o^\star = 1 - \text{PFA}_u^\star$, and we denote our final criterion

(10)

In summary, the proposed data consistency criterion (Dcc) for the model class $\mathcal{P}_\Theta$ is the minimum of $\text{PFA}_u^\star$ and $\text{PFA}_o^\star$ ((8)). A value close to 0 indicates that the observed data $\underline{\mathbf{y}}$ is atypical for the best models in $\mathcal{P}_\Theta$, and thereby we consider the model class $\mathcal{P}_\Theta$ to be inconsistent with $\underline{\mathbf{y}}$.

Implementation

The averaged $\text{PFA}^\star$ is available in closed-form only in very few special cases, since there are non-trivial integrals in (5), (8), and (9). However, the integrals can be efficiently approximated by Monte Carlo integration techniques, provided that data $\tilde{\mathbf{y}}$ can be generated from $p(\mathbf{y} \mid \boldsymbol{\theta})$ and that $p(\mathbf{y}_i \mid \mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \boldsymbol{\theta})$ can be evaluated pointwise. We outline such a generic Monte Carlo-based implementation[3] in Algorithm 1, where N parameters are first drawn using $w(\boldsymbol{\theta} \mid \mathbf{y})$, and then (for each such draw)

[3]Where $\mathbb{I}[\,\cdot\,]$ denotes the indicator function.

Algorithm 1: Monte Carlo implementation of DCC (8)

1 Construct $w(\boldsymbol{\theta} \mid \underline{\mathbf{y}})$
2 Draw N samples $\boldsymbol{\theta}^{(j)} \sim w(\boldsymbol{\theta} \mid \underline{\mathbf{y}})$, $j = 1, \ldots, N$
3 **for** $j = 1, \ldots, N$ **do**
4 Simulate M' data sets $\tilde{\mathbf{y}}'^{(k)} \sim p(\mathbf{y} \mid \boldsymbol{\theta}^{(j)})$, $k = 1, \ldots, M'$
5 Compute $\tilde{z}_i'^{(k)}$ as (4) for all generated data points $\tilde{\mathbf{y}}_i'^{(k)}$
6 Compute sample mean $\hat{m}_i$ and variance $\hat{v}_i$ of $\tilde{z}_i'$, $i = 1, \ldots, n$
7 Simulate M data sets $\tilde{\mathbf{y}}^{(\ell)} \sim p(\mathbf{y} \mid \boldsymbol{\theta}^{(j)})$, $\ell = 1, \ldots, M$
8 Compute $\tilde{z}_i^{(\ell)}$ as (4) for all generated data points $\tilde{\mathbf{y}}_i^{(\ell)}$
9 Compute $\underline{z}_i$ as (4) for all observed data points $\underline{\mathbf{y}}_i$
10 Compute $\underline{T}^{(j)} = \frac{1}{n} \sum_{i=1}^{n} \frac{\left(\underline{z}_i - \hat{m}_i\right)^2}{\hat{v}_i}$
11 Compute $\tilde{T}^{(j,\ell)} = \frac{1}{n} \sum_{i=1}^{n} \frac{\left(\tilde{z}_i^{(\ell)} - \hat{m}_i\right)^2}{\hat{v}_i}$
12 Set $\hat{\text{PFA}}_u^{(i)} = \frac{1}{M} \sum_{\ell=1}^{M} \mathbb{I}\left[\tilde{T}^{(j,\ell)} > \underline{T}^{(j)}\right]$
13 **end**
14 Set $\hat{\text{PFA}}_u^{\star} = \frac{1}{N} \sum_{i=1}^{N} \hat{\text{PFA}}_u^{(i)}$ and $\hat{\text{PFA}}^{\star} = \min\left(\hat{\text{PFA}}_u^{\star}, 1 - \hat{\text{PFA}}_u^{\star}\right)$

$M + M'$ samples of $\tilde{\mathbf{y}}$ are generated from $p(\mathbf{y} \mid \boldsymbol{\theta})$, giving a computational complexity on the order of $N(M + M')$.

The operations needed to execute Algorithm 1 are common in most statistical software packages. The weights $w(\boldsymbol{\theta} \mid \underline{\mathbf{y}})$ can be computed (at least approximately) by methods that estimate or learn $\boldsymbol{\theta}$. Numerical evaluation of the (incremental) likelihoods $p(\mathbf{y}_i \mid \mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \boldsymbol{\theta})$, which in Algorithm 1 has to be performed both for generated data $\tilde{\mathbf{y}}$ (line 5) and observed data $\underline{\mathbf{y}}$ (line 6), is possible for many statistical models. If n is large compared to the number of parameters, it can be justified to use the following weights

$$w(\boldsymbol{\theta} \mid \underline{\mathbf{y}}) = \begin{cases} 1, & \boldsymbol{\theta} = \hat{\boldsymbol{\theta}} \\ 0, & \boldsymbol{\theta} \neq \hat{\boldsymbol{\theta}} \end{cases},$$

where $\hat{\boldsymbol{\theta}}$ is the maximum likelihood/maximum a posteriori point estimate (in this case $N = 1$).

Examples

Earthquake counts (cont'd)

To assess whether the Poisson or the negative binomial distribution is best suited for the data (partly) presented in Fig. 1, we are now ready to apply the DCC. We use the Monte Carlo approach in Algorithm 1 with $N = 200$ and $M = M' = 200$, and obtain the results in Table 3. From this table we draw the conclusion that the Poisson distribution and the negative binomial distribution are both consistent with the data for earthquakes with magnitude ≥ 7. However, only the negative binomial distribution is consistent with the data for smaller magnitudes. This result supports

Magnitude	Poisson distribution	Negative binomial distribution
≥8	$\hat{\text{PFA}}^\star = 0.40$	$\hat{\text{PFA}}^\star = 0.39$
≥7	$\hat{\text{PFA}}^\star = 0.29$	$\hat{\text{PFA}}^\star = 0.38$
≥6	$\hat{\text{PFA}}^\star = 0.00$	$\hat{\text{PFA}}^\star = 0.30$
≥5	$\hat{\text{PFA}}^\star = 0.00$	$\hat{\text{PFA}}^\star = 0.13$

Figure 3: *Assessment of the two earthquake count models (Poisson and negative binomial) for earthquakes of different magnitudes. The low average false alarm probabilities $\hat{\text{PFA}}^\star$ for the Poisson distribution suggests that this model class is not consistent with the observed data for magnitudes* ≤ 6.

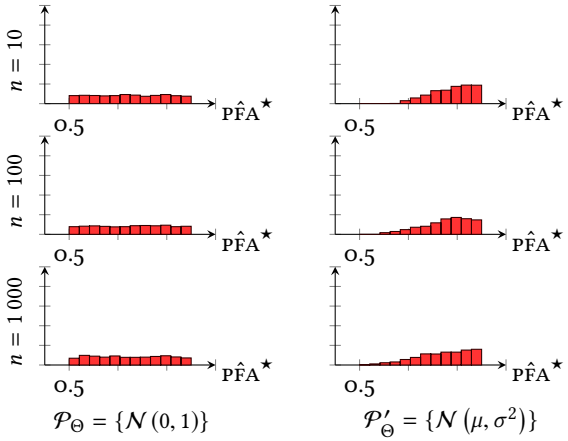
the qualitative reasoning in the literature (Kagan 2013): the Poisson distribution does not take the clustering effects into account, but since each cluster typically does not contain more than one major earthquake, the number of really big earthquakes can still follow the Poisson distribution.

Synthetic data: Gaussian models

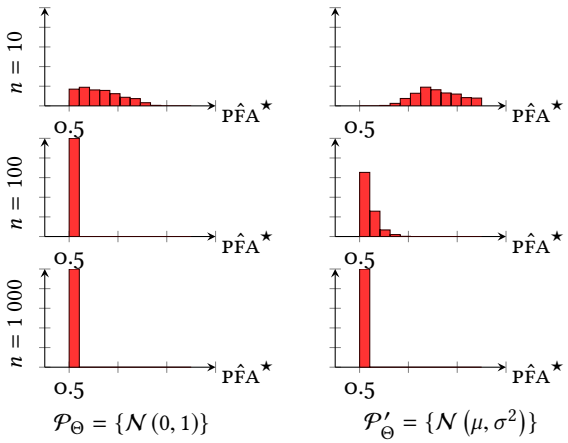
To further illustrate the behavior of PFA*, we conduct a simulation study with two toy model classes $\mathcal{P}_\Theta = \{\mathcal{N}(0,1)\}$ and $\mathcal{P}'_\Theta = \{\mathcal{N}(\mu, \sigma^2) : \sigma^2 > 0\}$ with $\boldsymbol{\theta} = \{\mu, \sigma^2\}$. Note that $\mathcal{P}_\Theta$ contains only a single model. We consider data sets $\underline{\mathbf{y}} \sim p_0(\mathbf{y})$ of different size n and consider two cases: a standard Gaussian distribution $p_0 = \mathcal{N}(0,1)$ and a standard uniform distribution $p_0 = \mathcal{U}[0,1]$, respectively. We use $N = 50$ and $M = M' = 100$, and evaluate Dcc in 1 000 experiments. The results are summarized as histograms in Fig. 4.

In the case when data comes from $p_0 = \mathcal{N}(0,1)$ (Fig. 4a), both $\mathcal{P}_\Theta$ and $\mathcal{P}'_\Theta$ contain p_0. However, the only model in $\mathcal{P}_\Theta$ is p_0. Thus the averaged PFA* is uniformly distributed across experiments (cf. the proof of uniform distribution for PFA). By contrast, the weights $w(\boldsymbol{\theta} \mid \underline{\mathbf{y}})$ for $\mathcal{P}'_\Theta$ concentrate at the best model p_0 only as $n \to \infty$. Thus the averaged PFA* approaches a uniform distribution only asymptotically. Neither model class is falsely rejected more frequently than the PFA indicates.

In the case when data comes from $p_0 = \mathcal{U}[0,1]$ (Fig. 4b), neither $\mathcal{P}_\Theta$ nor $\mathcal{P}'_\Theta$ contains p_0. Unlike $\mathcal{P}_\Theta$, however, the best model in $\mathcal{P}'_\Theta$ matches the mean and variance of p_0. The inconsistency of the best model are discernible only for data points generated from the distribution tails. Consequently, it takes more samples n to reject the larger model class $\mathcal{P}'_\Theta$ than the $\mathcal{P}_\Theta$. As n increases, both model classes are clearly rejected.



(a) *Data generating process $p_0 = \mathcal{N}(0,1)$. Both model classes contain p_0. As n increases, PFA* approaches a uniform distribution also for $\mathcal{P}'_\Theta$.*



(b) *Data generating process $p_0 = \mathcal{U}[0,1]$. Neither model class contain p_0. As n increases, PFA* concentrates at 0 and both model classes are rejected.*

Figure 4: *Histograms of DCC obtained from 1000 experiments, when p_0 is the standard Gaussian distribution (a) and the standard uniform distribution (b). Two different model classes are considered in the left and right columns, respectively. DCC is able to correctly identify the cases where the model class is inconsistent with the data.*

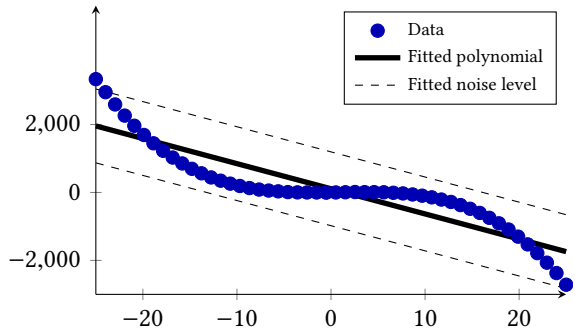


Figure 5: *Data points from p_0 (3rd order polynomial, blue dots) and model class $\mathcal{P}_\Theta$ = {1st order polynomial + Gaussian noise} where $\boldsymbol{\theta}$ contains the polynomial coefficients and noise variance. A fitted model is shown with $\boldsymbol{\theta}$ estimated using the maximum likelihood method (black solid line: estimated polynomial, dashed lines: 2 estimated noise standard deviations). While the estimated noise variance produces a good fit in terms of likelihoods, this model class should ideally be rejected. Using Dcc for $\mathcal{P}_\Theta$, we obtain $\text{PFA}^\star = 0.01$ and can reject $\mathcal{P}_\Theta$. Similarly, for a model class $\mathcal{P}'_\Theta$ containing 2nd order polynomials we obtain $\text{PFA}^\star = 0.01$. By contrast, for 3rd order polynomials, $\mathcal{P}''_\Theta$, we have $\text{PFA}^\star = 0.37$ and this class is correctly found not to be inconsistent with the data.*

Synthetic data: Regression models

We illustrate the capability of Dcc to reject model classes with an inappropriate model order, using the example of polynomial regression. If the observed data is well described by a high-order polynomial, the best model in $\mathcal{P}_\Theta$, which contains models of lower-order polynomials plus noise, will yield a good fit in the likelihood sense because the noise variance is scaled to match the residuals. Such an example is shown in Fig. 5, where $\mathcal{P}_\Theta$ contains 1st order polynomial models with independent Gaussian noise. When assessing $\mathcal{P}_\Theta$ using Dcc, we obtain $\hat{\text{PFA}}^\star = 0.01$ (with $N = M = M' = 100$), which clearly indicates an inconsistency. On the other hand, for the model class containing 3rd order polynomials we obtain $\hat{\text{PFA}}^\star = 0.37$, which (correctly) indicates no inconsistency.

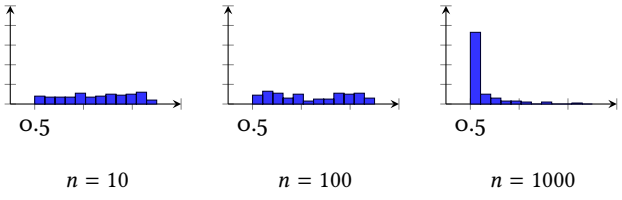
Synthetic data: Time-series models

For the case of time-series models driven by white noise, whiteness tests such as the Ljung-Box test (G. Ljung and Box 1978; see also Stoica 1977) are common validation techniques. The Ljung-Box test constructs a p-value from the fact that the statistic $n(n+2)\sum_{k=1}^{h} \frac{\hat{r}_k^2}{n-k}$ will follow a χ^2_{h-d} distribution if the model is correct, with $\hat{r}_k$ being the lag k sample correlation of the prediction residuals, and d the dimension of $\boldsymbol{\theta}$. We set the upper lag limit h to $\lfloor \log n \rceil$.

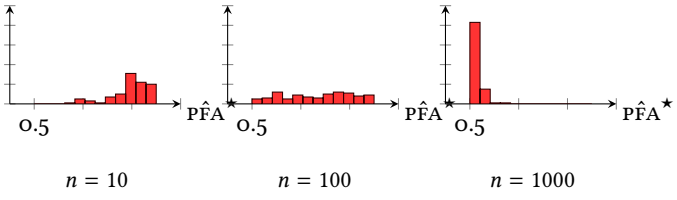
We conduct a simulation study with data from the saturated first order autoregressive model $y_i = \max(\, 0.7y_{i-1} + e_i,\ -0.3\,)$, where $e_i \sim \mathcal{N}(0,1)$. We assume a misspecified model class which consists of first-order linear autoregressive models $\mathcal{P}_\Theta = \{p(\mathrm{y}|\boldsymbol{\theta}) : y_i = ay_{i-1} + e_i,\ e_i \sim \mathcal{N}\left(0, \sigma^2\right) : \sigma^2 > 0\}$, with unknown parameters

$\theta = \{a, \sigma^2\}$. We consider cases where $\underline{\mathrm{y}}$ contains different amount of data samples n, and use $N = 200$ and $M = M' = 200$ in Algorithm 1.

The results are shown in Fig. 6. As the amount of data n grows, both methods correctly reject the misspecified model, with the proposed Dcc needing slightly less data to do so than the much more specialized Ljung-Box method.



(a) *Results for the Ljung-Box method.*



(b) *Results for the Dcc.*

Figure 6: *Histograms for the Ljung-Box p-value (top) and $\text{PFA}^\star$ (bottom). For both methods, small values indicate an inconsistency. The data $\underline{\mathrm{y}}$ is generated by a saturated (non-linear) autoregressive model, but the assumed model class $\mathcal{P}_\Theta$ consists of linear autoregressive models. In this example, the amount of data required by the Ljung-Box method to detect the mismatch is slightly larger than for the proposed criterion.*

Latent-variable models: Evolution of a kangaroo population

Certain models are too complex to be described using closed-form expressions. Instead these models are often parameterized using latent variables $\boldsymbol{\eta}$, with a prior distribution $p(\boldsymbol{\eta} \mid \boldsymbol{\theta})$. This includes, e.g., hidden Markov or state-space models, mixed-effect models, latent topic models, etc. Then the data distribution can be written as the integral

$$p(\mathrm{y} \mid \boldsymbol{\theta}) = \int p(\mathrm{y} \mid \boldsymbol{\eta}, \boldsymbol{\theta}) p(\boldsymbol{\eta} \mid \boldsymbol{\theta}) d\boldsymbol{\eta}. \tag{11}$$

If the (incremental) likelihoods $p(\mathrm{y}_i \mid \mathrm{y}_1, \ldots, \mathrm{y}_{i-1}, \boldsymbol{\theta})$ can be evaluated or well approximated, Dcc can be computed also for this model class, as we will illustrate in the following.

We consider the dynamics of a population of red kangaroos (*Macropus rufus*) in New South Wales, Australia. The data $\underline{\mathrm{y}}$, from Bayliss (1987, Appendix 8.2; available in Fig. 8), is a time series of $n = 41$ bi-variate observations from double transect counts at irregular time intervals between 1973 and 1984. In Knape and Valpine (2012) the authors propose three different models for this data. These models are then compared in a pairwise fashion using the Bayes factor. The comparison has recently been

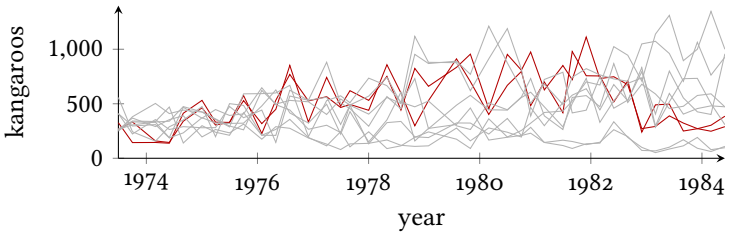


Figure 7: *The kangaroo time series data ($\underline{\mathbf{y}}$, red), together with a few time series (grey) that are generated from the model in* (12) *(parameters sampled from $w(\boldsymbol{\theta} \mid \underline{\mathbf{y}})$; cf. Step 7 of Algorithm 1). Dcc indicates no inconsistency between the data and the model ($\hat{\text{pfa}}^{\star} = 0.28$), which resonates with the intuition since the generated data behaves 'similarly' to the observed data.*

repeated in Shao et al. (2017) using the Hyvärinen score. Both Knape and Valpine (2012) and Shao et al. (2017) conclude that among the three different models, the preferred model is the following continuous-time stochastic differential equation

$$x_1 = \mathcal{LN}(0, 5), \tag{12a}$$

$$\frac{dx_t}{x_t} = \frac{\sigma^2}{2}dt + \sigma dW_t, \tag{12b}$$

$$y_{1,t}, y_{2,t} \mid x_t \overset{\text{i.i.d.}}{\sim} \mathcal{NB}\left(\frac{\tau - 1}{\tau}, \frac{\tau x_t}{\tau x_t + 1}\right), \tag{12c}$$

where $\mathcal{LN}$ is the log-normal distribution, W_t is the standard Brownian motion, $\mathcal{NB}$ is the negative binomial distribution, and $\boldsymbol{\theta} = \{\sigma, \tau\}$ are unknown parameters. The latent variables are $\boldsymbol{\eta} = \{x_t\}_{t \geq 1}$. Note that this model describes a bi-variate time-series $\{\mathbf{y}_i\}$.

While Knape and Valpine (2012) and Shao et al. (2017) favor the model given in (12) over other alternatives, we consider the different question whether the model in (12) is consistent with the observed data $\underline{\mathbf{y}}$ or not. We first solve (12) analytically to obtain a discrete-time nonlinear state-space model, and use the particle marginal Metropolis-Hastings method (Andrieu et al. 2010) to sample the unknown parameters. A standard particle filter is used to approximate $p(\mathbf{y}_i \mid \mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \boldsymbol{\theta})$. We use $N = 1000$ and $M' = M = 200$ and obtain $\hat{\text{pfa}}^{\star} = 0.28$. Thus the model in (12) is deemed to be consistent with the observed kangaroo population data (also see Fig. 7 for intuitive support of this conclusion).

Discussion

We have proposed a data consistency criterion (Dcc) to assess the consistency of a model class $\mathcal{P}_\Theta$ with respect to the observed data $\underline{\mathbf{y}}$. By comparing the observed (incremental) likelihoods $p(\underline{\mathbf{y}}_i \mid \underline{\mathbf{y}}_1, \ldots, \underline{\mathbf{y}}_{i-1}, \boldsymbol{\theta})$ to the ones of generated data $\tilde{\mathbf{y}}$, Dcc rejects a model class for which $\underline{\mathbf{y}}$ is atypical for the best models in the class. The criterion follows automatically from the specification of $\mathcal{P}_\Theta$ and does not require additional application-specific choices. It yields an (approximate) false alarm probability $\text{pfa}^{\star}$ of erroneously declaring the best models to be inconsistent. When $\text{pfa}^{\star}$ falls below some set threshold, there is a sound ground for ruling out the model class $\mathcal{P}_\Theta$.

By exploiting properties of the Fisher information matrix of $\mathcal{P}_\Theta$, it is possible to construct misspecification tests, cf. White (1982). That is, decide whether $p(\mathbf{y} \mid \boldsymbol{\theta}_\star) = p_0(\mathbf{y})$ is true or not. Since all practical models are incomplete in some respect, such tests may not always be relevant. Instead, what matters in many applications is whether the model is accurate or not, and the proposed Dcc provides a practically useful criterion in this respect.

Using a Bayesian interpretation of (8), Dcc can be understood as a certain posterior predictive check (Box 1980; Gelman et al. 2014; Rubin 1984). Posterior predictive checks are, however, not available for plug-and-play since they require the user to specify a 'discrepancy variable', in contrast to the fully automatic Dcc. The Dcc also readily admits a frequentist interpretation in terms of false alarm probability as a sampling property of $p_0(\mathbf{y})$.

The computational cost of the criterion increases indeed with the dimension of $\boldsymbol{\theta}$ and n. The operations required are, however, available for many models and well devloped in most statistical software packages. Dcc could therefore be used as a routine check in existing statistical modeling methods, in order to better guide the end user to well-grounded scientific conclusions.

References

Hirotugu Akaike (1974). "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6, pp. 716–723.

Theodore W. Anderson and Donald A. Darling (1952). "Asymptotic Theory of Certain "Goodness of Fit" Criteria Based on Stochastic Processes". In: *The Annals of Mathematical Statistics* 23.2, pp. 193–212.

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

Peter Bayliss (1987). "Kangaroos - Their Ecology and Management in the Sheep Rangelands of Australia". In: ed. by Graeme Caughley, Neil Shepher, and Jeff Short. Cambridge University Press. Chap. Kangaroo dynamics.

Robert H. Berk (1966). "Limiting behavior of posterior distributions when the model is incorrect". In: *The Annals of Mathematical Statistics* 37.1, pp. 51–58.

Pier Giovanni Bissiri and Stephen G. Walker (2012). "Converting information into probability measures with the Kullback–Leibler divergence". In: *Annals of the Institute of Statistical Mathematics* 64.6, pp. 1139–1160.

George E. P. Box (1980). "Sampling and Bayes' inference in scientific modelling and robustness". In: *Journal of the Royal Statistical Society, Series A* 143.4, pp. 383–430.

George Casella and Roger L. Berger (2002). *Statistical inference*. 2nd ed. Pacific Grove, CA, USA: Duxbury.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin (2014). *Bayesian data analysis*. 3rd ed. Boca Raton, FL, USA: Chapman & Hall/ CRC Press.

Yan Y. Kagan (2013). *Earthquakes: models, statistics, testable forecasts*. American Geophysical Union.

Jonas Knape and Perry de Valpine (2012). "Fitting complex population models by combining particle filters with Markov chain Monte Carlo". In: *Ecology* 93.2, pp. 256–263.

Solomon Kullback (1959). *Information Theory and Statistics*. Dover Publications.

Erich L. Lehmann (1975). *Nonparametrics: statistical methods based on ranks*. San Francisco, CA, USA: McGraw-Hill.

Greta Ljung and George E. P. Box (1978). "On a measure of lack of fit in time series models". In: *Biometrika* 65.2, pp. 297–303.

Lennart Ljung and Peter E. Caines (1979). "Asymptotic normality of prediction error estimators for approximate models". In: *Stochastics* 3, pp. 29–46.

Donald B. Rubin (1984). "Bayesianly justifiable and relevant frequency calculations for the applied statistician". In: *The Annals of Statistics* 12.4, pp. 1151–1172.

Gideon Schwarz (1978). "Estimating the dimension of a model". In: *Annals of Statistics* 6.2, pp. 461–464.

Stephane Shao, Pierre E. Jacob, Jie Ding, and Vahid Tarokh (2017). "Bayesian model comparison with the Hyvärinen score: computation and consistency". In: *arXiv:1711.00136*.

Torsten Söderström and Petre Stoica (1989). *System identification*. Hemel Hempstead, UK: Prentice-Hall, Inc.

Petre Stoica (1977). "A test for whiteness". In: *IEEE Transactions on Automatic Control* 22.6, pp. 992–993.

Petre Stoica and Randolph Moses (2004). *Spectral Analysis of Signals*. Upper Saddle River, NJ, USA: Prentice Hall.

Halbert White (1982). "Maximum likelihood estimation of misspecified models". In: *Econometrica* 50.1, pp. 1–25.

Magnitude	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998
≥8	0	0	0	0	0	2	1	0	0	1	0	0	0	0	2	2	1	0	1
≥7	6	10	7	14	14	15	11	13	11	8	18	17	13	12	13	20	15	16	12
≥6	96	88	90	139	141	162	140	173	126	139	154	137	179	148	159	201	164	136	121
≥5	1408	1265	1505	1802	1683	1806	1765	1572	1598	1561	1765	1583	1675	1564	1693	1501	1373	1234	1074

Magnitude	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
≥8	0	1	1	0	1	2	1	2	4	0	1	1	1	2	2	1	1	0	1
≥7	18	15	16	13	15	16	11	11	18	12	17	24	20	16	19	12	19	16	7
≥6	136	160	137	139	156	157	151	153	196	179	161	175	207	133	142	155	146	146	111
≥5	1192	1495	1352	1309	1364	1672	1843	1877	2283	1965	2075	2395	2692	1680	1596	1729	1558	1696	1560

(a) *Data for the earthquake count example: The number of earthquakes above a certain magnitude (left column) in the entire world for 1980-2017, retrieved from the U. S. Geological Survey earthquake catalog.*

Date	1973 Jul	1973 Oct	1974 Mar	1974 Jun	1974 Sep	1975 Jan	1975 Apr	1975 Jul	1975 Oct	1976 Feb	1976 May	1976 Aug	1976 Dec	1977 Apr
Counts	267	333	159	145	340	463	305	329	575	227	532	769	526	565
	326	144	145	138	413	531	331	329	529	318	449	852	332	742

Date	1977 Jul	1977 Sep	1978 Jan	1978 May	1978 Aug	1978 Nov	1979 Feb	1979 Aug	1979 Nov	1980 Mar	1980 Jul	1980 Oct	1980 Dec	1981 Mar
Counts	466	494	440	858	599	298	529	912	703	402	669	796	483	700
	479	620	531	751	442	824	660	834	955	453	953	808	975	627

Date	1981 Jul	1981 Sep	1981 Dec	1982 Mar	1982 Jun	1982 Sep	1982 Dec	1983 Mar	1983 Jun	1983 Sep	1983 Dec	1984 Mar	1984 Jun
Counts	418	979	757	755	517	710	240	490	497	250	271	303	386
	851	721	1112	731	748	675	272	292	389	323	272	248	290

(b) *Data for the kangaroo population example, adopted from Bayliss (1987, Appendix 8.2)*

Figure 8: *Complete data sets for the earthquake and kangaroo counting examples.*

Paper III

Title

Learning dynamical systems with particle stochastic approximation EM

Authors

Andreas Svensson and Fredrik Lindsten

Edited version of

Andreas Svensson and Fredrik Lindsten (2018). "Learning dynamical systems with particle stochastic approximation EM". Submitted for publication.

Digital identity

https://arxiv.org/abs/1806.09548

Parts of the content in this paper have previously been presented in

Fredrik Lindsten (2013). "An efficient stochastic approximation EM algorithm using conditional particle filters". In: *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vancouver, BC, Canada, pp. 6274–6278.

Financial support

Swedish Foundation for Strategic Research (SSF) via the projects *ASSEMBLE* (contract number: RIT15-0012) and *Probabilistic Modeling and Inference for Machine Learning* (contract number: ICA16-0015), and the Swedish Research Council via the project *Learning of Large-Scale Probabilistic Dynamical Models* (contract number: 2016-04278).

Learning Dynamical Systems with Particle Stochastic Approximation EM

Abstract

We present the particle stochastic approximation EM (PSAEM) algorithm for learning of dynamical systems. The method builds on the EM algorithm, an iterative procedure for maximum likelihood inference in latent variable models. By combining stochastic approximation EM and particle Gibbs with ancestor sampling (PGAS), PSAEM obtains superior computational performance and convergence properties compared to plain particle-smoothing-based approximations of the EM algorithm. PSAEM can be used for plain maximum likelihood inference as well as for empirical Bayes learning of hyperparameters. Specifically, the latter point means that existing PGAS implementations easily can be extended with PSAEM to estimate hyperparameters at almost no extra computational cost. We discuss the convergence properties of the algorithm, and demonstrate it on several machine learning applications.

1 Introduction

Learning of dynamical systems, or state-space models, is central to many machine learning problems, such as reinforcement learning, sequence modeling, and autonomous systems. Furthermore, state-space models are at the core of recent model developments within the machine learning area, such as Gaussian process state-space models (Frigola, Chen, et al. 2014; Mattos et al. 2016; etc.), infinite factorial dynamical models (Gael et al. 2009; Valera et al. 2015), and stochastic recurrent neural networks (Fraccaro et al. 2016, for example). A strategy to learn state-space models, independently suggested by Digalakis et al. (1993) and Ghahramani and Hinton (1996), is the use of the Expectation Maximization (EM, Dempster et al. 1977) method. Even though originally proposed only for maximum likelihood estimation of linear models with Gaussian noise, the strategy can be generalized to the more challenging non-linear and non-Gaussian cases, as well as the empirical Bayes setting. Many contributions have been made during the last decade, and this paper takes another step along the path towards a more computationally efficient method with a solid theoretical ground for learning of nonlinear dynamical systems.

To set the notation for this article, we write a general (discrete-time, non-linear and non-Gaussian) state-space model, or dynamical system, as

$$x_t \sim p_\theta(x_t \mid x_{t-1}), \tag{1a}$$

$$y_t \sim p_\theta(y_t \mid x_t), \tag{1b}$$

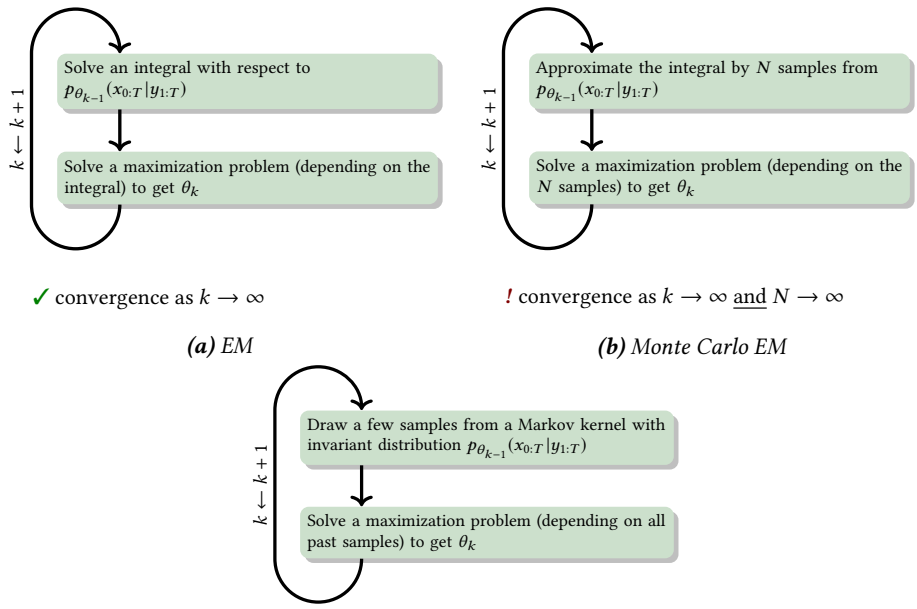
with transition density function $p_\theta(x_t \mid x_{t-1})$ and observation density function $p_\theta(y_t \mid x_t)$, parameterized by some unknown parameter $\theta \in \Theta$. Here, $\{x_t \in \mathsf{X}\}_{t=0,1,\ldots}$ denotes the unobserved state and $\{y_t \in \mathsf{Y}\}_{t=1,2\ldots}$ denotes the observations, and the index t is referred to as 'time'. The initial state x_0 is distributed according to $p(x_0)$[1]. We consider θ as unknown and the focus of this paper is to *learn* it from recorded data $(y_1, \ldots, y_T) \triangleq y_{1:T}$.

The EM algorithm, which is the strategy we will follow, iteratively solves an integration problem and a maximization problem. When using EM for learning non-linear state-space models (1), the integral includes the posterior distribution over the unobserved states $x_{0:T}$, and possibly also the parameter θ. This distribution is in general analytically intractable, but it can be approximated using computational methods such as particle filters/sequential Monte Carlo (SMC).

The combination of EM and SMC, as suggested by Cappé et al. 2005; Olsson et al. 2008; Schön et al. 2011, has provided a principled solution to the challenging problem of learning general state-space models (1), but is unfortunately 'doubly asymptotic'; to ensure convergence, it requires (i) an infinite number of particles/Monte Carlo samples in the approximation of the posterior of $x_{0:T}$ for *each iteration* of the EM algorithm, and (ii) the EM algorithm itself converges only as its number of iterations goes to infinity. This is a theoretical as well as a practical issue, and we will in this paper explore a solution where particle Markov chain Monte Carlo (PMCMC), rather than plain SMC, is used, which allows the two asymptotical convergences to be 'entangled'. This will give us an algorithm which relies on asymptotics only in one dimension (its number of iterations, not the number of particles), and thereby enjoys a significantly reduced computational cost and superior convergence properties compared to the predecessors. This overall picture is also briefly summarized in Figure 1. Further discussion of related work is postponed to Section 3.

Throughout the paper we assume that the reader is familiar with Markov chain Monte Carlo (MCMC, Robert and Casella 2004; Tierney 1994) as well as particle filters/sequential Monte Carlo (SMC, Doucet, Freitas, et al. 2001; Doucet and Johansen 2011).

[1]For notational brevity we assume that the initial density $p(x_0)$ is fully specified and not parameterized by θ, but the extension to an unknown initial density is straightforward.



(a) *EM* ***(b)*** *Monte Carlo EM*

(c) *SAEM*

Figure 1: *A brief summary of the different flavors of Expectation Maximization (EM) methods for maximum likelihood estimation of θ. When it is possible to solve the state inference problem (computing $p_{\theta_k}(x_{0:T}|y_{1:T})$) analytically, vanilla EM (a) is the preferred solution. For general models of the form (1)) this is not possible and numerical approximations are required. Monte Carlo EM (b) can be used with a particle smoother (an extension of the particle filter/SMC) to generate approximate samples from $p_{\theta_k}(x_{0:T}|y_{1:T})$, which has been proposed independently in the literature several times. Alternatively, stochastic approximation EM (SAEM, (c)) can be used (whose convergence properties are superior to Monte Carlo EM), and if the required Markov kernel is built up using the particle filter in Algorithm 1 we refer to it as PSAEM, the main contribution of this paper.*

2 Problem formulation and conceptual solution

Given a batch of observations $y_{1:T}$ we wish to learn the unknown parameters θ as well as the unobserved states $x_{0:T}$ of the model (1). For the states $x_{0:T}$ we are interested in their posterior distribution. For the parameters θ, we consider two cases: In the frequentistic, or rather Fisherian, setting we are interested in a (possibly regularized) maximum likelihood estimate $\widehat{\theta}$. In the Bayesian setting, we assign a prior distribution to the parameters, $\theta \sim p_\eta(\theta)$. The prior, in turn, is assumed to be parameterized by some *hyperparameter* η, which needs to be estimated. Thus, we will address both of the following two problems:

1. **(Fisherian setting)** Compute the *maximum likelihood estimate* of the model parameters,
$$\widehat{\theta} = \arg\max_\theta \log p_\theta(y_{1:T}),$$
where the likelihood function is $p_\theta(y_{1:T}) = \int p_\theta(y_{1:T} \mid x_{0:T}) p_\theta(x_{0:T}) \mathrm{d}x_{0:T}$. If desired, a regularization term, such as $\|\theta\|_1$ (Tibshirani 1996), may also be included in the maximization criterion.

2. **(Bayesian setting)** Compute the *posterior distribution* of the model parameters $p_{\widehat{\eta}}(\theta \mid y_{1:T})$, where the hyperparameters are estimated using empirical Bayes (a.k.a. type-II maximum likelihood)
$$\widehat{\eta} = \arg\max_\eta \log p_\eta(y_{1:T}),$$
where the *marginal* likelihood function is
$$p_\eta(y_{1:T}) = \int p_\theta(y_{1:T} \mid x_{0:T}) p_\theta(x_{0:T}) p_\eta(\theta) \mathrm{d}\theta \mathrm{d}x_{0:T}.$$

These two problems are in fact strongly related and the computational algorithm that we will propose can be used to address either one of the two problems. In either case, the algorithm will compute a Monte Carlo/sampling-based approximation of the posterior distribution over the latent variables. In the Fisherian setting, the latent variables are the states $x_{0:T}$, and their posterior is the smoothing distribution $p_{\widehat{\theta}}(x_{0:T} \mid y_{1:T})$. In the Bayesian approach, however, both the states $x_{0:T}$ and the parameters θ are considered as latent variables, and we will compute the joint state and parameter posterior distribution $p_{\widehat{\eta}}(\theta, x_{0:T} \mid y_{1:T})$. From this latter distribution, the parameter posterior $p_{\widehat{\eta}}(\theta \mid y_{1:T})$ can be obtained by marginalization.

