UPPSALA
UNIVERSITET

IT Licentiate theses
2016-011

Learning probabilistic models of dynamical phenomena using particle filters

ANDREAS SVENSSON

Uppsala University
Department of Information Technology

UPPSALA
UNIVERSITET

Learning probabilistic models of dynamical phenomena using particle filters

Andreas Svensson
andreas.svensson@it.uu.se

November 2016

Division of Systems and Control
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

Dissertation for the degree of Licentiate of Philosophy in Electrical Engineering
with specialization in Automatic Control

ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden

Abstract

Dynamical behavior can be seen in many real-life phenomena, typically as a dependence over time. This thesis studies and develops *methods* and *probabilistic models* for *statistical learning* of such dynamical phenomena.

A probabilistic model is a mathematical model expressed using probability theory. Statistical learning amounts to constructing such models, as well as adjusting them to data recorded from real-life phenomena. The resulting models can be used for, e.g., drawing conclusions about the phenomena under study and making predictions.

The methods in this thesis are primarily based on the particle filter and its generalizations, sequential Monte Carlo (SMC) and particle Markov chain Monte Carlo (PMCMC). The model classes considered are nonlinear state-space models and Gaussian processes.

The following contributions are included. Starting with a Gaussian-process state-space model, a general, flexible and computationally feasible nonlinear state-space model is derived in Paper I. In Paper II, a benchmark is performed between the two alternative state-of-the-art methods SMC^2 and PMCMC. Paper III considers PMCMC for solving the state-space smoothing problem, in particular for an indoor positioning application. In Paper IV, SMC is used for marginalizing the hyperparameters in the Gaussian-process state-space model, and Paper V is concerned with learning of jump Markov linear state-space models. In addition, the thesis also contains an introductory overview covering statistical inference, state-space models, Gaussian processes and some advanced Monte Carlo methods, as well as two appendices summarizing some useful technical results.

Thanks to Thomas Schön, Fredrik Lindsten, Johan Dahlin, Niklas Wahlström, Anna Wigren, Carl Andersson and Lawrence Murray for proofreading help and useful comments on the abstract and the introductory chapters. This research has partially been supported by the Swedish Foundation for Strategic Research (SSF) via the project ASSEMBLE.

Contents

List of Papers

The following published work is included in the thesis:

Paper I Andreas Svensson and Thomas B. Schön (2016a). "A flexible state space model for learning nonlinear dynamical systems". In: *Automatica*. Provisionally accepted.

Paper II Andreas Svensson and Thomas B. Schön (2016b). *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.

Paper III Andreas Svensson, Thomas B. Schön, and Manon Kok (2015b). "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. (Including the tech. rep. 2015-019 *Some details on state space smoothing using the conditional particle filter* from Department of Information Technology, Uppsala University, by the same authors.)

Paper IV Andreas Svensson, Johan Dahlin, and Thomas B. Schön (2015a). "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.

Paper V 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 following published work is of relevance to the thesis, but not included:

Paper A 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)*. Cadiz, Spain, pp. 213–221.

Paper B Andreas Svensson, Thomas B. Schön, Arno Solin, and Simo Särkkä (2015c). "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). Cancun, Mexico, pp. 493–496.

Paper C 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.

1 Introduction

Will this thesis be of any use to you? Well, if you ever find yourself in the following situation, there might be something relevant on the next 179 pages for you:

You have access to some data, consisting of numerical values which exhibits some dynamical behavior (i.e., there is some correlation between nearby data points). Your insights about the data source is also limited, and you would therefore like to have a mathematical model (i.e., a set of equations) true to the data, so that the model can be used to draw conclusions about the data generating mechanism.

This might happen, for instance, if you are in the business of, e.g., **automatic control** (modern control methodology often relies on a model that describes what would happen to the plant, if different actions are taken), **meteorology** (can the temperature record from last week tell us anything about what is likely to happen tomorrow?), **economics** (given the inflation rate and house prices, can we obtain any indication on when the next bubble will burst?), or simply is curious about some dynamical behavior you have seen.

Of course, statistical learning of dynamical behavior is a well-studied topic, and if this is the first time you come across the topic, you might prefer one of the several extensive textbooks that summarizes the researched made over the years (e.g., Box et al. 2015; Hastie et al. 2009; Ljung 1999; Pintelon and Schoukens 2012; Schön and Lindsten 2016; Söderström and Stoica 1989; Tsay 2010). This thesis is only a contribution to the field with a particular focus on how to make use of the computer-intensive sequential Monte Carlo algorithm, and makes no claim on covering the topic completely.

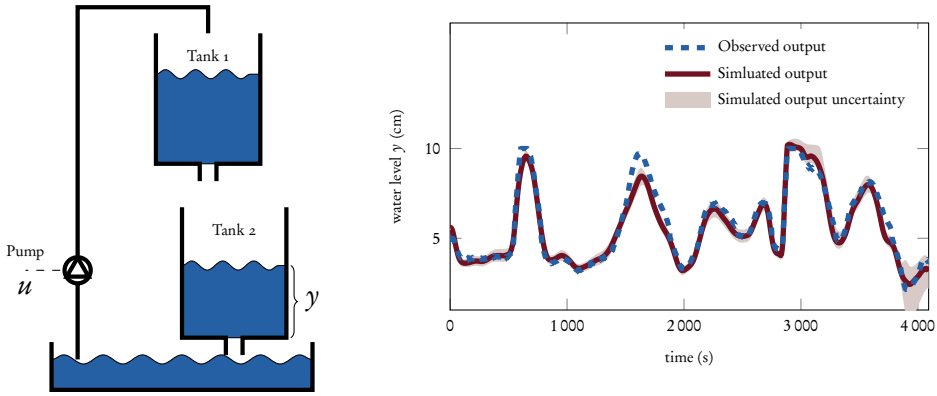


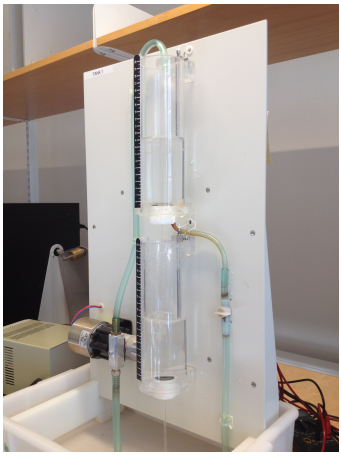
Figure 1.1. The water flow into tank 1 is controlled by the pump, which can be controlled by changing the voltage u (the input). (A picture is shown in Figure 1.2.) The water flows from tank 1 into tank 2, and from tank 2 into the basin, and *only the water level in the second tank is measured* as the output. Sometimes overflow happens in one or another of the tanks, and the overflow water may go partly from tank 1 into tank 2 and partly directly into the basin. First, a data set was recorded (not in the figure), with observations of u and y during one hour. This data set, called training data, was used by the algorithm designed in Paper I to learn a model of the system (without using any deeper physical insight into the experimental setup or fluid dynamics). Thereafter, another data set was recorded during another hour, of which only the input u (not in the figure) were fed into the model, and a *simulated output* (red line and light area in the plot) was obtained, which can be compared to the actual, measured, output (dashed blue). As can be seen, the model is able to predict the output rather well.

1.1 The papers: Contributions & a range of applications

In the following chapters and papers, the main focus will be on theory and algorithms. Unfortunately, the ultimate goal—its use in practice—will only be exemplified briefly at the very end of each paper. In order not to forget the motivation behind all work, we will kick off by a brief resume of the included papers, with a highlight on examples of how they can be applied. We will also in parallel review the research contribution of each papers.

Paper I

In Paper I, we devise a framework for learning a very flexible dynamical model from data. The key for working with the flexibility is to beforehand make assumptions about smoothness in the model, in order to help the model to extrapolate and generalize the behavior seen in the data in a sensible way. This is illustrated by the coupled water tanks in Figure 1.1. The contribution of the paper is twofold: the model itself (which can be seen as an approximate Gaussian-process state-space model), as well as its tailored learning algorithm.



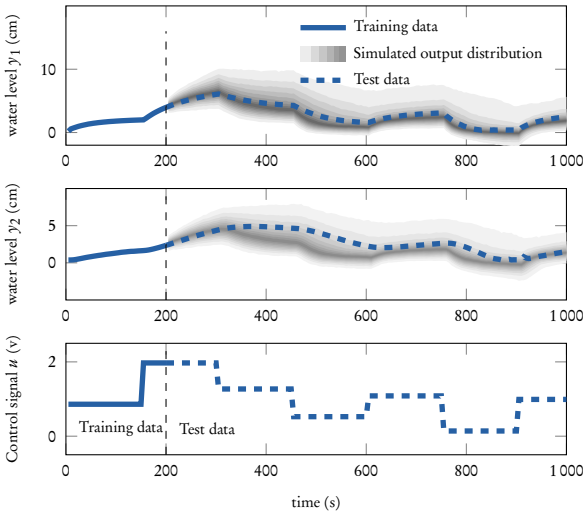


Figure 1.2. The same water tanks as in Figure 1.1, but this time (in Paper II), no overflow occurs, and we have measurements of the water level in both tanks, y_1 and y_2, respectively. In this paper, knowledge about the setup, fluid dynamics, etc., was used to devise a set of equations that could be expected to describe the system well. What remained unknown was a set of 6 parameters, representing some physical quantities, such as the size of the tanks and their outlets, etc. We used only the relatively short and seemingly not very informative data in the leftmost part of the plot (solid lines) as training data, to infer all unknown parameters. For the rest of the data set in the plot, the test data, only the input u was fed into the model, and the shaded areas were obtained as a simulated output. These may be compared to the actual measurements (dotted lines). *Photo: Maarten Schoukens.*

Paper II

Two different algorithms for finding unknown parameters in so-called state-space models are reviewed and compared in Paper II. We use the same water tanks as in Paper I, but with a slightly different setup, as described in Figure 1.2. The contributions in this paper lies in the comparison and application of the two state-of-the-art algorithms, none of which (to the best of the author's knowledge) yet has been extensively used for real world problems.

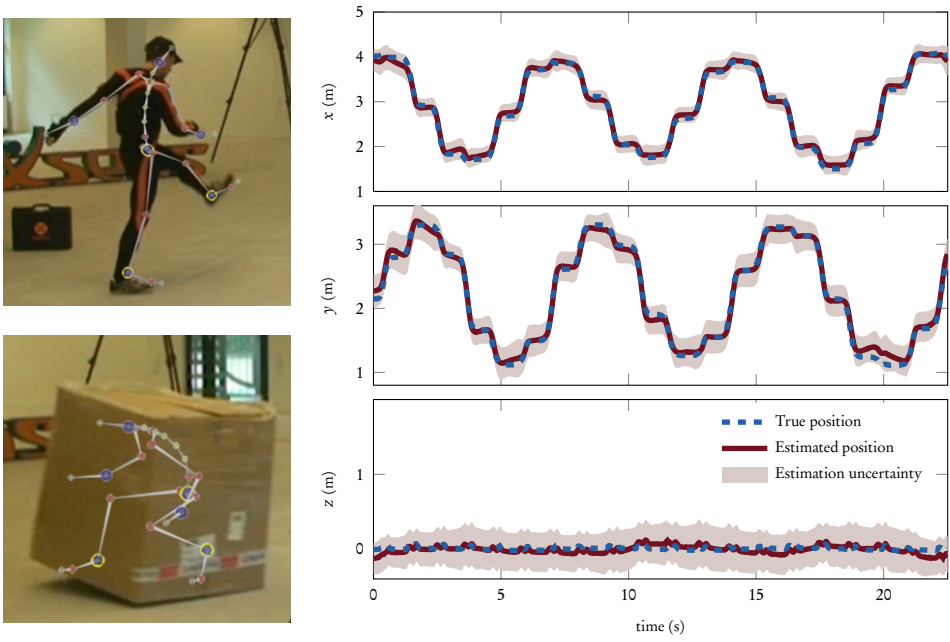


Figure 1.3. The position of a human foot is measured using ultrawideband measurements, a technology somewhat resembling the idea behind the GPS system: by accurately measuring the time the ultrawideband signal takes to travel from the sender (on the foot) to the receiver (at a fixed position), the distance between the sender and receiver can be determined. If placing many such receivers in the room at known positions, the position of the foot can be inferred 'backwards' by triangulation. All measurements are, however, corrupted by some noise, and in Paper III we consider a method capable of not only determining the foot position (solid red line), but also translating the noise in the measurements into uncertainty about the foot position, as shown in the shaded areas in the plot. The true position (dashed blue line) is measured as a reference with a much more expensive vision-based system (which cannot handle situations when there is no line-of-sight, as in the lower photo). *Photo courtesy of Xsens Technologies.*

Paper III

In Paper III, a method for determining the states in a nonlinear state-space model is considered. In the example described in Figure 1.3, this boils down to finding the position of a foot (on a human body), when measurements of the foot position are only available indirectly via ultrawideband measurements. The contribution of this paper is to consider and evaluate (both theoretically and on a realistic user case) the so-called particle Gibbs with ancestor sampling algorithm (Lindsten et al. 2014) for the particular case of state-space smoothing (when the model is known).

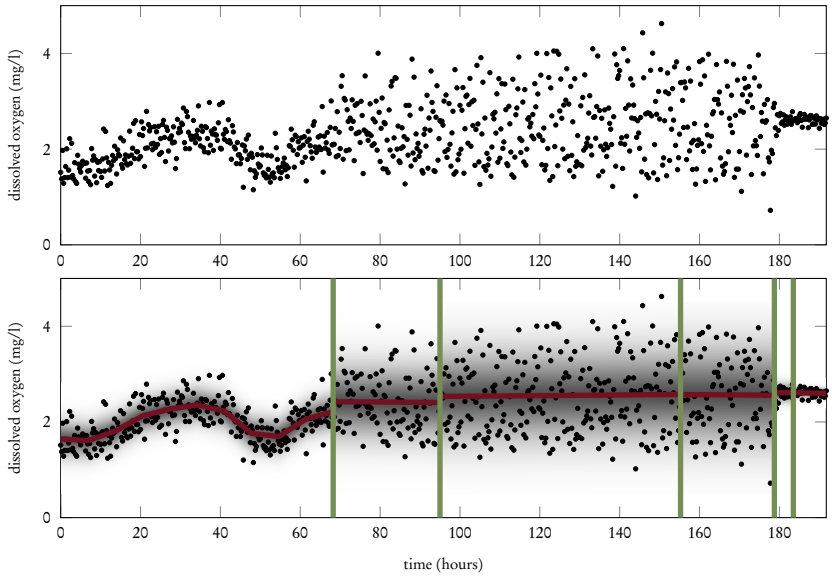


Figure 1.4. The upper plot shows measurements of dissolved oxygen recorded at a wastewater treatment plant. The cleaning of the sensor have, however, been suspended, and it gradually gets clogged. The challenge here is to automatically detect the clogging event, so that a warning can be issued to an operator. In Paper IV, we apply a change detection scheme to automatically subdivide the data into sequences of 'similar behavior' (without having any detailed technical knowledge about the sensor or the wastewater treatment plant). The segments obtained are illustrated by vertical green lines in the lower plot. The idea of 'similar behavior' within each segment is defined via a Gaussian-process model. The obtained models for the different segments are illustrated by the the red lines (the mean) and the shaded gray areas (the variance).

Paper IV

A method to marginalize hyperparameters in the Gaussian-process model is constructed in Paper IV. The perhaps most common way to handle the hyperparameters is the empirical Bayes approach, which (as discussed in the paper) can be very sensitive to small changes in data. With marginalization, instead, a more robust model can be obtained. This can be used for, e.g., data-driven change detection, as laid out in Figure 1.4. The contribution of the paper is to use an SMC sampler (Del Moral et al. 2006) to marginalize the hyperparameters in the Gaussian-process model.

$$
\begin{aligned}
s_{t+1} &= \varsigma \text{ with probability } \pi_{s_t,\varsigma} \\
x_{t+1} &= A_{s_t} x_t + B_{s_t} u_t + w_t, \quad w_t \sim \mathcal{N}(0, Q_{s_t}) \\
y_t &= C_{s_t} x_t + D_{s_t} u_t + e_t, \quad e_t \sim \mathcal{N}(0, R_{s_t})
\end{aligned}
$$

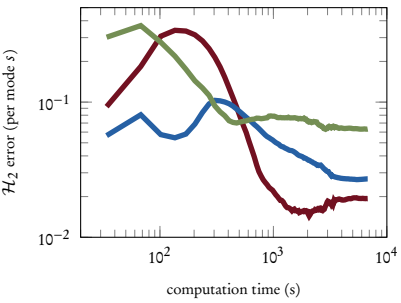


Figure 1.5. In Paper V, we derive a method for finding the parameters A_s, B_s, C_s, D_s, Q_s, R_s, $\pi_{s,\varsigma}$ for $s = 1, \ldots, K$ in the jump Markov state-space model equations to the left. It is an iterative method (i.e., it is repeated until it has reached a solution). The right plot shows how the estimated parameters (grouped per mode s, $K = 3$) approaches the true ones as the iterations goes on, for a simulated case.

Paper V

In Paper V, a method for finding the parameters of a jump Markov state-space model (Figure 1.5) is constructed. The jump Markov state-space model has a set of different modes, and behaves linearly within each mode: a deliberate generalization of the classical and very well-studied linear state-space model. The contribution of the paper is the application of the particle stochastic approximation EM algorithm (Lindsten 2013) to parameter estimation in the particular model.

1.2 Outline of the introductory chapters

The first half of the thesis contains five introductory chapters (including this chapter), one concluding chapter and two appendices with technical details. The purpose of the introductory chapters is to summarize the background and put the papers in a broader perspective. The concluding chapter, number 6, is meant to be read after reading the papers. The appendices contain some useful technical results (already existing in the literature), adapted to the notation and framework of this thesis.