Both approaches, as seen above, involve the computation of a maximum likelihood estimate; of the model parameters in the first case and of the hyperparameters in the second case. A conceptual solution to these problems is given by the expectation maximization (EM, Dempster et al. 1977) algorithm. EM is a data augmentation method, meaning that it is based on the notion of a *complete data*, comprising the observed data as well as the latent (or missing) variables. The EM algorithm iteratively updates the (hyper-)parameters, and each iteration consists of two steps:

(E) Compute the expected value of the complete data log-likelihood for fixed (hyper-)parameters

(M) Maximize the Q-function (which will be defined below) with respect to the (hyper-)parameters

The observed data is always $y_{1:T}$, and what differs between the Fisherian and the Bayesian problem is what constitutes the latent variables. For the Fisherian problem, the latent variables are $x_{0:T}$ and we obtain, at iteration k,

$$\text{(E)} \qquad \text{Let } Q_k^{\text{Fish}}(\theta) := \int \log p_\theta(y_{1:T}, x_{0:T}) p_{\theta_{k-1}}(x_{0:T} \mid y_{1:T}) \mathrm{d}x_{0:T}, \tag{2a}$$

$$\text{(M)} \qquad \text{Solve } \theta_k \leftarrow \arg\max_\theta Q_k^{\text{Fish}}(\theta). \tag{2b}$$

Note that the expectation in the (E)-step is w.r.t. the smoothing distribution $p_{\theta_{k-1}}(x_{0:T} \mid y_{1:T})$ parameterized by the previous parameter iterate θ_{k-1}. (If a regularization term is present, it is added to $\log p_\theta(y_{1:T}, x_{0:T})$.) It is well known that iterating this procedure results in a monotone increase of the likelihood $p_\theta(y_{1:T})$, and the iterates θ_k will under weak assumptions converge to a stationary point of the likelihood function as $k \to \infty$ (see e.g., Wu 1983).

In the empirical Bayes setting we obtain similar expressions, but the latent variables now comprise both $x_{0:T}$ and θ. The E-step is thus modified, and iteration k is

$$\begin{aligned} \text{(E)} \qquad \text{Let } Q_k^{\text{Bay}}(\eta) &:= \int \log p_\eta(y_{1:T}, x_{0:T}, \theta) p_{\eta_{k-1}}(\theta, x_{0:T} \mid y_{1:T}) \mathrm{d}\theta \mathrm{d}x_{0:T} \\ &= \int \log p_\eta(\theta) p_{\eta_{k-1}}(\theta \mid y_{1:T}) \mathrm{d}\theta + \text{const.}, \end{aligned} \tag{3a}$$

$$\text{(M)} \qquad \text{Solve } \eta_k \leftarrow \arg\max_\eta Q_k^{\text{Bay}}(\eta), \tag{3b}$$

where the second line of (3a) follows from the fact that in the factorization of the complete data likelihood, only the prior density $p_\eta(\theta)$ depends on the hyperparameter η. The M-step remains unchanged. In complete analogy to the Fisherian setting, (3) will also under weak assumptions converge to a stationary point of the marginal likelihood as $k \to \infty$.

Both (2) and (3) can be implemented and iterated until convergence, as long as the integrals can be computed and the maximization problem solved. However, in most cases—specifically for the type of dynamical systems we consider in this paper—the integrals can *not* be solved analytically, and the topic for the rest of this paper is essentially to design an efficient method for approximating the integrals. The solution will be based on PMCMC (Andrieu, Doucet, et al. 2010), but also a stochastic approximation of the Q-function (Delyon et al. 1999) to ensure a computationally efficient solution with good convergence properties. Our solution will therefore be more involved than just replacing the integrals in (2) or (3) with vanilla Monte Carlo estimators.

A short word on notation: we will use subscripts to denote sequences of variables for which we are seeking a maximum, like η_k, and brackets for samples of variables for which we are seeking a posterior distributions, like $x_{0:T}[k]$.

3 Related work and contributions

The use of EM for learning linear state-space models appears to have been independently suggested by, at least, Digalakis et al. (1993) and Ghahramani and Hinton (1996). In the linear case the state inference problem can be solved exactly using a Kalman filter, which is not the case for nonlinear models. To this end, the extended Kalman filter has been proposed (Delattre and Lavielle 2013; Ghahramani and Roweis 1998), as well as SMC-based solutions (Cappé et al. 2005; Olsson et al. 2008; Schön et al. 2011), the latter leading to a so-called Monte Carlo EM solution.

EM is a general strategy for latent variable models, and the standard choice in the application of EM to state-space models is to select the states as the latent variable. That is, however, not the only possible choice, and Umenberger et al. (2018) shows that by considering the process noise realization (instead of the states) as latent variables, it is possible to introduce stability guarantees for the learned model, at the cost of a more involved maximization problem.

Stochastic approximation EM (SAEM, Delyon et al. 1999; Kuhn and Lavielle 2004) can be used to improve the convergence properties and reduce the computational cost, compared to Monte Carlo EM. This is particularly true when the Monte Carlo simulation is computationally involved, which is the case for SMC-based solutions. In the context of state-space models, SAEM appears to first have been proposed by Donnet and Samson (2011) and Andrieu and Vihola (2014), who suggest to combine it with a particle independent Metropolis–Hastings procedure (PIMH, Andrieu, Doucet, et al. 2010) to infer the latent states. The idea to combine SAEM with particle Gibbs with ancestor sampling (PGAS, Lindsten, Jordan, et al. 2014), which often has a much lower computational cost compared to PIMH, was first suggested in a brief conference paper by Lindsten (2013)—the present article is an extension of this paper. Since its first publication, this method—which we will refer to as PSAEM—has found applications in system identification (Paper I and IV), causal inference (Gong et al. 2017), and econometrics (Singor et al. 2017), to mention a few. In this paper, we will introduce and study PSAEM more thoroughly, formulate it explicitly for empirical Bayes learning, present a new theoretical result, and illustrate the method's applicability to some contemporary dynamical systems models from the machine learning literature (Gaussian process state-space models (Frigola, Chen, et al. 2014; Svensson, Solin, et al. 2016) and infinite factorial dynamical models (Valera et al. 2015)).

4 Particle stochastic approximation EM

We will now build up and present the contribution of this paper, the particle stochastic approximation EM (PSAEM) algorithm. The two main components are (i) an MCMC kernel for simulating the latent variables from either $p_\theta(x_{0:T} \mid y_{1:T})$ or $p_\eta(\theta, x_{0:T} \mid y_{1:T})$, and (ii) a stochastic approximation version of the EM algorithm (SAEM, Delyon et al. 1999; Kuhn and Lavielle 2004), to update the (hyper-)parameter estimate. We will start with the former (Section 4.1) and thereafter turn to the latter (Section 4.2).

4.1 Sampling the latent variables using PGAS

At the core of the EM algorithm is the posterior inference of the latent variables, which is needed for the integrals in (2a) or (3a). For general non-linear or non-Gaussian state-space models these posterior distributions are intractable and we are therefore forced to numerical approximations. Fortunately, much research has been focused on developing computational algorithms for addressing these problems over the past decades and there are by now many powerful tools available. We will focus on one class of methods which we believe is particularly well suited for the problem at hand (as discussed below), namely PMCMC (Andrieu, Doucet, et al. 2010). This is a framework for using particle filters to construct efficient high-dimensional Markov kernels. Specifically, we will make use of the method PGAS, (Lindsten, Jordan, et al. 2014), which has been shown to have good empirical performance in many situations (Linderman et al. 2014; Marcos et al. 2015; Meent et al. 2015; Valera et al. 2015, etc).

To start we assume that the parameters to estimate (θ or η) are fixed at some value, and consider how $p_\theta(x_{0:T} \mid y_{1:T})$ or $p_\eta(\theta, x_{0:T} \mid y_{1:T})$ can be approximated using PGAS. Consider first the Fisherian setting. Just like any MCMC method would do, PGAS makes use of an Markov kernel on the space X^{T+1} with $p_\theta(x_{0:T} \mid y_{1:T})$ as its unique stationary distribution. This kernel is then applied iteratively, and if certain ergodicity assumptions hold, this procedure will eventually produce samples from $p_\theta(x_{0:T} \mid y_{1:T})$. In PGAS this Markov kernel is constructed using a particle filter, or more precisely a *conditional particle filter with ancestor sampling*, given in Algorithm 1. One execution of the entire Algorithm 1 will correspond to one iteration of the Markov kernel. The conditional particle filter resembles a standard particle filter with $N - 1$ particles, with the addition that there is also a *conditional* particle trajectory (for convenience numbered N, line 2 and 8) which is specified a priori. In the resampling step (line 5), this conditional trajectory can be replicated, but never discarded. At the end, one single trajectory is extracted, which will be used as the new conditional trajectory for the next iteration. The ancestor sampling (line 7) assigns ancestors to the contional trajectory, similar to resampling but 'backwards' in time and only for the conditional trajectory. We refer to Lindsten, Jordan, et al. (2014) for further details.

Remark: Algorithm 1 is formulated in its 'bootstrap' version, but a more general SMC formulation is also possible, see Lindsten, Jordan, et al. (2014).

Algorithm 1: Conditional particle filter with ancestor sampling

Input: Conditional trajectory $x'_{0:T}$, parameter θ.
Output: Output: Trajectory $x^\star_{0:T}$.

Draw $x_0^i \sim p(x_0), i = 1, \ldots, N-1$.
Set $x_0^N \leftarrow x'_0$.
Set $w_0^i \leftarrow 1, i = 1, \ldots, N$.
for $t = 1, 2, \ldots, T$ **do**
  Draw a_t^i with $\Pr(a_t^i = j) \propto w_{t-1}^j$ for $i = 1, \ldots, N-1$.
  Draw $x_t^i \sim p_\theta(x_t \mid x_{t-1}^{a_t^i})$ for $i = 1, \ldots, N-1$.
  Draw a_t^N with $\Pr(a_t^N = j) \propto w_{t-1}^j p_\theta(x'_t \mid x_{t-1}^j)$.
  Set $x_t^N \leftarrow x'_t$.
  Set $w_t^i \leftarrow p_\theta(y_t \mid x_t^i)$ for $i = 1, \ldots, N$.
end
Draw I with $\Pr(I = i) \propto w_T^i$.
Set $x_T^\star = x_T^I$.
for $t = T-1, T-2, \ldots, 0$ **do**
  Set $I \leftarrow a_{t+1}^I$.
  Set $x_t^\star \leftarrow x_t^I$.
end

Formally we let Algorithm 1 define a Markov kernel Π_θ on the space of state trajectories X^{T+1} given by

$$\Pi_\theta(x'_{0:T}, B) = \mathbb{E}\left[\mathbb{1}(x^\star_{0:T} \in B)\right] \tag{4}$$

where the expectation is w.r.t. the random variables used in Algorithm 1. The Markov kernel constructed by Algorithm 1 takes a state trajectory $x_{0:T}[j-1] = x'_{0:T} \in \mathsf{X}^{T+1}$ as input and outputs another state trajectory $x_{0:T}[j] = x^\star_{0:T} \in \mathsf{X}^{T+1}$. Put differently, a sample $x_{0:T}[j] \sim \Pi_\theta(x_{0:T}[j-1], \cdot)$ can be generated by executing Algorithm 1 with fixed θ and $x_{0:T}[j-1]$ as input reference trajectory. If this is iterated, an MCMC procedure on the space X^{T+1} is obtained, and the trajectories $x_{0:T}[0]$, $x_{0:T}[1]$, $x_{0:T}[2]$, …, will eventually (as $j \to \infty$) be samples from the sought smoothing distribution $p_\theta(x_{0:T} \mid y_{1:T})$. MCMC methods like this, which uses Markov kernels based on particle filters, are called PMCMC.

It is far from obvious that Π_θ admits $p_\theta(x_{0:T} \mid y_{1:T})$ as its stationary distribution. However, its properties (as well as those of its older sibling based on conditional particle filtering without ancestor sampling by Andrieu, Doucet, et al. 2010) have been extensively studied, see for example Andrieu, Doucet, et al. 2010; Andrieu, Lee, et al. 2018; Chopin and Singh 2015; Del Moral et al. 2014; Lindsten, Douc, et al. 2015; Lindsten, Jordan, et al. 2014. The main results are: (i) $p_\theta(x_{0:T} \mid y_{1:T})$ is a stationary distribution of Π_θ, and (ii) Π_θ is uniformly geometrically ergodic for any $N \geq 2$ under (weak) boundedness conditions on w_t^i in Algorithm 1. We summarize this PMCMC procedure to infer $p_\theta(x_{0:T} \mid y_{1:T})$ in Algorithm 2 (still assuming θ is fixed).

Algorithm 2: PMCMC inference for $p_\theta(x_{0:T} \mid y_{1:T})$ (Fisherian, fixed θ)

1 Initialize $x_{0:T}[0]$ arbitrarily, e.g., by running a standard particle filter targeting $p_\theta(x_{0:T} \mid y_{1:T})$.
2 **for** $j = 1, 2, \ldots, J$ **do**
3 | Sample $x_{0:T}[j] \sim \Pi_\theta(x_{0:T}[j-1], \cdot)$ (that is, run Algorithm 1 once)
4 **end**

So far we have only considered the Fisherian setting, in which the latent variables only comprise the state trajectory. Turning to the Bayesian setting, we assume that η (instead of θ) is fixed, and we see from (3a) that we have to compute the model parameter posterior distribution $p_\eta(\theta \mid y_{1:T})$. We will do this by first computing the joint posterior over both the model parameters and the states, $p_\eta(\theta, x_{0:T} \mid y_{1:T})$, using Gibbs sampling. That is, we split the simulation problem into two steps, one in which we sample $x_{0:T}$ conditionally on θ (and $y_{1:T}$) and one in which we sample θ conditionally on $x_{0:T}$ (and $y_{1:T}$). The first step, sampling $x_{0:T}$ conditionally on θ, is equivalent to the problem discussed for the Fisherian setting, and we can use the Markov kernel Π_θ defined in (4) and Algorithm 1. For the second step, sampling θ conditionally on $x_{0:T}$, exact solutions are often possible, leading to Gibbs smapling. Otherwise, rejection sampling is possible, leading to Hastings-within-Gibbs sampling (see, for instance, Tierney (1994, Section 2.4)). The particular choice depends on the actual model, and we will later illustrate it by an example. Let the Markov kernel used to simulate θ be denoted by $\Pi_{\eta, x_{0:T}}(\theta', \cdot)$. The resulting MCMC procedure used in the Bayesian setting (still assuming a fixed value for η) is summarized in Algorithm 3, and converges (in the same sense as Algorithm 2) to $p_\eta(\theta, x_{0:T} \mid y_{1:T})$.

Algorithm 3: MCMC inference for $p_\eta(\theta, x_{0:T} \mid y_{1:T})$ (Bayesian, fixed η)

1 Initialize $\theta[0]$ and $x_{0:T}[0]$ arbitrarily. For the latter e.g., by a particle filter targeting $p_{\theta[0]}(x_{0:T} \mid y_{1:T})$
2 **for** $j = 1, 2, \ldots$ **do**
3 | Sample $x_{0:T}[j] \sim \Pi_{\theta[j-1]}(x_{0:T}[j-1], \cdot)$ (that is, run Algorithm 1 once)
4 | Sample $\theta[j] \sim \Pi_{\eta, x_{0:T}[j]}(\theta[j-1], \cdot)$
5 **end**

4.2 Combining PGAS and EM

We have so far assumed that the (hyper-)parameters are fix. The objective in this paper is, however, to learn those, and we will for this purpose use a stochastic approximation version of the EM algorithm.

A naive solution using EM and PMCMC

The problem with the preliminary EM solutions outlined in (2) and (3), respectively, is the analytically intractable integrals in their Q-functions. A first idea would be to replace the integrals with sums over J Monte Carlo samples. For the Fisherian setting, this means replacing the (E)-step of (2) with a simulation (Si) step as follows:

$$\text{(Si)} \qquad \begin{cases} \text{Draw} & \{x_{0:T}[j]\}_{j=1}^{J} \sim p_{\theta_{k-1}}(x_{0:T} \mid y_{1:T}) \\ \text{and let} & \widehat{Q}_k^{\text{Fish}}(\theta) := \frac{1}{J} \sum_{j=1}^{J} \log p_\theta(y_{1:T}, x_{0:T}[j]). \end{cases} \tag{5a}$$

$$\text{(M)} \qquad \text{Solve } \theta_k \leftarrow \arg\max_\theta \widehat{Q}_k^{\text{Fish}}(\theta). \tag{5b}$$

Note that this is our initial EM scheme (2), but with the analytically intractable integral over $\log p_\theta(y_{1:T}, x_{0:T})$ approximated by a sum. This algorithm is commonly referred to as Monte Carlo EM (Wei and Tanner 1990) or, if $J = 1$, stochastic EM (Diebolt and Ip 1996). To draw the samples in the (Si)-step we can use PGAS from Section 4.1, which would give Algorithm 4.

Algorithm 4: A Monte Carlo EM implementation for the Fisherian problem

1 Initialize θ_0
2 **for** $k = 1, 2, \ldots$ **do**
3 Run Alg. 4 with θ_{k-1} 'until convergence' to obtain J samples $\{x_{0:T}[j]\}_{j=1}^{J}$
4 Solve $\theta_k \leftarrow \arg\max_\theta \frac{1}{J} \sum_{j=1}^{J} \log p_\theta(y_{1:T}, x_{0:T}[j])$
5 **end**

A similar algorithm could be devised for the Bayesian setting. Even though Algorithm 4 might look promising, there are two issues with this solution:

(i) To guarantee that Algorithm 2 has converged to its stationary distribution, we cannot bound its number of iterations at line 3.

(ii) For the sum in (5a)/line 4 to converge to the integral it approximates, we must let $J \to \infty$.

Indeed, these two issues are related. We basically need to allow $J \to \infty$ to ensure convergence, whilst the convergence of the EM iteration happens as $k \to \infty$. This is not desirable since it, intuitively, gives a computational complexity of "$\infty \times \infty$"; see further Fort and Moulines 2003. Existing methods based on various types of particle smoothing for approximating the integral with respect to $p_\theta(x_{0:T} \mid y_{1:T})$, for instance (Olsson et al. 2008; Schön et al. 2011), suffer from the same issues. Indeed, these methods can also be viewed as (SMC-based) instances of Monte Carlo EM.

We will now first address issue (ii) with stochastic approximation EM, and thereafter handle issue (i) by 'entangling' the convergence of Algorithm 2 ($J \to \infty$) with the convergence of the EM algorithm ($k \to \infty$).

SAEM: Handling sample approximations within EM

Stochastic approximation, as introduced by Robbins and Monro (1951), is an averaging procedure to solve a (deterministic) equation which can only be evaluated through noisy (stochastic) observations. In stochastic approximation a step length $\gamma_k \in [0, 1]$ is used, which has to fulfill

$$\sum_{k=1}^{\infty} \gamma_k = \infty, \qquad \sum_{k=1}^{\infty} \gamma_k^2 < \infty, \qquad \gamma_k = 1. \tag{6}$$

Following Delyon et al. (1999), the SAEM algorithm can be introduced by making a stochastic approximation of the Q-function. In SAEM, we transform Monte Carlo EM (5) by introducing a stochastic approximation (SA)-step. For simplicity we only use one sample ($J = 1$) in the simulation (Si)-step, but in practice it can be favorable to use a small batch of samples. For iteration k this becomes

$$\text{(Si)} \qquad \text{Draw } x_{0:T}[k] \sim p_{\theta_{k-1}}(x_{0:T} \mid y_{1:T}). \tag{7a}$$

$$\text{(SA)} \qquad \text{Let } \mathbb{Q}_k^{\text{Fish}}(\theta) \leftarrow (1-\gamma_k)\mathbb{Q}_{k-1}^{\text{Fish}}(\theta) + \gamma_k \log p_\theta(y_{1:T}, x_{0:T}[k]). \tag{7b}$$

$$\text{(M)} \qquad \text{Solve } \theta_k \leftarrow \arg\max_\theta \mathbb{Q}_k^{\text{Fish}}(\theta). \tag{7c}$$

To intuitively understand the stochastic approximation, let us first ignore the (M)-step and assume $\gamma_k = \frac{1}{k}$. In such a case, the (SA)-step would simply be online averaging, equivalent to $\mathbb{Q}_k^{\text{Fish}}(\theta) = \frac{1}{k}\sum_{\ell=1}^{k} \log p_\theta(y_{1:T}, x_{0:T}[\ell])$, where $x_{0:T}[\ell] \sim p_\theta(x_{0:T} \mid y_{1:T})$, which converges to $\int \log p_\theta(y_{1:T}, x_{0:T}) p_\theta(x_{0:T} \mid y_{1:T}) \mathrm{d}x_{0:T}$ when $k \to \infty$ by the law of large numbers. The introduction of the (M) step complicates the picture, but assuming that θ_k will eventually converge to a stationary point, the influence from the transient phase will vanish as $k \to \infty$, and the averaging argument can still be applied. In Section 5 we discuss the convergence properties in detail. Before that, in Section 4.3, we will consider the important special case of exponential family models, for which the (SA) step reduces to a convenient recursive update of sufficient statistics.

With (7) in place, we can make stronger theoretical claims (even though we are using only a single sample, $J = 1$, from $p_{\theta_{k-1}}(x_{0:T} \mid y_{1:T})$, at each iteration!) thanks to the use of stochastic approximation (Delyon et al. 1999). However, for the problem under study it is still of limited practical use since we cannot generate samples from $p_{\theta_{k-1}}(x_{0:T} \mid y_{1:T})$ by other means than using Algorithm 2 with an infinite number of iterations (in order to ensure that it has converged). Thus, our final step is to use the method studied by Kuhn and Lavielle (2004) to combine SAEM with an MCMC procedure in a more intricate way than (7).

PSAEM: Combining SAEM with PGAS

As suggested and analyzed by Kuhn and Lavielle (2004), the draw from $p_{\theta_{k-1}}(x_{0:T} \mid y_{1:T})$ in (7a) can be replaced with a draw from a Markov kernel which has $p_{\theta_{k-1}}(x_{0:T} \mid y_{1:T})$ as its invariant distribution. As discussed, this is exactly what the PGAS kernel Π_θ from (4) is, and we can thus assemble

$$\text{(Si)} \quad \text{Draw } x_{0:T}[k] \sim \Pi_{\theta_{k-1}}(x_{0:T}[k-1], \cdot) \text{ (that is, run Algorithm 1 } \textit{once}\text{).} \tag{8a}$$

$$\text{(SA)} \quad \text{Let } \mathbb{Q}_k^{\text{Fish}}(\theta) \leftarrow (1-\gamma_k)\mathbb{Q}_{k-1}^{\text{Fish}}(\theta) + \gamma_k \log p_\theta(y_{1:T}, x_{0:T}[k]). \tag{8b}$$

$$\text{(M)} \quad \text{Solve } \theta_k \leftarrow \arg\max_\theta \mathbb{Q}_k^{\text{Fish}}(\theta). \tag{8c}$$

Note that we do not make use of Algorithm 2 anymore, but only Algorithm 1. This means that we do not run the Makov kernel "until convergence" at each iteration, but it will (intuitively speaking) converge in parallel with the SAEM iterations indexed with k. We summarize and present this as Algorithms 5 and 6, for the Fisherian and Bayesian case, respectively.

Algorithm 5: Particle stochastic approximation EM for the Fisherian setting

1 Initialize $x_{0:T}[0], \theta_0$
2 **for** $k = 1, 2, \ldots$ **do**
3 Run Algorithm 1 conditional on $x_{0:T}[k-1]$ and θ_{k-1} to sample $x_{0:T}[k]$
4 Update $\mathbb{Q}_k^{\text{Fish}}(\theta) \leftarrow (1-\gamma_k)\mathbb{Q}_{k-1}^{\text{Fish}}(\theta) + \gamma_k \log p_\theta(y_{1:T}, x_{0:T}[k])$
5 Solve and update parameters $\theta_k \leftarrow \arg\max_\theta \mathbb{Q}_k^{\text{Fish}}(\theta)$
6 **end**

Algorithm 6: Particle stochastic approximation EM for the Bayesian setting

1 Initialize $x_{0:T}[0], \theta[0], \eta_0$
2 **for** $k = 1, 2, \ldots$ **do**
3 Run Algorithm 1 conditional on $x_{0:T}[k-1]$ and $\theta[k-1]$ to sample $x_{0:T}[k]$
4 Sample $\theta[k] \sim \Pi_{\eta_{k-1}, x_{0:T}[k]}(\theta[k-1], \cdot)$
5 Update $\mathbb{Q}_k^{\text{Bay}}(\eta) \leftarrow (1-\gamma_k)\mathbb{Q}_{k-1}^{\text{Bay}}(\eta) + \gamma_k \log p_\eta(\theta[k])$
6 Solve and update hyperparameters $\eta_k \leftarrow \arg\max_\eta \mathbb{Q}_k^{\text{Bay}}(\eta)$
7 **end**

We have now obtained an algorithm which only relies on asymptotics as $k \to \infty$, by 'entangling' the convergence of PGAS with the convergence of SAEM. As we will see in Section 5, convergence can be shown under certain assumptions. We will now consider the important special case of models (1) in the exponential family, for which the recursively defined function $\mathbb{Q}_k$ reduces to a much simpler expression.

4.3 PSAEM for exponential family models

Studying Algorithm 5 or 7, one may expect the computational cost of all computations involving the $\mathbb{Q}$-function to increase as $k \to \infty$, since $\mathbb{Q}_k$ is a function of all past samples. This is, however, not the case if the model belongs to the exponential family, which is an important special case discussed below.

When we write "the model belongs to the exponential family", we mean that the joint distribution for the latent and observed variables, $p_\theta(x_{0:T}, y_{1:T})$ or $p_\eta(\theta, x_{0:T}, y_{1:T})$, belongs to the exponential family with θ or η as its parameter, respectively. For the Fisherian case, this is fulfilled if both equations in (1) can, with some choice of $S_x : \mathsf{X} \times \mathsf{X} \mapsto \mathbb{R}^\ell, \psi_x : \Theta \mapsto \mathbb{R}, \phi_x : \Theta \mapsto \mathbb{R}^\ell$ (for some ℓ), and similarly for some S_y, ψ_y, ϕ_y, be written as

$$p_\theta(x_t \mid x_{t-1}) \overset{\theta}{\propto} \exp\left\{-\psi_x(\theta) + \langle S_x(x_{t-1}, x_t), \phi_x(\theta)\rangle\right\}, \tag{9a}$$

$$p_\theta(y_t \mid x_t) \overset{\theta}{\propto} \exp\left\{-\psi_y(\theta) + \langle S_y(x_t, y_t), \phi_y(\theta)\rangle\right\}. \tag{9b}$$

Here, $\overset{\theta}{\propto}$ reads "proportional (with respect to θ) to" and $\langle \cdot, \cdot \rangle$ is an inner product. The subscripts (x, y and θ) do not denote dependencies in this context, but are only names.

For the Bayesian case, the requirements are weaker, and it is enough that the prior distribution for θ belongs to the exponential family,

$$p_\eta(\theta) \overset{\eta}{\propto} \exp\left\{-\psi_\theta(\eta) + \langle S_\theta(\theta), \phi_\theta(\eta)\rangle\right\}. \tag{10}$$

We will now see how the $\mathbb{Q}$ function from the (SA)-step simplifies for models which can be written on one of these forms. First consider the Fisherian case. Using the Markovian structure of (1), we can write

$$\log p_\theta(y_{1:T}, x_{0:T}) = \sum_{t=1}^{T} \log p_\theta(y_t \mid x_t) + \log p_\theta(x_t \mid x_{t-1}) + \text{const.} \tag{11}$$

$$= -\psi(\theta) + \langle S(x_{0:T}, y_{1:T}), \phi(\theta)\rangle + \text{const.} \tag{12}$$

where

$$\psi(\theta) = T\left\{\psi_x(\theta) + \psi_y(\theta)\right\}, \quad S(x_{0:T}, y_{1:T}) = \sum_{t=1}^{T} \begin{pmatrix} S_x(x_{t-1}, x_t) \\ S_y(x_t, y_t) \end{pmatrix}, \quad \phi(\theta) = \begin{pmatrix} \phi_x(\theta) \\ \phi_y(\theta) \end{pmatrix}.$$

Here we have used the fact that the initial distribution $p(x_0)$ is independent of θ (for notational simplicity). It follows that

$$\mathbb{Q}_k^{\text{Fish}}(\theta) = -\psi(\theta) + \langle \mathbb{S}_k, \phi(\theta)\rangle + \text{constant}, \tag{13a}$$

where

$$\mathbb{S}_k = (1-\gamma_k)\mathbb{S}_{k-1} + \gamma_k S(x_{0:T}[k], y_{1:T}). \tag{13b}$$

Note that this is a non-recursive definition of $\mathbb{Q}_k^{\text{Fish}}(\theta)$, but instead recursive in $\mathbb{S}_k$. For an algorithmic point of view, this means that we can compute and store $\mathbb{S}_k$ as (13b), and solve the maximization problem for (13a) instead of the more intricate and computationally challenging (8b). In fact, the maximizing argument to (13a) can be expressed on closed form in many cases.

Analogously the Bayesian case is obtained as,

$$\mathbb{Q}_k^{\text{Bay}}(\eta) = -\psi_\theta(\eta) + \langle \mathbb{S}_k, \phi_\theta(\eta)\rangle + \text{constant}. \tag{14a}$$

where

$$\mathbb{S}_k = (1-\gamma_k)\mathbb{S}_{k-1} + \gamma_k S_\theta(\theta[k]). \tag{14b}$$

We summarize in Algorithms 7 and 8.

Algorithm 7: PSAEM for exponential family models, Fisherian setting

Initialize $x_{0:T}[0]$ and θ_0
for $k = 1, 2, \ldots$ **do**
  Run Algorithm 1 conditional on $x_{0:T}[k-1]$ and θ_{k-1} to sample $x_{0:T}[k]$
  Update sufficient statistics $\mathbb{S}_k$ according to (13b)
  Solve and update parameters $\theta_k \leftarrow \arg\max_\theta \mathbb{Q}_k^{\text{Fish}}(\theta)$ using (13a)
end

Algorithm 8: PSAEM for exponential family models, Bayesian setting

Initialize $X[0]$, $\theta[0]$ and η_0
for $k = 1, 2, \ldots$ **do**
  Run Algorithm 1 conditional on $x_{0:T}[k-1]$ and $\theta[k-1]$ to sample $x_{0:T}[k]$
  Sample $\theta[k] \sim \Pi_{\eta_{k-1}, x_{0:T}[k]}(\theta[k-1], \cdot)$
  Update sufficient statistics $\mathbb{S}_k$ according to (14b)
  Solve and update hyperparameters $\eta_k \leftarrow \arg\max_\eta \mathbb{Q}_k^{\text{Bay}}(\eta)$ using (14a)
end

5 Convergence

The convergence of SAEM and its extensions, including MCMC-based implementations, has received a lot of attention in the research community (Andrieu, Moulines, et al. 2005; Andrieu and Vihola 2014; Delyon et al. 1999; Kuhn and Lavielle 2004). In Section 5.1 we present a basic convergence result for PSAEM. This is essentially an application of Kuhn and Lavielle (2004, Theorem 1), however, we also add a missing piece regarding the continuity of the PGAS Markov kernel. This will under certain (strong) assumptions on X, Θ and the model (1) imply convergence of PSAEM as $k \to \infty$ (with finite $N \geq 2$ fixed in Algorithm 1). Some of these conditions could possibly be weakened by using the algorithmic modifications proposed by Andrieu and Vihola 2014, but we do not pursue this further here. We will also, in Section 5.2, discuss some practical considerations regarding the choice of N and γ_k.

In the presentation below we write $\|f\|_\infty = \sup_x |f(x)|$ for the supremum norm of function f and $\Pi f(x) = \int \Pi(x, dx') f(x')$ for the Markov kernel Π acting on f.

5.1 Theoretical results

We will for brevity present this section in the Fisherian setting, but the results extend also to the Bayesian setting by considering $\{x_{0:T}, \theta\}$ as the latent variables instead of $x_{0:T}$, if ergodicity and continuity of the joint Markov kernel for $x_{0:T}$ and θ can be shown.

Convergence of the SAEM algorithm has only been established for models in the exponential family. The essence of the assumptions used by Kuhn and Lavielle (2004) are:

(A1) The model belongs to the exponential family, and the log-likelihood function and its atoms ϕ, ψ and S are sufficiently smooth, differentiable and integrable.

(A2) A unique solution to the maximization problem in the (M)-step exists.

(A3) X and Θ are compact.

(A4) The Markov kernel Π_θ for sampling $x_{0:T}$ is uniformly ergodic and its density is Lipschitz continuous w.r.t θ for all $x_{0:T}$.

For a more precise statement of the assumptions, we refer to (SAEM3') in Kuhn and Lavielle (2004) and (M1-M5), (SAEM1)-(SAEM2) and, if applicable, (MAX1)-(MAX2) in Delyon et al. (1999). Under (A(A1))-(A(A4)), Kuhn and Lavielle (2004) show that SAEM converges to a stationary point of the likelihood surface. Assumption (A(A1))-(A(A3)) define the class of models (1) for which convergence is proven. The compactness assumptions on X and Θ are strong, but not strictly necessary, see Delyon et al. (1999, Section 5), Andrieu, Moulines, et al. (2005) and also the more recent development by Andrieu and Vihola (2014). Assumption (A(A4)) puts requirements (uniform ergodicity and Lipschitz continuity) on the MCMC kernel that is used, which is PGAS in our case. Uniform ergodicity has been shown for PGAS under a boundedness assumption on the weights of the conditional particle filter (Lindsten, Jordan, et al. 2014, Theorem 3). What has not previously been shown, though, is Lipschitz continuity of the PGAS Markov kernel. To establish this we introduce the following additional assumptions.

(A5) There exists constants $L_1, L_2 < \infty$, $\delta_1, \delta_2 > 0$ and $\kappa_1, \kappa_2 < \infty$ such that, for all $(x_{t-1}, x_t) \in \mathsf{X}^2$ and all $t = 1, \ldots, T$,

(a) *Lipschitz continuity of transition and likelihood densities:*

$$|p_\theta(x_t \mid x_{t-1}) - p_{\widetilde{\theta}}(x_t \mid x_{t-1})| \leq L_1 \|\theta - \widetilde{\theta}\|,$$
$$|p_\theta(y_t \mid x_t) - p_{\widetilde{\theta}}(y_t \mid x_t)| \leq L_2 \|\theta - \widetilde{\theta}\|.$$

(b) *Strong mixing:*
For all $\theta \in \Theta$, $\delta_1 \leq p_\theta(x_t \mid x_{t-1}) \leq \kappa_1$ and $\delta_2 \leq p_\theta(y_t \mid x_t) \leq \kappa_2$.

Remark: The lower bound on the state transition and likelihood functions in (A(A5)b), commonly referred to as the *strong mixing condition*, are indeed strong but standard for many theoretical results on SMC, see for instance Del Moral (2004). Furthermore, this assumption essentially boils down to compactness of the state and parameter spaces, which is assumed in (A(A3)) already.

Theorem 1 (Lipschitz continuity of PGAS)**.** *Assume (A(A5)) and let Π_θ denote the PGAS Markov kernel* (4)*. Then there exists a constant $C < \infty$ such that for any bounded function $f : \mathsf{X}^{T+1} \mapsto \mathbb{R}$, it holds that*

$$\|\Pi_\theta f - \Pi_{\widetilde{\theta}} f\|_\infty \leq C\|f\|_\infty \|\theta - \widetilde{\theta}\|.$$

Proof. See Appendix A. □

We may now piece all results together into the main theorem of this section, which establishes the convergence of PSAEM.

Theorem 2 (Convergence of PSAEM)**.** *Assume (A(A1))-(A(A3)) and (A(A5)) and let θ_k be computed by Algorithm 7. Then $\lim_{k\to\infty} d(\theta_k, \mathcal{L}) = 0$, where $d(\theta, \mathcal{L})$ denotes the distance from θ to the set $\mathcal{L} = \{\theta \in \Theta : \frac{\partial}{\partial\theta} p_\theta(y_{1:T}) = 0\}$.*

Proof. The big picture is that Theorem 1 (together with existing ergodicity results) implies (A4), and therefore Theorem 2 follows from Kuhn and Lavielle (2004, Theorem 1). The technical details of (A4) are, however, more intricate, and are found in Appendix B. □

5.2 Practical considerations

Even though Theorem 2 gives a reassuring theoretical foundation for using PSAEM, it does not give any practical advice on some of the (few) tuning parameters available: the choice of step length $\{\gamma_k\}_{k=1}^\infty$ or the number of particles N in Algorithm 1.

A common choice for step length is $\gamma_k = k^{-\alpha}$, and the requirements (6) are fulfilled for any $\alpha \in (\frac{1}{2}, 1]$. In our experience, it is often advisable to choose $\alpha < 1$, perhaps $\alpha = 0.7$, not to constrain the steps too much. Even though not necessary, the initial convergence speed can sometimes be improved by setting some initial step lengths to constant 1, before starting the sequence of decreasing step lengths.

For N, we have to make a balance between a well mixing Markov kernel (large N) and the computational load (small N). Let K denote the number of iterations of PSAEM, and assume that the computational budget available is such that the product KN is limited. In such a situation, the general advice would be to take N 'small' and K 'large'. However, if N is too small, the Markov kernel will not mix well, affecting the convergence speed. To monitor the mixing, the overlap between two consecutive state trajectories $x_{0:T}[k-1]$ and $x_{0:T}[k]$ could be computed, and if it exceeds a certain threshold, say 90%, a warning could be raised that the mixing is not sufficient and N should be increased.

6 Experiments and applications

We will in this section first (Section 6.1) illustrate the behavior of PSAEM on a small toy example (where the maximum likelihood estimate can be found exactly), and study the advantage over a standard Monte Carlo EM implementation for the same problem. We will thereafter turn to three different machine learning applications, namely parameter estimation in a non-linear state-space model (the Fisherian setting, Section 6.2), and hyperparameter estimation (Bayesian setting) in infinite factorial dynamical models (Section 6.3) and Gaussian process state-space models (Section 6.4), respectively. Full details for all examples are found in Appendix C.