After this first chapter, we will start in Chapter 2 by discussing the basic concepts of data, models and how to use data for deducing unknown quantities in a model, a topic called (statistical) inference. Two concrete models will thereafter be introduced, the state-space model in Chapter 3 and the Gaussian-process model in Chapter 4. We will thereafter return to the topic of inference again (Chapter 5) and focus on Monte Carlo methods for the computations that are required for doing inference. Appendix A contains a central result in the context of the so-called PMCMC method (Chapter 5), and Appendix B contains a derivation of the conjugate prior for linear regression: important expressions for Paper I.

1.3 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 71). The notation in the included papers is, however, slightly different, and introduced in each paper separately.

In general we use a probabilistic language and notation, and use the word 'model' mainly to refer to a probability distribution, which we assume somehow describes the phenomenon under study. 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 in focus. 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 integral variable.

2

Statistical learning: Data, models & inference

The topic of this thesis is statistical learning, for the special case of dynamical phenomena. We understand statistical learning as the *processing of recorded data into a statistical model*. The key elements in this process are the recorded *data* y (Section 2.1), the *model assumptions* $p(y \mid \theta)$ with some degrees of freedom expressed via a parameter θ (Section 2.2), and the *inference* which links the model to the data by drawing conclusions about θ from y (Sections 2.3–2.5). In this chapter, we will in the first place think of θ as being finite-dimensional and real-valued. However, in Chapter 4 we will consider a model where θ is infinite-dimensional.

The purpose of this chapter is to give an introduction to the way we will think about data and models, and also to cover some background of statistical inference in order to put the thesis in a context and perspective. This chapter will be kept on a rather general level mostly talking about data and models, and particular examples of such are mostly deferred to the following chapters and the papers. We will also leave integrals and optimization problems hanging in the midair without attempting to actually compute them, and refer to Chapter 5 for some Monte Carlo-based methods for these problems.

2.1 Data y

The first and foremost thing in all statistical methods is the data y. The data could in principle be anything that allows to be recorded, but in this thesis, we will 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 society, the data is recorded from some real phenomenon which is not yet completely understood. Examples of typical data could be logs with outdoor temperatures, measured forces in a mechanical system, or stock prices. We will 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 the thesis, we will assume the data is already recorded and available to us. Thus, the topic on how to design experiments and record useful data containing 'as much information as possible' is not treated in this thesis (see, e.g., Chaloner and Verdinelli 1995; Hjalmarsson 2009; Pukelsheim 1993).

Throughout the first part of this chapter, we will carry on with a toy example in order to illustrate the concepts. We now introduce the data to the toy example by saying that we have observed two data points $y_1 = 1.54$ and $y_2 = 3.72$.

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

Next, we introduce a model for the data y, which we denote by $p(y \mid \theta)$. With the notation $p(y \mid \theta)$, we mean a (probabilistic) description of the data y, possibly depending on some parameter θ. For the final result of the inference procedure to be useful, all present knowledge (and ignorance) about how the data was generated should be included in the model choice: if we suspect that the phenomenon under study exhibits some strong saturation effects, a linear model will perhaps not be useful. Also, if the present knowledge is very limited, a flexible model could be preferred in order not to be too limiting on beforehand, etc. In many cases, first principles (such as Newton's laws of motions, Leavitt's law or the ideal gas law) can be used to derive a model, typically known up to some (yet) undetermined constants or coefficients.

The abstract notion of a model is probably demystified most effectively by the toy example: let us say that we decide (*with or without* insight into where the data comes from) 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, i.e., $\theta \triangleq \{\mu, \sigma^2\}$ of the Gaussian distribution, and we write the model as

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

[1] We will 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.

Note that we have not, neither in the toy example nor elsewhere, limited ourselves to data actually generated by a system with exactly the same behavior as the model: the model is only an assumption within the inference procedure. We are, in fact, even free to make model assumptions that we know are inconsistent with how the data was generated (perhaps in the interest of feasible computations)! Inference results should, for this reason, *always be read with the model assumptions in mind*.

2.3 Two paradigms for deducing unknown parameters

In the model notation, we prepared for the inference itself 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 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 reasonings,

(i) which estimate $\widehat{\theta}$ fits 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 estimate* and (ii) the *Bayesian* approach. The distinction between the different paradigms is not always entirely clear in the literature, but we will below give an explanation of the way we will 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 inference approach we review, is that of finding a *point estimate* $\widehat{\theta}$ of θ that fits the observed data as good as possible. Which particular point estimate $\widehat{\theta}$ to choose, i.e., what 'fit' in the rhetoric question above means, 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 the likelihood function is a function of θ, whereas $p(y \mid \theta)$ is the likelihood of y (not θ). 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 point estimates 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 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, but 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 uses 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,

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

[3]Note that $p(y)$, the denominator, can be written as an integral over the numerator with respect to θ, $p(y) = \int p(y \mid \theta) p(\theta) d\theta$. If θ is the variable of interest, $p(y)$ 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.

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 the 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 acquire a useful result, the choice of prior should preferably reflect all present ignorance and knowledge, in the very same way as the model $p(y \mid \theta)$ preferably should be chosen with all present knowledge in mind.

It is indeed possible to extract point estimates of θ also from the posterior $p(\theta \mid y)$. Popular such estimates 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. 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 perform some algebraic manipulations, we find the posterior $p(\mu, \sigma^2 \mid y) = \mathcal{NIG}\left(\mu, \sigma^2; 1.75, 3, 2, 4.49\right)$, a distribution 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 even admit a closed form, and conjugate priors only exist for a limited set of models.

[4]The (inevitably subjective) prior choice is, according to the authors personal experience, one of the main criticism 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.*'

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. That a point estimate $\widehat{\theta}$ and a distribution $p(\theta \mid y)$ are very different objects is, instead, most likely the major difference for a user: A point estimate $\widehat{\theta}$ is a number, whereas $p(\theta \mid y)$ is a distribution. If the user interest, for example, is to predict a future observation $y^\star$, the point estimation approach is typically 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

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

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 alternatives are the variational approach (e.g., Blei et al. 2016) and Monte Carlo methods (Chapter 5).

Which approach to take, point estimates or Bayesian, may depend on several aspects. Often, but not always, is point estimation less computationally intense than the Bayesian approach, a reason that itself might be a ground 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)

Also personal preferences may of course 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 diminish. 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

[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.

exchangeable 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 manipulation 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 towards 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. It is relevant in many situations when $T \to \infty$, not only the 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. Albeit the prior has a central role in the Bayesian approach, but is not even present when doing point estimates, the differences sometimes diminishes 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))

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

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', a situation occurring in a wide class of models (clearly with exception of pathological cases with Dirac priors etc.) (J. O. Berger 1985, Section 4.7.8; Vaart 1998, Section 10.2).

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 in the following way:

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

where $\theta = \{n, c_0, \ldots, c_n, \sigma_n^2\}$. 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 wish 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, 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 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 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:

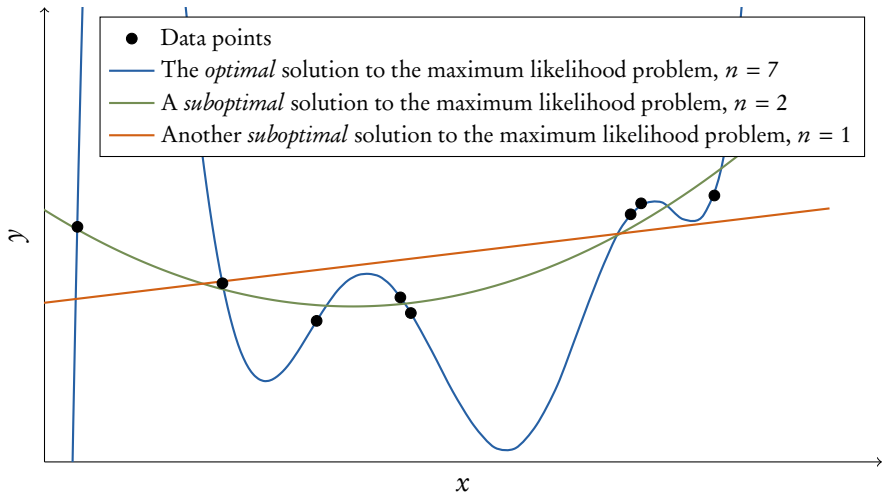


Figure 2.1. Eight data points marked with black dots, modeled using nth order polynomials and Gaussian noise. The optimal maximum likelihood solutions is $n = 7$, with the 8 polynomial coefficients chosen such that its blue curve fits the 8 data points perfectly. Two suboptimal solutions are $n = 2$ (green curve) and $n = 1$ (orange curve), which—despite their suboptimality in a maximum likelihood sense—both might appear to be a more sensible model, in terms of inter- and extrapolating the behavior seen in the data. The key aspect here is that the maximum likelihood finds the parameter explaining the data the best, exactly as it is seen: indeed, the blue curve fits the data perfectly. There is, however, *no* notion of 'likely parameters', as opposed to the Bayesian approach. The green and orange curves could 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).

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 maximum likelihood when we actually have different prior beliefs in different parameter values: we prefer[6] the predictable behavior of a low order polynomial to avoid the strange behavior of a higher order polynomial. Note, however, that we *could* have fixed the model class to $n = 2$, and searched for the maximum likelihood solution only among $\{c_0, \ldots, c_2, \sigma_2^2\}$. In such a case, the green line would have been the optimal solution, and we had not faced any problem with overfit. However, restricting the model to $n = 2$ would at the same time make the model more rigid and less flexible, with no possibilities of, e.g., describing any odd nonlinearities. To

[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.

summarize the example, we make the two observations:

- Maximum likelihood means searching for an estimate $\widehat{\theta}$ that maximizes the likelihood of the data y as it is exactly seen, which may differ significantly from any intuitive idea about 'likely parameters'.
- The (inevitably subjective) model choice plays a crucial role for the result.

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 (and indeed subjective) 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[7], an advantage that comes at the price of choosing a prior and working with probability distributions rather than point estimates.

Either inspired by the Bayesian approach or heuristically motivated, an increasingly 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. In the signal processing and machine learning literature, 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 $\mathcal{N}(\theta; 0, \sigma^2)$, the logarithm of the posterior becomes

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

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

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

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 constitutes another connection between the point estimation and the Bayesian approach.

[7]There are two different perspective one can take when understanding the non-existence of overfit 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 overfit since the posterior *by definition* represents our (subjective) beliefs about the situation, and therefore contains nothing but useful information (and hence no overfit to non-informative noise).

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 by 1960) regularization is probably of limited interest. However, if the model class under consideration is more complex[8] and contains a huge number of parameters, overfit 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 (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.

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 design $p(y \mid \theta)$) and thereby stay as objective as possible. Several alternatives for avoiding, or at least minimizing the influence of the prior choice, have therefore been researched.

'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 $p(\sigma^2) \propto 1$ for $\sigma^2 > 0$, it implies that $p(\sigma) \propto \sigma$ for $\sigma > 0$, a prior choice that does not appear very 'noninformative' at all.

To avoid this undesired effect, a prior that is invariant under re-parametrizations have 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*

[8] The study of model flexibility is the core in the statistical learning theory or Vapnik–Chervonenkis theory, which we will leave out from this thesis.

"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 informative than not performing any Bayesian inference at all.

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[9] as *empirical Bayes*, a popular and currently emerging method. 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., Bishop 2006; Efron 2013 and references therein).

Due to the point estimation, overfit may occur when using empirical Bayes, 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 hyperparameters, 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 defer it to the level $p(\eta)$ 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

[9] Another term sometimes seen is 'maximum likelihood type II'.

obtained for, e.g., the Gaussian-process (Chapter 4) model, even with relatively simple choices of hyperpriors: Heinonen et al. (2016), Shah et al. (2014) and Paper IV. 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 $p(\theta)$.

3
State space models

The state-space model is a popular and widely used model. In this chapter, we will introduce the general state-space model, and thereafter turn the focus to four important special cases, namely the linear, jump-Markov linear, and Wiener and Hammerstein state-space models. We also devote a section to discuss statistical inference in the particular context of state-space models.

3.1 The general state-space model

At the core of the state-space model is 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 the state transition function. We refer to x_t as the *state*, and $t = 0, \ldots, T$ is an index typically representing time in time-series data, but other interpretations are also possible. We assume that $x_t \in \mathbb{R}^{n_x}$ where n_x is the dimension of the state space.

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 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 key for the efficiency of several algorithms that we will introduce in the next chapter.

The state-space model also includes the observation function $g(\cdot \mid \cdot)$, 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 need to describe a density $p(x_0)$ for the initial state x_0. Once again, we have expressed the model in terms of its densities. This is merely a matter of notation, and the extension to degenerate models (such as a deterministic relationship between some components of x_t and x_{t+1}) is often possible.

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 $u_t \in \mathbb{R}^{n_u}$, a case explicitly considered in Paper I and V by adding a conditioning on u_t in $f(\cdot \mid \cdot)$ and $g(\cdot \mid \cdot)$. Another commonly seen flavor of (3.2) is the so-called 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 is the data $\{y_1, \ldots, y_T\}$ far from obeying the Markov assumption, and a state-space model is fitted to the data so that a (more or less artificial) state sequence $\{x_1, \ldots, x_T\}$ (with the Markov property) can be (re)constructed. 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 always exists, which accurately describes any data set recorded from the same process? 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 inference problems can be carried out relatively easy, if not on 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 popular 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, which we will discuss in the next sections, are the jump-Markov linear state-space models, 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 V develops a particular inference algorithm tailored for models on the form (3.4).

Various versions of jumping/switching linear models are present in the literature, of which (3.4) is one.

3.4 Wiener and Hammerstein models

Another common extension of (3.3) is the Wiener and Hammerstein models. The models are named after Norbert Wiener (1958) and Adolf Hammerstein (1930), respectively. These models preserve the linear dynamics of (3.3), but add a nonlinear transformation $h(\cdot)$ at either the input or the output, as

$$x_{t+1} = Ax_t + h(u_t) + w_t, \qquad w_t \sim \mathcal{N}(0, Q), \tag{3.5a}$$
$$y_t = Cx_t + Du_t + e_t, \qquad e_t \sim \mathcal{N}(0, R). \tag{3.5b}$$

for the Hammerstein model, and

$$x_{t+1} = Ax_t + Bu_t + w_t, \qquad w_t \sim \mathcal{N}(0, Q), \tag{3.6a}$$
$$y_t = h(x_t, u_t) + e_t, \qquad e_t \sim \mathcal{N}(0, R). \tag{3.6b}$$

for the Wiener model. In both models, $h(\cdot)$ is some nonlinear function $h : \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_u}$ or $h : \mathbb{R}^{n_x \cdot n_u} \mapsto \mathbb{R}^{n_y}$, respectively.

Both the Wiener and the Hammerstein model have received much attention in the literature, perhaps due to their versatileness while still avoiding nonlinear dynamics á la (3.2a) – a showstopper for many inference methods. The Wiener and the Hammerstein models lay the foundation of the block oriented system identification (Giri and Bai 2010; Schoukens et al. 2003), and the Wiener structure has also gained recent interest in the machine learning literature (Johnson et al. 2016). A reason for the usefulness of the Wiener model can perhaps be found in that a broad class of nonlinear state-space models can be well approximated by a Wiener model with a finite, however sometimes large, dimension n_x (Boyd and Chua 1985).

3.5 Statistical inference in state-space models

Due to the particular 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, II, III and V are all concerned with particular aspects of inference in state-space models.

3.5.1 Quantities to infer: states and model parameters

When we discussed inference 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 infer, 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 an inference perspective: they are both unknown quantities in the state-space model. However, depending on the user's case, different inference problems may be relevant. We will consider two alternative cases:

(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 inference*, 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 and the model parameter inference algorithms will have to be designed differently, in order to gain computationally feasible algorithms.

The model parameter inference problem contains a spectrum of settings, ranging from inference of a single parameter value to determining the entire functional forms of $f(\cdot \mid \cdot)$ or $g(\cdot \mid \cdot)$. In this thesis, Paper V 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. Inference concerning n_x is another problem, not considered in this thesis.

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

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

3.5.2 A Bayesian approach or point estimates?

Given the two inference 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 inference 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.

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 will 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 can be written (2.4) as

$$p(x_{1:T} \mid y_{1:T}) = \frac{p(y_{1:T} \mid x_{1:T})p(x_{1:T})}{p(y_{1:T})}. \tag{3.7}$$

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, (3.7) is of very limited use. Instead the recursion (see, e.g., Särkkä 2013)

$$p(x_t \mid y_{1:t}) = \frac{1}{p(y_t \mid y_{1:t-1})} g(y_t \mid x_t) \int f(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1} \tag{3.8}$$

[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.

has proven useful for computing the (marginal) posterior distributions $p(x_t \mid y_{1:t})$. The denominator in (3.8) 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 (3.8). We will refer to (3.8) 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 (see, e.g., Lindsten and Schön 2013; Särkkä 2013 for an overview) and Paper III.

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 (3.8) written down for the special case of the linear Gaussian state-space 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 likelihood for $y_{1:T}$ given ϑ, i.e., $p(y_{1:T} \mid \vartheta)$. When we later, in Chapter 5 will discuss numerical methods for model parameter inference, this expression will be at the center of attention.

$$p(y_{1:T} \mid \vartheta) = \prod_{t=1}^{T} p(y_t \mid y_{1:t-1}, \vartheta) = \prod_{t=1}^{T} \int p(y_t \mid x_{t-1}, \vartheta) p(x_{t-1} \mid y_{1:t-1}, \vartheta) dx_{t-1}, \tag{3.9}$$

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 estimate of ϑ is sought!

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

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

4

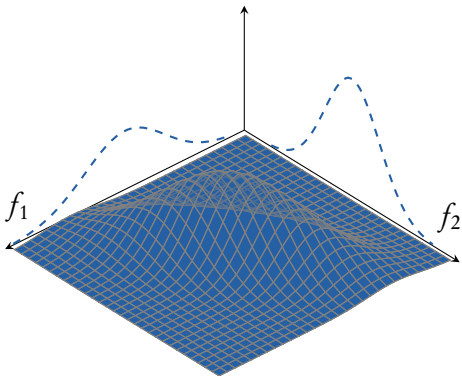
Gaussian processes

The Gaussian process (GP) defines a probability distribution over functions f, and is commonly used as a probabilistic model of functions. 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.