6.1 Linear Gaussian state-space model

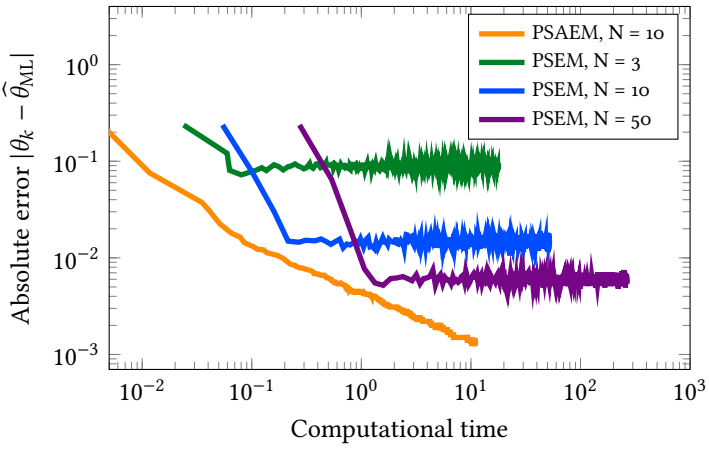
We start with considering $T = 300$ data points from the model

$$x_{t+1} = \theta x_t + w_t, \qquad w_t \sim \mathcal{N}(0, 1), \tag{15a}$$

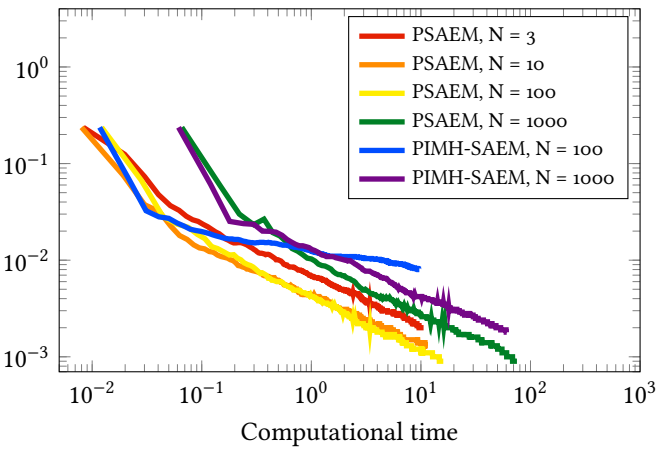
$$y_t = x_t + e_t, \qquad e_t \sim \mathcal{N}(0, 0.3), \tag{15b}$$

with $\theta \in [-1, 1]$. We apply PSAEM as well as PSEM (a Monte Carlo EM solution using a particle smoother presented by Schön et al. 2011), and PIMH-SAEM (similar to PSAEM, but using particle independent Metropolis–Hastings instead of PGAS; Andrieu and Vihola 2014; Donnet and Samson 2011) with different numbers of particles N. This model is indeed toy, but is interesting since we can find the maximum likelihood estimate $\widehat{\theta}_{\mathrm{ML}}$ of θ exactly and use as a reference.

In Figure 2, the evolution of the absolute error of the applied methods is shown as a function of computational time on the same standard desktop computer with comparable implementations, averaged over 100 realizations of each algorithm. Note that PSAEM and PIMH-SAEM converge as $k \to \infty$ (for fixed N), whereas PSEM has a non-vanishing bias which only decreases as $N \to \infty$, that is, $k \to \infty$ is not sufficient for convergence in PSEM. Comparing PSAEM and PIMH-SAEM, the latter requires a significantly larger number of particles than PSAEM, and has therefore a higher computational cost. This difference is believed to be even more pronounced for larger values of T, due to superior scaling properties of PGAS compared to PIMH.

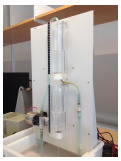


(a) *Besides the superior computational time, PSAEM (orange) does not suffer from the bias present in PSEM (green, blue, purple). The bias in PSEM is caused by the Monte Carlo approximation of the integral in the Q-function. Because of this, PSEM converges, and the bias vanishes, only as $N \to \infty$, which is in contrast to PSAEM, for which it sufficies that only $k \to \infty$.*



(b) *For this problem the optimal N for PSAEM is between 10 and 100 (orange, yellow); smaller N causes poor mixing (red) and thereby slower convergence; larger N (green) increases computational cost without improving convergence. PIMH-SAEM with $N = 100$ (blue) struggles because of poor mixing, whereas $N = 1000$ (purple) mixes well but has an unfavorably high computational cost.*

Figure 2: *Estimation of θ in (15) with three different methods; the proposed PSAEM, PSEM (a Monte Carlo EM solution using a particle smoother) and PIMH-SAEM (similar to PSAEM, but with a PIMH Markov kernel instead of PGAS). The methods are run with various number of particles N. Their average evolution of the absolute error $|\theta_k - \widehat{\theta}_{ML}|$ (over 100 runs), where $\widehat{\theta}_{ML}$ is the exact maximum likelihood estimate, is shown as a function of the wall clock time.*



Model	Simulation (test data)
Initial model to PSAEM	2.85
Estimated with PSAEM	**0.29**
Relan et al. (2017)	0.34

Table 9.1: *The cascaded water tank setup and modeling results. We initialize the 9 unknown parameters with an ad-hoc educated guess (top row), and then optimize them with PSAEM (middle row). We also include the best performing result previously published (last row). The figure of merit is root-mean-squared error for simulation on the test data.*

6.2 Cascaded water tanks

We consider the benchmark problem of learning a model for a cascaded water tank system, using the data presented by[2] Schoukens and Noël 2017. A training and a test data set of input-output data samples $\{u_t, y_t\}$, each with $T = 1024$ data points, are provided. The data is recorded from an experimental setup where water is pumped into an upper water tank, from which it flows through a small opening into a lower water tank, and from there through another small opening into a basin. During the data collection, overflow occasionally occurred in the tanks, and the excess water from the upper tank partially flowed into the lower tank. Only the pump voltage (input) and the water level in the lower tank (output) is measured each $T_s = 4$ second, and the problem is to predict the water level in the lower tank given only the pump voltage. A discrete-time nonlinear state-space model (partly adopted from Holmes, Rogers, et al. 2016) based on physical principles is

$$\begin{aligned} x^u_{t+1} &= 10 \wedge x^u_t + T_s(-k_1\sqrt{10 \wedge x^u_t} - k_2\{10 \wedge x^u_t\} + k_5 u_t) + w^u_t \\ x^l_{t+1} &= 10 \wedge x^l_t + T_s(k_1\sqrt{10 \wedge x^u_t} + k_2\{10 \wedge x^u_t\} - k_3\sqrt{10 \wedge x^l_t} - k_4\{10 \wedge x^l_t\} + k_6\{(x^u_t - 10) \vee 0\}) + w^l_t \\ y_t &= 10 \wedge x^l_t + e_t, \end{aligned} \tag{16}$$

where the states $x^u_t \in \mathbb{R}$ and $x^l_t \in \mathbb{R}$ are the water levels plus the inflow in the upper and lower tank, respectively. The parameters k_1, k_2, k_3, k_4, k_5 represent unknown physical quantities, such as tank and hole diameters, flow constants, pump efficiency, etc. Each tank has height 10 (in the scale of the sensor), and k_6 and $10 \wedge (\dots)$ is motivated by the overflow events. The initial level of the upper water tank is modeled as $x^u_0 \sim \mathcal{N}(\xi_0, \sqrt{0.1})$, with ξ_0 unknown. Furthermore, w^1_t, w^2_t and e_t are assumed to be zero mean white Gaussian noise with unknown variances σ^2_w and σ^2_e, respectively. All in all, the unknown parameters are $\theta = \{k_1, k_2, k_3, k_4, k_5, k_6, \sigma^2_e, \sigma^2_w, \xi_0\}$.

The model belongs to the exponential family, and we can thus apply PSAEM as presented in Algorithm 6 to find a maximum likelihood estimate of θ. We initialize θ randomly around physically reasonable values, and run PSAEM with $N = 100$ and $K = 50$ (taking a few seconds on a standard desktop computer). The obtained results are reported in Table 9.1 together with the best performing result previously published (to the best of the author's knowledge). Most of the previously published methods take a more data-driven approach, but the relatively small amount of data available makes the encoding of physical knowledge important, as we have done here by using (16) and PSAEM.

[2] See also http://www.nonlinearbenchmark.org

6.3 Hyperparameter estimation in infinite factorial dynamical models

The infinite factorial dynamical model (iFDM), proposed by Valera et al. (2015), is a Bayesian non-parametric model for separation of an aggregated time-series into independent sources. By using a Markov Indian buffet process, the number of sources (dimensionality of the hidden state) does not have to be upper bounded a priori. Each source is modeled as a (discrete or continuous) Markov chain which evolves independently of the other. To solve the inference problem, that is performing the actual source separation, PGAS has proven useful (Valera et al. 2015). There is, however, a multitude of hyperparameters in this Bayesian setting, and we will demonstrate how the procedure by Valera et al. (2015) easily can be extended with PSAEM to automatically estimate hyperparameters on-the-fly, reducing the need for extensive manual tuning.

We will consider the cocktail party problem originating from Gael et al. (2009), to which iFDM has been applied (Valera et al. 2015, Section 4). The voices from 15 different speakers is aggregated into a $T = 1085$ long sequence, together with some noise, and the problem is to jointly infer (i) the number of speakers (dimension of x_t), (ii) when each speaker is talking (the trajectory x_t) and (iii) the dynamics of each speaker (how prone s/he is to talk).

Each speaker is modeled as a Markov chain with two states, 'talking' or 'quiet', and the posterior distribution over its transition probabilities is inferred individually for each speaker. The Beta distribution is used as prior for these probabilities, and the hyperparameters for the Beta distribution are manually chosen by Valera et al. We outline in Algorithm 9 how the inference procedure can be extended with PSAEM (new lines are marked with blue). In addition to the lessened burden of manual hyperparameter tuning, we can also report slightly improved results: The hyperparameters automatically found by PSAEM are such that the average number of switches between 'quiet' and 'talking' in the posterior samples are closer to the ground truth (84 instead of 86, ground truth: 62) and the average value of the complete data likelihood of the posterior samples increases, compared to the choice of hyperparameters by Valera et al. (2015). Of course, PSAEM could be applied also to other hyperparameters in the problem, following the very same pattern. Posterior samples of x_t are shown in Figure 3.

Algorithm 9: PSAEM for iFDM: the cocktail party example

1 **for** $k = 0$ *to* K **do**
2 Update the state dimensionality (number of speakers) M_+ using slice sampling.
3 Sample a state trajectory using PGAS.
4 Gibbs update of the transition probabilities $\{b^m\}_{m=1}^{M_+}$ for each speaker.
5 $\mathbb{S}_k \leftarrow (1-\gamma_k)\mathbb{S}_{k-1} + \gamma_k S(\{b^m\}_{m=1}^{M_+})$, with S being sufficient statistics for the Beta distribution.
6 $\eta_k \leftarrow \arg\max_\eta \mathbb{Q}_k^{\text{Bay}}(\eta) = \arg\max_\eta \langle \mathbb{S}_k, \phi(\eta)\rangle - \psi(\eta)$.
7 Gibbs update of noise variance parameters.
8 **end**

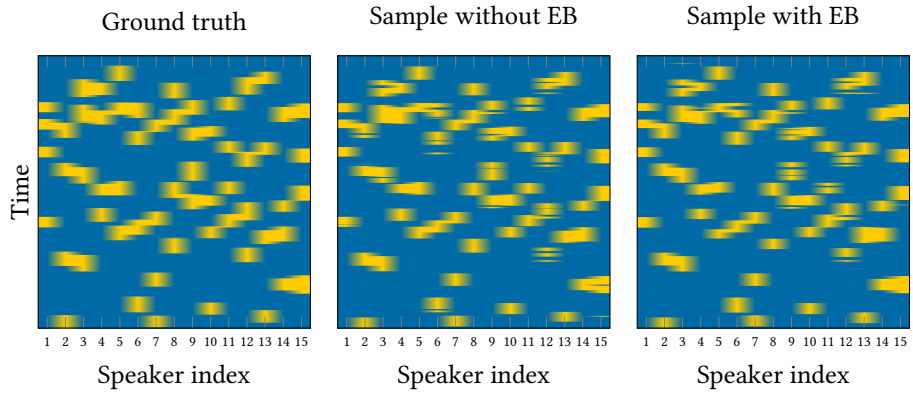


Figure 3: *The cocktail problem, introduced by Gael et al. (2009) and Valera et al. (2015), amounts to inferring the number of speakers (columns) and their periods of talking and being quiet (yellow and blue, respectively) from an aggregated observation (not shown) during a time sequence (along y-axis). The middle panel is a sample from the solution by Valera et al. (2015), and the right panel a sample after we have extended that solution with PSAEM to automatically estimate some hyperparameters. Both solutions infer the correct number of speakers (15), but the solution with PSAEM is slightly less prone to switch between quiet and talking (closer to the ground truth). The main advantage of PSAEM, not present in the plot, is however the lessened need for manual tuning of hyperparameters, at almost no extra computational cost. (The problem is unsupervised, but the columns are manually sorted to enhance comparisons between the panels.)*

6.4 Hyperparameter estimation in GP state-space models

Gaussian process state-space models are a class of models constructed as a combination of the state-space model and the Gaussian process (GP) model as

$$x_{t+1} = f(x_t) + w_t, \qquad f \sim \mathcal{GP}\left(m_\eta^f, K_\eta^f\right), \qquad w_t \sim \mathcal{N}\left(0, \Sigma_n^f\right), \tag{17a}$$

$$y_t = g(x_t) + e_t, \qquad g \sim \mathcal{GP}\left(m_\eta^g, K_\eta^g\right), \qquad e_t \sim \mathcal{N}\left(0, \Sigma_n^g\right), \tag{17b}$$

or variations thereof. As in any state-space model, only $y_{1:T}$ is observed and not $x_{0:T}$, and standard GP regression methods (Rasmussen and Williams 2006) can therefore not be used to learn the posterior $p_\eta(f, g \mid y_{1:T})$. Consequently, learning of the GP hyperparameters η—usually done via empirical Bayes $\widehat{\eta} = \arg\max_\eta p_\eta(f, g \mid y_{1:T})$—is not straightforward either.

Despite the computational challenges, it has been argued that the model is versatile and powerful by its combination of the dynamic state-space model and the nonparametric and probabilistic GP, and has for this reason achieved attention in the machine learning literature. One proposed solution is to use PGAS for learning the model (Frigola, Lindsten, Thomas B. Schön, et al. 2013; Frigola, Lindsten, Thomas B Schön, et al. 2014; Svensson, Solin, et al. 2016), and we extended that solution with PSAEM to also include estimation of the hyperparameters at almost no extra computational cost.

We consider the solution proposed by Svensson, Solin, et al. (2016), in which the nonparametric GP is approximated with a reduced-rank representation with a

Algorithm 10: PSAEM for hyperparameters in GP state-space models

1 Initialize $x_{1:T}[0], \theta[0], \eta_0$.
2 **for** $k = 0$ *to* K **do**
3 Sample $x_{0:T}[k] \mid \theta_{k-1}, \eta_{k-1}$ using Algorithm 1.
4 Sample $\theta_k \mid \eta_{k-1}, x_{0:T}[k]$ with a closed-form expression.
5 Update $\mathbb{S}_k \leftarrow (1 - \gamma_k)\mathbb{S}_{k-1} + \gamma_k S(\theta[k])$.
6 Solve $\eta_k \leftarrow \arg\max_\eta \mathbb{Q}_k^{\text{Bay}}(\eta) = \arg\max_\eta \langle \mathbb{S}_k, \phi(\eta) \rangle - \psi(\eta)$.
7 **end**

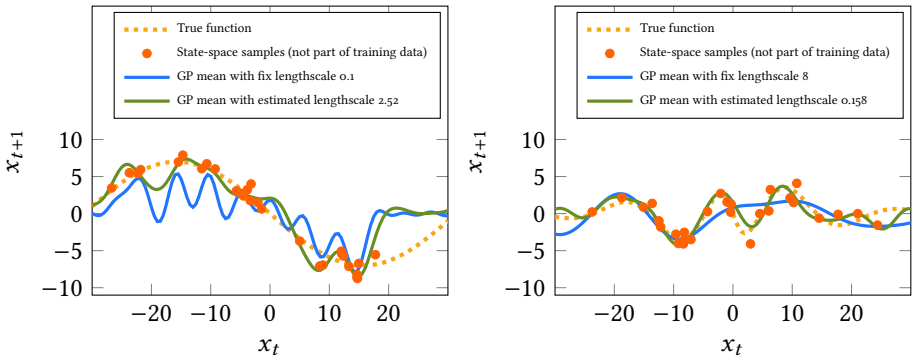


Figure 4: *Estimation of hyperparameters in two Gaussian process state-space models. Since the states x_t are unobserved in the GP-SSM, the learning is more challenging than standard GP regression. We have extended an PGAS procedure (Svensson, Solin, et al. 2016) to also include estimatation of the hyperparameters with PSAEM. The blue line is the posterior mean of $p(f \mid y_{1:T}, \eta)$ with η being a fixed lengthscale, whereas the length scale η is estimated with PSAEM for the green line. The true function is dashed yellow, alongside with the $T = 40$ samples $x_{0:T}$ (orange dots) underlying the training data (only $y_{1:T}$, not shown, is available when learning the model). Indeed, the effect of the length scale hyperparameter is very similar to standard GP regression, but the main point of this example is merely a proof of concept for hyperparameter estimation in the challenging GP-SSM model.*

finite parameter set θ. We introduce PSAEM for this solution in Algorithm 7 (new lines in blue). Since the computational burden in practice is dominated by running the conditional particle filter (line 2), the inclusion of PSAEM adds very little extra computational cost. An example of estimation of the length scale in a Gaussian process state-space model is shown in Figure 4, where the space of x_t is one-dimensional and $g(x) = x$ is considered known, but the noise level is significant with $\sigma_n^f = \sigma_n^g = 1$.

7 Conclusions

We have presented PSAEM (Algorithm 5 and 7) for learning nonlinear state-space models, both in a maximum likelihood setting and in an empirical Bayes setting. We have also summarized the available theoretical results, and added a missing piece about continuity of the PGAS Markov kernel, in order to show convergence for the case of exponential family models. We have, furthermore, illustrated how it can be applied to some contemporary machine learning models. We believe that the proposed PSAEM method can be particularly useful for models where PGAS is currently used. Indeed, with small modifications of existing code and little computational overhead it enables automatic estimation of hyperparameters, thereby avoiding the need of difficult and tedious manual tuning.

A Proof of Theorem 1, Lipschitz continuity of PGAS

This appendix contains a proof of Theorem 1. It is based on the construction of a coupling between the Markov kernels Π_θ and $\Pi_{\widetilde{\theta}}$. A similar technique has previously been used by Chopin and Singh (2015) to prove uniform ergodicity of the Particle Gibbs kernel. Jacob et al. (2017) explicitly use couplings of conditional particle filters to construct (practical) algorithms for, among other things, likelihood estimation and unbiased estimates of smoothing functionals.

We first review some basic properties of couplings and total variation. Let P and Q be two probability measures with densities p and q, respectively, with respect to some reference measure λ. Let C be the set of couplings of P and Q, that is, joint probability measures with marginals P and Q. We can then write the total variation distance between P and Q in the following equivalent ways:

$$\|P - Q\|_{\mathrm{TV}} = \frac{1}{2} \sup_{|f| \leq 1} |Pf - Qf| \tag{18a}$$

$$= \lambda(\max\{p - q, 0\}) \tag{18b}$$

$$= 1 - \lambda(\min\{p, q\}) \tag{18c}$$

$$= \inf_{\xi \in C} \iint \mathbb{1}(x \neq y) \xi(dx, dy). \tag{18d}$$

Note also that it is possible to explicitly construct a coupling attaining the infimum in (18d): let $\alpha = \lambda(\min\{p, q\})$, $\nu(dx) = \alpha^{-1} \min\{p(x), q(x)\}\lambda(dx)$, and

$$\xi(dx, dy) = \alpha\nu(dx)\delta_x(dy) + (1 - \alpha)^{-1}(P(dx) - \alpha\nu(dx))(Q(dy) - \alpha\nu(dy)). \tag{19}$$

A coupling ξ which attains the infimum, or equivalently which maximizes the probability of X and Y being identical when $(X, Y) \sim \xi$, is referred to as a *maximal coupling*. Finally, for a coupling ξ, the quantity $\iint \mathbb{1}(x = y)\xi(dx, dy)$—that is, the probability that X and Y are identical under ξ—is referred to as the *coupling probability* under ξ.

Now, to prove the Lipschitz continuity of the PGAS Markov kernel as stated in Theorem 1 we will construct a coupling $\xi_{\theta,\widetilde{\theta}}(x'_{0:T}, dx^\star_{0:T}, d\widetilde{x}^\star_{0:T})$ of the Markov kernels $\Pi_\theta(x'_{0:T}, dx^\star_{0:T})$ and $\Pi_{\widetilde{\theta}}(x'_{0:T}, d\widetilde{x}^\star_{0:T})$. This coupling is defined via Algorithm 11, which takes $x'_{0:T}$ as input and produces $x^\star_{0:T}$ and $\widetilde{x}^\star_{0:T}$ as outputs, such that the marginal distributions of the output trajectories are $\Pi_\theta(x'_{0:T}, dx^\star_{0:T})$ and $\Pi_{\widetilde{\theta}}(x'_{0:T}, d\widetilde{x}^\star_{0:T})$, respectively. For ease of notation in Algorithm 11, we write $\mathsf{M}[P, Q]$ for any maximal coupling (for instance the one given by (19)) of some distributions P and Q. With slight abuse of notation, we also write $\mathsf{M}[p, q]$ for probability density functions p and q, even if they are not normalized in which case it is understood that the coupling is between the *normalized* probability distributions.

Note that for any bounded function f,

$$\begin{aligned}
\|\Pi_\theta f - \Pi_{\widetilde{\theta}} f\|_\infty &\leq \|f\|_\infty \sup_{x'_{0:T}} \sup_{|g| \leq 1} |\Pi_\theta g(x'_{0:T}) - \Pi_{\widetilde{\theta}} g(x'_{0:T})| \\
&\leq 2\|f\|_\infty \sup_{x'_{0:T}} \iint \mathbb{1}(x^\star_{0:T} \neq \widetilde{x}^\star_{0:T}) \xi_{\theta,\widetilde{\theta}}(x'_{0:T}, dx^\star_{0:T}, d\widetilde{x}^\star_{0:T}) \\
&= 2\|f\|_\infty \sup_{x'_{0:T}} \left(1 - \iint \mathbb{1}(x^\star_{0:T} = \widetilde{x}^\star_{0:T}) \xi_{\theta,\widetilde{\theta}}(x'_{0:T}, dx^\star_{0:T}, d\widetilde{x}^\star_{0:T})\right)
\end{aligned}$$

Algorithm 11: Coupled CPFs with ancestor sampling defining $\xi_{\theta,\widetilde{\theta}}$.

Input: Conditional trajectory $x'_{0:T}$, parameters θ and $\widetilde{\theta}$.
Output: Trajectories $x^\star_{0:T}$ and $\widetilde{x}^\star_{0:T}$.

1 Draw $x_0^i \sim p(x_0)$ and set $\widetilde{x}_0^i \leftarrow x_0^i$, $i = 1, \ldots, N-1$.
2 Set $x_0^N \leftarrow x'_0$ and $\widetilde{x}_0^N \leftarrow x'_0$.
3 Set $w_0^i \leftarrow 1$ and $\widetilde{w}_0^i \leftarrow 1$, $i = 1, \ldots, N$.
4 **for** $t = 1, 2, \ldots, T$ **do**
5 Draw $(a_t^i, \widetilde{a}_t^i) \sim \mathsf{M}\Big[\{w_{t-1}^j\}_{j=1}^N, \{\widetilde{w}_{t-1}^j\}_{j=1}^N\Big]$ for $i = 1, \ldots, N-1$.
6 Draw $(x_t^i, \widetilde{x}_t^i) \sim \mathsf{M}\Big[p_\theta(\cdot \mid x_{t-1}^{a_t^i}), p_{\widetilde{\theta}}(\cdot \mid \widetilde{x}_{t-1}^{\widetilde{a}_t^i})\Big]$ for $i = 1, \ldots, N-1$.
7 Draw $(a_t^N, \widetilde{a}_t^N) \sim \mathsf{M}\Big[\{w_{t-1}^j p_\theta(x'_t \mid x_{t-1}^j)\}_{j=1}^N, \{\widetilde{w}_{t-1}^j p_{\widetilde{\theta}}(x'_t \mid \widetilde{x}_{t-1}^j)\}_{j=1}^N\Big]$.
8 Set $x_t^N \leftarrow x'_t$ and $\widetilde{x}_t^N \leftarrow x'_t$
9 Set $w_t^i \leftarrow p_\theta(y_t \mid x_t^i)$ and $\widetilde{w}_t^i \leftarrow p_{\widetilde{\theta}}(y_t \mid \widetilde{x}_t^i)$ for $i = 1, \ldots, N$.
10 **end**
11 Draw $(J, \widetilde{J}) \sim \mathsf{M}\big[\{w_T^i\}_{i=1}^N, \{\widetilde{w}_T^i\}_{i=1}^N\big]$.
12 Set $x_T^\star = x_T^J$ and $\widetilde{x}_T^\star = \widetilde{x}_T^{\widetilde{J}}$.
13 **for** $t = T-1, T-2, \ldots, 0$ **do**
14 Set $J \leftarrow a_{t+1}^J$ and $\widetilde{J} \leftarrow \widetilde{a}_{t+1}^{\widetilde{J}}$.
15 Set $x_t^\star \leftarrow x_t^J$ and $\widetilde{x}_t^\star \leftarrow \widetilde{x}_t^{\widetilde{J}}$.
16 **end**

where we have used (18a) and (18d) for the first and second lines, respectively. Hence, it is sufficient to show that

$$\iint \mathbb{1}(x^\star_{0:T} = \widetilde{x}^\star_{0:T}) \xi_{\theta,\widetilde{\theta}}(x'_{0:T}, dx^\star_{0:T}, d\widetilde{x}^\star_{0:T}) \geq 1 - \frac{C}{2}\|\theta - \widetilde{\theta}\|, \tag{20}$$

where C is the same constant as in the statement of the theorem.

Let α_{t-1} denote the coupling probability for the coupling at line 4 of Algorithm 11 (and thus α_T is the coupling probability on line 10). On the set $\{x_t^{1:N} = \widetilde{x}_t^{1:N}\}$ we have by (18c)

$$\begin{aligned}\alpha_t &= \sum_{i=1}^N \min\left\{\frac{w_t^i}{\sum_k w_t^k}, \frac{\widetilde{w}_t^i}{\sum_k \widetilde{w}_t^k}\right\} \geq \frac{\sum_{i=1}^N \min\left\{w_t^i, \widetilde{w}_t^i\right\}}{\sum_{i=1}^N \max\left\{w_t^i, \widetilde{w}_t^i\right\}} \\ &\geq \frac{\sum_{i=1}^N \left(\max\left\{w_t^i, \widetilde{w}_t^i\right\} - L_2\|\theta - \widetilde{\theta}\|\right)}{\sum_{i=1}^N \max\left\{w_t^i, \widetilde{w}_t^i\right\}} \geq 1 - \frac{L_2}{\delta_2}\|\theta - \widetilde{\theta}\|,\end{aligned} \tag{21}$$

where we have used the Lipschitz continuity of the likelihood (A6a) for the penultimate inequality, and the lower bound on the likelihood (A6b) for the last inequality.

Similarly, let β_t denote the coupling probability for the coupling on line 7. Under assumption (A6), the product $p_\theta(y_{t-1} \mid x_{t-1}) p_\theta(x_t \mid x_{t-1})$ (which constitutes the unnormalized ancestor sampling weights) is bounded from below by $\delta_1\delta_2$. The product is also Lipschitz continuous in θ: since $|ab-cd| = |ab-ad+ad-cd| \leq |a||b-d|+|d||a-c|$

we have

$$|p_\theta(y_{t-1} \mid x_{t-1})p_\theta(x_t \mid x_{t-1}) - p_{\widetilde{\theta}}(y_{t-1} \mid x_{t-1})p_{\widetilde{\theta}}(x_t \mid x_{t-1})| \leq (\kappa_1 L_2 + \kappa_2 L_1)\|\theta - \widetilde{\theta}\|.$$

Therefore, on the set $\{x_{t-1}^{1:N} = \widetilde{x}_{t-1}^{1:N}\}$, we have by a computation analogous to above,

$$\beta_t \geq 1 - \frac{\kappa_1 L_2 + \kappa_2 L_1}{\delta_1 \delta_2}\|\theta - \widetilde{\theta}\|. \tag{22}$$

Finally, let γ_t^i denote the coupling probability for the coupling at line 5, for the ith particle. By (18b) and (18d) we have, on the set $\{x_{t-1}^{1:N} = \widetilde{x}_{t-1}^{1:N}, a_t^{1:N} = \widetilde{a}_t^{1:N}\}$,

$$\gamma_t^i = 1 - \lambda(\max\{p_\theta(x_t \mid x_{t-1}^{a_t^i}) - p_{\widetilde{\theta}}(x_t \mid x_{t-1}^{a_t^i}), 0\}) \geq 1 - L_1\lambda(\mathrm{X})\|\theta - \widetilde{\theta}\|,$$

where λ denotes Lebesgue measure and where the inequality follows by (A6a). By (A3), $\lambda(\mathrm{X}) < \infty$. Note that the bound on γ_t^i is independent of i.

Now, if we write $\mathbb{P}$ for probability with respect to the random variables generated by Algorithm 11, we can crudely bound (20) by

$$\begin{aligned}
\mathbb{P}(\{x_t^{1:N} = \widetilde{x}_t^{1:N}, a_t^{1:N} = \widetilde{a}_t^{1:N} : t = 1, \ldots, T\}, J = \widetilde{J}) &\geq \mathbb{E}\left[\alpha_T \prod_{t=1}^{T}\left(\beta_t \prod_{i=1}^{N-1} \alpha_t \gamma_t^i\right)\right] \\
&\geq \left(1 - \frac{L_2}{\delta_2}\|\theta - \widetilde{\theta}\|\right)^{T(N-1)+1} \times \left(1 - \frac{\kappa_1 L_2 + \kappa_2 L_1}{\delta_1 \delta_2}\|\theta - \widetilde{\theta}\|\right)^{T} \times \left(1 - L_1\lambda(\mathrm{X})\|\theta - \widetilde{\theta}\|\right)^{T(N-1)} \\
&\geq (1 - D\|\theta - \widetilde{\theta}\|)^{2T(N-1)+T+1}
\end{aligned}$$

where $D = \max\{\frac{L_2}{\delta_2}, \frac{\kappa_1 L_2 + \kappa_2 L_1}{\delta_1 \delta_2}, L_1\lambda(\mathrm{X})\}$.

Finally, we note that the coupling probability is trivially bounded from below by 0 and that $(1 - D\|\theta - \widetilde{\theta}\|)^m \geq 1 - Dm\|\theta - \widetilde{\theta}\|$ on $\|\theta - \widetilde{\theta}\| \in [0, D^{-1}]$. Thus,

$$\begin{aligned}
&\iint \mathbb{1}(x_{0:T}^\star = \widetilde{x}_{0:T}^\star)\xi_{\theta,\widetilde{\theta}}(x'_{0:T}, dx_{0:T}^\star, d\widetilde{x}_{0:T}^\star) \\
&\quad \geq \mathbb{P}(\{x_t^{1:N} = \widetilde{x}_t^{1:N}, a_t^{1:N} = \widetilde{a}_t^{1:N} : t = 1, \ldots, T\}, J = \widetilde{J}) \\
&\quad \geq 1 - D(2T(N-1) + T + 1)\|\theta - \widetilde{\theta}\|,
\end{aligned}$$

which proves Theorem 1.

B Proof of Theorem 2, convergence of PSAEM

Theorem 2 follows from Kuhn and Lavielle (2004, Theorem 1), by using the established Lipschitz continuity of the PGAS kernel from Theorem 1. A slight difference, however, is that Kuhn and Lavielle (2004, Theorem 1) assumes that the Markov transition kernel admits a density with respect to Lebesgue measure and that this density function is Lipschitz continuous; see their assumption (SAEM3')2. Our continuity result is instead expressed in terms of total variation distance. The condition (SAEM3')2 is used by Kuhn and Lavielle to prove their Lemma 2, see Kuhn and Lavielle (2004, p. 129). Thus, to complete the picture we provide a lemma which replaces Kuhn and Lavielle (2004, Lemma 2). The result—which extends the continuity of the PGAS Markov kernel to the k-fold kernel—is a special case of Andrieu, Moulines, et al. (2005, Proposition B.2), but for completeness we repeat the proof here.

Lemma 1. *Assume that the conditions of Theorem 1 hold. Then, there exists a constant $D \leq \infty$ such that for any $k \geq 0$ and any bounded function f,*

$$\|\Pi_\theta^k f - \Pi_{\widetilde{\theta}}^k f\|_\infty \leq D\|f\|_\infty \|\theta - \widetilde{\theta}\|.$$

Proof. Define $\bar{f}_{\widetilde{\theta}}(x_{0:T}) = f(x_{0:T}) - \int f(x_{0:T}) p_{\widetilde{\theta}}(x_{0:T} \mid y_{0:T}) \mathrm{d}x_{0:T}$. Since $\bar{f}_{\widetilde{\theta}}$ differs from f by a constant (depending on $\widetilde{\theta}$) we can write,

$$\|\Pi_\theta^k f - \Pi_{\widetilde{\theta}}^k f\|_\infty = \|\Pi_\theta^k \bar{f}_{\widetilde{\theta}} - \Pi_{\widetilde{\theta}}^k \bar{f}_{\widetilde{\theta}}\|_\infty \leq \sum_{j=1}^{k} \|\Pi_\theta^{k-j}(\Pi_\theta - \Pi_{\widetilde{\theta}})\Pi_{\widetilde{\theta}}^{j-1} \bar{f}_{\widetilde{\theta}}\|_\infty.$$

We have,

$$\begin{aligned}
\|\Pi_\theta^{k-j}(\Pi_\theta - \Pi_{\widetilde{\theta}})\Pi_{\widetilde{\theta}}^{j-1} \bar{f}_{\widetilde{\theta}}\|_\infty &= \sup_{x_{0:T}} \left| \int \Pi_\theta^{k-j}(x_{0:T}, dx_{0:T}^\star)(\Pi_\theta - \Pi_{\widetilde{\theta}})\Pi_{\widetilde{\theta}}^{j-1} \bar{f}_{\widetilde{\theta}}(x_{0:T}^\star) \right| \\
&\leq \sup_{x_{0:T}} \int \Pi_\theta^{k-j}(x_{0:T}, dx_{0:T}^\star) \left| (\Pi_\theta - \Pi_{\widetilde{\theta}})\Pi_{\widetilde{\theta}}^{j-1} \bar{f}_{\widetilde{\theta}}(x_{0:T}^\star) \right| \\
&\leq \sup_{x_{0:T}} \int \Pi_\theta^{k-j}(x_{0:T}, dx_{0:T}^\star) \|(\Pi_\theta - \Pi_{\widetilde{\theta}})\Pi_{\widetilde{\theta}}^{j-1} \bar{f}_{\widetilde{\theta}}\|_\infty \\
&= \|(\Pi_\theta - \Pi_{\widetilde{\theta}})\Pi_{\widetilde{\theta}}^{j-1} \bar{f}_{\widetilde{\theta}}\|_\infty.
\end{aligned}$$

Now, consider the function $\Pi_{\widetilde{\theta}}^\ell \bar{f}_{\widetilde{\theta}}(x_{0:T})$ for some $\ell \geq 0$. Recall that $\bar{f}_{\widetilde{\theta}}$ is centered around the posterior expectation of f with respect to $p_{\widetilde{\theta}}(x_{0:T} \mid y_{1:T})$, which is the limiting distribution of $\Pi_{\widetilde{\theta}}$. Thus, by uniform ergodicity of $\Pi_{\widetilde{\theta}}$ for any $\widetilde{\theta} \in \Theta$ it follows that

$$\sup_{\widetilde{\theta} \in \Theta} \|\Pi_{\widetilde{\theta}}^\ell \bar{f}_{\widetilde{\theta}}\|_\infty \leq M\rho^\ell \|f\|_\infty$$

for some constants $M < \infty$ and $\rho < 1$. Consequently, the function $\Pi_{\widetilde{\theta}}^\ell \bar{f}_{\widetilde{\theta}}(x_{0:T})$ satisfies the conditions of Theorem 1 and thus

$$\|(\Pi_\theta - \Pi_{\widetilde{\theta}})\Pi_{\widetilde{\theta}}^{j-1} \bar{f}_{\widetilde{\theta}}\|_\infty \leq CM\rho^{j-1}\|\theta - \widetilde{\theta}\|.$$

Plugging this into the expressions above completes the proof. □

From this, the results of Lemma 2 in Kuhn and Lavielle (2004) follows for our assumptions, and hence also Theorem 1 of Kuhn and Lavielle (2004) and, ultimately, Theorem 2 of our main article.

C Details about experiments

This section contains additional details regarding the experiments in Section 6.

Experiment 6.1–Linear Gaussian state-space model

The step length in PSAEM, as well as PIMH-SAEM, is choosen as $\gamma_k = k^{-0.99}$. PSEM is implemented as a particle filter with N particles and a backward simulator (Godsill et al. 2004) with N backward trajectories. The sufficient statistics, as derived by for instance, Ghahramani and Hinton (1996), are $\frac{1}{T}\sum_t x_t x_t^T$ and $\frac{1}{T}\sum_t x_{t-1} x_t^T$, and the maximization problem can be solved analytically.

Experiment 6.2–Cascaded water tanks

The step length in PSAEM is choosen as $\gamma_k = 1$ for $k = 1, \ldots, 30$, and $\gamma_k = (k-30)^{-0.7}$ for $k = 31, \ldots$. The initial parameter values are initialized randomly around $k_1 = k_2 = k_3 = k_4 = 0.05$, $k_5 = k_6 = 0$, $\sigma_e^2 = \sigma_w^2 = 0.1$, $\xi_0 = 6$, and a slight L_2-regularization (corresponding to a $\mathcal{N}(0, 10^3)$ prior) is used for k_4 to avoid problems if the state trajectory contains no overflow events in the lower tank. The sufficient statistics for a model on the form $x_{t+1} = a(x_t) + \theta^T b(x_t) + w_t, \quad w_t \sim \mathcal{N}(0, \sigma^2)$, where θ and σ^2 are unknown, are $\frac{1}{T}\sum_t (x_t - a(x_{t-1}))(x_t - a(x_{t-1}))^T$, $\frac{1}{T}\sum_t b(x_{t-1})(x_t - a(x_{t-1}))^T$ and $\frac{1}{T}\sum_t b(x_{t-1})b(x_{t-1})^T$, and x_0 for the initial value. The maximization problem can be solved analytically.

Experiment 6.3–Hyperparameter estimation in iFDM

The exact setup is a replica of Valera et al. (2015), to which we refer for details. We use $\gamma_k = k^{-0.7}$, but let the PMCMC run for 500 iterations (which, by a very quick look at the trace of PGAS, appears to be a rough estimate of the burn-in period) before starting PSAEM. The initial value of η are the ones chosen by Valera et al. (2015). The sufficient statistics for M number of Beta random variables θ_m is M, $\sum_{m=1}^M \log(\theta_m)$ and $\sum_{m=1}^M \log(1-\theta_m)$. The maximization problem lacks an analytical solution, and an off-the-shelf numerical optimization routine (`fmincon` in Matlab) was applied to solve the maximization problem.