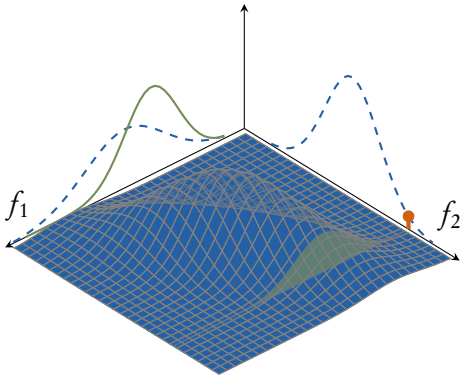
The GP is a so-called *nonparametric* model, in that it does not rely on a finite set of parameters θ. A parametric model involves a set of parameters θ acting as a 'mid-layer' between the data and the posterior over f, and finding the latter amount to first infer $p(\theta \mid y)$ and then $p(f \mid y) = \int p(f \mid \theta)p(\theta \mid y)d\theta$ (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', since there is no a priori limit on the data record size.

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).



(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.



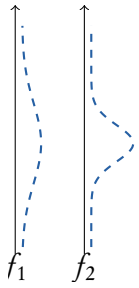
(b). The conditional distribution of f_1 (green line), when f_2 is observed (orange dot). The conditional distribution of f_1 is given by (4.3), which (apart from a normalizing constant) in this graphical representation also is the green 'slice' of the joint distribution (blue surface). The marginals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

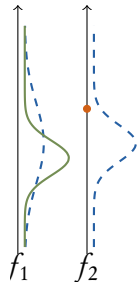
The density for the k-dimensional multivariate Gaussian distribution is

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

where $\overline{f} = [f_1 \ \cdots \ f_k]^\intercal$ 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 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 understood as the transition from the parametric Gaussian distribution to the nonparametric GP.



(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.

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

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

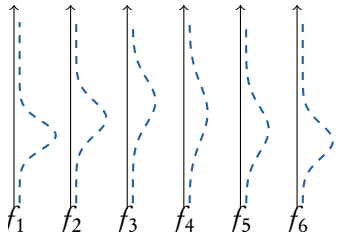
If some elements of $\overline{f}$, let us say the ones in $\overline{f}_2$, are observed, the conditional distribution for $\overline{f}_1$ given the observation of $\overline{f}_2$ is

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

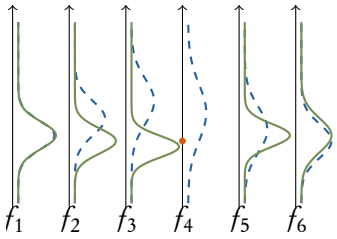
The conditional distribution is nothing but another Gaussian distribution with closed-form expressions for the mean and covariance. This is particularly useful.

Figure 4.1 shows a 2-dimensional example where a multivariate Gaussian distribution is conditioned on data. In Figure 4.2, we have now 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



(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 akin to Figure 4.2.

$\{1, 2, 3, 4, 5, 6\}$ 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 parameter 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 in the following way:

Definition (the Gaussian process)**.** *Let* $\{x_1, \ldots, x_n\}$ *be any 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 $\{x_1, \ldots, x_n\}$.

If we want to plot the GP, which we will 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). We will use the notation

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

to express this assumption, i.e., f is distributed according to a GP with mean function $\mu(\cdot)$ and covariance function $\kappa(\cdot, \cdot)$.

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,

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

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

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

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

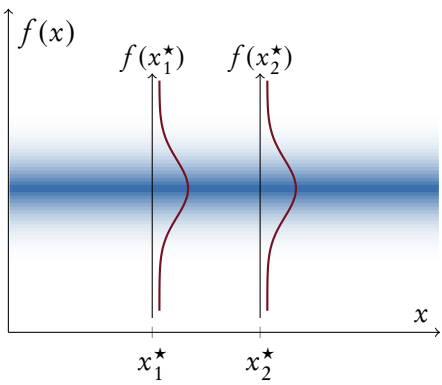
$$p\left(\begin{bmatrix} f(x^\star) \\ f(x^d) \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} f(x^\star) \\ f(x^d) \end{bmatrix}; \begin{bmatrix} \mu(x^\star) \\ \mu(x^d) \end{bmatrix}, \begin{bmatrix} K^{\star\star} & K^{\star d} \\ K^{d\star} & K^{dd} \end{bmatrix}\right). \tag{4.7}$$

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

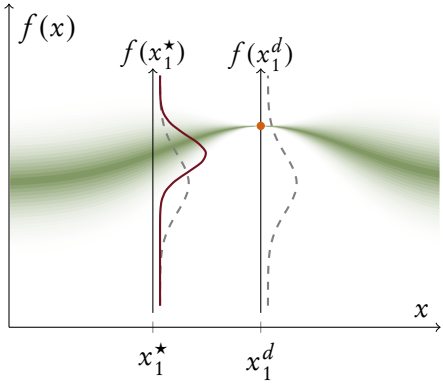
$$\begin{aligned} &p\left(f(x^\star) \,|\, f(x^d)\right) = \\ &\mathcal{N}\left(f(x^\star); \mu(x^\star) + K^{\star d}(K^{dd})^{-1}\left(f(x^d) - \mu(x^d)\right), K^{\star\star} - K^{\star d}(K^{dd})^{-1}K^{d\star}\right), \end{aligned} \tag{4.8}$$

i.e., again 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.



(a). A GP defined on the real line parametrized 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 adjust to the observation, both in terms of mean (closer to the observation) and (marginal) variance (smaller in the proximity of the observation, but 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).

The construction of the Gaussian process can alternatively also be introduced as a nonlinear and nonparametric generalization of linear regression. Of course, the final model is the same, but the introduction made in this thesis is perhaps less standard (cf. Bishop 2006, Section 6.4; MacKay 1998; Rasmussen and Williams 2006).

4.2 Choosing 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 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 Wahlström et al. (2013), where the magnetic field is modeled using a GP covariance function tailored to obey Maxwell's equations. In many situations, however, such detailed knowledge is not present, and one has to make a less informed choice of covariance function $\kappa(\cdot, \cdot)$. Two common choices are the exponentiated quadratic and the Matérn class[2]

[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).

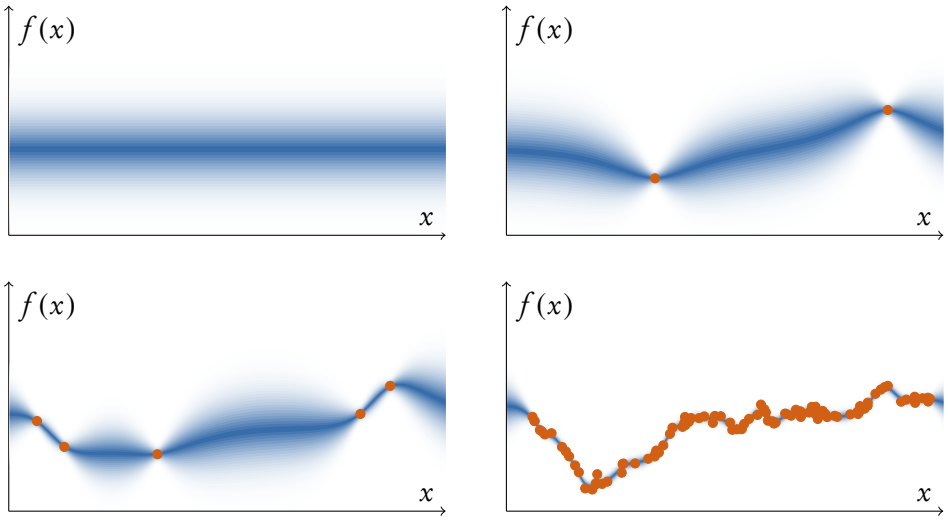


Figure 4.5. The GP as a modeling tool: the conditional distribution (shaded blue) for $f(x)$ after 0, 2, 5 and 100 observations (orange dots) of $y = f(x) + \text{noise}$. (We have now left our earlier convention of plotting the posterior distribution after conditioning on data in green, since the prior–posterior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

---

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 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 only a function of $\|x - x'\|$, *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., conditional on data) may be non-zero even though the prior is 0. In fact, $\mu(x) = 0$ appears to work well in many situations.

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.

Function	Meaning	Limitations	Examples
Mean $\mu(x)$	Prior assumption about mean	-	C (constant) $a \cdot x$ (linear)
Covariance $\kappa(x, x')$	Assumption on how tightly bounded 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.

4.3 Hyperparameter inference

Most mean functions, covariance functions, and noise distributions contains 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 distributed 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 inference chapter in Section 2.5.3, there are two common alternatives for how to deal with unknown hyperparameters: namely 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]

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

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,

[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: $p(y \mid \eta) = \int p(y \mid \eta)p(\theta \mid \eta)d\theta$.

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. A further discussion with examples is found in Paper IV.

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 IV.

4.4 Computational aspects

The computational load of (4.8), the main workhorse of the GP model, is governed by the matrix inversion of K^{dd}, an operation essentially of complexity $\mathcal{O}(m^3)$. 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., Chalupka et al. (2013);Rasmussen and Williams (2006, Chapter 8) and Snelson (2007). In particular, we will make use of the approximation proposed by Solin and Särkkä (2014b) in Paper I.

Essentially, all approximative methods amount to create 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 $\mathcal{O}(m^3)$ to $\mathcal{O}(p^3)$.

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 GP for modeling.

4.5.1 A posterior variance independent of the 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 inference 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 very unlikely sample of the GP, just as 0 is a very unlikely sample of a $\mathcal{N}(0, 1)$ distribution. Moreover does the GP also encode 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 was made when we considered the 2-dimensional Gaussian distribution in Figure 4.1, and only plotted its marginal distributions in 4.2.

[4] This point has its counterpart in the Kalman filter, Section 3.5.2, 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 2014a).

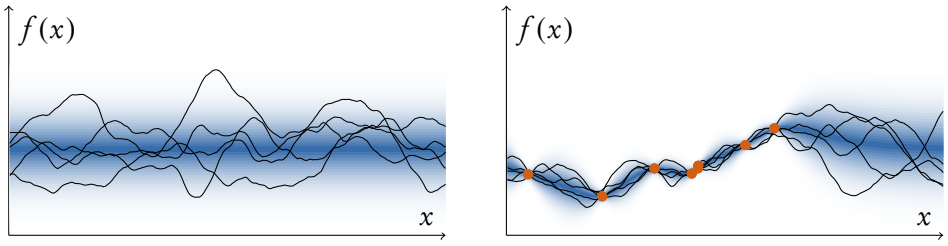


Figure 4.6. Five samples of the GP from Figure 4.5 (with a Matérn $\nu = 3/2$ covariance function). Note, in particular, that the samples are more wiggly than the mean function: a reminder that the blue shades do not contain all information, but is only the marginal distribution for each x (cf. Figure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

4.6 Extensions and generalizations

4.6.1 Heteroscedasticity and non-stationarity

The standard GP framework, as discussed so far, allows for non-stationary problems (i.e., having properties depending explicitly on x) if (i) the noise level is independent of x (homoscedastic), and (ii) a functional expression for the mean and covariance function is available. The extension of the GP framework beyond the limitations (i) and (ii) has received attention. Some work on the issue of heteroscedasticity, (i), (beyond the references in Rasmussen and Williams 2006), is Kersting et al. (2007) and Titsias and Lázaro-Gredilla (2011), and some recent work on (ii) is Heinonen et al. (2016) and Saul et al. (2016).

4.6.2 Student-t processes

If there is no reason for choosing a particular covariance function, one alternative would be to marginalize over all covariance functions. Perhaps surprisingly, it turns out that this approach actually is analytically tractable, and yields the student-t process. We refrain from the details here, but refer to Rasmussen and Williams (2006, Section 9.9) and Shah et al. (2014).

4.6.3 Dynamical GP models

The vanilla use of the GP model is typically to encode a smoothness assumption on $f(\cdot)$. However, the GP model can also be used to model more advanced dynamical behavior along the x-dimension in $f(x)$ (think of x as, e.g., representing time). For such a use, it is of interest to understand the covariance function $\kappa(\cdot, \cdot)$ as a prior assumption about the dynamical behavior of $f(\cdot)$. Such a covariance function design, motivated by the physical notion of a 'latent force', is found in M. A. Alvarez et al.

(2009) and M. A. Alvarez et al. (2013), whereas Solin and Särkkä (2014a) focuses on the connection between $\kappa(\cdot, \cdot)$ and a state-space model.

Another approach for dynamical modeling is the autoregressive GP, where $y_t = f(y_{t-1}, \ldots, y_{t-k})$ and $f \sim \mathcal{GP}$ (see, e.g., Frigola-Alcade 2015, Section 2.3 or Kocijan 2016 for an overview of such approaches).

4.6.4 Other nonparametric models

The GP is one member of the family of Bayesian nonparametric models, which also contains other nonparametric models, such as the Dirichlet process (Ferguson 1973). Some introductions to this topic are, e.g., Broderick (2016), Gershman and Blei (2012), and Hjort et al. (2010).

4.7 Gaussian-process state-space models

A relatively recent model is the GP state-space model,

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

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

a combination of the GP and the state-space model. 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 sate space model, allowing for complex and highly nonlinear dynamical phenomena to be described by the model. 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 et al. (2013) proposed a conceptually interesting but computationally brutal solution, and the subsequent Frigola et al. (2014) and Paper I (and in particular, its predecessor Svensson et al. 2016) present further developments in different directions. Both brings the computational load for an example requiring about 10 hours by Frigola et al. (2013) down to only a few minutes.

5

Monte Carlo methods for statistical inference

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 in 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 an established tool within many different scientific fields, and they are also present in the undergraduate education (Svensson 2016).

Monte Carlo methods are applicable in cases when an analysis of a mathematical model is not analytically tractable. There are also alternatives, such as the variational approach (Blei et al. 2016), where additional assumptions are imposed on the quantities of interest until the modified problem becomes analytically tractable. This thesis, however, focuses on the Monte Carlo approach.

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

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.*

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 its variance) is not analytically tractable. The Monte Carlo idea is to approximately represent π by random samples (an empirical measure). These samples are numerical values stored in the computer, which hopefully are easier to manipulate or analyze than the intractable distribution itself. The random samples should preferably be generated so that their properties with high probability resemble the properties of the distribution π.

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 other more involved alternatives, of which we will review some in this chapter.

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. This holds, e.g., for $p(y_{1:T} \mid \vartheta)$ but not for $p(x_t \mid y_{1:t}, \vartheta)$ when a particle filter has generated the samples.

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

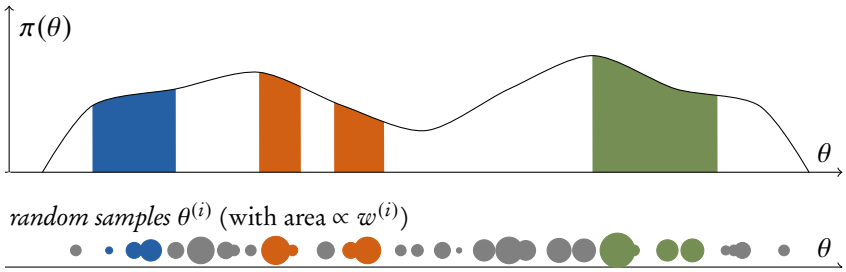


Figure 5.2. The Monte Carlo approximation: A probability density $\pi(\theta)$ at the top, and weighted random samples of $\pi(\theta)$ below. Each color is a choice of A in (5.1), and we thus expect the area in the upper part of the figure (i.e., $\int_A \pi(\theta)d\theta$) to be roughly proportional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

5.2 The bootstrap particle filter

As an example of a Monte Carlo-based algorithm, we start by introducing the popular 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 (3.8) solving the *state inference* problem, i.e., inferring 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[2] particle filter.

[2]The connection between the particle filter 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.

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.*

5.2.1 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 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 et al. (2015) for overviews. It is also possible to design resampling schemes where the duplicated particles are not assigned unit weights, as is implicitly done in Algorithm 1, see, Paige et al. (2014) for an example.

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 $p(y_{1:T} \mid \vartheta)$ (3.9) of the model. 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 it is a Monte Carlo-based estimate based on N_x particles in the notation. It can be shown (see, e.g., Appendix A) that (5.3) is an unbiased estimate of the data 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 inferring the model parameters ϑ. We will also mention a few more theoretical properties about Algorithm 1 later in Section 5.4.4.

5.3 The Markov chain Monte Carlo sampler

Let us now leave the state-space model, and return to the general problem we formulated in Section 5.1, where we were interested in drawing conclusions about some analytically intractable distribution $\pi(\theta)$, typically a posterior $p(\theta \mid y)$. We can use the MCMC methodology to generate samples from π if it is impossible to draw samples from π directly, but we can evaluate π pointwise (i.e., query the value of $\pi(\theta)$ for any θ), at least up to proportionality. 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 $K \to \infty$.

[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 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 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**

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*

We will 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 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 $\mathcal{K}(\cdot \mid \pi) \triangleq \pi(\cdot)$, where we use the shorthand notation $\mathcal{K}(\cdot \mid \pi) = \int \mathcal{K}(\cdot \mid \theta')\pi(\theta')d\theta'$. 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 $\mathcal{K}$ to θ'),

The transition kernel $\mathcal{K}$ in MCMC is often defined by an algorithm itself, rather than a closed form expression. There are two popular algorithms for designing $\mathcal{K}$, Metropolis-Hastings and Gibbs, which we will introduce next.

5.3.1 The Metropolis-Hastings kernel

The Metropolis-Hastings algorithm (named[4] after Nicholas Metropolis et al. 1953 and Wilfred K. Hastings 1970) is a popular plug-in kernel, only requiring that π can

[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.

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 $p(\theta_2 \mid \theta_1^{(k)})$

be evaluated pointwise 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 pointwise. 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 $x^{(k)}$ 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 right stationary distribution. If the support of 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 produce samples that are always accepted (but may on the other hand suffer from a high autocorrelation). It requires, however, that θ can be partitioned as $\theta = \{\theta_1, \theta_2, \ldots, \theta_M\}$ (preferably with low cross-dependence between the partitions) such 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 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) and $K \to \infty$.