Experiment 6.4–Hyperparameter estimation in GP state-space models

The true functions in the example are $x_{t+1} \sim \mathcal{N}\left(-7\arctan(\frac{x_t}{3})\cos(\frac{x_t}{3})\exp(-\frac{|x_t|}{10}), 1\right)$ and $x_{t+1} \sim \mathcal{N}\left(-7\sin(\frac{x_t}{10}, 1\right)$, respectively.

In the approximate GP-SSM model used, the unknown function f is approximated as a finite basis function expansion, whose coefficients θ (column vector) have a certain multivariate zero mean Gaussian prior distribution with a variance depending on η (see Svensson, Solin, et al. 2016 for details). Thus, the sufficient statistics is $\theta\theta^T$, and the maximization problem to solve is $\arg\max_\eta -\frac{1}{2}\mathrm{Tr}(\theta\theta^T V_\eta^{-1}) - \frac{1}{2}\log\det(V_\eta)$ (where V_η follows from the choice of covariance function, see again Svensson, Solin, et al. 2016), which requires a numerical approach.

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

Christophe Andrieu, Anthony Lee, and Matti Vihola (2018). "Uniform Ergodicity of the Iterated Conditional SMC and Geometric Ergodicity of Particle Gibbs samplers". In: *Bernoulli* 24.2, pp. 842–872.

Christophe Andrieu, Éric Moulines, and Pierre Priouret (2005). "Stability of Stochastic Approximation under Verifiable Conditions". In: *SIAM Journal on Control and Optimization* 44.1, pp. 283–312.

Christophe Andrieu and Matti Vihola (2014). "Markovian stochastic approximation with expanding projections". In: *Bernoulli* 20.2, pp. 545–585.

Olivier Cappé, Éric Moulines, and Tobias Rydén (2005). *Inference in hidden Markov models*. Springer Series in Statistics. New York, NY, USA: Springer.

Nicolas Chopin and S. Sumeetpal Singh (2015). "On Particle Gibbs Sampling". In: *Bernoulli* 21.3, pp. 1855–1883.

Pierre Del Moral (2004). *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. New York, NY, US: Springer.

Pierre Del Moral, Robert Kohn, and Frédéric Patras (2014). "On particle Gibbs Markov chain Monte Carlo models". In: *arXiv:1404.5733*.

Maud Delattre and Marc Lavielle (2013). "Coupling the SAEM algorithm and the extended Kalman filter for maximum likelihood estimation in mixed-effects diffusion models". In: *Statistics and Its Interface* 6, pp. 519–532.

Bernard Delyon, Marc Lavielle, and Éric Moulines (1999). "Convergence of a stochastic approximation version of the EM algorithm". In: *Annals of Statistics* 27.1, pp. 94–128.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–38.

Jean Diebolt and Eddie H. S. Ip (1996). "Stochastic EM: method and application". In: *Markov Chain Monte Carlo in Practice*. Ed. by W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. Boca Raton, FL, USA: Chapman & Hall/CRC, pp. 259–274.

Vassilios V. Digalakis, Jan Robin Rohlicek, and Mari Ostendorf (1993). "ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition". In: *IEEE Transactions on Speech and Audio Processing* 1.4, pp. 431–442.

Sophie Donnet and Adeline Samson (2011). *EM algorithm coupled with particle filter for maximum likelihood parameter estimation of stochastic differential mixed-effects models*. Tech. rep. hal-00519576, v2. Université Paris Descartes, MAP5.

Arnaud Doucet, Nando de Freitas, and Neil J. Gordon (2001). "An introduction to sequential Monte Carlo methods". In: *Sequential Monte Carlo methods in practice*. Springer, pp. 3–14.

Arnaud Doucet and Adam M. Johansen (2011). "A tutorial on particle filtering and smoothing: fifteen years later". In: *Nonlinear Filtering Handbook*. Ed. by D. Crisan and B. Rozovsky. Oxford, UK: Oxford University Press, pp. 656–704.

Gersende Fort and Éric Moulines (2003). "Convergence of the Monte Carlo Expectation Maximization for Curved Exponential Families". In: *Annals of statistics* 31.4, pp. 1220–1259.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther (2016). "Sequential Neural Models with Stochastic Layers". In: *Advances in Neural Information Processing Systems (NIPS) 29*. Barcelona, Spain, pp. 2199–2207.

Roger Frigola, Yutian Chen, and Carl Rasmussen (2014). "Variational Gaussian process state-space models". In: *Advances in Neural Information Processing Systems 27 (NIPS)*. Montréal, QC, Canada, pp. 3680–3688.

Roger Frigola, Fredrik Lindsten, Thomas B. Schön, and Carl Rasmussen (2013). "Bayesian inference and learning in Gaussian process state-space models with particle MCMC". In: *Advances in Neural Information Processing Systems 26 (NIPS)*. Lake Tahoe, NV, USA, pp. 3156–3164.

Roger Frigola, Fredrik Lindsten, Thomas B Schön, and Carl Rasmussen (2014). "Identification of Gaussian process state-space models with particle stochastic approximation EM". In: *Proceedings of the 19th IFAC World Congress*. Cape Town, South Africa, pp. 4097–4102.

Jurgen V. Gael, Yee W. Teh, and Zoubin Ghahramani (2009). "The Infinite Factorial Hidden Markov Model". In: *Advances in Neural Information Processing Systems (NIPS) 21*. Vancouver, BC, ada, pp. 1967–1704.

Zoubin Ghahramani and Geoffrey E. Hinton (1996). *Parameter Estimation for Linear Dynamical Systems*. Tech. rep. CRG-TR-96-2. Department of Computer Science, University of Toronto.

Zoubin Ghahramani and Sam T. Roweis (1998). "Learning nonlinear dynamical systems using an EM algorithm". In: *Advances in Neural Information Processing Systems (NIPS) 11*. Denver, CO, USA, pp. 431–437.

Simon J. Godsill, Arnaud Doucet, and Mike West (2004). "Monte Carlo Smoothing for Nonlinear Time Series". In: *Journal of the American Statistical Association* 99.465, pp. 156–168.

Mingming Gong, Kun Zhang, Bernhard Schölkopf, Clark Glymour, and Dacheng Tao (2017). "Causal discovery from temporally aggregated time series". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. Sydney, Australia.

Geoff Holmes, Tim Rogers, et al. (2016). *Cascaded Tanks Benchmark: Parametric and Nonparametric Identification*. Presentation at *Workshop on Nonlinear System Identification Benchmarks 2016*, Vrije Universiteit Brussel. Brussels, Belgium.

Pierre E. Jacob, Fredrik Lindsten, and Thomas B. Schön (2017). "Smoothing with Couplings of Conditional Particle Filters". In: *arXiv:1701.02002*.

Estelle Kuhn and Marc Lavielle (2004). "Coupling a stochastic approximation version of EM with an MCMC procedure". In: *ESAIM: Probability and Statistics* 8, pp. 115–131.

Scott W. Linderman, Christopher H. Stock, and Ryan P. Adams (2014). "A framework for studying synaptic plasticity with neural spike train data". In: *Advances in Neural Information Processing Systems (NIPS) 28*. Montreal, QC, Canada.

Fredrik Lindsten (2013). "An efficient stochastic approximation EM algorithm using conditional particle filters". In: *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vancouver, BC, Canada, pp. 6274–6278.

Fredrik Lindsten, Randal Douc, and Éric Moulines (2015). "Uniform ergodicity of the Particle Gibbs sampler". In: *Scandinavian Journal of Statistics* 42.3, pp. 775–797.
Fredrik Lindsten, Michael I. Jordan, and Thomas B. Schön (2014). "Particle Gibbs with ancestor sampling". In: *The Journal of Machine Learning Research (JMLR)* 15.1, pp. 2145–2184.
Marta Marcos, Francisco M. Calafat, Ángel Berihuete, and Sönke Dangendorf (2015). "Long-term variations in global sea level extremes". In: *Journal of Geophysical Research* 120.12, pp. 8115–8134.
César L. C. Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A. Barreto, and Neil D. Lawrence (2016). "Recurrent Gaussian processes". In: *4th International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.
Jan-Willem van de Meent, Yang Hongseok, Vikash Mansinghka, and Frank Wood (2015). "Particle Gibbs with Ancestor Sampling for Probabilistic Programs". In: *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*. San Diego, CA, USA, pp. 986–994.
Jimmy Olsson, Olivier Cappé, Randal Douc, and Éric Moulines (2008). "Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state-space models". In: *Bernoulli* 14.1, pp. 155–179.
Carl E. Rasmussen and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. Cambridge, MA, USA: MIT Press.
Rishi Relan, Koen Tiels, Aanna Marconato, and Johan Schoukens (2017). "An Unstructured Flexible Nonlinear Model for the Cascaded Water-Tanks Benchmark". In: *Proceedings of the 20th International Federation of Automatic Control World Congress (IFAC)*, pp. 454–459.
Herbert Robbins and Sutton Monro (1951). "A stochastic approximation method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407.
Christian P. Robert and George Casella (2004). *Monte Carlo statistical methods*. 2nd ed. New York, NY, USA: Springer.
Thomas B. Schön, Adrian Wills, and Brett Ninness (2011). "System identification of nonlinear state-space models". In: *Automatica* 47.1, pp. 39–49.
Maarten Schoukens and Jean-Philippe Noël (2017). "Three Benchmarks Addressing Open Challenges in Nonlinear System Identification". In: *Proceedings of the 20th World Congress of the International Federation of Automatic Control (IFAC)*. Toulouse, France.
Stefan N. Singor, Alex Boer, J. S. C. Alberts, and Cornelis W. Oosterlee (2017). "On the modelling of nested risk-neutral stochastic processes with applications in insurance". In: *Applied Mathematical Finance* 24.2, pp. 302–336.
Andreas Svensson, Arno Solin, Simo Särkkä, and Thomas B. Schön (2016). "Computationally efficient Bayesian learning of Gaussian process state space models". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cádiz, Spain, pp. 213–221.
Robert Tibshirani (1996). "Regression shrinkage and selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 58.1, pp. 267–288.
Luke Tierney (1994). "Markov chains for exploring posterior distributions". In: *Annals of Statistics* 22.4, pp. 1701–1728.

Jack Umenberger, Johan Wågberg, Ian Manchester, and Thomas B. Schön (2018). "Maximum likelihood identification of stable linear dynamical systems". In: *Automatica*. Accepted for publication.

Isabel Valera, Francisco J. R. Ruiz, Lennart Svensson, and Fernando Perez-Cruz (2015). "Infinite Factorial Dynamical Model". In: *Advances in Neural Information Processing Systems (NIPS) 28*. Montreal, QC, Canada, pp. 1666–1674.

Greg CG Wei and Martin A Tanner (1990). "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms". In: *Journal of the American Statistical Association* 85.411, pp. 699–704.

C. F. Jeff Wu (1983). "On the Convergence Properties of the EM Algorithm". In: *Annals of Statistics* 11.1, pp. 95–103.

Paper IV

Title

Identification of jump Markov linear models using particle filters

Authors

Andreas Svensson, Thomas B. Schön and Fredrik Lindsten

Edited version of

Andreas Svensson, Thomas B. Schön, and Fredrik Lindsten (2014). "Identification of jump Markov linear models using particle filters". In: *Proceedings of the 53rd IEEE Conference on Decision and Control (CDC)*. Los Angeles, CA, USA, pp. 6504–6509.

Digital identity

doi:10.1109/cdc.2014.7040409

Financial support

The Swedish Research Council (VR) via the project *Probabilistic modeling of dynamical systems* (contract number: 621-2013-5524).

Identification of jump Markov linear models using particle filters

Abstract

Jump Markov linear models consists of a finite number of linear state space models and a discrete variable encoding the jumps (or switches) between the different linear models. Identifying jump Markov linear models makes for a challenging problem lacking an analytical solution. We derive a new expectation maximization (EM) type algorithm that produce maximum likelihood estimates of the model parameters. Our development hinges upon recent progress in combining particle filters with Markov chain Monte Carlo methods in solving the nonlinear state smoothing problem inherent in the EM formulation. Key to our development is that we exploit a conditionally linear Gaussian substructure in the model, allowing for an efficient algorithm.

1 Introduction

Consider the following *jump Markov linear model* on state space form

$$s_{t+1} \mid s_t \sim p(s_{t+1}|s_t), \tag{1a}$$

$$z_{t+1} = A_{s_{t+1}} z_t + B_{s_{t+1}} u_t + v_t, \tag{1b}$$

$$y_t = C_{s_t} z_t + D_{s_t} u_t + e_t, \tag{1c}$$

where $\sim$ means distributed according to and the (discrete) variable s_t takes values in $\{1, \ldots, K\}$ (which can be thought of as different *modes* which the model is *jumping* between) and the (continuous) variable z_t lives in $\mathbb{R}^{n_z}$. Hence, the state variable consists of $x_t \triangleq (z_t,\ s_t)$. Furthermore, $e_t \in \mathbb{R}^{n_y}$ and $v_t \in \mathbb{R}^{n_z}$ are zero mean white Gaussian noise and $\mathbb{E} v_t v_t^T = Q_{s_{t+1}}$, $\mathbb{E} e_t e_t^T = R_{s_t}$ and $\mathbb{E} v_t e_t^T \equiv 0$. The output (or measurement) is $y_t \in \mathbb{R}^{n_y}$, the input is $u_t \in \mathbb{R}^{n_u}$. As K is finite, $p(s_{t+1}|s_t)$ can be defined via a matrix $\Pi \in \mathbb{R}^{K \times K}$ with entries $\pi_{mn} \triangleq p(s_{t+1} = n | s_t = m)$.

We are interested in off-line identification of jump Markov linear models on the form (1) for the case of an *unknown jump sequence*, but the number of modes K is known. More specifically, we will formulate and solve the maximum likelihood (ML) problem to compute an estimate of the static parameters θ of a jump Markov linear model based on a batch of measurements $y_{1:T} \triangleq \{y_1, \ldots, y_T\}$ and (if available) inputs $u_{1:T}$ by solving,

$$\widehat{\theta}_{\mathrm{ML}} = \arg\max \theta \in \Theta p_\theta(y_{1:T}). \tag{2}$$

Here $\theta \triangleq \{\{A_n, B_n, C_n, D_n, Q_n, R_n\}_{n=1}^K, \Pi\}$, i.e., all unknown static parameters in model (1). Here, and throughout the paper, the dependence on the inputs $u_{1:T}$ is implicit.

Solving (2) is challenging and there are no closed form solutions available. Our approach is to derive an expectation maximization (EM, Dempster et al. 1977) type of solution, where the strategy is to separate the original problem into two closely linked problems. The first problem is a challenging, but manageable nonlinear state smoothing problem and the second problem is a tractable optimization problem. The nonlinear smoothing problem we can solve using a combination of sequential Monte Carlo (SMC) methods (particle filters and particle smoothers, Doucet and Johansen 2011) and Markov chain Monte Carlo (MCMC) methods (Robert and Casella 2004). More specifically we will make use of particle MCMC (PMCMC), which is a systematic way of exploring the strengths of both approaches by using SMC to construct the necessary high-dimensional Markov kernels needed in MCMC (Andrieu, Doucet, et al. 2010; Lindsten, Jordan, et al. 2014).

Our main contribution is a new maximum likelihood estimator that can be used to identify jump Markov linear models on the form (1). The estimator exploits the conditionally linear Gaussian substructure that is inherent in (1) via Rao-Blackwellization. More specifically we derive a Rao-Blackwellized version of the particle stochastic approximation expectation maximization (PSAEM) algorithm recently introduced in Lindsten (2013).

Jump Markov linear models, or switching linear models, is a fairly well studied class of hybrid systems. For recent overviews of existing system identification methods for jump Markov linear models, see Garulli et al. (2012) and Paoletti et al. (2007). Existing approaches considering the problem under study here include two stage methods, where the data is first segmented (using e.g. change detection type of methods) and the individual models are then identified for each segment, see e.g. Borges et al. (2005) and Pekpe et al. (2004). There has also been approximate EM algorithms proposed for identification of hybrid systems (Blackmore et al. 2007; Gil and Williams 2009) and the very recent Ashley and Andersson (2014) (differing from our method in that we use stochastic approximation EM and Rao-Blackwellization). There are also relevant relationships to the PMCMC solutions introduced in Whiteley et al. (2010) and the SMC-based on-line EM solution derived in Yildirim et al. (2013).

There are also many approaches considering the more general problem with an unknown number of modes K and an unknown state dimension n_z, see e.g. Fox et al. (2011) and Bemporad et al. (2001), making use of Bayesian nonparametric models and mixed integer programming, respectively.

2 Expectation maximization algorithms

The EM algorithm (Dempster et al. 1977) provides an iterative method for computing maximum likelihood estimates of the unknown parameters θ in a probabilistic model involving latent variables. In the jump Markov linear model (1) we observe $y_{1:T}$, whereas the state $x_{1:T}$ is latent.

The EM algorithm maximizes the likelihood by iteratively maximizing the *intermediate quantity*

$$Q(\theta, \theta') \triangleq \int \log p_\theta(x_{1:T}, y_{1:T}) p_{\theta'}(x_{1:T} \mid y_{1:T}) \mathrm{d}x_{1:T}. \tag{3}$$

More specifically, the procedure is initialized in $\theta_0 \in \Theta$ and then iterates between computing an expected (E) value and solving a maximization (M) problem,

$$\begin{aligned} &\text{(E)} \quad \text{Compute } Q(\theta, \theta_{k-1}). \\ &\text{(M)} \quad \text{Compute } \theta_k = \arg\max_{\theta \in \Theta} Q(\theta, \theta_{k-1}). \end{aligned}$$

Intuitively, this can be thought of as 'selecting the new parameters as the ones that make the given measurements *and* the current state estimate as likely as possible'.

The use of EM type algorithms to identify dynamical systems is by now fairly well explored for both linear and nonlinear models. For linear models, there are explicit expressions for all involved quantities, see e.g. Gibson and Ninness (2005) and Shumway and Stoffer (1982). For nonlinear models the intermediate quantity $Q(\theta, \theta')$ is intractable and we are forced to approximate solutions; see e.g. Cappé et al. (2005), Lindsten (2013), Olsson et al. (2008), and Schön, Wills, et al. (2011). This is the case also for the model (1) under study in this work. Indeed, the maximization step can be solved in closed form for the model (1), but (3) is still intractable in our case.

It is by now fairly well established that we can make use of sequential Monte Carlo (SMC, Doucet and Johansen 2011) or particle Markov chain Monte Carlo (PMCM, Andrieu, Doucet, et al. 2010) methods to approximate the joint smoothing distribution for a general nonlinear model arbitrarily well according to

$$\widehat{p}(x_{1:T} \mid y_{1:T}) = \sum_{i=1}^{N} w_T^i \delta_{x_{1:T}^i}(x_{1:T}), \tag{4}$$

where $x_{1:T}^i$ are random samples with corresponding importance weights w_T^i, δ_x is a point-mass distribution at x and we refer to $\{x_{1:T}^i, w_T^i\}_{i=1}^N$ as a *weighted particle system*. The particle smoothing approximation (4) can be used to approximate the integral in (3). Using this approach within EM, we obtain the particle smoothing EM (PSEM) method (Olsson et al. 2008; Schön, Wills, et al. 2011). PSEM can be viewed as an SMC-analogue of the well known Monte Carlo EM (MCEM) algorithm (Wei and Tanner 1990).

However, it has been recognized that MCEM, and analogously PSEM, makes inefficient use of the generated samples (Delyon et al. 1999). This is particularly true when the simulation step is computationally expensive, which is the case when using SMC or PMCMC. To address this shortcoming, Delyon et al. (1999) proposed to use a *stochastic approximation* (SA, Robbins and Monro 1951) of the intermediate

quantity instead of a vanilla Monte Carlo approximation, resulting in the stochastic approximation EM (SAEM) algorithm. The SAEM algorithm replaces the intermediate quantity Q in EM with

$$\widehat{Q}_k(\theta) = (1 - \gamma_k)\widehat{Q}_{k-1}(\theta) + \gamma_k \log p_\theta(y_{1:T}, x_{1:T}[k]), \tag{5}$$

with $\{\gamma_k\}_{k=1}^{\infty}$ being a sequence of step sizes which fulfils $\sum_{k=1}^{\infty} \gamma_k = \infty$ and $\sum_{k=1}^{\infty} \gamma_k^2 < \infty$. In the above, $x_{1:T}[k]$ is a sample state trajectory, simulated from the joint smoothing distribution $p_{\theta_k}(x_{1:T} \mid y_{1:T})$. It is shown by Delyon et al. (1999) that the SAEM algorithm—which iteratively updates the intermediate quantity according to (5) and computes the next parameter iterate by maximizing this stochastic approximation—enjoys good convergence properties. Indeed, despite the fact that the method requires only a single sample $x_{1:T}[k]$ at each iteration, the sequence $\{\theta_k\}_{k\geq 1}$ will converge to a maximizer of $p_\theta(y_{1:T})$ under reasonably weak assumptions.

However, in our setting it is not possible to simulate from the joint smoothing distribution $p_{\theta_k}(x_{1:T} \mid y_{1:T})$. We will therefore make use of the particle SAEM (PSAEM) method (Lindsten 2013), which combines recent PMCMC methodology with SAEM. Specifically, we will exploit the structure of (1) to develop a Rao-Blackwellized PSAEM algorithm.

We will start our development in the subsequent section by considering the smoothing problem for (1). We derive a PMCMC-based Rao-Blackwellized smoother for this model class. The proposed smoother can, principally, be used to compute (3) within PSEM. However, a more efficient approach is to use the proposed smoother to derive a Rao-Blackwellized PSAEM algorithm, see Section 4.

3 Smoothing using Monte Carlo methods

For smoothing, that is, finding $p_\theta(x_{1:t}|y_{1:t}) = p_\theta(s_{1:T}, z_{1:T}|y_{1:T})$, various Monte Carlo methods can be applied. We will use an MCMC based approach, as it fits very well in the SAEM framework (see e.g. Andrieu, Moulines, et al. 2005; Kuhn and Lavielle 2004), which together shapes the PSAEM algorithm. The aim of this section is therefore to derive an MCMC-based smoother for jump Markov linear models.

To gain efficiency, the jump sequence $s_{1:T}$ and the linear states $z_{1:T}$ are separated using conditional probabilities as

$$p_\theta(s_{1:T}, z_{1:T}|y_{1:T}) = p_\theta(z_{1:T}|s_{1:T}, y_{1:T})p_\theta(s_{1:T}|y_{1:T}). \tag{6}$$

This allows us to infer the conditionally linear states $z_{1:T}$ using closed form expressions. Hence, it is only the jump sequence $s_{1:T}$ that has to be computed using approximate inference. This technique is referred to as Rao-Blackwellization (Casella and Robert 1996).

3.1 Inferring the linear states: $p(z_{1:T}|s_{1:T}, y_{1:T})$

State inference in linear Gaussian state space models can be performed exactly in closed form. More specifically, the Kalman filter provides the expressions for the filtering density $p_\theta(z_t|s_{1:t}, y_{1:t}) = \mathcal{N}(z_t|\widehat{z}_{f;t}, P_{f;t})$ and the one step predictor density $p_\theta(z_{t+1}|s_{1:t+1}, y_{1:t}) = \mathcal{N}(z_t|\widehat{z}_{p;t+1}, P_{p;t+1})$. The marginal smoothing density

Algorithm 1: MCMC smoother

Initialize $s_{1:T}[0]$ arbitrarily
for $k \geq 1$ **do**
  Generate $s_{1:T}[k] \sim \mathcal{K}_\theta(\cdot|s_{1:T}[k-1])$
end

$p_\theta(z_t|s_{1:T}, y_{1:T}) = \mathcal{N}(z_t|\widehat{z}_{s;t}, P_{s;t})$ is provided by the Rauch-Tung-Striebel (RTS) smoother (Rauch et al. 1965). See, e.g., Kailath et al. (2000) for the relevant results. Here, we use $\mathcal{N}(x \mid \mu, \Sigma)$ to denote the density for the (multivariate) normal distribution with mean μ and covariance matrix Σ.

3.2 Inferring the jump sequence: $p(s_{1:T}|y_{1:T})$

To find $p(s_{1:T}|y_{1:T})$, an MCMC approach is used. First, the concept of using Markov kernels for smoothing is introduced, and then the construction of the kernel itself follows.

MCMC makes use of ergodic theory for statistical inference. Let $\mathcal{K}_\theta$ be a Markov kernel (to be defined below) on the T-fold product space $\{1, ..., K\}^T$. Note that the jump sequence $s_{1:T}$ lives in this space. Furthermore, assume that $\mathcal{K}_\theta$ is *ergodic* with unique stationary distribution $p_\theta(s_{1:T}|y_{1:T})$. This implies that by simulating a Markov chain with transition kernel $\mathcal{K}_\theta$, the marginal distribution of the chain will approach $p_\theta(s_{1:T}|y_{1:T})$ in the limit.

Specifically, let $s_{1:T}[0]$ be an arbitrary initial state with $p_\theta(s_{1:T}[0]|y_{1:T}) > 0$ and let $s_{1:T}[k] \sim \mathcal{K}_\theta(\cdot|s_{1:T}[k-1]))$ for $k \geq 1$, then by the ergodic theorem (Robert and Casella 2004):

$$\frac{1}{n}\sum_{k=1}^{n} h(s_{1:T}[k]) \to \mathbb{E}_\theta\left[h(s_{1:T})|y_{1:T}\right], \tag{7}$$

as $n \to \infty$ for any function $h : \{1, ..., K\}^T \mapsto \mathbb{R}$. This allows a smoother to be constructed as in Algorithm 1.

We will use the *conditional particle filter with ancestor sampling* (CPF-AS, Lindsten, Jordan, et al. 2014) to construct the Markov kernel $\mathcal{K}_\theta$. The CPF-AS is similar to a standard particle filter, but with the important difference that one particle trajectory (jump sequence), $s'_{1:T}$, is specified *a priori*.

The algorithm statement for the CPF-AS can be found in, e.g., Lindsten, Jordan, et al. (2014). Similar to an auxiliary particle filter (Doucet and Johansen 2011), the propagation of $p_\theta(s_{1:t-1}|y_{1:t-1})$ (approximated by $\{s^i_{1:t-1}, w^i_{t-1}\}_{i=1}^N$) to time t is done using the *ancestor indices* $\{a^i_t\}_{i=1}^N$. To generate s^i_t, the ancestor index is sampled according to $\mathbb{P}\left(a^i_t = j\right) \propto w^j_{t-1}$, and s^i_t as $s^i_t \sim p_\theta(s_t|s^{a^i_t}_{t-1})$. The trajectories are then augmented as $s^i_{1:t} = \{s^{a^i_t}_{1:t-1}, s^i_t\}$.

This is repeated for $i = 1, \ldots, N-1$, whereas s^N_t is set as $s^N_t = s'_t$. To 'find' the history for s^N_t, the ancestor index a^N_t is drawn with probability

$$\mathbb{P}\left(a^N_t = i\right) \propto p_\theta(s^i_{1:t-1}|s'_{t:T}, y_{1:T}). \tag{8}$$

The probability density in (8) is proportional to

$$p_\theta(y_{t:T}, s'_{t:T}|s^i_{1:t-1}, y_{1:t-1})p_\theta(s^i_{1:t-1}|y_{1:t-1}), \tag{9}$$

where the last factor is the importance weight w^i_{t-1}.

By sampling $s_{1:T}[k+1] = s^J_{1:T}$ from the rendered set of trajectories $\{s^i_{1:T}, w^i_T\}_{i=1}^N$ with $\mathbb{P}(J=j) = w^j_T$, a Markov kernel $\mathcal{K}_\theta$ mapping $s_{1:T}[k] = s'_{1:T}$ to $s_{1:T}[k+1]$ is obtained. For this Markov kernel to be useful for statistical inference we require that *(i)* it is ergodic, and *(ii)* it admits $p_\theta(s_{1:T}|y_{1:T})$ as its unique limiting distribution. While we do not dwell on the (rather technical) details here, we note that these requirements are indeed fulfilled; see Lindsten, Jordan, et al. (2014).

3.3 Rao-Blackwellization

Rao-Blackwellization of particle filters is a fusion of the Kalman filter and the particle filter based on (6), and it is described in, e.g., Schön, Gustafsson, et al. (2005). However, Rao-Blackwellization of a particle smoother is somewhat more involved since the process $x_t|y_{1:T}$ is Markovian, but not $s_t|y_{1:T}$ (with z_t marginalized, see, e.g., Whiteley et al. (2010) and Lindsten and Schön (2013) for various ways to handle this).

A similar problem as for the particle smoothers arises in the ancestor sampling (8) in the CPF-AS. In the case of a non-Rao-Blackwellized CPF-AS, (8) reduces to $w^i_{t-1}p(x'_t|x^i_{t-1})$ (Lindsten, Jordan, et al. 2014). This does not hold in the Rao-Blackwellized case.

To handle this, (8) can be rewritten as

$$w^i_{t-1}p(y_{t:T}, s'_{t:T}|s^i_{1:t-1}, y_{1:t-1}). \tag{10}$$

Using the results from Section 4.4 in Lindsten and Schön (2013) (adapted to model (1)), this can be written (omitting w^i_{t-1}, and with the notation $\|z\|^2_\Omega \triangleq z^T\Omega z$, $P \triangleq \Gamma\Gamma^T$, i.e. the Cholesky factorization, $Q_t \triangleq F_tF_t^T$ and $A_t \triangleq A_{s'_t}$ etc.)

$$p(y_{t:T}, s'_{t:T}|s^i_{1:t-1}, y_{1:t-1}) \propto Z_{t-1}|\Lambda_{t-1}|^{-1/2}\exp(-\frac{1}{2}\eta_{t-1}),$$

with

$$\Lambda_t = \Gamma^{i,T}_{f;t}\Omega_t\Gamma^i_{f;t} + I, \tag{11a}$$

$$\eta_t = \|\widehat{z}^i_{f;t}\|^2_{\Omega_t} - 2\lambda_t^T\widehat{z}^{i,T}_{f;t} - \|\Gamma^n_{f;t}(\lambda_t - \Omega_t\widehat{z}^n_{f;t})\|^2_{M_t^{-1}}, \tag{11b}$$

where

$$\Omega_t = A^T_{t+1}\left(I - \widehat{\Omega}_{t+1}F_{t+1}M^{-1}_{t+1}F^T_{t+1}\right)\widehat{\Omega}_{t+1}A_{t+1}, \tag{11c}$$

$$\widehat{\Omega}_t = \Omega_t + C_t^TR_t^{-1}C_t,\ M_t = F_t^T\widehat{\Omega}F_t + I, \tag{11d}$$

$$\lambda_t = A^T_{t+1}\left(I - \widehat{\Omega}_{t+1}F_{t+1}M^{-1}_{t+1}F^T_{t+1}\right)m_t, \tag{11e}$$

$$\widehat{\lambda}_t = \lambda_t + C_t^TR_t^{-1}(y_t - D_tu_t), \tag{11f}$$

$$m_t = (\widehat{\lambda}_{t+1} - \widehat{\Omega}_{t+1}B_{t+1}u_{t+1}). \tag{11g}$$

Algorithm 2: Rao-Blackwellized CPF-AS

Input: $s'_{1:T} = s_{1:T}[k]$.

Output: $s_{1:T}[k+1]$ (A draw from $\mathcal{K}_\theta(\cdot|s_{1:T}[k])$ and $\{s^i_{1:T}, w^i_T\}_{i=1}^N$.

Draw $s^i_1 \sim p_1(s_1|y_1)$ for $i = 1, \ldots, N-1$.

Compute $\{\Omega_t, \lambda_t\}_{t=1}^T$ for $s'_{1:T}$ according to (11c) - (11g).

Set $(s^N_1, \ldots, s^N_T) = (s'_1, \ldots, s'_T)$.

Compute $\widehat{z}^i_{f,1}$ and $P^i_{f,1}$ for $i = 1, \ldots, N$.

Set $w^i_1 \propto p_\theta(y_1|s^i_1)$ (12) for $i = 1, \ldots, N$ s.t. $\sum_i w^i_1 = 1$.

for $t = 2$ **to** T **do**

  Draw a^i_t with $\mathbb{P}\left(a^i_t = j\right) = w^j_{t-1}$ for $i = 1, \ldots, N-1$.

  Draw s^i_t with $\mathbb{P}\left(s^i_t = n\right) = \pi_{s^i_{t-1},n}$ for $i = 1, \ldots, N-1$.

  Compute $\{\Lambda^i_{t-1}, \eta^i_t\}$ according to (11a)-(11b).

  Draw a^N_t with $\mathbb{P}\left(a^N_t = i\right) \propto w^i_{t-1}\pi_{s^i_{t-1},s^N_t}|\Lambda^i_{t-1}|^{-1/2}\exp(-\frac{1}{2}\eta^i_{t-1})$.

  Set $s^i_{1:t} = \{s^{a^i_t}_{1:t-1}, s^i_t\}$ for $i = 1, \ldots, N$.

  Set $\widehat{z}^i_{f,1:t-1} = \widehat{z}^{a^i_t}_{f,1:t-1}$, $P^i_{f,1:t-1} = P^{a^i_t}_{f,1:t-1}$, $\widehat{z}^i_{p,1:t-1} = \widehat{z}^{a^i_t}_{p,1:t-1}$ and $P^i_{p,1:t-1} = P^{a^i_t}_{p,1:t-1}$ for $i = 1, \ldots, N$.

  Compute $\widehat{z}^i_{p;t}$, $P^i_{p;t}$, $\widehat{z}^i_{f;t}$ and $P^i_{f;t}$ for $i = 1, \ldots, N$.

  Set $w^i_t \propto p_\theta(y_t|s^i_t, y_{1:t-1})$ for $i = 1, \ldots, N$ s.t. $\sum_i w^i_t = 1$.

end

for $t = T$ *to* 1 **do**

  Compute $\widehat{z}^i_{s;t}$, $P^i_{s;t}$ for $i = 1, \ldots, N$.

end

Set $s_{1:T}[k+1] = s^J_{1:T}$ with $\mathbb{P}(J = j) = w^j_T$.

and $\Omega_T = 0$ and $\lambda_T = 0$. The Rao-Blackwellization also includes an RTS smoother for finding $p_\theta(z_{1:T}|s_{1:T}, y_{1:T})$.

Summarizing the above development, the Rao-Blackwellized CPF-AS (for the jump Markov linear model (1)) is presented in Algorithm 2, where

$$p_\theta(y_t|s^i_{1:t}, y_{1:t-1}) = \mathcal{N}(y_t; C_{s^i_t}\widehat{z}^n_{p;t} + D_{s^i_t}u_t, C_{s^i_t}P_{p;t}C^T_{s^i_t} + R_{s^i_t})$$

is used. Note that the discrete state s_t is drawn from a discrete distribution defined by Π, whereas the linear state z_t is handled analytically. The algorithm implicitly defines a Markov kernel $\mathcal{K}_\theta$ that can be used in Algorithm 1 for finding $p(s_{1:T}|y_{1:T})$, or, as we will see, be placed in an SAEM framework to estimate θ (both yielding PMCMC constructions).

4 Identification of jump Markov linear models

In the previous section, an ergodic Markov kernel $\mathcal{K}_\theta$ leaving $p_\theta(s_{1:T}|y_{1:T})$ invariant was found as a Rao-Blackwellized CPF-AS summarized in Algorithm 2. This will be used together with SAEM, as it allows us to make one parameter update at each step

Algorithm 3: Rao-Blackwellized PSAEM

1 Initialize $\widehat{\theta}_0$ and $s_{1:T}[0]$, and $\widehat{Q}_0(\theta) \equiv 0$.
2 **for** $k \geq 1$ **do**
3 Run Algorithm 2 to obtain $\{s^i_{1:T}, w^i_T\}_{i=1}^N$
4 and $s_{1:T}[k]$.
5 Compute $\widehat{Q}_k(\theta)$ according to (12).
6 Compute $\widehat{\theta}_k = \arg\max_{\theta\in\Theta} \widehat{Q}_k(\theta)$.
7 **end**

of the Markov chain smoother in Algorithm 1, as presented as PSAEM in Lindsten (2013). (However, following Lindsten (2013), we make use of all the particles generated by CPF-AS, and not only $s_{1:T}[k+1]$, to compute the intermediate quantity in the SAEM.)

This leads to the approximation (cf. (5))

$$\widehat{Q}_k(\theta) = (1-\gamma_k)\widehat{Q}_{k-1}(\theta) + \gamma_k \sum_{i=1}^{N} w^i_T \mathbb{E}_{\theta_k}\left[\log p_\theta(y_{1:T}, z_{1:T}, s^i_{1:T}) | s^i_{1:T}, y_{1:T}\right], \qquad (12)$$

where the expectation is with respect to $z_{1:T}$. Putting this together, we obtain a Rao-Blackwellized PSAEM (RB-PSAEM) algorithm presented in Algorithm 3. Note that this algorithm is similar to the MCMC-based smoother in Algorithm 1, but with the difference that the model parameters are updated at each iteration, effectively enabling simultaneous smoothing and identification.

(For notational convenience, the iteration number k is suppressed in the variables related to $\{s^i_{1:T}, w^i_T\}_{i=1}^N$.)

With a strong theoretical foundation in PMCMC and Markovian stochastic approximation, the RB-PSAEM algorithm presented here enjoys very favourable convergence properties. In particular, under certain smoothness and ergodicity conditions, the sequence of iterates $\{\theta_k\}_{k\geq 1}$ will converge to a maximizer of $p_\theta(y_{1:T})$ as $k \to \infty$, regardless of the number of particles $N \geq 2$ used in the internal CPF-AS procedure (see Proposition 1 of Lindsten 2013 together with Kuhn and Lavielle 2004 for details). Furthermore, empirically it has been found that a small number of particles can work well in practice as well. For instance, in the numerical examples considered in Section 5, we run Algorithm 3 with $N = 3$ with accurate identification results.

For the model structure (1), there exists infinitely many solutions to the problem (2); all relevant involved matrices can be transformed by a linear transformation matrix and the modes can be re-ordered, but the input-output behaviour will remain invariant. The model is therefore over-parametrized, or lacks identifiability, in the general problem setting. However, it is shown in Pintelon et al. (1996) that the Cramér-Rao Lower Bound is not affected by the over-parametrization. That is, the estimate quality, in terms of variance, is unaffected by the over-parametrization.

4.1 Maximizing the intermediate quantity

When making use of RB-PSAEM from Algorithm 3, one major question arises from Step 6, namely the maximization of the intermediate quantity $\widehat{Q}_k(\theta)$. For the jump Markov linear model, the expectation in (12) can be expressed using sufficient statistics, as will be shown later, as an inner product

$$\sum_{i=1}^{N} w_T^i \mathbb{E}_{\theta_k}\left[\log p_\theta(y_{1:T}, z_{1:T}, s_{1:T}^i)|s_{1:T}^i, y_{1:T}\right] = \langle S^k, \eta(\theta)\rangle, \tag{13}$$

for a sufficient statistics S and corresponding natural parameter $\eta(\theta)$. Hence $\widehat{Q}_k$ can be written as

$$\widehat{Q}_k(\theta) = (1-\gamma_k)\widehat{Q}_{k-1}(\theta) + \gamma_k\langle S^k, \eta(\theta)\rangle = \langle \mathbb{S}^k, \eta(\theta)\rangle \tag{14}$$

if the transformation

$$\mathbb{S}^k = (1-\gamma_k)\mathbb{S}^{k-1} + \gamma_k S^k \tag{15}$$

is used. In detail,

$$\sum_{i=1}^{N} w_T^i \mathbb{E}_{\theta_k}\left[\log p_\theta(y_{1:T}, z_{1:T}, s_{1:T}^i)|s_{1:T}^i, y_{1:T}\right] =$$

$$\sum_{n=1}^{K}\sum_{m=1}^{K} S_{n,m}^{(1)} \log \pi_{n,m} - \sum_{n=1}^{K} \frac{1}{2}\left(S_n^{(2)} \log(|Q_n||R_n|) + \mathrm{Tr}(H_n^\theta S_n^{(3)})\right) \tag{16a}$$

neglecting constant terms in the last expression. This can be verified to be an inner product (as indicated in (13)) in $S = \{S^{(1)}, S^{(2)}, S^{(3)}\}$. Here the sufficient statistics

$$S_{n,m}^{(1)} = \sum_{i=1}^{N} w_T^i \sum_{t=1}^{T} \mathbb{I}_{s_t^i=m, s_{t-1}^i=n}, \tag{16b}$$

$$S_n^{(2)} = \sum_{i=1}^{N} w_T^i \sum_{t=1}^{T} \mathbb{I}_{s_t^i=n}, \tag{16c}$$

$$S_n^{(3)} = \sum_{i=1}^{N} w_T^i \sum_{t=1}^{T} \mathbb{I}_{s_t^i=n}(\widehat{\xi}_t^i \widehat{\xi}_t^{i,T} + M_{t|T}^i), \tag{16d}$$

with

$$\widehat{\xi}_t^i = \left(\widehat{z}_{s;t}^{i,T} \begin{bmatrix}\widehat{z}_{s;t-1}^{i,T} & u_{t-1}^T\end{bmatrix} y_t^T \begin{bmatrix}\widehat{z}_{s;t}^{i,T} & u_t^T\end{bmatrix}\right)^T, \tag{16e}$$

and

$$H_n^\theta = \begin{pmatrix} \begin{bmatrix} I & A_n^T & B_n^T\end{bmatrix} Q_n^{-1} \begin{bmatrix} I \\ A_n \\ B_n\end{bmatrix} & 0 \\ 0 & \begin{bmatrix} I & C_n^T & D_n^T\end{bmatrix} R_n^{-1} \begin{bmatrix} I \\ C_n \\ D_n\end{bmatrix} \end{pmatrix} \tag{16f}$$

have been used. Further notation introduced is $\mathbb{I}_{\cdot}$ as the indicator function, and

$$M^i_{t|T} = \begin{pmatrix} P^i_{s;t} & P^i_{s;t,t-1} & 0 & 0 & P^i_{s;t} & 0 \\ P^i_{s;t,t-1} & P^i_{s;t-1} & 0 & 0 & P^i_{s;t,t-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ P^i_{s;t} & P^i_{s;t,t-1} & 0 & 0 & P^i_{s;t-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{16g}$$

For computing this, the RTS-smoother in step 17 in Algorithm 2 has to be extended by calculation of $P_{s;t+1,t} \triangleq \operatorname{cov}\left[\widehat{z}_{s,t+1}\widehat{z}^T_{s;t}\right]$, which can be done as follows (Shumway and Stoffer 2006, Property P6.2)

$$P_{s;t,t-1} = P_{f;t}J^T_{t-1} + J_t(P_{s;t+1,t} - A_{t+1}P_{f;t})J^T_{t-1}, \tag{17}$$

initialized with $P_{T,T-1|T} = (I - K_T C_T)A_T P_{f;t-1}$.

For notational convenience, we will partition $S^{(3)}_n$ as

$$S^{(3)}_n = \begin{pmatrix} \Phi_n & \Psi_n & & \\ \Psi^T_n & \Sigma_n & & \\ & & \Omega_n & \Lambda_n \\ & & \Lambda^T_n & \Xi_n \end{pmatrix}. \tag{18}$$

Lemma 1. *Assume for all modes $n = 1, \ldots, K$, that all states z are controllable and observable and $\sum_t \mathbb{I}_{s_t=n} u^T_t u_t > 0$. The parameters θ maximizing $\widehat{Q}_k(\theta)$ for the jump Markov linear model* (1) *are then given by*

$$\pi^j_{n,m} = \frac{\mathbb{S}^{(1),k}_{n,m}}{\sum_l \mathbb{S}^{(1),k}_{n,l}}, \tag{19a}$$

$$\begin{bmatrix} A_n & B_n \end{bmatrix} = \Psi_n \Sigma^{-1}_n, \tag{19b}$$

$$\begin{bmatrix} C_n & D_n \end{bmatrix} = \Lambda_n \Xi^{-1}_n, \tag{19c}$$

$$\begin{bmatrix} Q_n \end{bmatrix} = (\mathbb{S}^{(2),k}_n)^{-1}\left(\Phi_n - \Psi_n \Sigma^{-1}_n \Psi^T_n\right), \tag{19d}$$

$$\begin{bmatrix} R_n \end{bmatrix} = (\mathbb{S}^{(2),k}_n)^{-1}\left(\Omega_n - \Lambda_n \Xi^{-1}_n \Lambda^T_n\right), \tag{19e}$$

for $n, m = 1, \ldots, K$.

$\Phi_n, \Psi_n, \ldots$ are the partitions of $\mathbb{S}^{(3),k}_n$ indicated in (18), and $\mathbb{S}^{(i)}$ are the 'SA-updates' (15) of the sufficient statistics (16b)-(16d).

Remark: If $B \equiv 0$, the first square bracket in (16e) can be replaced by $\left[\widehat{z}^{i,T}_{s;t-1}\right]$, and (19b) becomes $\begin{bmatrix} A_n \end{bmatrix} = \Psi_n \Sigma^{-1}_n$. The case with $D \equiv 0$ is fully analogous.

Proof. With arguments directly from Gibson and Ninness (2005, Lemma 3.3), the maximization of the last part of (16a) for a given $s_t = n$ (for any sufficient statistics Z in the inner product, and in particular $Z = \mathbb{S}^k$), is found to be (19b)-(19e).

Using Lagrange multipliers and that $\sum_i \pi_{n,m} = 1$, the maximum w.r.t. Π of the first part of (16a) is obtained as

$$\pi_{n,m} = \frac{\mathbb{S}^{(1),k}_{n,m}}{\sum_l \mathbb{S}^{(1),k}_{n,l}}. \tag{20}$$

□

4.2 Computational complexity

Regarding the computational complexity of Algorithm 3, the most important result is that it is linear in the number of measurements T. It is also linear in the number of particles N.

5 Numerical examples

Some numerical examples are given to illustrate the properties of the Rao-Blackwellized PSAEM algorithm. The Matlab code for the examples is available via the homepage of the first author[1].

5.1 Example 1 - Comparison to related methods

The first example concerns identification using simulated data ($T = 3\,000$) for a one-dimensional ($n_z = 1$) jump Markov linear model with 2 modes ($K = 2$) (with parameters randomly generated according to $A_n \sim U_{[-1,1]}$, $B_n \sim U_{[-5,5]}$, $C_n \sim U_{[-5,5]}$, $D_n \equiv 0, Q_n \sim U_{[0.01,0.1]}, R_n \sim U_{[0.01,0.1]}$) with low-pass filtered white noise as u_t. The following methods are compared:

1. RB-PSAEM from Algorithm 3, with (only) $N = 3$ particles,
2. PSAEM as presented in Lindsten (2013) with $N = 20$,
3. PSEM (Schön, Wills, et al. 2011) with $N = 100$ forward particles and $M = 20$ backward simulated trajectories.

The initial parameters $\widehat{\theta}_0$ are each randomly picked from $[0.5\theta^\star, 1.5\theta^\star]$, where $\theta^\star$ is the true parameter value. The results are illustrated in Figure 1, which shows the mean (over all modes and 7 runs) $\mathcal{H}_2$ error for the transfer function from the input u to the output y.

From Figure 1 (note the log-log scale used in the plot) it is clear that our new Rao-Blackwellized PSAEM algorithm has a significantly better performance, both in terms of mean and in variance between different runs, compared to the previous algorithms.

5.2 Example 2 - Identification of multidimensional systems

Let us now consider a two-dimensional system ($n_z = 2$) with $K = 3$ modes. The eigenvalues for A_n are randomly picked from $[-1, 1]$. The other parameters are randomly picked as $B_n \sim U_{[-5,5]}$, $C_n \sim U_{[-5,5]}$, $D_n \equiv 0, Q_n \sim I_2 \cdot U_{[0.01,0.1]}, R_n \sim U_{[0.01,0.1]}$, and the system is simulated for $T = 8\,000$ time steps with input u_t being a low-pass filtered white noise. The initialization of the Rao-Blackwellized PSAEM algorithm is randomly picked from $[0.6\theta^\star, 1.4\theta^\star]$ for each parameter. The number of particles used in the particle filter is $N = 3$. Figure 2a shows the mean (over 10 runs) $\mathcal{H}_2$ error for each mode, similar to Figure 1. Figure 2b shows the estimated Bode plots after 300 iterations. As is seen from Figure 2b, the RB-PSAEM algorithm has the ability to catch the dynamics of the multidimensional system fairly well.

[1] http://www.it.uu.se/katalog/andsv164

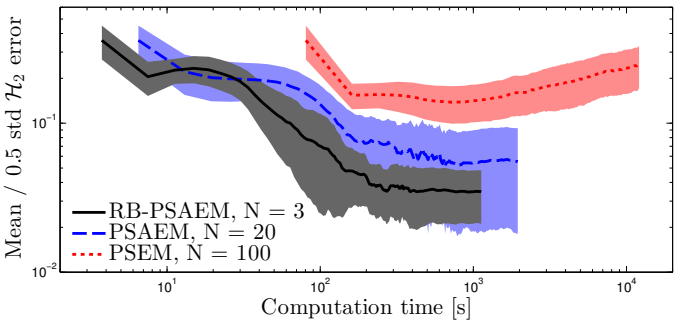
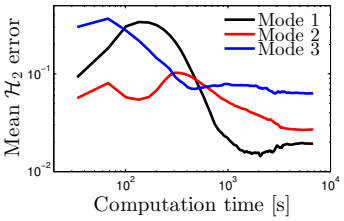
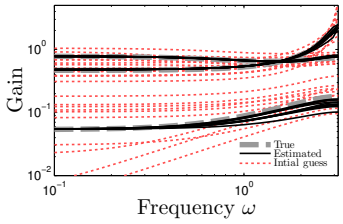


Figure 1: *Numerical example 1. Mean (lines) and 0.5 standard deviation (fields) $\mathcal{H}_2$ error for 7 runs of our RB-PSAEM using $N = 3$ particles (black) PSAEM (Lindsten 2013) using $N = 20$ particles (blue) and PSEM (Schön, Wills, et al. 2011) using $N = 100$ particles and $M = 20$ backward trajectories (red).*



(a) *Mean $\mathcal{H}_2$ error for each mode.*



(b) *Bode plots of the estimates (black), true (dashed grey) and the initializations (dotted red).*

Figure 2: *Plots from Numerical example 2.*

6 Conclusions and future work

We have derived a maximum likelihood estimator for identification of jump Markov linear models. More specifically an expectation maximization type of solution was derived. The nonlinear state smoothing problem inherent in the expectation step was solved by constructing an ergodic Markov kernel leaving the joint state smoothing distribution invariant. Key to this development was the introduction of a Rao-Blackwellized conditional particle filter with ancestor sampling. The maximization step could be solved in closed form. The experimental results indicate that we obtain significantly better performance both in terms of accuracy and computational time when compared to previous state of the art particle filtering based methods. The ideas underlying the smoother derived in this work have great potential also outside the class of jump Markov linear models and this is something worth more investigation. Indeed, it is quite possible that it can turn out to be a serious competitor also in finding the joint smoothing distribution for general nonlinear state space models.

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

Christophe Andrieu, Éric Moulines, and Pierre Priouret (2005). "Stability of stochastic approximation under verifiable conditions". In: *SIAM Journal on control and optimization* 44.1, pp. 283–312.

Trevor T. Ashley and Sean B. Andersson (2014). "A sequential Monte Carlo framework for the system identification of jump Markov state space models". In: *Proceedings of the 2014 American Control Conference (ACC)*. Portland, OR, USA, pp. 1144–1149.

Alberto Bemporad, Jacob Roll, and Lennart Ljung (2001). "Identification of hybrid systems via mixed-integer programming". In: *Proceedings of the 40th IEEE Conference on Decision and Control (CDC)*. Orlando, FL, USA, pp. 786–792.

Lars Blackmore, Stephanie Gil, Seung Chung, and Brian Williams (2007). "Model learning for switching linear systems with autonomous mode transitions". In: *Proceedings of the 46th IEEE Conference on Decision and Control (CDC)*. New Orleans, LA, USA, pp. 4648–4655.

José Borges, Vincent Verdult, Michel Verhaegen, and Miguel Ayala Botto (2005). "A switching detection method based on projected subspace classification". In: *Proceedings of the 44th IEEE Conference on Decision and Control (CDC)*. Sevilla, Spain, pp. 344–349.

Olivier Cappé, Éric Moulines, and Tobias Rydén (2005). *Inference in hidden Markov models*. Springer Series in Statistics. New York, NY, USA: Springer.

George Casella and Christian P. Robert (1996). "Rao-Blackwellisation of sampling schemes". In: *Biometrika* 83.1, pp. 81–94.

Bernard Delyon, Marc Lavielle, and Éric Moulines (1999). "Convergence of a stochastic approximation version of the EM algorithm". In: *Annals of Statistics* 27.1, pp. 94–128.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–38.

Arnaud Doucet and Adam M. Johansen (2011). "A tutorial on particle filtering and smoothing: fifteen years later". In: *Nonlinear Filtering Handbook*. Ed. by D. Crisan and B. Rozovsky. Oxford, UK: Oxford University Press, pp. 656–704.

Emily Fox, Erik B. Sudderth, Michael I. Jordan, and Alan Willsky (2011). "Bayesian nonparametric inference of switching dynamic linear models". In: *IEEE Transactions of Signal Processing* 59.4, pp. 1569–1585.

Andrea Garulli, Simone Paoletti, and Antonio Vicino (2012). "A survey on switched and piecewise affine system identification". In: *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*. Brussels, Belgium, pp. 344–355.

Stuart Gibson and Brett Ninness (2005). "Robust maximum-likelihood estimation of multivariable dynamic systems". In: *Automatica* 41.10, pp. 1667–1682.

Stephanie Gil and Brian Williams (2009). "Beyond local optimality: an improved approach to hybrid model learning". In: *Proceedings of the 48th IEEE Conference on Decision and Control (CDC)*. Shanghai, China, pp. 3938–3945.

Thomas Kailath, Ali H. Sayed, and Babak Hassibi (2000). *Linear estimation*. Upper Saddle River, NJ, USA: Prentice Hall.

Estelle Kuhn and Marc Lavielle (2004). "Coupling a stochastic approximation version of EM with an MCMC procedure". In: *ESAIM: Probability and Statistics* 8, pp. 115–131.

Fredrik Lindsten (2013). "An efficient stochastic approximation EM algorithm using conditional particle filters". In: *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vancouver, BC, Canada, pp. 6274–6278.

Fredrik Lindsten, Michael I. Jordan, and Thomas B. Schön (2014). "Particle Gibbs with ancestor sampling". In: *The Journal of Machine Learning Research (JMLR)* 15.1, pp. 2145–2184.

Fredrik Lindsten and Thomas B. Schön (2013). "Backward simulation methods for Monte Carlo statistical inference". In: *Foundations and Trends in Machine Learning* 6.1, pp. 1–143.

Jimmy Olsson, Olivier Cappé, Randal Douc, and Éric Moulines (2008). "Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state-space models". In: *Bernoulli* 14.1, pp. 155–179.

Simone Paoletti, Aleksandar Lj. Juloski, Giancarlo Ferrari-Trecate, and René Vidal (2007). "Identification of hybrid systems: a tutorial". In: *European Journal of Control* 13.2, pp. 242–260.

Komi Midzodzi Pekpe, Gilles Mourot, Komi Gasso, and José Ragot (2004). "Identification of switching systems using change detection technique in the subspace framework". In: *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*. Vol. 4. Paradise Island, Bahamas, pp. 3838–3843.

Rik Pintelon, Joannes Schoukens, Tomas McKelvey, and Yves Rolain (1996). "Minimum variance bounds for overparameterized models". In: *IEEE Transactions on Automatic Control* 41.5, pp. 719–720.

Herbert E. Rauch, Frank F. Tung, and Charlotte T. Striebel (1965). "Maximum likelihood estimates of linear dynamic systems". In: *AIAA journal* 3.8, pp. 1445–1450.

Herbert Robbins and Sutton Monro (1951). "A stochastic approximation method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407.

Christian P. Robert and George Casella (2004). *Monte Carlo statistical methods*. 2nd ed. New York, NY, USA: Springer.

Thomas B. Schön, Fredrik Gustafsson, and Per-Johan Nordlund (2005). "Marginalized particle filters for mixed linear/nonlinear state-space models". In: *IEEE Transactions on Signal Processing* 53.7, pp. 2279–2289.

Thomas B. Schön, Adrian Wills, and Brett Ninness (2011). "System identification of nonlinear state-space models". In: *Automatica* 47.1, pp. 39–49.

Robert H. Shumway and David S. Stoffer (1982). "An approach to time series smoothing and forecasting using the EM algorithm". In: *Journal of Time Series Analysis* 3.4, pp. 253–264.

Robert H. Shumway and David S. Stoffer (2006). *Time series analysis and its applications: with R examples*. 2nd ed. New York, NY, USA: Springer.

Greg C.G. Wei and Martin A. Tanner (1990). "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms". In: *Journal of the American Statistical Association* 85.411, pp. 699–704.

Nick Whiteley, Christophe Andrieu, and Arnaud Doucet (2010). "Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods". In: *arXiv:1011.2437*.

Sinan Yildirim, Sumeetpal S. Singh, and Arnaud Doucet (2013). "An online expectation–maximization algorithm for changepoint models". In: *Journal of Computational and Graphical Statistics* 22.4, pp. 906–926.

Paper V

Title

Learning of state-space models with highly informative observations: a tempered Sequential Monte Carlo solution

Authors

Andreas Svensson, Thomas B. Schön and Fredrik Lindsten

Edited version of

Andreas Svensson, Thomas B. Schön, and Fredrik Lindsten (2018). "Learning of state-space models with highly informative observations: a tempered Sequential Monte Carlo solution". In: *Mechanical Systems and Signal Processing* 104, pp. 915–928.

Digital identity

doi:10.1016/j.ymssp.2017.09.016

Financial support

The Swedish Research Council via the projects *Probabilistic modeling of dynamical systems* (contract number: 621-2013-5524), *Learning of Large-Scale Probabilistic Dynamical Models* (contract number: 2016-04278), and *NewLEADS - New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079) and the Swedish Foundation for Strategic Research (SSF) via the project *ASSEMBLE* (contract number: RIT15-0012).

Learning of state-space models with highly informative observations: a tempered Sequential Monte Carlo solution

Abstract

Probabilistic (or Bayesian) modeling and learning offers interesting possibilities for systematic representation of uncertainty using probability theory. However, probabilistic learning often leads to computationally challenging problems. Some problems of this type that were previously intractable can now be solved on standard personal computers thanks to recent advances in Monte Carlo methods. In particular, for learning of unknown parameters in nonlinear state-space models, methods based on the particle filter (a Monte Carlo method) have proven very useful. A notoriously challenging problem, however, still occurs when the observations in the state-space model are highly informative, i.e. when there is very little or no measurement noise present, relative to the amount of process noise. The particle filter will then struggle in estimating one of the basic components for probabilistic learning, namely the likelihood $p(\text{data}|\text{parameters})$. To this end we suggest an algorithm which initially assumes that there is substantial amount of artificial measurement noise present. The variance of this noise is sequentially decreased in an adaptive fashion such that we, in the end, recover the original problem or possibly a very close approximation of it. The main component in our algorithm is a sequential Monte Carlo (SMC) sampler, which gives our proposed method a clear resemblance to the SMC^2 method. Another natural link is also made to the ideas underlying the approximate Bayesian computation (ABC). We illustrate it with numerical examples, and in particular show promising results for a challenging Wiener-Hammerstein benchmark problem.

1 Introduction

Probabilistic (or Bayesian) modeling and learning offers interesting and promising possibilities for a coherent and systematic description of model and parameter *uncertainty* based on probability theory (Peterka 1981; Robert 2001). The computational tools for probabilistic learning in state-space models have lately been developed. In this paper, we study probabilistic learning based on measured data $\{y_1, \ldots, y_T\} \triangleq y_{1:T}$, which we assume to be well described by a nonlinear state-space model with (almost) no

measurement noise,

$$x_t \mid (x_{1:t-1}, \theta) \sim f(x_t \mid x_{t-1}, u_{t-1}, \theta), \tag{1a}$$

$$y_t = g(x_t), \tag{1b}$$

with some unknown parameters $\theta \in \Theta$ which we want to learn. The lack of measurement noise in (1b) gives a deterministic mapping $g : \mathsf{X} \mapsto \mathsf{Y}$ from the unobserved states $x_t \in \mathsf{X}$ to the measurement $y_t \in \mathsf{Y}$, on the contrary to (1a) which encodes uncertainty about x_t, mathematically represented as a probability density f over x_t conditional on x_{t-1} and possibly an exogenous input u_{t-1}. We refer to this uncertainty as *process noise*, but its origin does not have to be a physical noise, but possibly originating from lack of information or model errors. The reasoning and contributions of this paper will be applicable also to the case where the relationship (1b) does contain uncertainty, *measurement noise*, but its variance is much smaller than the process noise. As a general term, we refer to the model as having *highly informative observations*. Furthermore, g could also be allowed to depend on θ and u_t, but we omit that possibility for notational clarity.

Models on the form (1) may arise in several practical situations, for instance in a mechanical system where the measurements can be made with good precision but some unobserved forces are acting on the system. The situation may also appear if the measurements, yet again, can be made with good precision, but the user's understanding of the physical system is limited, which in the probabilistic framework can be modeled as a stochastic element in f.

The model (1) defines, together with priors on θ, a joint probabilistic model $p(y_{1:T}, x_{1:T}, \theta)$. Probabilistic learning of the parameters θ amounts to computing the parameter posterior $p(\theta \mid y_{1:T})$, where we have conditioned on data $y_{1:T}$ and marginalized over all possible states $x_{1:T}$ (we omit the known $u_{1:T}$ to ease the notation). Although conceptually clear, the computations needed are typically challenging, and almost no cases exist that admit closed-form expressions for $p(\theta \mid y_{1:T})$.

For probabilistic learning, Monte Carlo methods have proven useful, as outlined in the accompanying paper Schön, Svensson, et al. (2018). The idea underlying these Monte Carlo methods is to represent the distributions of interest, such as the posterior $p(\theta \mid y_{1:T})$, with samples. The samples can later be used to estimate functions of the parameters θ, such as their mean, variance, etc., as well as making predictions of future outputs y_{T+1}, etc. For state-space models, the particle filter is a tailored algorithm for handling the unknown states x_t, and in particular to compute an unbiased estimate z of the likelihood

$$p(y_{1:T} \mid \theta) = \int p(y_{1:T}, x_{1:T} \mid \theta) dx_{1:T}, \tag{2}$$

which is a central object in probabilistic learning, see the accompanying paper Schön, Svensson, et al. (2018) for a more thorough introduction (or, e.g., Kantas et al. 2015; Schön, Lindsten, et al. 2015). The peculiarity in the problem studied in this paper is the (relative) absence of measurement noise in (1) compared to the process noise level. This seemingly innocent detail is, as we will detail in Section 2.2, a show-stopper for the standard algorithms based on the particle filter, since the quality of the likelihood estimate z tends to be very poor if the model has highly informative observations.

The problem with highly informative observations has a connection to the literature on approximate Bayesian computations (ABC, Beaumont et al. 2002), where

some observations y are available, as well as a model (not necessarily a state-space model) with some unknown parameters θ. In ABC problems, however, the model is only capable of *simulating* new synthetic observations $\widehat{y}(\theta)$ and the likelihood $p(y \mid \theta)$ cannot be evaluated. The ABC idea is to construct a distance metric between the real observations y and the simulated synthetic observations $\widehat{y}(\theta)$, and take this distance (which becomes a function of y and θ) as a substitute for $p(y \mid \theta)$. The accuracy of the approximation is controlled by the metric with higher accuracy corresponding to more informative observations, providing a clear link to the present work.

We propose in this paper a novel algorithm for the purpose of learning θ in (1). Our idea is to start the algorithm by assuming that there *is* a substantial amount of measurement noise which mitigates the computational problems, and then gradually decrease this artificial measurement noise variance simultaneously as the parameters θ are learned. The assumption of artificial measurement noise resembles the ABC methodology. The sequence of gradually decreasing measurement noise variance can be seen as tempering, which we will combine with a sequential Monte Carlo (SMC) sampler (Del Moral, Doucet, et al. 2006) to obtain a theoretically sound algorithm which generates samples from the posterior $p(\theta \mid y_{1:T})$.

In a sense, our proposed algorithm is a combination of the work by (Dean et al. 2015) on ABC for state-space models and the use of SMC samplers for ABC by Del Moral, Doucet, et al. (2012), resulting in a SMC^2-like algorithm (Chopin et al. 2013).

2 Background on particle filtering and tempering

In this section we will provide some background on particle filters, Markov chain Monte Carlo (MCMC) and related methods. For a more elaborate introduction, please refer to, e.g., Dahlin and Schön (2016), Robert and Casella (2004), and Schön, Svensson, et al. (2018). We will in particular discuss why models on the form (1) are problematic for most existing methods, and also introduce the notion of tempering.

2.1 Particle filtering, PMCMC and SMC^2

The bootstrap particle filter was presented in the early 1990's (Doucet and Johansen 2011; Gordon et al. 1993) as a solution to the state filtering problem (computing $p(x_t \mid y_{1:t})$) in nonlinear state-space models. The idea is to propagate a set of N_x Monte Carlo samples $\{x_t^n\}_{n=1}^{N_x}$ along the time dimension $t = 1, 2, \ldots, T$, and for each t the algorithm follows a 3-stage scheme with resampling (sampling ancestor indices a_t^n based on weights w_{t-1}^n), propagation (sampling x_t^n from $x_{t-1}^{a_t^n}$ using (1a)) and weighting (evaluate the 'usefulness' of x_t^n using (1b) and store it as the weight w_t^n). This algorithm will be given as Algorithm 2, and a more elaborate introduction can be found in Schön, Svensson, et al. (2018). The samples are often referred to as particles, and provide an empirical approximation $\widehat{p}(x_t \mid y_{1:t}) = \frac{1}{N_x} \sum_{n=1}^{N_x} \delta_{x_t^n}(x_t)$ (with δ the Dirac measure) of the filtering distribution $p(x_t \mid y_{1:t})$. Since the particle filter itself builds on Monte Carlo ideas, the outcome of the algorithm will be different every time the algorithm is run.

The particle filter itself is only applicable when the state-space model does not contain any unknown parameters. It has, however, been realized that the particle filter does

not only solve the filtering problem, but it can also be used to estimate the likelihood $p(y_{1:T} \mid \theta)$ of a state-space model by using the empirical approximation $\widehat{p}(x_t \mid y_{1:t})$ in (2) and hence approximate the integral with a sum. We will denote the obtained estimate with z, and it can be shown (Del Moral 2004) that z is in fact an unbiased estimator of the likelihood, $\mathbb{E}[z - p(y_{1:T} \mid \theta)] = \mathbb{E}[z] - p(y_{1:T} \mid \theta) = 0$. That is, unbiased means that the average of the estimate z (if the particle filter algorithm is run many times for the same model, the same parameters and the same data) will be close to the true (but intractable) $p(y_{1:T} \mid \theta)$.

The use of the particle filter as an estimator of the likelihood has opened up possibilities for combining it with another branch of Monte Carlo methods, namely Markov chain Monte Carlo (MCMC). This combination allows for inferring not only unobserved states x_t but *also* unknown parameters θ in nonlinear state-space models. One such successful idea is to construct a high-level MCMC procedure concerned with θ, and then run the particle filter to estimate the likelihood for different θ. The high-level procedure can be a Metropolis-Hastings algorithm (Metropolis et al. 1953, Schön, Svensson, et al. 2018, Section 5), essentially an informed random walk in Θ. The Metropolis-Hastings algorithm is constructed such that after sufficiently long time, the trace of the 'walk' (the 'chain') in Θ will be samples from the distribution we are interested in, $p(\theta \mid y_{1:T})$.

The original Metropolis-Hastings algorithm assumes that the target distribution can be evaluated exactly. In our state-space learning problem it would mean that the stochastic estimate z from the particle filter would not be sufficient for a valid Metropolis-Hastings algorithm. However, it has lately been shown (Andrieu and Roberts 2009) that valid algorithms can be constructed based also on stochastic estimates with certain properties, which provides the ground for the particle (marginal) Metropolis-Hastings (PMH, or PMMH, Andrieu, Doucet, et al. 2010) algorithm. We will not go into further details here, but refer to Schön, Svensson, et al. (2018).

An alternative partner for the particle filter, instead of MCMC, is SMC. Interestingly enough, SMC is a family of methods that has been developed as a generalization of the particle filter. One SMC method is the SMC sampler (Del Moral, Doucet, et al. 2006), which can be employed to handle the unknown θ instead of Metropolis-Hastings. The SMC sampler will then query the particle filter for likelihood estimates z for different values of θ. The SMC sampler itself is similar to a particle filter, propagating its N_θ samples $\{\theta^{(j)}\}_{j=1}^{N_\theta}$ through a sequence of distributions ending up in the posterior $p(\theta \mid y)$. The sequence through which the samples of θ are propagated can be a so-called tempering sequence. With a certain choice of tempering sequence, the nested construction of the particle filter and SMC sampler has been termed SMC^2 (Chopin et al. 2013). The method that we propose in this paper bears close resemblance to SMC^2, but makes use of a different tempering sequence.

2.2 Challenges with highly informative observations

The bootstrap particle filter is often used to provide estimates z of the likelihood $p(y_{1:T} \mid \theta)$ in probabilistic learning methods. However, when there is (almost) no measurement noise relative to the amount of process noise, and thus highly informative observations, these estimates become poor due to the importance sampling mechanism inherent in the particle filter. In the bootstrap particle filter, N_x particles

$\{x_t^n\}_{n=1}^{N_x}$ are drawn from (1a), and then weighted by evaluating (1b). As long as there is at least one particle x_t^n wich gives a reasonably high probability for the measurement y_t (and consequently gets assigned a large weight), the particle filter will provide a reasonable result, and the more such high-weight particles, the better (in terms of variance of the estimate z). However, if no samples x_t are drawn under which y_t could have been observed with reasonably high probability, the estimate z will be very poor. If there is very little measurement noise but non-negligible process noise in the model, the chance of drawing any useful particles x_t^n by simulating the system dynamics is typically small. The problem may become even more articulated if the bootstrap particle filter is run with a parameter θ which does not explain the measurements $y_{1:T}$ well. The bottom line is that a model with highly informative observations causes the bootstrap particle filter to provide estimates z with high variance. This is in particular true for values of θ that do not explain the measurements well. Considering that high variance of z implies bad performance in the high-level MCMC or SMC sampler for θ, the model (1) is problematic to learn.

To this end, research has been done on how to improve the particle filter by drawing particles $\{x_t^n\}_{n=1}^{N_x}$ not from (1a) but instead from a tailored proposal which also depend on y_t, in order to better adapt to the measurement y_t and make more 'well-informed' particle draws. In the interest of a maintained consistency, the weight update is modified accordingly. Such adaptation is not always simple, but proposed methods include the fully adapted auxiliary particle filter (Pitt and Shephard 1999) (only possible for a limited set of model structures), the alive particle filter (Del Moral, Jasra, et al. 2015) and the bridging particle filter (Del Moral and Murray 2015) (both computationally more costly). In this work, we will not focus on this aspect, but rather on how inference about θ can be constructed in order to (as far as possible) avoid running the particle filter for models with highly informative observations. Ultimately, our suggested approach could be combined with methods like the fully adapted, alive or bridging particle filter to push the limits even further.

2.3 Tempering

To construct inference algorithms, the computational trick of tempering (or annealing) has proven useful. The name tempering was originally used for a certain heat treatment method within metallurgy, but the term is also used in a figurative sense for a set of computational methods. The idea is to construct a 'smooth' sequence $\{\pi_p(\theta)\}_{p=0}^{P}$ starting in a user-chosen initial function $\pi_0(\theta)$ and ending in the target function $\pi_P(\theta)$. In our case, these functions are probability densities, and our target is $\pi_P(\theta) = p(\theta \mid y_{1:T})$, as illustrated in Figure 1a. There are several ways in which such a sequence can be constructed. We do not have a formal definition of 'smooth', but understand it as a sequence where every adjacent pair $\{\pi_p(\theta), \pi_{p+1}(\theta)\}$ are similar in, e.g., total variation sense. By tracking the evolution from the typically simple and unimodal $\pi_0(\theta)$ to the potentially intricate and multimodal target $\pi_P(\theta)$, the risk of getting stuck in zero-gradient regions or in local optima is reduced, compared to standard methods starting directly in $\pi_P(\theta) = p(\theta \mid y_{1:T})$.

For state-space models, there are several generic choices for constructing tempering sequences ending up in a posterior $p(\theta \mid y_{1:T})$. One choice (with $P = T$) is the data-tempered sequence $\pi_p(\theta) = p(\theta \mid y_{1:p})$, which gives a sequence starting in

the prior $p(\theta)$ and, by sequentially including one additional measurement y_p, eventually ending up in the posterior $p(\theta \mid y_{1:T})$. Typically, the landscape of $p(\theta \mid y_{1:p})$ does not change dramatically when including one extra measurement, and smoothness of the sequence is thus ensured. Another choice is found by first noting that $p(\theta \mid y_{1:T}) \propto p(y_{1:T} \mid \theta)p(\theta)$, and then making the choice $\pi_p(\theta) \propto p(y_{1:T} \mid \theta)^{p/P}p(\theta)$. Such a sequence also starts, with $p = 0$, in the prior $p(\theta)$ and ends, with $p = P$, in the posterior $p(\theta \mid y_{1:T})$. We will in this paper, Section 3.1, introduce a new tempering sequence that is tailored for state-space models with highly informative observations, inspired by the ABC approach.

2.4 Using a tempering sequence in an SMC sampler

A tempering sequence $\{\pi_p(\theta)\}_{p=0}^P$ can be used in an SMC sampler to produce samples from $\pi_P(\theta) = p(\theta \mid y_{1:T})$. The idea underlying the SMC sampler is to propagate a set of N_θ samples $\{\theta^j\}_{j=1}^{N_\theta}$ along the tempering sequence, and—thanks to the smoothness of the sequence—gain a high computational efficiency by generating samples primarily in the most relevant part of Θ, compared to more basic sampling schemes such as importance sampling. One version of the SMC sampler is a sequential iteration of importance sampling and resampling on the sequence $\{\pi_p(\theta)\}_{p=0}^P$, proceeding as follows: samples $\{\theta^j\}_{j=1}^{N_\theta}$ are initially drawn from $\pi_0(\theta)$, and assigned importance weights $W_1^{(j)}$ from the ratio $\frac{\pi_1(\theta^j)}{\pi_0(\theta^j)}$. The samples are then resampled and moved around in the landscape of $\pi_1(\theta)$ with one or a few steps with Metropolis-Hastings. They are then weighted according to $\frac{\pi_2(\theta^j)}{\pi_1(\theta^j)}$, and the procedure is repeated. After P such iterations, samples from $\pi_P(\theta)$ are obtained. An illustration can be found in Figure 1b.

A reader familiar with PMH may understand this use of the SMC sampler as a manager of N_θ parallel PMH chains, which aborts and duplicates the chains in order to optimize the overall performance[1].

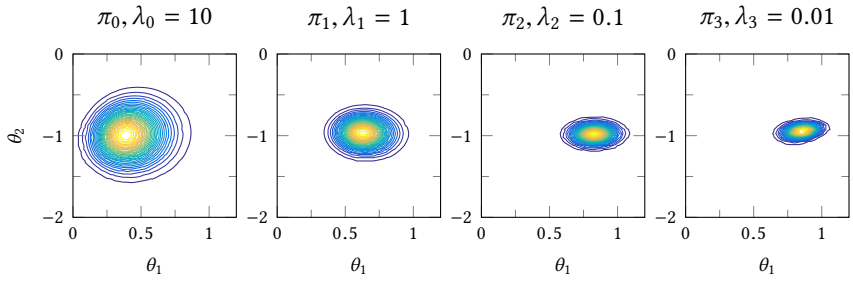
3 Solution strategy

Provided the background on particle filters, tempering and SMC samplers, we are now ready to assemble our proposed solution. We will first propose our novel tempering idea suited for learning parameters θ in models on the form (1), and then explore how the tempering pace can be automatically adapted to the problem. Thereafter we will provide an overview of the proposed algorithm, and detail some additional connections to existing literature.

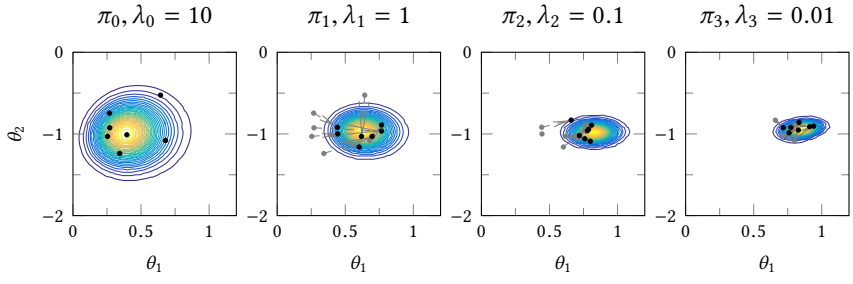
3.1 A tempering sequence for our problem

Our aim is to infer the posterior $p(\theta \mid y_{1:T})$ for the model (1). The absence of measurement noise in (1b) gives the likelihood estimate z from the bootstrap particle filter a high variance (in particular for values of θ not explaining the data well), which is

[1] A subtle but important difference to vanilla PMH is that a PMH chain is typically initialized arbitrarily, run until it converges, and thereafter its transient behavior (the burn-in period) is discarded. In the SMC sampler, however, all chains are 'warm-started' thanks to the resampling mechanism, and it is therefore *not* relying on asymptotics to avoid burn-in periods.