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 has to 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 $h(\theta)$, we may compare the true expectation $\mathbb{E}\left[h(\theta)\right]$ (when θ is distributed according to π) and the sample-based estimate of it $h_K(\{\theta^{(k)}\}_{k=1}^K) = \frac{1}{K}\sum_{k=1}^K h(\theta^{(k)})$, as

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

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 3.5.1, the state inference in a state-space model is a particular inference problem. Similarly, the particle filter can be seen as a particular instance of the more general sequential Monte Carlo (SMC) method. Other SMC-based methods are particle smoothers (see, e.g., Paper III or Lindsten and Schön 2013), for inferring the smoothing distribution $p(x_t \mid y_{1:T}, \vartheta)$ instead of the filtering distribution $p(x_t \mid y_{1:t}, \vartheta)$. SMC can also be formulated for other type of models, such as graphical models (Naesseth et al. 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 the SMC algorithm is the SMC sampler (Del Moral et al. 2006), here presented as Algorithm 5. The SMC sampler is formulated for the same problem as the MCMC sampler, namely sample from a static density π which only can be evaluated pointwise up to proportionality.

The particle filter targets the filtering distributions (3.8) 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.

5.4.1 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

[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 pointwise.

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.*

proposal density, and the weighting is adjusted accordingly (see, e.g., Doucet and Johansen 2011 for an overview). The aim with 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 contrary, 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 π_{p-1} and π_p should be 'small' in order to guide the particles towards $\pi_P = \pi$. This idea resembles simulated annealing (also introduced by Metropolis 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.

5.4.3 Propagating the particles

For the SMC sampler case, 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 et al. 2006, Section 3.3), but one choice is

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

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

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 IV and another in Del Moral et al. (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 et al. 2013).

5.4.4 Convergence

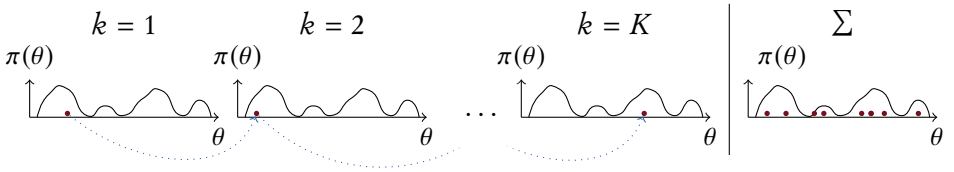
We have already discussed an important property of the particle filter (Algorithm 1), namely that $\widehat{p}_{N_x}(y_{1:T} \mid \vartheta)$ is unbiased for any finite $N_x \geq 1$. An analogous unbiased estimator is possible to construct for the normalizing constants Z_p in the SMC sampler. Results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

Akin to the MCMC case, results are also available 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

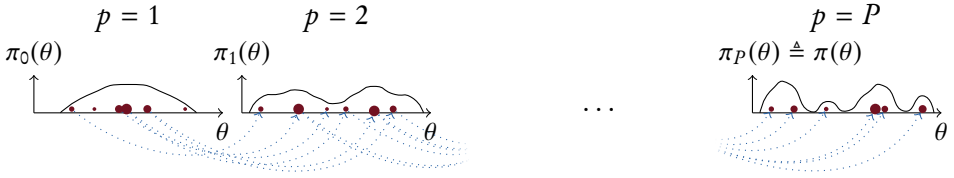
$$\sqrt{N_\theta}\left(h_{N_\theta}(\{\theta_p^{(i)}, w_p^{(i)}\}_{i=1}^{N_\theta}) - \mathbb{E}\left[h(\theta)\right]\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2_{\text{SMC}}(h, \pi)\right), \tag{5.9}$$

when $N_x \to \infty$, where $\sigma^2_{\text{SMC}}(h)$ is a bounded function of h and π (Del Moral et al. 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 et al. (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 (the red dot) stochastically through the landscape of π, such that the realization of the so-called chain (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. Clearly, it has to visit every mode of π: it will most likely happen 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 Paper II for a particular case) that the SMC sampler requires more user effort, in terms of implementation time. It is also worth to highlight 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.

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

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: due to the historical timeline, MCMC is perhaps unjustifiably favored for certain problems.

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 Paper III.) 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 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 data likelihood $p(y_{1:T} \mid \vartheta)$ nor the conditional distribution $p(x_{1:T} \mid \vartheta, y_{1:T})$ are available on 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 et al. 2010). PMCMC can also be used for solely solving the state inference problem, as discussed in Paper III for particle Gibbs.

[7] If the unknown parameters has a low dimensionality, they can however be considered as part of the state x_t (and modeled to be slowly time-varying), and thereby is the problem 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.

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 the 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!

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 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 et al. (2010), and some design methods for q can be found in Dahlin et al. (2015). A recent tutorial for getting started with particle Metropolis-Hastings is provided by Dahlin and Schön (2016).

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 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 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 et al. (2014), the conditional particle filter with ancestor sampling. We will not detail this construction any further here, but an introduction is found in Paper III, and we refer to Andrieu et al. (2010) and Lindsten 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) 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 only make us of particle Gibbs 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 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 by Papers I and V, where the method is also used 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 be used. 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 $P = T$, $\pi_0(\vartheta) = p(\vartheta)$, $\pi_1(\vartheta) = p(\vartheta \mid y_1)$, $\ldots$, $\pi_T(\vartheta) = p(\vartheta \mid y_{1:T})$. 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 et al. 2013; Fulop and Li 2013). For propagating the particles in step 3 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 et al. (2013).

[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).

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 et al. (2015) have suggested how to automatically adapt these numbers. A more detailed introduction to SMC^2, as well as a comparison to particle Metropolis-Hastings, is given in Paper II.

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

6 Conclusions and future work

This chapter is a brief summary with overall conclusions of the research presented in Paper I–V, as well as an outlook into further possible research directions, in addition to the corresponding section in each paper.

6.1 Conclusions

The main contributions of this thesis are not the novelty of any SMC methods, nor the novelty of any models. The contributions are, instead, the application of state-of-the-art SMC methods to the nontrivial models and problems at hand: the GP state-space model in Paper I, a watertank model in Paper II, the state-space smoothing problem (and in particular the indoor positioning application) in Paper III, the hyperparameters in a GP in Paper IV and the jump Markov linear state-space model in Paper V. The resulting learning methods are arguably powerful, at least on par with the existing state-of-the-art methods not based on SMC. This indeed suggests that SMC is a promising construction, and its full potential is not yet utilized. However, this also suggest that despite being a powerful method, SMC is also far from trivial to implement and use for several models.

6.2 Future work

As discussed above, the main contribution of this thesis is the application of SMC methods to various models. The work is indeed rather involved—and obviously publishable—but in a perfect world it should perhaps be less so. A natural future research direction is therefore to investigate how the procedure of combining advanced inference methods with complex models can be automated, for instance via a tailored high level programming language.

There are also several other (in a sense more detailed) possible directions for further research:

Parts of this thesis are focused around Bayesian inference in somewhat complicated state-space models. However, to the best of the author's knowledge, little research has been done on how to interpret and efficiently convert the posterior $p(\vartheta \mid y)$ in terms of a 'posterior' for the dynamical behavior (such as the input-output relationship). Such results would most likely be of big interest in situation when the parameters themselves do not bear a physical meaning of interest (such as in Paper I). A further challenge is also how such information can be used for designing automatic control strategies.

As briefly discussed at the end of Paper I, the classical system identification framework has the concept of 'prediction error method' and 'output error method' (Söderström and Stoica 1989, Chapter 7). The former of these approaches can be seen as optimizing the predictive performance of a model, and the latter as optimizing the simulation performance of a model. In particular if the data generating mechanism is different from the model, the different focuses can make a big difference. The statistical modeling approach, however, only has the concept of one data likelihood model (which for some model structures is close to the prediction error approach), regardless of the intended use for the model. This discrepancy is clearly not satisfying, and – to the best of the author's knowledge – not properly addressed in the literature either. An interesting perspective was recently presented by Rasmussen (2016).

Related to most algorithms used, and in particular the recent and little explored PSAEM algorithm (used in Paper I and V), there are several unanswered questions, such as the choice of the γ_k-sequence, initialization, etc.

A future direction could also be based on the question how to apply PMCMC methods to models where the particle filter struggles, such as state-space models with degenerate noise structure.

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 data likelihood $p(y_{1:T} \mid \vartheta)$ (3.9) 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 different realizations of the particle filter algorithm itself.

The proof follows closely that of Pitt et al. (2012), which is written for the more general case of the auxiliary particle filter. Another proof can be found in Del Moral 2004, Section 7.4.2, using the Feyman-Kac framework.

In the sequel, ϑ will be suppressed in the notation, since every expression is conditioned on ϑ. We start by introducing the estimator[1]

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

[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.