(a) *A tempering sequence shown by level curves for $\pi_p(\theta)$. The tempering sequence used in this figure is the sequence (4) proposed in this paper, for a linear state-space model with two unknown parameters θ_1 and θ_2. With decreasing λ_p, which is the variance of an artificial measurement noise, tempering is obtained, starting in a distribution with a broad support, and ending in a more narrow and peaky distribution.*



(b) *The problem of localizing the peak of $\pi_3(\theta)$ can be solved with an SMC sampler which propagates samples (black dots) through the smooth evolution of the sequence from $p = 0$ to 3. This is instead of starting to search directly in $\pi_3(\theta)$, which would be challenging because of the large 'flat' areas. The problem of large uninteresting regions becomes particularly articulated in high dimensional problems.*

Figure 1: *An illustration of the tempering idea. The model used here will later be properly introduced as an example in Section 5.1.*

a problem when seeking $p(\theta \mid y_{1:T})$. We therefore suggest to introduce the modified model

$$p(x_t \mid x_{1:t-1}, \theta) = f(x_t \mid x_{t-1}, u_{t-1}, \theta), \tag{3a}$$

$$y_t = g(x_t) + e_t, \qquad e_t \sim \mathcal{N}(0, \lambda_p). \tag{3b}$$

This model has an artificial Gaussian[2] measurement noise with variance λ_p, and our original model (1) is recovered for $\lambda_p = 0$. We denote the posterior distribution under this model as $p(\theta \mid y_{1:T}, \lambda_p)$, and the corresponding likelihood $p(y_{1:T} \mid \theta, \lambda_p)$. Furthermore, we will define a decreasing sequence of λ_p, such that $\lambda_P = 0$, and get a tempering sequence

$$\pi_p(\theta) = p(\theta \mid y_{1:T}, \lambda_p) \propto p(y_{1:T} \mid \theta, \lambda_p) p(\theta), \tag{4}$$

[2]The choice of Gaussian noise is for convenience and clarity only. Other choices, for example heavy-tailed distributions, are also possible. The only requirement is that its density can be evaluated point-wise.

which we have illustrated in Figure 1. In this sequence, the target distribution (at $p = P$) indeed becomes $\pi_P(\theta) = p(\theta \mid y_{1:T}, \lambda_P = 0) = p(\theta \mid y_{1:T})$, i.e., the posterior for θ in the original model (1), the problem we study in this paper. Such a tempering sequence bears clear resemblance to the ABC methodology proposed in Del Moral, Doucet, et al. (2012). However, to the best of the authors knowledge, such a tempering sequence has not previously been studied in the context of state-space models.

We will use the tempering sequence (4) in an SMC sampler. To this end, we need to be able to evaluate $\pi_p(\theta)$ up to proportionality. For this purpose, we propose to use the particle filter to estimate the likelihood $p(y_{1:T} \mid \theta)$ (and assume that the prior $p(\theta)$ can be evaluated). The algorithm will thus have a nested construction of SMC algorithms: the particle filter is used to generate likelihood estimates z_p for different values of θ and λ_p, and the SMC sampler is used to infer θ by keeping track of the samples $\{\theta^j\}_{j=1}^{N_\theta}$ and deciding for which values of θ to run the particle filter. However, to ease the presentation, we will throughout the rest of this section assume that we do have access to the likelihood $p(y_{1:T} \mid \theta, \lambda_p)$ exactly. That is indeed the case if, for example, (1) is a linear Gaussian state-space model or a finite discrete hidden Markov model, in which cases the Kalman filter (Rugh 1993) or the forward-backward algorithm (Cappé et al. 2005) would provide $p(y_{1:T} \mid \theta, \lambda_p)$ exactly. We will later return (in Section 4) to the situation where we only have access to stochastic estimates z_p, and expand the algorithm with a few more details to ensure theoretical soundness also for the general (and practically interesting) case (1).

3.2 Automatically determining the tempering pace

Choosing a good sequence $\{\lambda_p\}_{p=1}^P$ is fundamental to the performance of the proposed algorithm. A sequence $\{\lambda_p\}_{p=0}^P$ that is decreasing too fast will lead to rapid changes in the landscape of $\pi_p(\theta) = p(\theta \mid y_{1:T}, \lambda_p)$, obstructing the SMC sampler and adding to the variance of the final results. On the other hand, a sequence $\{\lambda_p\}_{p=0}^P$ that is decreasing too slowly will be a waste of computational power. To this end, we suggest to take inspiration from Del Moral, Doucet, et al. (2012), where they tackle a somewhat similar problem with the same version of the SMC sampler. They argue that a good tempering sequence would yield an effective sample size (ESS, Kong et al. 1994) somewhat constant throughout the sequence $p = 0, \ldots, P$. The ESS is defined as

$$\mathrm{ESS}\left(\{W_p^j\}_{j=1}^{N_\theta}\right) = \left(\sum_{j=1}^{N_\theta}\left(\frac{W_p^{(j)}}{\sum_{k=1}^{N_\theta} W_p^{(k)}}\right)^2\right)^{-1}, \tag{5}$$

where W_p^j denotes the importance weight of sample j from $\pi_p(\theta)$. The ESS takes values between 1 and N_θ, with the interpretation that inference based on the N_θ weighted samples is approximately equivalent to inference based on $\mathrm{ESS}\left(\{W_p^j\}_{j=1}^{N_\theta}\right)$ equally weighted samples. Consequently, if the weight of a single sample dominates all the other, the ESS is 1, and if all samples have equal weights, the ESS is N_θ. Intuitively, it is natural to expect that a smaller value of λ_p gives a lower ESS, if the particles x_t^n are fixed in the particle filter: the smaller λ_p, the fewer particles x_t^n are likely to explain the measurement y_t, yielding a higher variance in the particle weights, and thus a low ESS. Furthermore, Del Moral, Doucet, et al. (2012) note that on their problem it is

possible to solve the equation of setting λ_p (note that W_p^j depends on λ_p) such that

$$\mathrm{ESS}\left(\{W_p^j\}_{j=1}^{N_\theta}\right) = \alpha N_\theta, \tag{6}$$

where α is some user-chosen coefficient between 0 and 1. (A similar adaption can also be found in Jasra et al. 2011.) It turns out, perhaps a bit surprisingly, that it is in fact possible to solve (6) also in our case when W_p^j depends on z_p^j from the particle filter, which in turn depends on λ_p. We postpone the details to the subsequent section where we discuss the details of the inner particle filter algorithm which defines the estimate z_p^j. By solving (6), we obtain an automated way to determine the tempering pace 'on the fly', i.e., automatically determining the value of each λ_p with the aim to achieve a constant 'quality' (constant ESS) of the Monte Carlo approximation in runtime.

3.3 Termination

The variance of the estimates z_p is likely to increase as p increases and λ_p approaches 0 (Section 2.2). The implications of an increased variance of z_p will be that fewer of the proposed samples will be accepted in the Metropolis-Hastings step, and the overall performance of the SMC sampler will deteriorate. It may therefore be necessary to terminate the sampler prematurely (at, say, $\lambda_p = 0.01$ instead of the desired $\lambda_p = 0$), and take the obtained samples as an approximate solution. One heuristic suggested by Del Moral, Doucet, et al. (2012) for determining a suitable termination point is to monitor the rejection rate in the Metropolis-Hastings steps, and trigger a termination when it reaches a certain threshold. The effect of such a premature termination is analyzed (in a slightly different setting) by, e.g., Dean et al. (2015).

3.4 Proposed algorithm – preliminary version

In Algorithm 1 we outline our proposed algorithm. Here, q is the proposal in the Metropolis-Hastings sampler, proposing new values of the parameter θ which in a later stage are either accepted or rejected. We have in Algorithm 1 assumed that $p(y_{1:T} \mid \theta, \lambda_p)$ can be evaluated exactly. However, in the general case of a nonlinear state-space model the particle filter has to be used, which results in an unbiased stochastic estimate $z_p \approx p(y_{1:T} \mid \theta, \lambda_p)$. We will address this fully in Section 4.

As mentioned earlier, a parallel to our problem with no measurement noise can be found in the literature under the heading approximate Bayesian computations (ABC, Beaumont et al. 2002). In ABC, the idea is to simulate data $\widehat{y}(\theta)$ from a model and compare it to the recorded data y. ABC, however, is originally not formulated for state-space models, even though recent such contributions have been made (Dean et al. 2015; Jasra 2015). The introduction of an artificial measurement noise in our problem can be seen as an ABC-type of idea, but since the artificial measurement noise interacts with the particle filter (our analogy to simulate new data $\widehat{y}(\theta)$), our method does not qualify as a standard ABC solution.

Another closely related algorithm is the SMC^2 algorithm (Chopin et al. 2013). SMC^2 is also an SMC sampler using the particle filter to estimate the likelihood z, but it makes use of a data-tempered sequence (Section 2.3) instead of tempering based

Algorithm 1: Strategy for particle filter based learning of θ in (1)

Output: Samples $\{\theta^{(j)}\}_{j=1}^{N_\theta}$ from $p(\theta \mid y_{1:T}, \lambda_p)$.

Set $p \leftarrow 0$ and λ_0 large.
Sample initial $\{\theta^{(j)}\}_{j=1}^{N_\theta} \sim p(\theta \mid y_{1:T}, \lambda_0)$ using, e.g., Metropolis-Hastings.
while λ *not sufficiently small (Section 3.3)* **do**
  Update $p \leftarrow p + 1$.
  Let $\omega^{(j)} \leftarrow p(y_{1:T} \mid \theta^{(j)}, \lambda_{p-1})p(\theta^{(j)})$.
  Find λ_p such that $\texttt{ESS}(\{\omega^{(j)}\}_{j=1}^{N_\theta}, \{\widetilde{\omega}^{(j)} = p(y_{1:T} \mid \theta^{(j)}, \lambda_p)p(\theta^{(j)})\}_{j=1}^{N_\theta}) = \alpha \cdot N_\theta$.
  Let $\widetilde{\omega}^{(j)} \leftarrow p(\theta^{(j)} \mid y_{1:T}, \lambda_p)$.
  Draw $a^{(j)}$ with $\mathbb{P}\left(a^{(j)} = k\right) \propto \frac{\widetilde{\omega}^{(k)}}{\omega^{(k)}}$.
  Sample $\theta^{(j)} \leftarrow \texttt{Metropolis-Hastings}(\lambda_p, \theta^{(j)})$.
end
Function $\texttt{ESS}(\{\omega^{(j)}\}_{j=1}^{N_\theta}, \{\widetilde{\omega}^{(j)}\}_{j=1}^{N_\theta})$
  Let $W^{(j)} \leftarrow \frac{\widetilde{\omega}^{(j)}}{\omega^{(j)}}$.
  return $\left(\sum_{j=1}^{N_\theta} \left(W^{(j)} / \sum_{k=1}^{N_\theta} W^{(k)}\right)^2\right)^{-1}$
Function $\texttt{Metropolis-Hastings}(\lambda_p, \theta^{(j)})$
  Propose a new $\theta' \sim q(\cdot \mid \theta^{(j)})$.
  Sample $d \leftarrow \mathcal{U}_{[0,1]}$, i.e., uniformly on the interval $[0, 1]$.
  if $d < \frac{p(y_{1:T} \mid \theta', \lambda_p)p(\theta')}{p(y_{1:T} \mid \theta^{(j)}, \lambda_p)p(\theta^{(j)})} \frac{q(\theta^{(j)} \mid \theta')}{q(\theta' \mid \theta^{(j)})}$ **then**
    Accept $\theta^{(j)} \leftarrow \theta'$.
  end
  return $\theta^{(j)}$

All operations are for $n = 1, \ldots, N_x$.

on artificial measurement noise (4). For the problem of learning the parameters θ in (1), the particle filter is likely to face troubles for small values of the measurement noise λ_p. For our proposed algorithm, this can be handled by terminating the algorithm prematurely if necessary. Such a resort is not possible with a data-tempered sequence in SMC^2, since the problems with poor estimates z from the particle filter would be faced already from the first step of a data-tempered sequence.

4 Full algorithm and details

In this section, we will first consider how to initialize the algorithm, and thereafter the details concerning the particle filter required for the adaptation of λ_p. Next, we present the proposed algorithm in detail and fully address the fact that the particle filter only provides stochastic estimates z, whereas Algorithm 1 requires that $p(y_{1:T} \mid \theta)$ can be evaluated exactly. The key is to consider the proposed algorithm to be sampling from an extended space explicitly encoding all randomness in the estimator z, and

Algorithm 2: Bootstrap particle filter

Input: State space model $f(\cdot \mid \cdot, \theta)$, $g(\cdot)$, λ_p, $p(x_1)$, number of particles N_x, and data $y_{1:T}$.

Output: $x_{1:T}, a_{2:T}$

1 Sample $x_1^{(i)} \sim p(x_1)$.
2 Compute $w_1^{(i)} \leftarrow \mathcal{N}(y_1 | g(x_1^{(i)}), \lambda_p)$.
3 **for** $t = 2$ **to** T **do**
4 Sample $a_t^{(i)}$ with $\mathbb{P}\left(a_t^{(i)} = j\right) \propto w_{t-1}^{(j)}$.
5 Sample $x_t^{(i)} \sim f(x_t | x_{t-1}^{a_t^{(i)}}, \theta)$.
6 Compute $w_t^{(i)} \leftarrow \mathcal{N}(y_t | g(x_t^{(i)}), \lambda_p)$.
7 **end**

All operations are for $n = 1, \ldots, N_x$.

thereby reduce the problem to a standard SMC algorithm operating on an extended space.

4.1 Initialization

To initialize the SMC sampler properly, samples $\{\theta^{(j)}\}_{j=1}^{N_\theta}$ from $p(\theta \mid y_{1:T}, \lambda_0)$ are required. However, that distribution is typically not available to draw samples from directly. To this end, PMH (Schön, Svensson, et al. 2018) (or SMC^2) can be used. Since λ_0 is user-chosen, we can choose it big enough such that $p(y_{1:T} \mid \theta, \lambda_0)$ has a broad support and we can obtain low-variance estimates z_0. However, in practice the use of Metropolis-Hastings inside the SMC sampler makes the algorithm somewhat 'forgiving' with respect to initialization, and it may for practical purposes suffice to initialize the algorithm with samples $\{\theta^{(j)}\}_{j=1}^{N_\theta}$ that are only approximate samples from $p(\theta \mid y_{1:T}, \lambda_0)$ obtained using, e.g., some suboptimal optimization-based method.

4.2 Re-visiting the particle filter

We have so far not fully justified the use of the particle filter inside the proposed algorithm. The particle filter provides a *stochastic* estimate z_p of $p(y_{1:T} \mid \theta, \lambda_p)$, and the λ_p-adaptation requires that we can solve (6), $\mathrm{ESS}\left(\{W_p^j\}_{j=1}^{N_\theta}\right) = \alpha N_\theta$, where W_p^j depends on the ratio between z_p and z_{p-1}, in turn depending on λ_p and λ_{p-1}, respectively. Both estimates, z_p and z_{p-1}, are stochastic, which seems not to allow for a well-defined numerical solution to (6). This also implies that the weights $W^{(j)}$ in the SMC sampler are random themselves. The latter problem of stochastic weights within SMC is, however, already studied in the literature (Fearnhead et al. 2010), whereas solving (6) is novel in this work.

The key point for solving (6) in our context with particle filters, and also to theoretically justify the random weights, is to consider the outcome of the particle filter (Algorithm 2) to be all its internal random variables, $\{x_{1:T}^n, a_{2:T}^n\}_{n=1}^{N_x}$, rather than only z. By doing so, we can explicitly handle all randomness in the particle filter,

and understand our proposed algorithm as a standard algorithm on the non-standard extended space $\Theta \times \mathsf{X}^{N_x T} \times \mathsf{A}^{N_x(T-1)}$ (instead of only θ), where X is the space in which x_t lives, and similar for A and a_t. We will come back to this formalism, but let us first give a more intuitive view on the construction.

In solving (6), we would like to run the particle filter once (using λ_{p-1}), and afterwards decide on a λ_p such that (6) is fulfilled. The random variables in the particle filter, $\{x^n_{1:T}, a^n_{2:T}\}_{n=1}^{N_x}$, are drawn with a certain distribution determined by the particle filter (Algorithm 2) and λ_p. That is, if we were given samples $\{x^n_{1:T}, a^n_{2:T}\}_{n=1}^{N_x}$, we could compute the probability (density) of $\{x^n_{1:T}, a^n_{2:T}\}_{n=1}^{N_x}$ to be drawn by the particle filter. In particular, by inspection of Algorithm 2, we realize that *if* the ancestor variables $\{a^n_{2:T}\}_{n=1}^{N_x}$ were fixed, λ_p would not affect $\{x^n_{1:T}\}_{n=1}^{N_x}$, but only the computation of z. Thus, if we run a particle filter with a measurement noise model with variance λ_{p-1} and save $\{x^n_{1:T}, a^n_{2:T}\}_{n=1}^{N_x}$, we may afterwards compute the probability (density) of the resampling (i.e., the draw of $\{a^n_{2:T}\}_{n=1}^{N_x}$) to have happened had it been run with a measurement noise model with variance λ_p instead[3]. This turns out to be enough for evaluating ESS $\left(\{W^j_p\}_{j=1}^{N_\theta}\right)$ conditionally on $\{x^n_{1:T}, a^n_{2:T}\}_{n=1}^{N_x}$, which can be used to solve (6) using a numerical search, such as a bisection method.

This idea bears clear resemblances to the work by Le Gland (2007), but is not identical. Whereas Le Gland (2007) considers fixed resampling weights across different models (in our context different λ_p), the resampling weights are not fixed in our approach, but changes with λ_p.

A useful perspective is to understand our idea as importance sampling of $\{x^n_{1:T}, a^n_{2:T}\}_{n=1}^{N_x}$, using a particle filter with λ_{p-1} as proposal and a particle filter with λ_p as target.

We summarize our proposed method in Algorithm 3. Continuing the extended space motivation, Algorithm 3 can in its most compact form be seen as a standard SMC sampler on the extended space $\Theta \times \mathsf{X}^{N_x T} \times \mathsf{A}^{N_x(T-1)}$ with target distribution at iteration p

$$p(\theta, \{x_{1:T}, a_{2:T}\}_{n=1}^{N_x} \mid y_{1:T}, \lambda_p) \propto \tag{7}$$
$$p(\theta) \left(\prod_{t=1}^{T} \sum_{n=1}^{N_x} g(y_t \mid x_t^n, \lambda_p) \right) \left(\prod_{t=1}^{T-1} \prod_{n=1}^{N_x} \underbrace{\mathbb{P}\left(a_{t+1}^{(i)} \mid \{w_t^{(i)}\}_{n=1}^{N_x}\right)}_{= \frac{g(y_t \mid x_t^{a^n_{t+1}}, \lambda_p)}{\sum_{n=1}^{N_x} g(y_t \mid x_t^n, \lambda_p)}} \right) \left(\prod_{t=1}^{T-1} \prod_{n=1}^{N_x} f(x_{t+1}^n \mid x_t^{a^n_{t+1}}, \theta) \right).$$

From this, Algorithm 3, and in particular the particle filter (Algorithm 2) as well as the weighting function `w` in Algorithm 3, can be derived.

The previous paragraph can be understood as follows. First of all, the particle filter algorithm itself contains random elements. If we consider all randomness in the particle filter explicitly as random variables, i.e., consider $\{x^n_{1:T}, a^n_{2:T}\}_{n=1}^{N_x}$ and not just z, Algorithm 3 is a standard SMC sampler (Del Moral, Doucet, et al. 2006) for the distribution (7). This implies that available theoretical guarantees and convergence results (e.g., Chopin 2004; Del Moral 2004; Del Moral, Doucet, et al. 2006) apply also to our construction when the λ_p sequence is fixed. When λ_p is selected adaptively these

[3]For this answer not to be exactly 0 forbiddingly often, multinomial resampling has to be used.

Algorithm 3: Particle-filter based learning of θ in (1)

Output: Samples $\{\theta^{(j)}\}_{j=1}^{N_\theta}$ from $p(\theta \mid y_{1:T}, \lambda)$.

Set $p \leftarrow 0$ and λ_0 large.

Sample initial $\{\theta^{(j)}\}_{j=1}^{N_\theta} \sim p(\theta \mid y_{1:T}, \lambda_0)$ using, e.g., particle Metropolis-Hastings.

Run a particle filter with λ_0 for each $\theta^{(j)}$, and save $\xi^{(j)} \triangleq \{x_{1:T}^{(i)}, a_{2:T}^{(i)}\}_{n=1}^{N_x}$.

while λ_p *not sufficiently small (Section 3.3)* **do**

  Update $p \leftarrow p + 1$.

  Let $\omega^{(j)} \leftarrow \mathtt{w}(\lambda_{p-1}, \theta^{(j)}, \xi^{(j)})$.

  Find λ_p such that $\mathtt{ESS}(\{\omega^{(j)}\}_{j=1}^{N_\theta}, \{\mathtt{w}(\lambda_p, \theta^{(j)}, \xi^{(j)})\}_{j=1}^{N_\theta}) = \alpha \cdot N_\theta$.

  Let $\widetilde{\omega}^{(j)} \leftarrow \mathtt{w}(\lambda_p, \theta^{(j)}, \xi^{(j)})$.

  Resample the (θ, ξ)-particles using weights $\propto \frac{\widetilde{\omega}^{(k)}}{\omega^{(k)}}$

  Sample $\{\theta^{(j)}, \xi^{(j)}\} \leftarrow \mathtt{Particle\ Metropolis\text{-}Hastings}(\lambda_p, \theta^{(j)}, \xi^{(j)})$.

end

Function $\mathtt{w}(\lambda_p, \theta^{(j)}, \xi^{(j)})$

  Let $w_{t-1}^{(i)} \leftarrow g(y_t | x_t^{(i)}, \lambda_p)$

  return $p(\theta^{(j)}) \left(\prod_{t=1}^{T} \sum_{n=1}^{N_x} w_{t-1}^{(i)}\right) \left(\prod_{t=1}^{T-1} \prod_{n=1}^{N_x} \mathbb{P}\left(a_{t+1}^{(i)} \mid \{w_t^{(i)}\}_{n=1}^{N_x}\right)\right)$

Function $\mathtt{ESS}(\{\omega^{(j)}\}_{j=1}^{N_\theta}, \{\widetilde{\omega}^{(j)}\}_{j=1}^{N_\theta})$

  Let $W^{(j)} \leftarrow \frac{\widetilde{\omega}^{(j)}}{\omega^{(j)}}$ for every j

  return $\left(\sum_{j=1}^{N_\theta} \left(W^{(j)} / \sum_{k=1}^{N_\theta} W^{(k)}\right)^2\right)^{-1}$

Function $\mathtt{Particle\ Metropolis\text{-}Hastings}(\lambda_p, \theta^{(j)}, \xi^{(j)})$

  Let $z_p^j \leftarrow \prod_{t=1}^{T} \sum_{n=1}^{N_x} g(y_t \mid x_t^i, \lambda_p)$ (with x_t^i from $\xi^{(j)}$)

  Propose a new $\theta' \sim q(\cdot \mid \theta^{(j)})$

  Run a particle filter with λ_p and θ' and save ξ'

  Let $z_p' \leftarrow \prod_{t=1}^{T} \sum_{n=1}^{N_x} g(y_t \mid x_t^i, \lambda_p)$ (with x_t^i from ξ')

  Sample $d \leftarrow \mathcal{U}_{[0,1]}$, i.e., uniformly on the interval $[0, 1]$.

  if $d < \frac{z_p' p(\theta')}{z_p^{(j)} p(\theta^{(j)})} \frac{q(\theta^{(j)} \mid \theta')}{q(\theta' \mid \theta^{(j)})}$ **then**

    Update $\theta^{(j)} \leftarrow \theta'$, $\xi^{(j)} \leftarrow \xi'$

  end

  return $\theta^{(j)}, \xi^{(j)}$

results do not readily apply, but Beskos et al. (2016) have established convergence results for adaptive SMC algorithms in a related setting, and these results could possibly be extended to the adaptive scheme proposed in this article. Note also that no practical problems caused by the proposed adaptation have been encountered in the numerical examples.

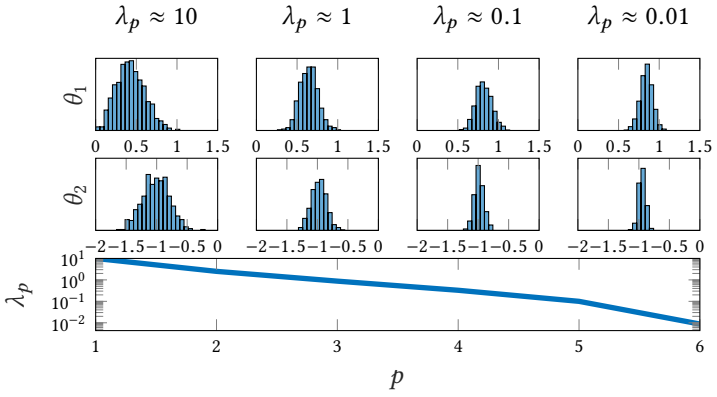


Figure 2: *Result from applying Algorithm 1 to $T = 200$ data points from the model (8). The upper panels show how the marginals of the samples contracts as $\lambda_p \to 0$ (cf. Figure 1), and the lower panel shows the sequence $\{\lambda_p\}_{p=1}^{P}$ automatically determined by our algorithm in an adaptive manner.*

5 Numerical experiments

In this section, we provide three numerical experiments illustrating and evaluating the proposed method from various perspectives. First, we start with a simple numerical example with a linear state-space model subject to Gaussian noise (implicitly introduced by Figure 1) to illustrate the main ideas presented by Algorithm 1. We then consider a more challenging nonlinear example, where we compare our proposed method to the PMH algorithm (Andrieu, Doucet, et al. 2010) and SMC² (Chopin et al. 2013), as well as a study on the influence of T. Finally we consider the challenging Wiener-Hammerstein benchmark problem from M. Schoukens and Noël (2016). The code for the examples is available via the first author's homepage.

5.1 Toy example

We consider the linear state-space model on the form

$$x_{t+1} = \begin{bmatrix} 1 & \theta_1 \\ 0 & 0.1 \end{bmatrix} x_t + \begin{bmatrix} \theta_2 \\ 0 \end{bmatrix} u_t + v_t, \qquad v_t \sim \mathcal{N}(0, I_2), \tag{8a}$$

$$y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} x_t, \tag{8b}$$

where $\theta \triangleq \{\theta_1, \theta_2\}$ are the unknown parameters (true values: $\theta_1 = 0.8$, $\theta_2 = -1$) and I_2 denotes the identity matrix of dimension 2. This model was used to produce Figure 1, where the propagation of samples $\{\theta^j\}_{j=1}^{N_\theta}$ was illustrated. Since this model is linear and Gaussian, the computation of the likelihood $p(y_{1:T} \mid \theta, \lambda_p)$ can be done exactly[4] and no particle filter (with its potential problem due to small measurement noise variance) is needed. Thus, Algorithm 1 can be applied directly, by using the Kalman filter to exactly compute $p(y_{1:T} \mid \theta, \lambda_p)$. We now demonstrate Algorithm 1 by

[4]The choice of a linear and Gaussian model also made it possible to exactly plot the contours in Figure 1.

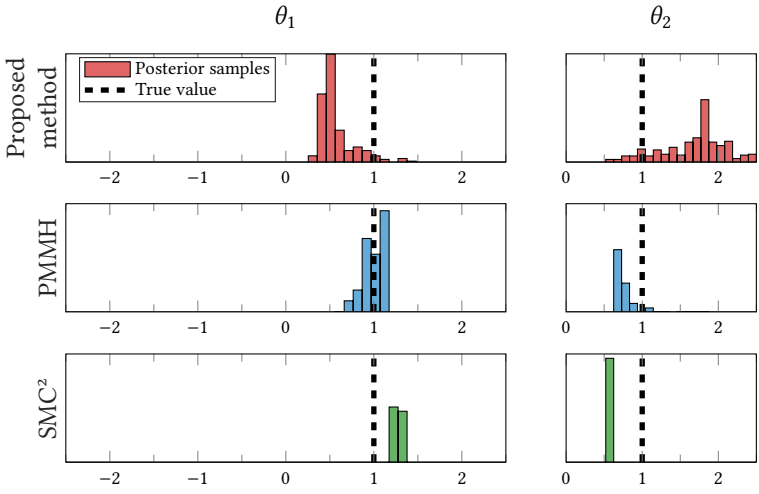


Figure 3: *Posterior samples from the problem in Section 5.2. The probability of acceptance (empirically 0.026%) in the Metropolis-Hastings mechanism is very low due to the model (9) with highly informative observations, which gives a high variance in the estimates z. Neither PMH nor SMC² therefore explore the posterior well, whereas the proposed method shows a better result due to the proposed tempering scheme, which adds an artificial measurement noise to the model giving it computational advantages with less variance in z_p.*

applying it to $T = 200$ data points simulated from (8). The artificial measurement noise λ_p is automatically adapted such that ESS $\approx 0.5N_\theta$ at each step. The priors for both parameters are taken to be uniform on $[0, 2.5] \times [0, 2.5]$. The resulting (marginal) posteriors are summarized in Figure 2, which shows that the automatic tempering seems to work as expected. Figure 1 shows the true (joint) posteriors, and we can indeed confirm that their marginals resembles Figure 2.

The main motivation behind our work was indeed to overcome the computational difficulties for the particle filter when the variance of the measurement noise is very small. However, for probabilistic learning of θ also in linear Gaussian models where the exact Kalman filter can be applied, sampling methods can still be useful for learning θ, see, e.g, Ninness and Henriksen (2010) and Wills et al. (2012) for the use of Metropolis-Hastings and Gibbs samplers, respectively. Our proposed tempering scheme for an SMC sampler thus presents yet another alternative for these models.

5.2 A more challenging nonlinear example

We now consider the following state-space model

$$x_{t+1} = \operatorname{atan}(x_t) + \theta_1 u_t + v_t, \qquad v_t \sim \mathcal{N}(0, 1), \tag{9a}$$

$$y_t = |x_t| + \theta_1\theta_2 + e_t, \qquad e_t \sim \mathcal{N}(0, 10^{-2}). \tag{9b}$$

This model, with a 1-dimensional state space, has as an exogenous input u_t, a significant amount of process noise v_t and an almost negligible measurement noise e_t. From this model, $T = 300$ data points were simulated and the two unknown parameters

$\theta = \{\theta_1, \theta_2\}$ are to be learned from the measured data $\{y_{1:T}, u_{1:T}\}$ with uniform priors. The input $u_{1:T}$ is taken as a realization of a white noise random process.

The relatively short data record together with the presence of θ_2 only in the product $\theta_1\theta_2$ in (9b) suggest there is a certain amount of uncertainty present in the problem, which we expect to be reflected in the posterior. However, the highly informative observations makes this a rather challenging problem for the standard methods.

We apply our proposed method, and compare it to PMH (Del Moral, Doucet, et al. 2006) and SMC^2 (Chopin et al. 2013) on this problem. In all algorithms, we use the bootstrap particle filter (Algorithm 2) with $N_x = 300$, and a simple random walk proposal. Furthermore, we let $N_\theta = 300$, $K = 40$ and $\alpha = 0.3$ in Algorithm 3, as well as their counterparts in SMC^2, and we run PMH until 100 000 samples are obtained. We use the same Metropolis-Hastings proposal in all algorithms. For our proposed algorithm, we adopt a similar heuristic as Del Moral, Doucet, et al. (2012) and terminate the tempering once the acceptance rate in the Metropolis-Hastings procedure goes below 5%.

The obtained posterior samples are shown in Figure 3. The mixing of PMH is rather poor (the acceptance rate was recorded as 0.026%), and it has consequently not managed to explore the posterior as well as our proposed method. A similar problem occurs for SMC^2, which performs even worse on this problem. Since the tempering in our proposed method, however, follows a sequence of decreasing artificial measurement noise, which terminates once the mixing becomes too bad, it does not suffer from the same problem.

The settings of PMH can indeed be optimized by using more clever proposals than random walks (see, e.g., Dahlin and Schön 2016 for an overview) and methods for reducing the variance of z (such as adapted or bridging particle filter Del Moral, Jasra, et al. 2015; Del Moral and Murray 2015; Pitt and Shephard 1999). Such adaption would indeed push the performance further. However, the adaption could be applied to all three methods, and such tuning is therefore not crucial in a relative comparison between the methods.

5.3 Evaluating the performance with growing T

As discussed in Section 4.2, the proposed method can essentially be understood as an importance sampler producing $\{x^n_{1:T}, a^n_{2:T}\}_{n=1}^{N_x}$. Because of this, we could expect the method to be less efficient as the number of measurements T grows, since T is one of the dimensions in the importance sampling space. We study this effect with a similar state-space model as above, and record how many steps P that are required for the artificial output noise to transition from the initial starting point $\lambda_0 = 1$ to the true $\lambda_P = 0.05$, when α is set to 0.4, and different number of data points T are included in the data set. The results are shown in Figure 4 and suggest a linear growth in the number of steps P as T grows. In combination with a computational load growing linearly with T for the particle filter, the total computational load is $\propto T^2$ for N_x constant[5].

[5]For optimal performance, possibly also N_x should be scaled with T, this is a question for further research.

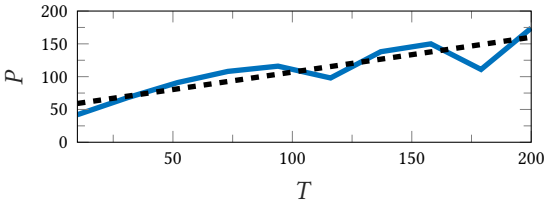
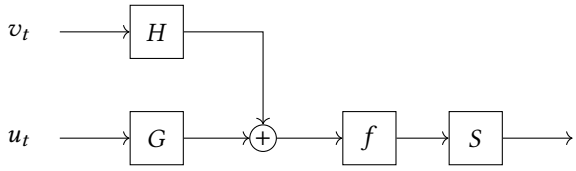
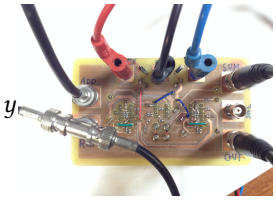


Figure 4: *The number of steps (solid blue) required in Example 5.3 to transition from the initial $\lambda_0 = 1$ to the true final $\lambda_P = 0.05$ when different number of data points T are included in the data set, and α is fixed at 0.4. As expected the number of steps, P, required grows with the number of data points included, T, seemingly in a rather linear way (dashed black).*



(a) *A block diagram describing the system. The blocks with uppercase letters (G, H, S) are all linear dynamical systems, whereas the block with lowercase f is a static nonlinearity. Further, u_t is a known input signal, v_t is (white) process noise and y_t is the measured output.*



(b) *A picture of the electronic circuit*

Figure 5: *The Wiener-Hammerstein benchmark system.*

5.4 The Wiener-Hammerstein benchmark with process noise

The Wiener-Hammerstein benchmark (M. Schoukens and Noël 2016) is a recent benchmark problem for system identification which is a particular special case of the model (1). This problem has also served as the motivating problem for us to propose this method. The benchmark is implemented as an electronic circuit, and the challenge is to use recorded data from the system to estimate a model which is able to imitate the behavior of the electric circuit well. The system can be described as a Wiener-Hammerstein system, i.e., a linear dynamical system, a static nonlinearity, and then another linear dynamical system in series. This is by now a fairly well-studied model, see e.g. Bershad et al. (2001), Billings and Fakhouri (1982), Giri and Bai (2010), and J. Schoukens et al. (2009) for earlier work. There was also a relatively recent special section devoted to an earlier Wiener-Hammerstein benchmark problem in Control Engineering Practice in 2012 Hjalmarsson et al. (2012). The key difference is the significantly higher process noise level in this newly proposed benchmark.

The input to the system is a (known) signal entering into the first linear system. There is also an (unknown) colored process noise present, which enters directly into the nonlinearity. The measurements are of rather high quality, so there is very little measurement noise (when compared to the process noise) which makes the measurements highly informative. The system is summarized in Figure 5.

The structure of the Wiener-Hammerstein benchmark system can be brought into the state-space formalism as

$$\begin{bmatrix} x^G_{t+1} \\ x^H_{t+1} \\ x^S_{t+1} \end{bmatrix} = \begin{bmatrix} A^G & 0 & 0 \\ 0 & A^H & 0 \\ 0 & 0 & A^S \end{bmatrix} \begin{bmatrix} x^G_t \\ x^H_t \\ x^S_t \end{bmatrix} + \begin{bmatrix} B^G \\ 0 \\ 0 \end{bmatrix} u_t + \begin{bmatrix} 0 \\ B^H \\ 0 \end{bmatrix} v_t$$
$$+ \begin{bmatrix} 0 \\ 0 \\ B^S \end{bmatrix} \sum_{m=1}^{M} c^{(m)} \phi^{(m)} (C^G x^G_t + C^H x^H_t + D^G u_t), \tag{10a}$$
$$y_t = C^S x^S_t + D^S v^S_t, \tag{10b}$$

where all As are 3×3-matrices, Bs are 3×1-matrices, Cs are 1×3-matrices, Ds are scalars, $\{\phi^{(m)}\}$ is a Fourier basis function expansion (truncated at $M = 10$), and v_t is a zero-mean scalar-valued white Gaussian process noise with unknown variance. Adjusting for the overparametrization of the linear state-space model, the effective number of unknown parameters is 30. For learning the parameters, a data set with $T = 8192$ samples and u_t a faded multisine input[6] was used. We applied Algorithm 3 with $N_\theta = 50$ and $K = 10$. For initialization purposes, an approximate model was found essentially using the ideas by Paduart et al. (2010) (which is computationally lighter, but cannot fully handle the presence of process noise, on the contrary to Algorithm 3).

The obtained results are presented in Table 11.1 and Figure 6, where they are reported according to the benchmark instructions M. Schoukens and Noël (2016), i.e., the simulation error for two test data sets measured on the system with no process noise present and a swept sine and a multisine as input, respectively. For reference, the performance of the model used to initialize Algorithm 3 is also included. The results reported were obtained within a few hours on a standard personal computer.

The essentially non-existing measurement noise makes PMH (as well as SMC^2) incompatible with this problem.

Table 11.1: *Wiener-Hammerstein benchmark: The root mean square error (RMSE) of the simulation error on the provided test data sets.*

	RMSE of proposed method	RMSE of initial model
Swept sine	0.014	0.039
Multisine	0.015	0.038

[6] Available as `WH_MultisineFadeOut` at http://homepages.vub.ac.be/~mschouke/benchmarkWienerHammerstein.html

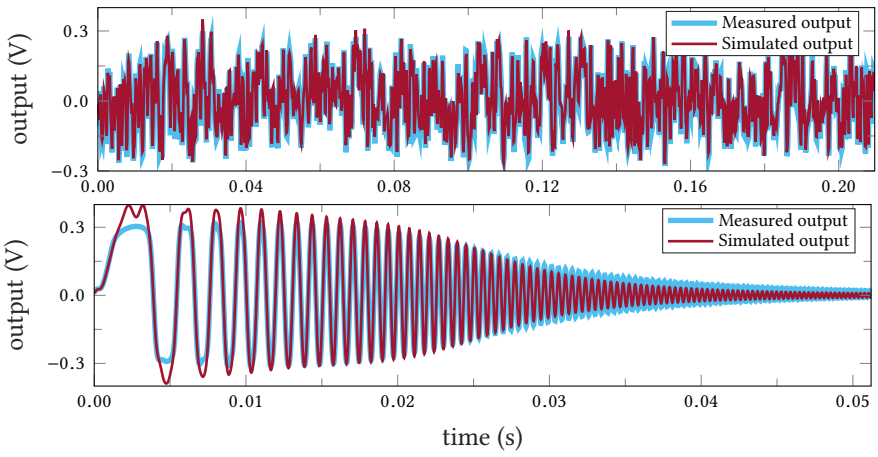


Figure 6: *Simulated output from the model (red line) versus the recorded output (blue line) for the Wiener-Hammerstein benchmark, for a multisine (top) and swept sine signal (bottom). The test data sets are recorded with no process noise, as opposed to the training data sets which were used for learning the model.*

6 Discussion

We have proposed an algorithm for probabilistic learning of unknown parameters in models of the structure (1), i.e., state-space models with highly informative observations. Our proposed algorithm can be understood as either an ABC-inspired methodology (Dean et al. 2015; Del Moral, Doucet, et al. 2012), or as an alternative tempering in an SMC sampler (akin to SMC^2 Chopin et al. 2013). Its theoretical justification follows from viewing it as a standard SMC sampler on an extended space, and well established theoretical guarantees are thus available.

The importance sampling perspective (Section 4.2) raises the question of how well the proposed adaptation of λ_p scales with dimensionality, in particular the number of measurements T. This is partly investigated in Example 5.3 suggesting (at least) a computational load $\propto T^2$, but we also note that the method performs well in the benchmark example containing $T = 8192$ samples. A more systematic numerical evaluation of the proposed method, which however is beyond the scope of this paper, would certainly be of interest.

For further research, connections with the idea of variational tempering (Mandt et al. 2016) could possibly also be of interest to explore. It is also not obvious that the ESS criterion (6) is the best criterion for deciding a well-performing tempering within the SMC sampler, and other alternatives could be studied and compared.

For optimal performance, our proposed method could be combined with methods for variance reduction of the estimate z, such as the adapted or bridging particle filter (Del Moral, Jasra, et al. 2015; Del Moral and Murray 2015; Pitt and Shephard 1999). The combination with such methods would indeed be interesting to explore further. However, while the use of such methods may indeed push the limits, for most cases they will not remove the fundamental problem.

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

Christophe Andrieu and Gareth O. Roberts (2009). "The pseudo-marginal approach for efficient Monte Carlo computations". In: *Annals of Statistics* 37.2, pp. 967–725.

Mark A. Beaumont, Wenyang Zhang, and David J. Balding (2002). "Approximate Bayesian computation in population genetics". In: *Genetics* 162.4, pp. 2025–2035.

Neil J. Bershad, Patrick Celka, and Stephen McLaughlin (2001). "Analysis of stochastic gradient identification of Wiener-Hammerstein systems for nonlinearities with Hermite polynomial expansions". In: *IEEE Transactions on Signal Processing* 49.5, pp. 1060–1072.

Alexandros Beskos, Ajay Jasra, Nikolas Kantas, and Alexandre Thiery (2016). "On the convergence of adaptive sequential Monte Carlo algorithms". In: *The Annals of Applied Probability* 26.2, pp. 1111–1146.

Stephen A. Billings and S. Y. Fakhouri (1982). "Identification of systems containing linear dynamic and static nonlinear elements". In: *Automatica* 18.1, pp. 15–26.

Olivier Cappé, Éric Moulines, and Tobias Rydén (2005). *Inference in hidden Markov models*. Springer Series in Statistics. New York, NY, USA: Springer.

Nicolas Chopin (2004). "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference". In: *Annals of Statistics* 36.6, pp. 2385–2411.

Nicolas Chopin, Pierre E. Jacob, and Omiros Papaspiliopoulos (2013). "SMC^2: an efficient algorithm for sequential analysis of state space models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75.3, pp. 397–426.

Johan Dahlin and Thomas B. Schön (2016). "Getting started with particle Metropolis-Hastings for inference in nonlinear models". In: *arXiv:1511:01707*.

Thomas A. Dean, Sumeetpal S. Singh, Ajay Jasra, and Gareth W. Peter (2015). "Parameter estimation for hidden Markov models with intractable likelihoods". In: *Scandinavian Journal of Statistics* 41.4, pp. 970–987.

Pierre Del Moral (2004). *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. New York, NY, US: Springer.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra (2006). "Sequential Monte Carlo samplers". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3, pp. 411–436.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra (2012). "An adaptive sequential Monte Carlo method for approximate Bayesian computation". In: *Statistics and Computing* 22.5, pp. 1009–1020.

Pierre Del Moral, Ajay Jasra, Anthony Lee, Christopher Yau, and Xiaole Zhang (2015). "The alive particle filter and its use in particle Markov chain Monte Carlo". In: *Stochastic Analysis and Applications* 33.6, pp. 943–974.

Pierre Del Moral and Lawrence M. Murray (2015). "Sequential Monte Carlo with highly informative observations". In: *SIAM/ASA Journal on Uncertainty Quantification* 3.1, pp. 969–997.

Arnaud Doucet and Adam M. Johansen (2011). "A tutorial on particle filtering and smoothing: fifteen years later". In: *Nonlinear Filtering Handbook*. Ed. by D. Crisan and B. Rozovsky. Oxford, UK: Oxford University Press, pp. 656–704.

Paul Fearnhead, Omiros Papaspiliopoulos, Gareth O. Roberts, and Andrew Stuart (2010). "Random-weight particle filtering of continuous time processes". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 72.4, pp. 497–512.

Fouad Giri and Er-Wei Bai, eds. (2010). *Block-oriented nonlinear system identification*. Berlin, Germany: Springer-Verlag.

Neil J. Gordon, David J. Salmond, and Adrian F.M. Smith (1993). "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". In: *IEE Proceedings F - Radar and Signal Processing*, pp. 107–113.

Håkan Hjalmarsson, Cristian R. Rojas, and Daniel E. Rivera (2012). "System identification: A Wiener-Hammerstein benchmark". In: *Control Engineering Practice* 20.11, pp. 1095–1096.

Ajay Jasra (2015). "Approximate Bayesian computation for a class of time series models". In: *International Statistical Review* 83.3, pp. 405–435.

Ajay Jasra, David A. Stephens, Arnaud Doucet, and Theodoros Tsagaris (2011). "Inference for Lévy-driven stochastic volatility models via adaptive sequential Monte Carlo". In: *Scandinavian Journal of Statistics* 38.1, pp. 1–22.

Nikolas Kantas, Arnaud Doucet, Sumeetpal S. Singh, Jan M. Maciejowski, and Nicolas Chopin (2015). "On particle methods for parameter estimation in state-space models". In: *Statistical Science* 30.3, pp. 328–351.

Augustine Kong, Jun S. Liu, and Wing Hung Wong (1994). "Sequential imputations and Bayesian missing data problems". In: *Journal of the American Statistical Association* 89.425, pp. 278–288.

Francois Le Gland (2007). "Combined use of importance weights and resampling weights in sequential Monte Carlo methods". In: *ESAIM: Proc.* 19, pp. 85–100.

Stephan Mandt, James McInerney, Farhan Abrol, Rajesh Ranganath, and David Blei (2016). "Variational tempering". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Caáiz, Spain, pp. 704–712.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller (1953). "Equation of state calculations by fast computing machines". In: *Journal of Chemical Physics* 21.6, pp. 1087–1092.

Brett Ninness and Soren Henriksen (2010). "Bayesian system identification via Markov chain Monte Carlo techniques". In: *Automatica* 46.1, pp. 40–51.

Johan Paduart, Lieve Lauwers, Jan Swevers, Kris Smolders, Johan Schoukens, and Rik Pintelon (2010). "Identification of nonlinear systems using polynomial nonlinear state space models". In: *Automatica* 46.4, pp. 647–656.

Václav Peterka (1981). "Bayesian system identification". In: *Automatica* 17.1, pp. 41–53.

Michael K. Pitt and Neil Shephard (1999). "Filtering via simulation: auxiliary particle filters". In: *Journal of the American Statistical Association* 94.446, pp. 590–599.

Christian P. Robert (2001). *The Bayesian choice: from decision-theoretic foundations to computational implementation*. 2nd ed. New York, NY, USA: Springer.

Christian P. Robert and George Casella (2004). *Monte Carlo statistical methods*. 2nd ed. New York, NY, USA: Springer.

Wilson J. Rugh (1993). *Linear system theory*. Englewood Cliffs, NJ, USA: Prentice Hall.

Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wågberg, Christian A. Naesseth, Andreas Svensson, and Liang Dai (2015). "Sequential Monte Carlo methods for system identification". In: *Proceedings of the 17th IFAC Symposium on System Identification (SYSID)*. Beijing, China, pp. 775–786.

Thomas B. Schön, Andreas Svensson, Lawrence M. Murray, and Fredrik Lindsten (2018). "Probabilistic learning of nonlinear dynamical systems using sequential Monte Carlo". In: *Mechanical Systems and Signal Processing* 104, pp. 866–883.

Johan Schoukens, Johan Suykens, and Lennart Ljung (2009). "Wiener-Hammerstein Benchmark". In: *Proceedings of the 15th IFAC Symposium on system identification (SYSID)*. St. Malo, France.

Maarten Schoukens and Jean-Philippe Noël (2016). "Wiener-Hammerstein benchmark with process noise". In: *Workshop on Nonlinear System Identification Benchmarks*. Brussels, Belgium, pp. 15–19.

Adrian Wills, Thomas B. Schön, Fredrik Lindsten, and Brett Ninness (2012). "Estimation of linear systems using a Gibbs sampler". In: *Proceedings of the 16th IFAC Symposium on System Identification (SYSID)*. Brussels, Belgium, pp. 203–208.

Paper VI

Title

Learning nonlinear state-space models using smooth particle-filter-based likelihood approximations

Authors

Andreas Svensson, Fredrik Lindsten and Thomas B. Schön

Edited version of

Andreas Svensson, Fredrik Lindsten, and Thomas B. Schön (2018). "Learning nonlinear state-space models using smooth particle-filter-based likelihood approximations". In: *Proceedings of the 18th IFAC symposium on system identification (SYSID)*. Stockholm, Sweden, pp. 652–657.

Digital identity

https://arxiv.org/abs/1711.10765

Financial support

The Swedish Foundation for Strategic Research (SSF) via the projects *ASSEMBLE* (contract number: RIT15-0012) and *Probabilistic Modeling and Inference for Machine Learning* (contract number: ICA16-0015), and the Swedish Research Council (VR) via the projects *NewLEADS - New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079) and *Learning of Large-Scale Probabilistic Dynamical Models* (contract number: 2016-04278)

Learning Nonlinear State-Space Models Using Smooth Particle-Filter-Based Likelihood Approximations

Abstract

When classical particle filtering algorithms are used for maximum likelihood parameter estimation in nonlinear state-space models, a key challenge is that estimates of the likelihood function and its derivatives are inherently noisy. The key idea in this paper is to run a particle filter based on a current parameter estimate, but then use the output from this particle filter to re-evaluate the likelihood function approximation also for other parameter values. This results in a (local) deterministic approximation of the likelihood and any standard optimization routine can be applied to find the maximum of this approximation. By iterating this procedure we eventually arrive at a final parameter estimate.

1 Introduction

Consider the nonlinear state-space model

$$x_t \mid x_{t-1} \sim f_\theta(x_t \mid x_{t-1}), \tag{1a}$$

$$y_t \mid x_t \sim g_\theta(y_t \mid x_t), \tag{1b}$$

where the hidden states $x_t \in \mathsf{X} \subset \mathbb{R}^{n_x}$ and observations $y_t \in \mathsf{Y} \subset \mathbb{R}^{n_y}$ evolves over time $t = 0, 1, \ldots$. We assume that the probability densities f_θ and g_θ are parameterized by unknown parameters θ, which we are interested to learn from data. More specifically we are seeking the maximum likelihood estimate of θ from given observations $\{y_1, \ldots, y_T\} \triangleq y_{1:T}$, i.e., finding

$$\widehat{\theta} = \arg\max_\theta p_\theta(y_{1:T}). \tag{2}$$

We use p_θ to denote probability densities conditional on θ, and shall refer to $p_\theta(y_{1:T})$ as the *likelihood function* when considered a function of θ. It follows from (1) and the initial state density $p(x_0)$ that

$$p_\theta(y_{1:T}) = \int p(x_0) \prod_{t=1}^{T} f_\theta(x_t \mid x_{t-1}) g_\theta(y_t \mid x_t) dx_{0:T}. \tag{3}$$

This can be evaluated analytically only for a few special cases. Approximations, such as the particle filter, are needed for the nonlinear case. For any given θ, the particle filter can be run to give a Monte Carlo based estimate $\widehat{z}_\theta$ of (3). It is well-known (Del Moral 2004) that $\widehat{z}_\theta$ is unbiased, i.e., $\mathbb{E}[\widehat{z}_\theta] = p_\theta(y_{1:T})$, where the expectation is over the stochasticity in the particle filter algorithm itself. Even though unbiasedness of $\widehat{z}_\theta$ may sound promising, it is still stochastic (i.e., a different value is obtained every time the particle filter algorithm is run, since it is a Monte Carlo solution), often with a considerable variance and non-Gaussian characteristics. Because of this, the particle filter outcome $\widehat{z}_\theta$ can not be used as objective function in standard optimization routines. Indeed, schemes using optimization for noisy function evaluations have been proposed (e.g., Dahlin and Lindsten 2014; Wills and Schön 2017), but the approach of this paper will be to modify the objective function (the particle filter) rather than the optimization routine.

We propose in this paper to turn the stochastic estimator $\widehat{z}_\theta$ into a deterministic function by first running a particle filter with some current parameter estimate θ_{k-1} and then scrutinize the particle filter algorithm to re-interpret $\widehat{z}_\theta$ as a *deterministic* function of arbitrary parameter values. In a sense we hereby manage to circumvent the stochasticity from the Monte Carlo procedure in a consistent way, and can apply standard optimization routines to search for a new parameter value which maximizes this approximation of the likelihood function. We refer to the new parameter value as θ_k, and proceed in an iterative fasion by running a new particle filter for θ_k, etc. The idea is outlined as Algorithm 1.

Algorithm 1: Identification idea

1 **for** $k = 1, \ldots$ **do**
2 Run a particle filter with θ_{k-1} and save all the generated random numbers $\{x^n_{0:T}, a^n_{1:T}\}_{n=1}^N$.
3 Re-write the likelihood estimator $\widehat{z}_\theta$ as a deterministic function of the particle system $\{x^n_{0:T}, a^n_{1:T}\}_{n=1}^N$, and use conventional optimization to find its maximizing argument θ_k.
4 **end**

2 Background on particle filtering

The particle filter was originally proposed as a Monte Carlo solution to the state filtering problem (Gordon et al. 1993), i.e., to compute $p_\theta(x_t \mid y_{1:t})$. It was soon (Kitagawa 1996) realized that the particle filter could also be used to *estimate the likelihood function* for given parameter values, essentially by inserting the particles (Monte Carlo samples) into the integral in (3) and thereby obtain an estimate $\widehat{z}_\theta$ of $p_\theta(y_{1:T})$. Thanks to this likelihood estimate, the particle filter can be used for system identification purposes.[1] As mentioned, $\widehat{z}_\theta$ is unbiased, but it often has a heavy-tailed and

[1] There are also several alternative ways in which the particle filter can be used for system identification, for example approaches based on the EM algorithm or Gibbs sampling.

asymmetric distribution with a non-negligible variance. Its exact properties depends on the particle filter settings and the model.

A textbook introduction to particle filters is given in, e.g., Doucet, Freitas, et al. (2001). We summarize a rather general formulation of the particle filter in Algorithm 2, a version of the auxiliary article filter (Pitt and Shephard 1999). We use $q(x_t \mid x_{t-1}, y_t)$ to denote an almost arbitrary[2] proposal distribution. Furthermore, v_t^n are the resampling weights, w_t^n the importance weights, and $C(\{v_t^n\}_{n=1}^N)$ denotes the categorical distribution on the set $\{1, \ldots, N\}$ with (possibly unnormalized) weights $\{v_t^n\}_{n=1}^N$, and N is the number of particles.

Let us list all stochastic elements ('draws') of Algorithm 2: Particles are initially drawn from the initial distribution $p(x_0)$ on line 1 (which we assume to be independent of θ). Furthermore are the ancestor indices a_t^n on line 3 drawn with respect to the resampling weights v_t^n, and for the propagation of particles x_t^n on line 4 are the new particles drawn from the proposal $q(x_t \mid x_{t-1}, y_t)$.

Whereas $f_\theta(x_t \mid x_{t-1})$, $g_\theta(y_t \mid x_t)$ and $p(x_0)$ in Algorithm 2 are given by the model specification and w_t^n follows from the algorithm, the choices for $q(x_t \mid x_{t-1}, y_t)$, v_t^n and N are left to the user. The number of particles N is usually taken as large as the computational budget permits, and two common choices for proposals and resampling weights

- the *bootstrap particle filter* with

  $$q(x_t \mid x_{t-1}, y_t) = f_\theta(x_t \mid y_t)$$

  and

  $$v_t^n = w_t^n,$$

  and consequently $w_t^n = g_\theta(y_t \mid x_t^n)$. This choice is generic and requires very little from the user, but has inferior performance compared to

- the *fully adapted particle filter* with

  $$q(x_t \mid x_{t-1}, y_t) = p_\theta(x_t \mid x_{t-1}, y_t)$$

  and

  $$v_t^n = p_\theta(y_t \mid x_{t-1}).$$

  This choice is superior to the bootstrap choice in terms of variance of the obtained approximation, but is only available for a quite limited set of models. The literature on approximations to this choice is therefore rich (e.g., Doucet, Godsill, et al. 2000; Naesseth, Lindsten, et al. 2015).

We will in this paper exploit the relatively large freedom that is available when it comes to choosing the proposal density and the resampling weights—by making a choice that only depends on a *current* parameter value θ_{k-1} it is possible to evaluate the likelihood estimator (which we will denote by $\widehat{z}_{\theta_{k-1}}(\theta)$) for *new* values of θ, while at the same time making use of the *same* realization of $\{x_{0:T}^n, a_{1:T}^n\}_{n=1}^N$.

[2] The support of q has to cover the support of f_θ.

Algorithm 2: The auxiliary particle filter

1 Draw $x_0^n \sim p(x_0)$ and set $w_0^n = 1, v_0^n = 1$.
2 **for** $t = 1$ **to** T **do**
3 Draw $a_t^n \sim C(\{v_{t-1}^j\}_{j=1}^N)$.
4 Propagate $x_t^n \sim q(x_t \mid x_{t-1}^{a_t^n}, y_t)$.
5 Set $w_t^n \leftarrow \frac{w_{t-1}^{a_t^n}/\sum_{j=1}^N w_{t-1}^j}{v_{t-1}^{a_t^n}/\sum_{j=1}^N v_{t-1}^{a_t^j}} \frac{f_\theta(x_t^n \mid x_{t-1}^{a_t^n})}{q(x_t^n \mid x_{t-1}^{a_t^n}, y_t)} g_\theta(y_t \mid x_t^n)$.
6 Set $z_t \leftarrow \frac{1}{N}\sum_{n=1}^N w_t^n$.
7 **end**
8 Compute $\widehat{z}_\theta = \prod_{t=1}^T z_t$.

All statements with n are for $n = 1, \ldots, N$.

3 Related work

The use of the likelihood estimator $\widehat{z}_\theta$ as an objective function in optimization, in the search for a maximum likelihood estimate $\widehat{\theta}$, has been subject to several studies: Doucet and Tadić (2003) differentiate $\widehat{z}_\theta$ and use it in a stochastic gradient descent scheme, whereas Dahlin and Lindsten (2014) and Wills and Schön (2017) use an optimization scheme based on Gaussian processes. Malik and Pitt (2011) use a fixed random seed and run the particle filter for different θ. For a fixed random seed, $\widehat{z}_\theta$ is indeed deterministic, however with discontinuous 'jumps' due to different resampling decisions being made for different θ. To this end, Malik and Pitt proposes an approximative smoothing to obtain a continuous function.

The idea used in this paper—to make the ancestor indices $a_{1:T}^n$ and particles $x_{0:T}^n$ depending only on the current, or reference, parameter value θ_{k-1} (instead of θ)—has been theoretically analyzed by Le Gland (2007), but has (to the best of the authors knowledge) not been applied in the iterative setting we propose. The work by Le Gland concerns central limit theorems for the likelihood estimator $\widehat{z}_{\theta_{k-1}}(\theta)$, and its application to our work is subject for further studies. The work presented in this paper shares similarities with the authors recent other work V, which however is concerned with the different topic of Bayesian identification for state-space models with highly informative observations. There are also potentially interesting connections to the recent work on using particle filters as proposal distributions in variational inference (Anh Le et al. 2018; Maddison et al. 2017; Naesseth, Linderman, et al. 2018), where the problem of parameter optimization conditional on a stochastic realization is similar to our setting.

The Expectation Maximization (EM) algorithm is another iterative approach to maximum likelihood estimation in nonlinear state-space models (Lindsten 2013; Olsson et al. 2008; Schön et al. 2011). However, despite serious efforts, we have not been able to make a connection between our proposed method and the EM algorithm.

4 The proposed solution

The key idea in this paper is to choose $q(x_t \mid x_{t-1}, y_t)$ and ν_t^n such that they are independent of θ. By such a choice, we note that all the random elements in Algorithm 2, $\{x_{0:T}^n, a_{1:T}^n\}_{n=1}^N$, also become independent of θ. If we then condition on a certain realization of $\{x_{0:T}^n, a_{1:T}^n\}_{n=1}^N$, the estimate $\widehat{z}_\theta$ becomes a deterministic function in θ, and any standard optimization routine can subsequently be applied to solve (3) and find $\widehat{\theta}$.

The strength of the particle filter, however, lies in the sequential build-up of the samples on the high-dimensional space X^{T+1}, where the resampling operation provides important 'feedback' on which parts of the state space to be explored further. With an arbitrary choice of θ-independent resampling weights ν_t^n, this feature will be lost, and we may expect an extremely high variance in the obtained estimate. In fact, a particle filter with θ-independent resampling weights ν_t^n can be understood as importance sampling on the space X^{T+1}, and we can in general not expect such an approach to be successful.

In order to obtain a deterministic function in θ, but avoid the bad consequences of a θ-independent resampling, we propose to let the resampling weights ν_t^n and proposal $q(x_t \mid x_{t-1}, y_t)$ depend on some current parameter estimate θ_{k-1}, as, e.g.,

$$q(x_t \mid x_{t-1}, y_t) = f_{\theta_{k-1}}(x_t \mid y_t) \tag{4a}$$

and

$$\nu_t^n = g_{\theta_{k-1}}(y_t \mid x_t^n), \tag{4b}$$

i.e., the bootstrap choice for θ_{k-1} (instead of θ). If then θ is somewhat close to θ_{k-1}, we can expect the variance of the corresponding estimate of the likelihood function, which we denote by $\widehat{z}_{\theta_{k-1}}(\theta)$, not to be forbiddingly large.

However, if the current θ_{k-1} is far from the sought $\widehat{\theta}$ (2), we cannot expect $\widehat{z}_{\theta_{k-1}}(\theta)$ to be a particularly good estimator at the value $\widehat{\theta}$, and in particular, not expect the maximum of $\widehat{z}_{\theta_{k-1}}(\theta)$ to lie at $\widehat{\theta}$. For this reason, we have to iterate the parameter values over k to arrive in the vicinity of $\widehat{\theta}$. By inserting (4) into Algorithm 2, combined with an outer optimization loop as discussed, we arrive (after some re-arrangement) at Algorithm 3. For numerical reasons, we work with the logarithm of the likelihood function. The conceptual idea is illustrated in Figure 1.

4.1 Solving the maximization problem

On line 4 in Algorithm 3, $\arg\max_\theta \widehat{z}_{\theta_{k-1}}(\theta)$ is to be solved. Importantly, this is now a completely deterministic problem and it can be handled by any standard numerical optimization tool. We will in the experiments demonstrate this by applying the general-purpose optimization tool `fminunc` in Matlab[3] out-of-the-box.

The particular structure of $\widehat{z}_{\theta_{k-1}}(\theta)$ (defined implicitly in the function `likelihood` in Algorithm 3) could possibly be utilized by a more tailored optimization scheme. Its structure can be written as

$$\widehat{z}_{\theta_{k-1}}(\theta) = \frac{1}{N} \prod_{t=1}^{T} \sum_{n=1}^{N} c_t^n \; \omega_t^n(\theta) \; f_\theta(x_t | x_{t-1}^{a_t^n}) g_\theta(y_t | x_t^n), \tag{5}$$

[3]Similar functions in other languages are `fminunc` (Octave), `scipy.optimize` (Python), `optim` (R) and `optimize` (Julia).

Algorithm 3: Proposed method

Set θ_0
for $k = 1, \ldots$ **do**
  Call $\{x_{0:T}^n, a_{1:T}^n\}_{n=1}^N \leftarrow$ `particle_filter`(θ_{k-1})
  Solve $\theta_k \leftarrow \arg\max_\theta$ `log_likelihood`$(\theta, \theta_{k-1}, \{x_{0:T}^n, a_{1:T}^n\}_{n=1}^N)$ using an off-the-shelf optimization routine.
end

Function `particle_filter`(θ_{k-1})
  Draw $x_0^n \sim p(x_0)$ and set $w_0^n = 1$.
  for $t = 1$ **to** T **do**
    Draw $a_t^n \sim C(\{w_{t-1}^j\}_{j=1}^N)$.
    Propagate $x_t^i \sim f_{\theta_{k-1}}(x_t \mid x_{t-1}^{a_t^n}, y_t)$.
    Set $w_t^n \leftarrow g_{\theta_{k-1}}(y_t \mid x_t^n)$.
  end
  return $\{x_{0:T}^n, a_{1:T}^n\}_{n=1}^N$

Function `log_likelihood`$(\theta, \theta_{k-1}, \{x_{0:T}^n, a_{1:T}^n\}_{n=1}^N)$
  for $t = 1$ **to** T **do**
    Set $w_t^i \leftarrow \frac{w_{t-1}^{a_t^n} / \sum_j w_{t-1}^{a_t^j}}{(\star)} \frac{f_\theta(x_t \mid x_{t-1}^{a_t^n})}{f_{\theta_{k-1}}(x_t \mid x_{t-1}^{a_t^n})} g_\theta(y_t \mid x_t^n)$.
    Set $z_t \leftarrow \frac{1}{N} \sum_{n=1}^N w_t^n$.
  end
  return $\log \widehat{z}_{\theta_{k-1}}(\theta) \leftarrow \sum_{t=1}^T \log z_t$.

$(\star) = g_{\theta_{k-1}}(x_{t-1}^{a_t^n} \mid y_{t-1}) / \sum_j g_{\theta_{k-1}}(x_{t-1}^{a_t^j} \mid y_{t-1})$

where c_t^n is a constant that is independent of θ, $\omega_t^n(\theta)$ depends on θ but always fulfil $\sum_{n=1}^N \omega_t^n(\theta) = 1$, and f_θ and g_θ depends on the model. Whether this function exhibits any particular properties that could be exploited in optimization is subject to further research.

5 Analysis

We will now present a brief analysis of the proposed Algorithm 3. First, we conclude in Section 5.1 that the proposed scheme has desirable asymptotic properties as $N \to \infty$. Second, we make an attempt in Section 5.2 to understand the behavior also for finite N, and third in Section 5.4 we discuss an alternative solution that would place the proposed method in the framework of stochastic gradient descent methods.

5.1 Convergence as $N \to \infty$ and $k = 1$

Asymptotically as $N \to \infty$ in the particle filter, the proposed method becomes exact and converges (in principle) in one iteration. This can be realized as follows: The

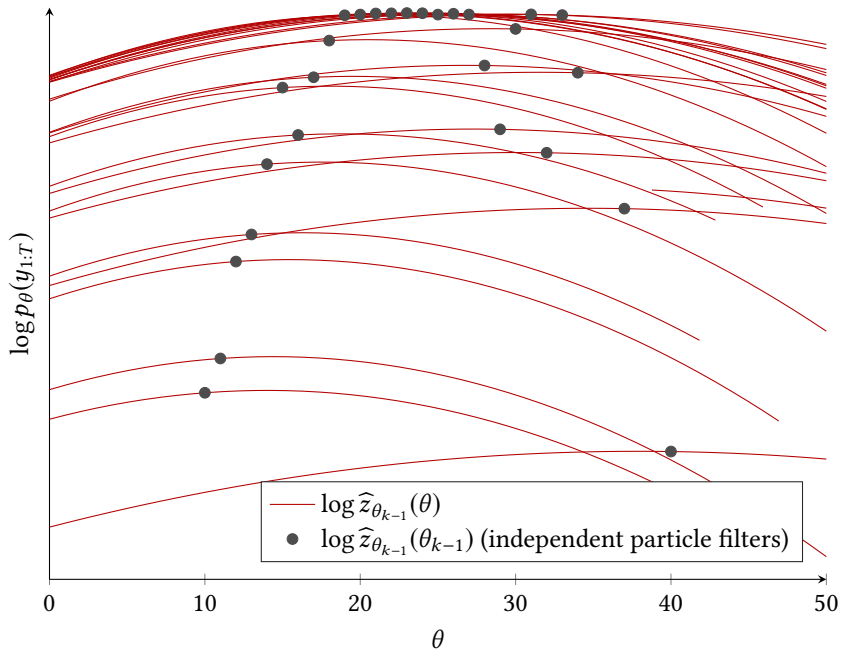


Figure 1: *The gray dots are log likelihood estimates obtained by running individual bootstrap particle filters with $N = 100$ particles and parameter value θ_{k-1}. Conditioned on the particle system underlying each particle filter (gray dots), the likelihood function is approximated also for other θ values (red lines), which we can expect to be practically useful in the vicinity of θ_{k-1}. The idea of Algorithm 3 is as follows: Start with some θ_{k-1} and sample a corresponding gray dot (*`particle_filter`*), and then apply a standard optimization scheme to find the maximum of the corresponding red line (*`log_likelihood`*). We save the result as θ_k, and start over again with a new particle filter for θ_k, etc.*

log-likelihood is estimated by

$$\log p_\theta(y_{1:T}) \approx \sum_{t=1}^{T} \log \left(\sum_{i=1}^{N} w_t^n p(y_t \mid x_t^n) \right). \tag{6}$$

Assuming that the proposal distribution used to generate the particles $\{x_t^n\}_{t=1}^T$ is everywhere non-zero, this log-likelihood approximation is consistent under weak conditions and converges point-wise in θ as $N \to \infty$ (Del Moral 2004; Doucet and Johansen 2011). Thus, as long as the global solution to $\arg\max_\theta$ can be found, it is indeed the likelihood that has been maximized and $\widehat{\theta}$ found, which happens in a single iteration $k = 1$.

5.2 Convergence as $k \to \infty$ and finite N

It is indeed reassuring that our proposed scheme is consistent as the number of particles $N \to \infty$, as discussed above. For practical purposes, however, the behavior for a finite N (which always is the case in an implementation) is probably more interesting.

We start by noting that for $N < \infty$, it holds that $\mathbb{E}\left[\widehat{z}_{\theta_{k-1}}(\theta)\right] = p_\theta(y_{1:T})$ (Del Moral 2004). Note, however, that this does *not* imply

$$\mathbb{E}\left[\arg\max_\theta \widehat{z}_{\theta_{k-1}}(\theta)\right] = \arg\max_\theta p_\theta(y), \tag{7}$$

so we do not have a theoretical justification to simply average the obtained sequence $\{\theta_k\}$ to obtain $\widehat{\theta}$.

To obtain some guidance on how to extract a final estimate from the obtained sequence $\{\theta_k\}$, we can make the simplifying assumption that the error $\log \widehat{z}_{\theta_{k-1}}(\theta) - \log p_\theta(y_{1:T})$, viewed as a stochastic process with index variable θ, is stationary. In such a case, we can (under some technical assumptions) expect that $\widehat{\theta}$ is equal to the maximum mode of the distribution for $\theta_k = \arg\max_\theta \log \widehat{z}_{\theta_{k-1}}(\theta)$. A proof sketch for this claim is found in Appendix A. This suggests that we should look at the maximum mode in the histogram of $\{\theta_k\}$ to find a good estimate of $\widehat{\theta}$ when we are using the method in practice (i.e., with a finite N). This will be illustrated in Example 2 and Figure 3c.

5.3 Stability

Stability, i.e., that the sequence $\theta_0, \theta_1, \ldots$ does not diverge, is another important property. We have not experienced any such issues with the proposed scheme. The key to a stable algorithm is that the solution to the maximization problem on line 4 in Algorithm 3 is often not too far away from θ_{k-1}. To motivate that this is likely to be the case, we note that while $\widehat{z}_\theta(\theta)$ is unbiased for all θ, its variance $\sigma^2(\theta)$ tends to increase as the distance between θ and θ_{k-1} increases. It is also known that $\log \widehat{z}_\theta(\theta)$ (rather than $\widehat{z}_\theta(\theta)$) has approximately a Gaussian distribution, which implies that $\log \widehat{z}_\theta(\theta)$ has a bias in the order of $-\sigma^2(\theta)$. This motivates that for θ far from θ_{k-1}, the value of $\log \widehat{z}_\theta(\theta)$ is likely to be small, and hence cause the algorithm not to deviate to much from the current iterate.

5.4 Stochastic gradient descent

An alternative approach would be not to solve the $\arg\max_\theta$ problem, but only use $\widehat{z}_{\theta_{k-1}}(\theta)$ to estimate a gradient around θ_{k-1} and take an (inevitable stochastic) gradient step. Indeed, this has already been proposed by Doucet and Tadić (2003). Designing step lengths based on stochastic approximation ideas (Robbins and Monro 1951) yields the well-studied stochastic gradient descent method. Our practical experience, however, is that (stochastic) gradient steps have inferior performance compared to the proposed $\arg\max_\theta$ scheme for our problem, including slower convergence and severe stability issues.

6 Numerical experiments

We will in this section apply our proposed method to two simulated examples, in order to illustrate and evaluate it. First a common example form the literature will be considered, and comparisons to alternative methods made. Thereafter a more difficult example will be studied. The source code is available via the first author's homepage.

6.1 Example 1

In this first example, we consider $T = 100$ measurements generated by the model

$$x_{t+1} = 0.5x_t + b\frac{x_t}{1+x_t^2} + 8\cos(1.2t) + qw_t, \tag{8a}$$

$$y_t = 0.05x_t^2 + e_t, \tag{8b}$$

where $w_t \sim \mathcal{N}(0,1)$, $e_t \sim \mathcal{N}(0,1)$, and $\theta = \{b, q\}$. The true values of θ are $\{25, \sqrt{0.1}\}$, and this example (with $q = \sqrt{0.1}$ and $\theta = b$) was also used to generate Figure 1. The proposed Algorithm 3 is implemented with $N = 100$, and employing the generic optimization routine `fminunc` in Matlab to solve the optimization problem on line 4 in Algorithm 3. The initial θ_0 is chosen randomly on the intervals $[10, 40]$ and $(0, 4]$, respectively, and the entire example is repeated 100 times. Each example took approximately 6 seconds on a standard laptop, and the results are found in Figure 2a. We compare with two alternative methods for maximum likelihood estimation in nonlinear state-space models: The results for particle stochastic approximation EM (PSAEM, Lindsten 2013) applied to the very same problem are reported in Figure 2b. The results for the same problem with a stochastic optimization approach using the particle filter to estimate the likelihood and a Gaussian process to model the likelihood surface and its derivatives (Mahsereci and Hennig 2015; Wills and Schön 2017) are found in Figure 2c. With the total number of iterations chosen as in the figures, the computational load are of the same order of magnitude for all three methods.

From this, we conclude that our proposed method tend to converge faster than the alternatives (counting the number of iterations needed), but that each iteration is computationally more involved. The computational load of our algorithm is partly due to the use of a generic optimization routine (`fminunc` in Matlab), which makes no use of the particular structure (5) of the objective function, as discussed in Section 4.1.

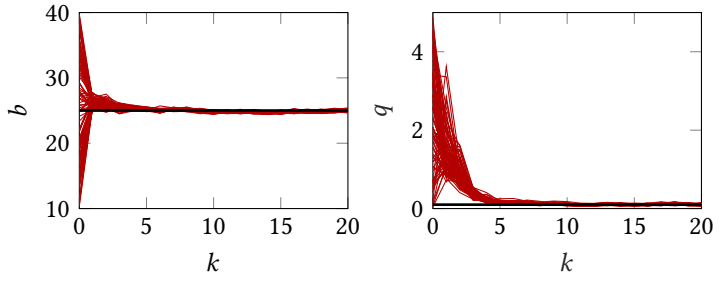
6.2 Example 2

Now, consider the following state-space model

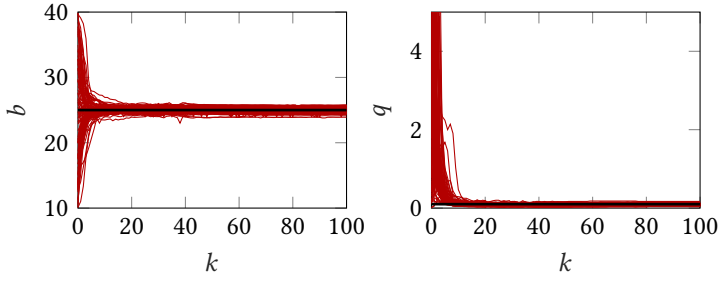
$$x_{t+1} = \frac{x}{a + x^2} + bu_t + w_t, \;\; w_t \sim \mathcal{N}(()\,0, 1), \tag{9a}$$

$$y_t = x + e_t, \qquad\qquad e_t \sim \mathcal{N}(()\,0, 1). \tag{9b}$$

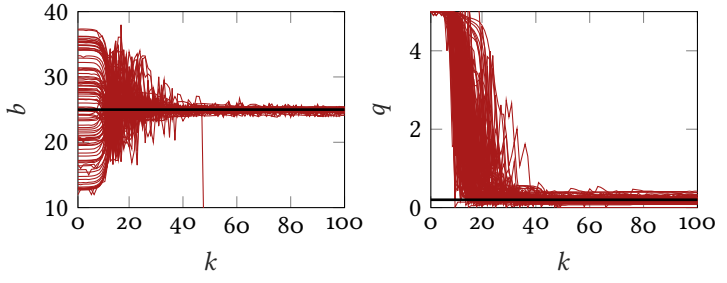
with $T = 1\,000$ and $\theta = \{a, b\}$. One data set is generated with $\theta = \{0.5, -2\}$, and our method is applied, with different initializations, 100 times to find $\widehat{\theta}$. This problem is significantly harder than Example 1 due to the location of a in the denominator (and not the numerator) in (9a). As an illustration, independent particles filter were run to estimate the log-likelihood for different values of a in Figure 3a, from which we conclude that the likelihood estimate is rather noisy. This can be compared to Example 1 and the gray dots in Figure 1, where the likelihood estimation is clearly less challenging. Again, we use the proposed Algorithm 3 with `fminunc` in Matlab to solve the optimization problem. The results are shown in Figure 3b and 3c. Despite the challenging likelihood estimation, our proposed method manages to eventually converge towards meaningful values, and following the guidance discussed in Section 5.2, we take the final estimates as the maximum of the histograms in Figure 3c, $\{0.59, -1.995\}$, which corresponds well to the true parameters.



(a) *The proposed method.*

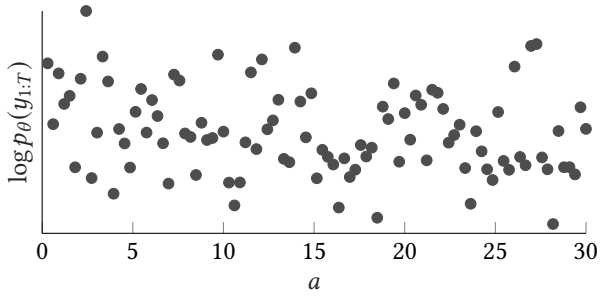


(b) *PSAEM (Lindsten 2013).*

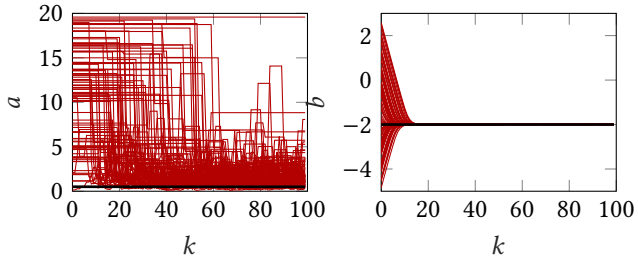


(c) *GP-based optimization (Wills and Schön 2017).*

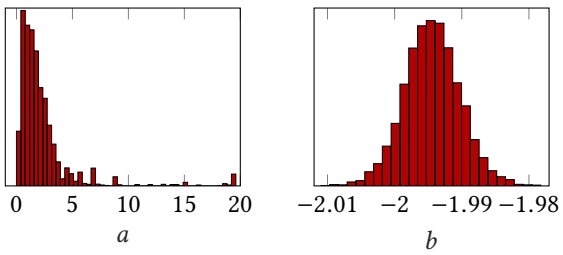
Figure 2: *Example 1. The results from each method is shown by red lines, and the true parameter values are shown in black. The practical convergence properties of the proposed method appears to be promising.*



(a) *Log-likelihood estimates (vertical axis) for the model* (9a) *in Example 2 for different a (horizontal axis, true* $a = 0.5$*) and* $b = -2$*, obtained with independent particle filters with* $N = 100$*. As can be seen, the variance in* $\widehat{z}_\theta$ *is rather high in this example, which is to be compared with the gray dots in Figure 1, the corresponding plot for Example 1. We thus expect maximum likelihood estimation to be significantly more challenging in this example.*



(b) *The results for our proposed method on Example 2. Our proposed method manages, despite the poor quality likelihood estimates (Figure 3a), to eventually converge towards sensible parameter values for a wide range of initializations. These traces are shown as histograms in the figure below.*



(c) *The traces from Figure 3b above as a histograms, after discarding the transient phase up to* $k = 50$*. Using the principle suggested in Section 5.2, the final estimate* $\widehat{\theta}$ *should be taken as the maximum of the histograms, i.e.,* $\{0.59, -1.995\}$ *, which corresponds well to the true parameter values* $\{0.5, -2\}$*.*

Figure 3: *Results for Example 2.*

7 Conclusions

We have proposed a novel method, Algorithm 3, to find maximum likelihood estimates of unknown parameters in nonlinear state-space models. The method builds on the particle filter, and allows for parameter inference in any model where also a particle filter can be applied. In fact, the method can be understood as a particular choice of θ-independent proposal and resampling weights in the auxiliary particle filter.

One key reason for the promising results is probably that we propose to solve an optimization problem at each iteration, instead of only taking a gradient step or similar: heuristically this seems to lead to fast convergence, and avoids instability issues. The theoretical analysis, however, becomes more involved. We have presented an attempt to such an analysis in Section 5.2, but all questions have not been answered. As mentioned, the work by Le Gland (2007) could potentially be useful in a more thorough analysis of the proposed method.

A tailored choice of optimization routine would be interesting for further work. Furthermore, the applicability of the proposed method for the particle marginal Metropolis-Hastings sampler (Andrieu et al. 2010) would be another interesting question.

A Appendix: Proof sketch

This is a sketch for a proof of the claim given in Section 5.2. Let θ_{k-1} be fixed and assume that $\varepsilon(\theta) = \log \widehat{z}_{\theta_{k-1}}(\theta) - \log p_\theta(y_{1:T})$ is a stationary stochastic process with index set $\theta \in \Theta$. For any $\theta' \in \Theta$ and $\delta > 0$, let $B_\delta(\theta')$ be a ball of radius δ centered at θ'. For notational simplicity, let $h(\theta) = \log p_\theta(y_{1:T})$ and note that this is a deterministic function of θ which is assumed to be Lipschitz continuous and attain its maximum for $\theta = \widehat{\theta}$. Now, take δ sufficiently small so that $\min_{\theta \in B_\delta(\widehat{\theta})} h(\theta) \geq \max_{\theta \notin B_\delta(\widehat{\theta})} h(\theta)$. For any θ' with $\|\theta' - \widehat{\theta}\| \geq \delta$ we then have

$$
\begin{aligned}
\mathbb{P}\left(\arg\max_\theta \{\varepsilon(\theta) + h(\theta)\} \in B_\delta(\theta')\right) &= \mathbb{P}\left(\max_{\theta \in B_\delta(\theta')} \{\varepsilon(\theta) + h(\theta)\} \geq \max_{\theta \in \Theta} \{\varepsilon(\theta) + h(\theta)\}\right) \leq \\
&\leq \mathbb{P}\left(\max_{\theta \in B_\delta(\theta')} \varepsilon(\theta) + \min_{\theta \in B_\delta(\widehat{\theta})} h(\theta) \geq \max_{\theta \in \Theta} \{\varepsilon(\theta) + h(\theta)\}\right) = \\
&= \mathbb{P}\left(\max_{\theta \in B_\delta(\widehat{\theta})} \varepsilon(\theta) + \min_{\theta \in B_\delta(\widehat{\theta})} h(\theta) \geq \max_{\theta \in \Theta} \{\varepsilon(\theta) + h(\theta)\}\right) \leq \\
&\leq \mathbb{P}\left(\max_{\theta \in B_\delta(\widehat{\theta})} \{\varepsilon(\theta) + h(\theta)\} \geq \max_{\theta \in \Theta} \{\varepsilon(\theta) + h(\theta)\}\right) = \mathbb{P}\left(\arg\max_\theta \{\varepsilon(\theta) + h(\theta)\} \in B_\delta(\widehat{\theta})\right)
\end{aligned}
$$

where the fourth step follows from the assumed stationarity of $\varepsilon(\theta)$. Now, since δ is arbitrary, it follows that if $X = \arg\max_\theta \{\varepsilon(\theta) + h(\theta)\}$ admits a density w.r.t. Lebesgue measure, its density function $p_X(x)$ is maximized at $x = \widehat{\theta}$.

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood (2018). "Auto-encoding sequential Monte Carlo". In: *International Conference on Learning Representations (ICLR)*. Vancouver, BC, Canada.

Johan Dahlin and Fredrik Lindsten (2014). "Particle filter-based Gaussian process optimisation for parameter inference". In: *Proceedings of the 19th IFAC World Congress*. Cape Town, South Africa, pp. 8675–8680.

Pierre Del Moral (2004). *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. New York, NY, US: Springer.

Arnaud Doucet, Nando de Freitas, and Neil J. Gordon (2001). "An introduction to sequential Monte Carlo methods". In: *Sequential Monte Carlo methods in practice*. Springer, pp. 3–14.

Arnaud Doucet, Simon J. Godsill, and Christophe Andrieu (2000). "On sequential Monte Carlo sampling methods for Bayesian filtering". In: *Statistics and Computing* 10.3, pp. 197–208.

Arnaud Doucet and Adam M. Johansen (2011). "A tutorial on particle filtering and smoothing: fifteen years later". In: *Nonlinear Filtering Handbook*. Ed. by D. Crisan and B. Rozovsky. Oxford, UK: Oxford University Press, pp. 656–704.

Arnaud Doucet and Vladislav B. Tadić (2003). "Parameter estimation in general state-space models using particle methods". In: *Annals of the Institute of Statistical Mathematics* 55.2, pp. 409–422.

Neil J. Gordon, David J. Salmond, and Adrian F.M. Smith (1993). "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". In: *IEE Proceedings F - Radar and Signal Processing*, pp. 107–113.

Genshiro Kitagawa (1996). "Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models". In: *Journal of Computational and Graphical Statistics* 5.1, pp. 1–25.

Francois Le Gland (2007). "Combined use of importance weights and resampling weights in sequential Monte Carlo methods". In: *ESAIM: Proc.* 19, pp. 85–100.

Fredrik Lindsten (2013). "An efficient stochastic approximation EM algorithm using conditional particle filters". In: *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vancouver, BC, Canada, pp. 6274–6278.

Chris J. Maddison, Dieterich Lawson, George Tucker, Heessm Nicolas, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Whye Theh (2017). "Filtering variational objectives". In: *Advances in Neural Information Processing Systems (NIPS) 31*. Long Beach, CA, USA.

Maren Mahsereci and Philipp Hennig (2015). "Probabilistic line searches for stochastic optimization". In: *Advances in Neural Information Processing Systems 28 (NIPS)*. Montréal, QC, Canada, pp. 181–189.

Sheheryar Malik and Michael K. Pitt (2011). "Particle filters for continuous likelihood evaluation and maximisation". In: *Journal of Econometrics* 165.2, pp. 190–209.

Christian A. Naesseth, Scott W. Linderman, Rajesh Ranganath, and David M. Blei (2018). "Variational sequential Monte Carlo". In: *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*. Lanzarote, Canary Islands, Spain, pp. 968–977.

Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön (2015). "Nested sequential Monte Carlo methods". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Lille, France, pp. 1292–1301.

Jimmy Olsson, Olivier Cappé, Randal Douc, and Éric Moulines (2008). "Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state-space models". In: *Bernoulli* 14.1, pp. 155–179.

Michael K. Pitt and Neil Shephard (1999). "Filtering via simulation: auxiliary particle filters". In: *Journal of the American Statistical Association* 94.446, pp. 590–599.

Herbert Robbins and Sutton Monro (1951). "A stochastic approximation method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407.

Thomas B. Schön, Adrian Wills, and Brett Ninness (2011). "System identification of nonlinear state-space models". In: *Automatica* 47.1, pp. 39–49.

Adrian Wills and Thomas B. Schön (2017). "On the construction of probabilistic Newton-type algorithms". In: *Proceedings of the 56th IEEE Conference on Decision and Control (CDC)*. Melbourne, Australia.

Title

Marginalizing Gaussian process hyperparameters using sequential Monte Carlo

Paper VII

Authors

Andreas Svensson, Johan Dahlin and Thomas B. Schön

Edited version of

Andreas Svensson, Johan Dahlin, and Thomas B. Schön (2015). "Marginalizing Gaussian process hyperparameters using sequential Monte Carlo". In: *Proceedings of the 6th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. Cancún, Mexico, pp. 489–492.

Digital identity

doi:10.1109/camsap.2015.7383840

Financial support

The Swedish Research Council (VR) via the project *Probabilistic modeling of dynamical systems* (contract number: 621-2013-5524).

Thanks to

Oscar Samuelsson and Dr. Jesús Zambrano for providing the sensor data in Section 3.3.

Marginalizing Gaussian process hyperparameters using sequential Monte Carlo

Abstract

Gaussian process regression is a popular method for non-parametric probabilistic modeling of functions. The Gaussian process prior is characterized by so-called hyperparameters, which often have a large influence on the posterior model and can be difficult to tune. This work provides a method for numerical marginalization of the hyperparameters, relying on the rigorous framework of sequential Monte Carlo. Our method is well suited for online problems, and we demonstrate its ability to handle real-world problems with several dimensions and compare it to other marginalization methods. We also conclude that our proposed method is a competitive alternative to the commonly used point estimates maximizing the likelihood, both in terms of computational load and its ability to handle multimodal posteriors.

1 Introduction

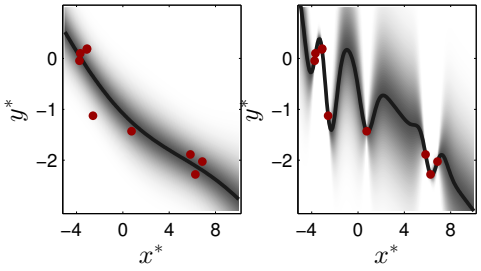
The Gaussian process (GP) is a non-parametric probabilistic model that can be used to model an unknown nonlinear function $f(\cdot)$ from observed input data x and (noisy) output data $y = f(x)$. No explicit form of $f(\cdot)$ is assumed, but some assumptions on $f(\cdot)$ are encoded through the GP prior and a mean function $m_\theta(x)$, a covariance function $\kappa_\theta(x, x')$, and their so-called hyperparameters $\theta \in \Theta$. In mathematical terms, f is a priori modeled to be distributed as

$$f(x) \sim \mathcal{GP}\Big(m_\theta(x), \kappa_\theta(x, x')\Big), \tag{1}$$

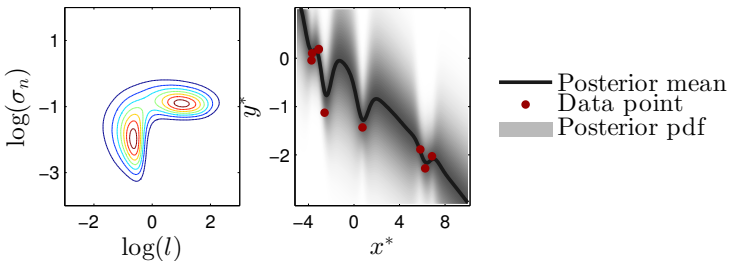
i.e., an infinite-dimensional Gaussian distribution. See Rasmussen and Williams (2006) for a more general introduction to GPs.

The posterior distribution over $f(\cdot)$ given data (y, x) is also a GP. This is due to the conjugacy property of the Gaussian distribution. The posterior is often greatly influenced by the choice of hyperparameters θ, which typically are unknown. We therefore propose a method to *marginalize* the hyperparameters in GPs. Marginalization can be seen as averaging over the range of hyperparameters supported by the data and by the prior; θ can be integrated out by treating it as a random variable with prior $p(\theta)$ and likelihood $p(y|x, \theta)$, giving rise to the posterior $p(\theta|y, x) \propto p(y|x, \theta)p(\theta)$. For example, the predictive distribution is computed by

$$p(y^*|x^*, y, x) = \int p(y^*|x^*, y, x, \theta)p(\theta|y, x)\mathrm{d}\theta, \tag{2}$$



(a) *Gaussian process regression for the data set defined by the red dots, using two different point estimates for the hyperparameters, each corresponding to a local minimum in (b, left).*



(b) *Left: the (multimodal) hyperparameter posterior conditional on the 9 data points. Right: the posterior using the proposed method (which marginalizes the hyperparameters, and thus handles the multimodality).*

Figure 1: *A small example illustrating the influence of the hyperparameters in the GP prior to the posterior estimate.*

which unfortunately is analytically intractable. However, using a Monte Carlo method to obtain N (weighted) samples $\{w^{(i)}, \theta^{(i)}\}_{i=1}^N$ of the distribution $p(\theta|y, x)$, the predictive distribution (2) can be approximated by

$$\widehat{p}(y^*|x^*, y, x) = \sum_{i=1}^{N} w^{(i)} p(y^*|x^*, y, x, \theta^{(i)}), \tag{3}$$

where the weights are normalized, i.e., $\sum_i w^{(i)} = 1$.

A common alternative to marginalization is to choose a point estimate of θ using an optimization procedure maximizing the likelihood $p(y|x, \theta)$ (sometimes referred to as empirical Bayes). This may be difficult if the likelihood is multimodal. See the small toy example in Figure 1 illustrating the robustness of marginalization compared to point estimates. There are also situations where point estimates are not sufficient, and marginalization is necessary, such as the change point detection problem in Section 3.3.

Our contribution is a method for sampling from the hyperparameter posterior distribution $p(\theta|y, x)$, based on sequential Monte Carlo (SMC) samplers. SMC samplers and their convergence properties are well studied (Whiteley 2012).

Several methods have previously been proposed in the literature for marginalization of the GP hyperparameters: Bayesian Monte Carlo (BMC) (Osborne et al. 2008), slice sampling (Agarwal and Gelfand 2005), Hamiltonian Monte Carlo (Neal 2011; Saatçi et al. 2010), and adaptive importance sampling (AIS) (Petelin et al. 2014). Particle learning which is closely related to SMC has been proposed by Gramacy and Polson (2011) for this purpose. The work by Gramacy and Polson, however, is not targeting the hyperparameters directly, and makes (possibly restrictive) assumptions on conjugate priors and model structure.

In this paper, we compare our proposed method to some of these methods, and apply it to two real-data problems: the first demonstrates that marginalization does not have to be more computationally demanding than finding point estimates. The second example, which deals with a fault detection problem from industry, is possible only with an efficient method for marginalization. Our proposed method (and all examples) are available as Matlab code via the first authors homepage[1].

From the experiments, we conclude that the advantages of the proposed method are (i) robustness towards multimodal hyperparameter posteriors, (ii) simplified tuning (compared to some other alternatives), (iii) competitive computational load, and (iv) online updating of hyperparameters as the data record grows.

2 Sampling hyperparameters using SMC

For the numerical marginalization (3), we require N samples, known as *particles*, from the posterior. In this section, we discuss how to use a SMC sampler (Del Moral et al. 2006) to generate such a particle system $\{\theta^{(i)}, w^{(i)}\}_{i=1}^{N}$, where $w^{(i)}$ is the weight of particle $\theta^{(i)}$. The underlying idea is to construct a sequence of probability distributions $(\{\pi_0, \ldots, \pi_P\})$, starting from the prior, and ending up in the posterior. The particles are then 'guided' through the sequence.

To construct a sequence $\{\pi_0, \ldots, \pi_P\}$, we use the fact that $p(\theta|y, x)$ depends on the data (y, x), by partitioning the data points into P disjoint batches $\{B_n\}_{n=1}^{P}$ and adding them sequentially as $\pi_n(\theta) \propto p(y_{B_{1:n}}|x_{B_{1:n}}, \theta)p(\theta)$.

To guide the particles through the smooth sequence $\{\pi_0, \ldots, \pi_P\}$, we will iteratively apply the three steps weighting, resampling and propagation, akin to a particle filter.

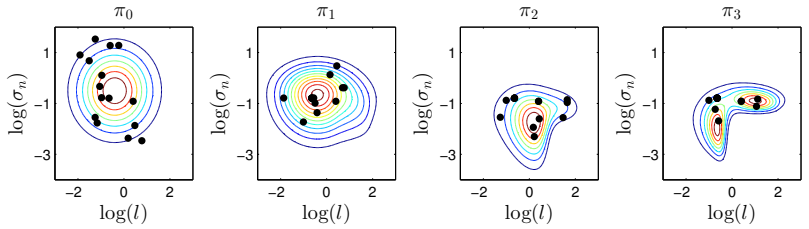
In the *weighting* step, the 'usefulness' of each particle is evaluated. To ensure convergence properties, the particles can be evaluated as (Del Moral et al. 2006, Section 3.3.2)

$$w_n^{(i)} = \frac{\pi_n(\theta_{n-1}^{(i)})}{\pi_{n-1}(\theta_{n-1}^{(i)})} w_{n-1}^{(i)}. \tag{4}$$

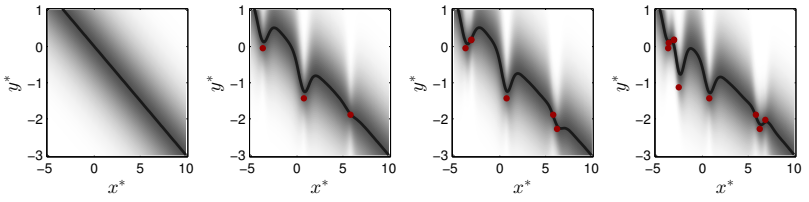
To avoid numerical problems, the particles have to be *resampled*. The idea is to duplicate particle with large weights, and discard particles with small weights.

To *propagate* the particles $\theta_{n-1}^{(i)}$ from π_{n-1} to π_n, a Metropolis-Hastings (MH) kernel $\mathcal{K} : \Theta \mapsto \Theta$ with invariant distribution π_n can be used. The procedure of propagating θ_{n-1} (a sample of π_{n-1}) to θ_n (a sample of π_n) by $\mathcal{K}$ is as follows:

[1] http://www.it.uu.se/katalog/andsv164



(a) *A transition from the prior $p(\theta)$ to the posterior $p(\theta|y, x)$ for the data in Figure 1b, obtained by adding 3 data points in each step to the likelihood. The particles are obtained from the SMC sampler.*



(b) *GP regression with marginalized hyperparameters from the corresponding posterior, obtained as a by-product from the particles depicted in (a). From left to right, 0 data points (i.e., the prior), 3 data points, 6 data points, and 9 data points. As we formulated the problem, only the rightmost figure is of interest. This illustrates however how this method can be used in online problem in a natural way.*

Figure 2: *A small example illustrating the influence of the hyperparameters in the GP prior to the posterior estimate.*

(i) Sample a new particle θ' from a proposal $q(\cdot|\theta_{n-1})$, e.g., a random walk with variance h. (ii) Compensate for the discrepancy between π_n and q by setting $\theta_n = \theta'$ with probability

$$\alpha(\theta_n, \theta') = \min\left\{1, \frac{\pi_n(\theta')}{\pi_n(\theta_n)} \frac{q(\theta_n|\theta')}{q(\theta'|\theta_n)}\right\}, \tag{5}$$

and otherwise $\theta_n = \theta_{n-1}$. To improve the mixing, this procedure can be repeated K times. For this, we use the notation $\theta_{n-1} = \theta_n^0 \to \theta_n^1 \to \cdots \to \theta_n^K = \theta_n$.

We now have an SMC sampler to obtain samples from the hyperparameter posterior, summarized in Algorithm 1 and illustrated by Figure 2. From the figure, the suitability to online applications is clear: If another data point is added to the data, the sequence can be extended to π_4 including the new data point, and only the transition from π_3 to π_4 has to be performed.

We make use of the adaptive SMC sampler by Fearnhead and Taylor (2013) in the numerical examples to adapt the proposal q automatically.

The computational cost of Algorithm 1 is in practice governed by the $2NPK$ evaluations of the likelihood $p(y|x, \theta)$. Hence, it is important to choose the number of samples N, SMC steps P, and MH-moves per SMC-step K sensibly. An idea of sensible numbers will be given along with the examples in the next section.

Algorithm 1: Hyperparameter posterior sampler

Input: Data (y, x), GP prior, and prior $p(\theta)$.
Output: N samples $\{\theta^{(i)}\}_{i=1}^{N}$ from $p(\theta|y, x) \propto p(y|x, \theta)p(\theta)$.
All statements with superscript (i) *are for* $i = 1, \ldots, N$.

Define $\pi_n(\theta) = p(y_{B_{1:n}}|x_{B_{1:n}}, \theta)p(\theta)$ by partitioning the data into P batches $\{B_n\}_{n=1}^{P}$.
Sample $\theta_0^{(i)}$ from $p(\theta)$ $(= \pi_0(\theta))$.
for $n = 1$ **to** P **do**
    Update weights according to (4).
    Resample $\{\theta_n^{(i)}, w_n^{(i)}\}_{i=1}^{N}$ if needed.
    for $k = 1$ *to* K **do**
        Propose $\theta'^{(i)}$ from $q(\theta'|\theta_n^{k-1,(i)})$.
        Set $\theta_n^{k,(i)} = \theta'^{(i)}$ with prob. $\alpha(\theta_n^{k-1,(i)}, \theta'^{(i)})$ (5).
    end
end

3 Examples and results

We consider three examples for demonstrating our proposed approach. First, we consider a small simulated example, also comparing to alternative sampling methods, and thereafter two applications with real-world data. The first real-data example is a benchmark problem to compare the marginalization approach in Algorithm 1 to the point estimates obtained using optimization. In the third example, we illustrate how we can make use of our solution within a GP-based online change point detection algorithm. To this end, we require marginalization of the hyperparameters, so an efficient hyperparameter posterior sampler is indeed a key enabler for this. The online nature of the problem also fits well to the possibility to update the samples in Algorithm 1 online, as discussed in Section 2.

3.1 Simulated example

We consider a small problem of 5 data points, and a covariance and mean function with 7 hyperparameters in total. We begin by considering the problem of marginalizing out 7 hyperparameters in a GP prior given 5 data points. Here, we are interested in comparing the performance of our SMC sampler (Algorithm 1) with some popular alternative methods; BMC (Osborne et al. 2008), AIS (Petelin et al. 2014), and (deterministic) griding.

The results for 15 runs are presented in Figure 3; it is indeed good if the variance between consecutive runs of the same algorithm gives similar results. The variations between the runs decrease faster for Algorithm 1 than for the comparable methods. When the GP prior has few hyperparameters, we conclude that the AIS and griding might be competitive methods. We have not managed to obtain competitive results with BMC for any problem size, but it should be noted that the computational load of BMC can be substantially decreased if the hyperparameter prior is independent between the dimensions.

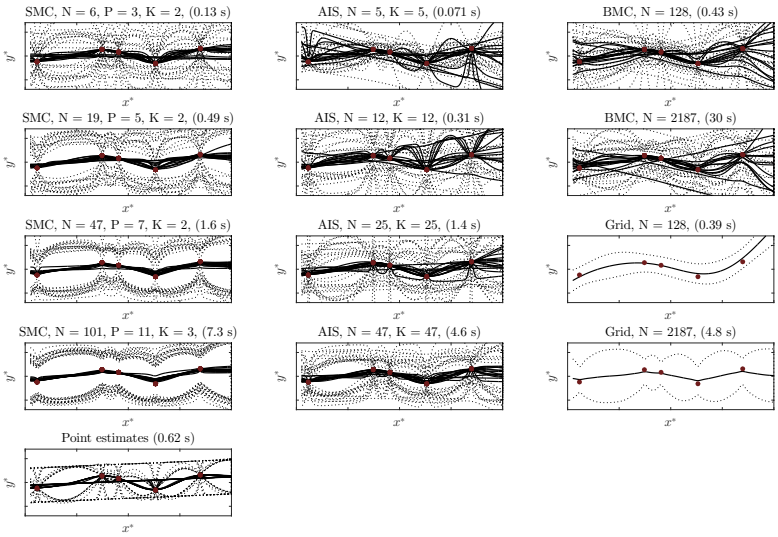


Figure 3: *Comparison between 15 runs of SMC (Algorithm 1), BMC, AIS, and griding, as well optimized point estimates. The predictions (mean, solid, and 3 standard deviations, dashed) are shown, together with the red data points. The number of particles/samples/grid points is denoted by N, while K and P are algorithm specific tuning parameters. The mean computation time is also shown. All axis are equally scaled.*

The quite 'messy' look in most of the plots indicates that the same method (with fixed settings) behaves differently on each run, which of course is an unwanted effect. However, the SMC sampler is not suffering from this problem for N, P, K large enough. This effect should also be expected for AIS and BMC, but apparently they need more samples/iterations (and thus computing time) than presented here before that effect can be seen.

The results for the conceptually different point estimates are also presented in Figure 3. The initialization point to the optimization algorithm is drawn from the prior: although it is a deterministic method, it is obviously very sensitive to the initialization.

3.2 Learning a robot arm model

We consider the problem of learning the inverse dynamics of a seven degrees-of-freedom SARCOS antromorphic robot arm (Rasmussen and Williams 2006; Vijayakumar and Schaal 2000). We use the same setup as Rasmussen and Williams (2006, Section 2.5), i.e., a non-trivial setting involving 23 hyperparameters.

To handle the size of the data set (44 484 training and 4 449 test data points), we make use of a subset of: (i) datapoints and (ii) regressors as discussed by Rasmussen and Williams (2006, Section 8.3.1). To use our method, we sample the hyperparameters from the posterior with a subset of m data points. For comparability, we have also reproduced the results using point estimates from Rasmussen and Williams (2006). The results are reported in Table 13.1. For Algorithm 1, $N = 15$, $P = 20$ and $K = 5$

Method	m	SMSE ($\times 10^{-2}$)	MSLL	Time (s)
Subset of datapoints				
Point est.	256	8.36 ± 0.80	-1.38 ± 0.04	6.8
SMC	**256**	**8.10 ± 1.32**	**-1.38 ± 0.56**	**7.1**
Point est.	512	6.36 ± 1.13	-1.51 ± 0.05	26.4
SMC	**512**	**6.13 ± 0.91**	**-1.49 ± 0.04**	**22.3**
Point est.	1024	4.31 ± 0.16	-1.66 ± 0.02	101
SMC	**1024**	**4.54 ± 0.33**	**-1.61 ± 0.03**	**92.5**
Point est.	2048	2.99 ± 0.08	-1.78 ± 0.03	423
SMC	**2048**	**3.33 ± 0.28**	**-1.69 ± 0.06**	**405**
Subset of regressors				
Point est.	256	3.67 ± 0.17	-1.63 ± 0.02	6.8
SMC	**256**	**3.55 ± 0.28**	**-1.65 ± 0.05**	**7.1**
Point est.	512	2.77 ± 0.44	-1.79 ± 0.07	26.4
SMC	**512**	**2.89 ± 0.20**	**-1.77 ± 0.03**	**22.3**
Point est.	1024	2.03 ± 0.11	-1.95 ± 0.03	101
SMC	**1024**	**2.00***	**-1.95***	**92.5**

Table 13.1: *Results for the SARCOS example in Section 3.2.*

was used. The priors to the logarithms of the length-scale and the signal variance are $\mathcal{N}(3, 3)$, and $\mathcal{N}(1, 1)$ for the noise variance.

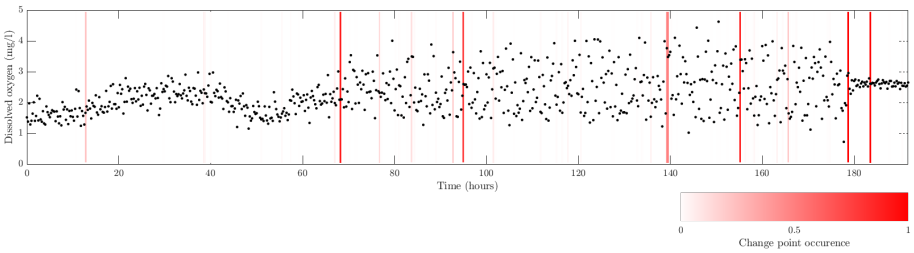
Table 13.1 presents the results in the same way as Rasmussen and Williams (2006, Table 8.1). SMSE is the standardized mean square error (i.e., mean square error normalized by the variance of the target), and MSLL is the mean standardized log loss; 0 if predicting using a Gaussian density with mean and variance of the training data, and negative if 'better'. The time is referring to the time required to sample and optimize the hyperparameters, respectively (not including the test evaluation). Numerical problems were experienced for large m, therefore * indicates runs where no interval can be reported.

Table 13.1 indicates no significant difference between the performance of our method and point estimates. It is however worth also to note the computational load: As Algorithm 1 apparently makes an equally good job in finding relevant hyperparameters as the optimization, it is a confirmation that our proposed method is indeed a competitive alternative to point estimates even for large problems.

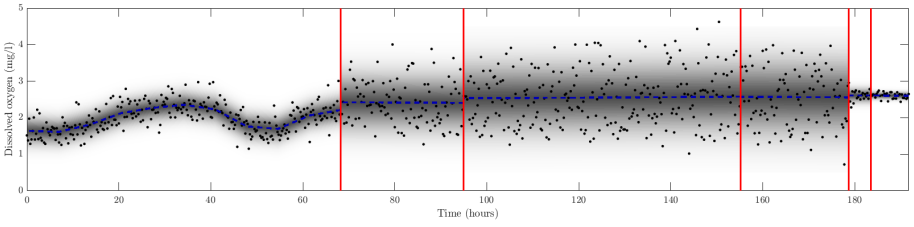
3.3 Fault detection of oxygen sensors

We now consider data from the wastewater treatment plant Käppalaverket, Sweden. An oxygen sensor measures the dissolved oxygen (in mg/l) in a bioreactor, but the sensor gets clogged because of suspended cleaning. The identification of such events is relevant to the control of wastewater treatment plants (Olsson et al. 2014). We apply the GP-based online change point detection algorithm by Saatçi et al. (2010), where the hyperparameters are marginalized using our proposed method.

The GP-based change point detection presented by Saatçi et al. (2010) can be summarized as follows: If data $y_{1:T}$ undergo a change at time r, it is of interest to



(a) *Measurements of dissolved oxygen (in mg/l) in a bioreactor with a sampling period of 15 minutes. The indicated change points are marked in red. Especially as the algorithm is fully Bayesian, the outcome is one probability distribution per data sample. This is comprehensively illustrated as the occurrence of change points in 'backwards simulations' through these distributions. A more intensive red color is a more likely change point.*



(b) *Left: the (multimodal) hyperparameter posterior conditional on the 9 data points. Right: the posterior using the proposed method (which marginalizes the hyperparameters, and thus handles the multimodality).*

Figure 4: *Results for the GP-based change point detection.*

(online) detect r, i.e., estimate $p(r|y_{1:t})$. The algorithmic idea is a recursive message passing scheme, updating the probability $p(r_t, y_{1:t})$, where $r_t \in \{1, \ldots, t\}$ is the last change point at time t.

To make predictions using a GP model, the hyperparameters either have to be fixed across all data segments, or marginalized. As it is not relevant to use fixed hyperparameters, an efficient sampling algorithm is a key enabler in solving this problem. The consecutive predictions $p(y_t|r_{t-1}, y_{r_t:t-1})$ and $p(y_{t+1}|r_{t-1}, y_{r_t:t})$ are both needed for the algorithm, hence our approach fit this problem well, as discussed in Section 2. We used $N = 25$ particles. On average, sampling the hyperparameters, i.e., one run of Algorithm 1, took 0.55 seconds on a standard desktop computer.

The results are presented in Figure 4a. The expected points, suspension and resuming of the cleaning, are indeed indicated. An interpretation of the result is obtained by converting the results to point estimates by thresholding, and plotting at the GP regression for each individual segment, see Figure 4b.

Note the data-driven nature of the algorithm, as no explicit model of the sensor was used at all. The tuning parameters are the covariance and mean functions, the prior of the change points and the hyperparameter priors.

4 Conclusion

We have proposed and demonstrated an SMC-based method to marginalize hyperparameters in GP models. The observed benefits are robustness towards multimodal posteriors (Figure 1) and a competitive computational load (Section 3.2), also compared to the commonly used point estimates of the hyperparameters. We have been able to cope with a hyperparameter space of dimension 23 (Section 3.2), and also concluded a sound convergence behavior (Section 3.1). Finally, the online update of the hyperparameters has been shown useful within the industry-relevant data-driven fault detection application (Section 3.3). As a future direction, it would be interesting to apply our method to the challenging GP optimization problem of system identification (Dahlin and Lindsten 2014).

References

Deepak K. Agarwal and Alan E. Gelfand (2005). "Slice sampling for simulation based fitting of spatial data models". In: *Statistics and Computing* 15.1, pp. 61–69.

Johan Dahlin and Fredrik Lindsten (2014). "Particle filter-based Gaussian process optimisation for parameter inference". In: *Proceedings of the 19th IFAC World Congress*. Cape Town, South Africa, pp. 8675–8680.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra (2006). "Sequential Monte Carlo samplers". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3, pp. 411–436.

Paul Fearnhead and Benjamin M. Taylor (2013). "An adaptive sequential Monte Carlo sampler". In: *Bayesian Analysis* 8.2, pp. 411–438.

Robert B. Gramacy and Nicholas G. Polson (2011). "Particle learning of Gaussian process models for sequential design and optimization". In: *Journal of Computational and Graphical Statistics* 20.1, pp. 102–118.

Radford M. Neal (2011). "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo*. Ed. by Steve Brooks, Andrew Gelman, Galin jones, and Xiao-Li Meng. Chapman & Hall/CRC Press.

Gustaf Olsson et al. (2014). "Instrumentation, control and automation in wastewater—from London 1973 to Narbonne 2013." In: *Water Science and Technology* 69.7, pp. 1373–1385.

Michael A. Osborne, Stephen J. Roberts, Alex Rogers, Sarvapali D. Ramchurn, and Nicholas R. Jennings (2008). "Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes". In: *Proceedings of the 7th international conference on information processing in sensor networks*. St. Louis, MO, USA, pp. 109–120.

Dejan Petelin, Matej Gašperin, and Václav Šmıdl (2014). "Adaptive importance sampling for Bayesian inference in Gaussian process models". In: *Proceedings of the 19th IFAC World Congress*. Cape Town, South Africa, pp. 5011–5015.

Carl E. Rasmussen and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. Cambridge, MA, USA: MIT Press.

Yunus Saatçi, Ryan D. Turner, and Carl E. Rasmussen (2010). "Gaussian process change point models". In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. Haifa, Israel, pp. 927–934.

Sethu Vijayakumar and Stefan Schaal (2000). "Locally weighted projection regression: Incremental real time learning in high dimensional space". In: *Proceedings of the 7th International Conference on Machine Learning (ICML)*. Stanford, CA, USA, pp. 1079–1086.

Nick Whiteley (2012). "Sequential Monte Carlo samplers: error bounds and insensitivity to initial conditions". In: *Stochastic Analysis and Applications* 30.5, pp. 774–798.