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

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

i.e., $\widehat{p}_{N_x}(y_t \mid y_{1:t-1})$ (A.2) (the contribution to (A.1) from the iteration of the particle filter at time t) 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 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,

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

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

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

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 $h = t - 1$)

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

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

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

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

Theorem 1. *The estimator $\widehat{p}_{N_x}(y_{1:T} \mid \vartheta)$, as defined by* (A.1) *and Algorithm 1 in Chapter 5, is unbiased in the sense*

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

for any finite $N_x \geq 1$.

B

The matrix normal inverse Wishart distribution in linear regression

This appendix gives some 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 $p(y \mid \mu, \sigma^2) = \mathcal{N}(y; \mu, \sigma^2)$, 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., $p(y \mid a, \sigma^2) = \mathcal{N}(y; ax, \sigma^2)$, 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,

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

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 $p(y_t \,|\, \theta) = \mathcal{N}(y_t; \mu, \sigma^2)$. If we decide to follow the Bayesian way of reasoning, we formulate a prior $p(\theta)$. A natural choice for the prior might be $p(\mu) = \mathcal{N}(\mu; m, \varsigma^2)$, and the posterior then becomes (after some algebra) $p(\theta \,|\, y_{1:T}) = \mathcal{N}\left(\mu; \left(\frac{m}{\varsigma^2} + \frac{\sum_t y_t}{\sigma^2}\right)\left(\frac{1}{\varsigma^2} + \frac{T}{\sigma^2}\right)^{-1}, \left(\frac{1}{\varsigma^2} + \frac{T}{\sigma^2}\right)^{-1}\right)$, 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,

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

The density for q can be written as

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

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 $\mathcal{G}(q; 1, \ell)$ 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

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

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

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

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:

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

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 $p(Z) = \mathcal{MNIW}(Z; M, U, V)$. We may write its density as

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

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 $ZZ^\intercal$ 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

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

matrix case, a $k \times k$ symmetric positive definite matrix Λ) and consider the inverse $(ZZ^\intercal)^{-1}$ (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 (elementwise) 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

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

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).

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., $y_t = ax_t + e_t$, $e_t \sim \mathcal{N}(0, \sigma^2)$ and x_t is known. That is, we have the model $p(y_{1:T} \mid a, \sigma^2) = \prod_{t=1}^T \mathcal{N}(y_t; ax_t, \sigma^2)$. 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 $\mathcal{NIG}(a, \sigma^2; m, v, \lambda, \ell)$. This yields the posterior

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

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

$$= (\sigma^2)^{-(\ell+T)/2-3/2} \exp\left(-\frac{\left(\frac{1}{v}+\sum_t x_t^2\right)\left(a-\frac{m/v+\sum_t y_t x_t}{1/v+\sum_t x_t^2}\right)^2 + \lambda + \sum_t y_t^2 + \frac{m^2}{v} - \frac{(m/v+\sum_t x_t y_t)^2}{\sum_t x_t^2+1/v}}{2\sigma^2} \right) \propto$$

$$\propto \Big/\text{cf. (B.6)}\Big/ \propto \mathcal{NIG}\left(a, \sigma^2; \overline{m}, \overline{v}, \overline{\lambda}, \overline{\ell}\right) \quad \text{(B.10)}$$

with

$$\overline{m} = \frac{m/v + \sum_t y_t x_t}{1/v + T}, \quad \text{(B.11a)}$$

$$\frac{1}{\overline{v}} = \frac{1}{v} + \sum_t x_t^2, \quad \text{(B.11b)}$$

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

$$\overline{\ell} = \ell + T. \quad \text{(B.11d)}$$

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

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

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

with known x_t. The data likelihood is given by

$$p(y_{1:T} \mid A, \Sigma) = \prod_{t=1}^T \mathcal{N}\left(y_t; Ax_t, \Sigma\right) =$$

$$= \prod_{t=1}^T |\Sigma|^{-k/2} \exp\left(-\tfrac{1}{2}(y_t - Ax_t)^\intercal \Sigma^{-1}(y_t - Ax_t)\right) \quad \text{(B.13)}$$

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

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

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

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

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

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

$$= |\Sigma|^{-(\ell+p+kn)/2-1} \exp\left(-\tfrac{1}{2}\mathrm{tr}\left((\star)+(\star\star)\right)\right) \propto \Big/\text{cf. (B.9)}\Big/ \propto$$
$$\propto \mathcal{NIW}\left(a, \Sigma; \overline{M}, \overline{V}, \overline{\Lambda}, \overline{\ell}\right) \quad \text{(B.14)}$$

with

$$\overline{M} = \left(MV^{-1} + \textstyle\sum_t y_t x_t^\mathsf{T}\right)\left(V^{-1} + \textstyle\sum_t x_t x_t^\mathsf{T}\right)^{-1}, \quad \text{(B.15a)}$$
$$\overline{V}^{-1} = V^{-1} + \textstyle\sum_t x_t x_t^\mathsf{T}, \quad \text{(B.15b)}$$
$$\overline{\Lambda} = (\star\star), \quad \text{(B.15c)}$$
$$\overline{\ell} = \ell + kn. \quad \text{(B.15d)}$$

Notation list

The notation used in the introductory chapters is summarized below. Note that the notation in the papers may vary, and is defined separately within 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 inference	
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 35

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

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.

Mauricio A. Alvarez, David Luengo, and Neil D. Lawrence (2009). "Latent force models". In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Clearwater Beach, FL, USA, pp. 9–16.

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.

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: *arXiv:1601.00670*.

George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung (2015). *Time series analysis: forecasting and control*. 5th ed. Hoboken, NJ, USA: Wiley.

Stephen Boyd and Leon O. Chua (1985). "Fading memory and the problem of approximating nonlinear operators with Volterra series". In: *IEEE Transactions on Circuits and Systems* 32.11, pp. 1150–1161.

Tamara Broderick (2016). *Tutorials*. URL: http://www.tamarabroderick.com/tutorials.html.

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.

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)*. Cadiz, 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.

Thomas S. Ferguson (1973). "A Bayesian analysis of some nonparametric problems". In: *Annals of Statistics* 1.2, pp. 209–230.

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

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.

Samuel J. Gershman and David M. Blei (2012). "A tutorial on Bayesian nonparametric models". In: *Journal of Mathematical Psychology* 56.1, pp. 1–12.

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.

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.

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.

Adolf Hammerstein (1930). "Nichtlineare Integralgleichungen nebst Anwendungen". In: *Acta Mathematica* 54.1, pp. 117–176.

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)*. Cadiz, 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.

Nils Lid Hjort, Chris Holmes, and Peter Müller, eds. (2010). *Bayesian nonparametrics*. Cambridge Series in Statistical and Probabilistic Mathematics 28. Cambridge, UK: Cambridge University Press.

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

Matthew J. Johnson, David Duvenaud, Alexander B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams (2016). "Composing graphical models with neural networks for structured representations and fast inference". In: *arXiv:1603:06277*.

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.

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard (2007). "Most likely heteroscedastic Gaussian process regression". In: *Proceedings of the 24th International Conference on Machine Learning (ICML)*. Corvallis, OR, USA, pp. 393–400.

Juš Kocijan (2016). *Modelling and control of dynamic systems using Gaussian process models*. Basel, Switzerland: Springer International.

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.

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.

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.

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.

Rik Pintelon and Johan Schoukens (2012). *System identification: a frequency domain approach*. 2nd ed. New York, NY, USA: John Wiley & Sons.

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 (2016). "The Bayesian method in system identification". In: *European Research Network on System Identification Workshop (ERNSI)*. Oral presentation. Available: `http://mlg.eng.cam.ac.uk/carl/talks/ernsi16.pdf`. Cison di Valmarino, TV, Italy.

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.

Alan D. Saul, James Hensman, Aki Vehtari, and Neil D. Lawrence (2016). "Chained Gaussian Processes". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Cadiz, Spain, pp. 1431–1440.

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 and Fredrik Lindsten (2016). "Learning of dynamical systems - particle filters and Markov chain methods".

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

Johan Schoukens, Jozsef G. Nemeth, Philippe Crama, Yves Riolain, and Rik Pintelon (2003). "Fast approximate identification of nonlinear systems". In: *Automatica* 39.7, pp. 1267–1274.

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 and Simo Särkkä (2014a). "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 (2014b). "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: http://www.youtube.com/watch?v=aUkBa1zMKv4.

Andreas Svensson (2016). *On the role of Monte Carlo methods in Swedish M. Sc. engineering education*. Tech. rep. 2016-009. Department of Information Technology, Uppsala University.

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)*. Cadiz, 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.

Michalis K. Titsias and Miguel Lázaro-Gredilla (2011). "Variational heteroscedastic Gaussian process regression". In: *Proceedings of the 28th International Conference on Machine Learning (ICML)*. Bellevue, WA, USA, pp. 841–848.

Ruey S. Tsay (2010). *Analysis of financial time series*. 3rd ed. Hoboken, NJ, USA: Wiley.

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

Niklas Wahlström, Manon Kok, Thomas B. Schön, and Fredrik Gustafsson (2013). "Modeling magnetic fields using Gaussian processes". In: *Proceedings of the 38th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, BC, Canada, pp. 3522–3526.

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

Norbert Wiener (1958). *Nonlinear problems in random theory*. Cambridge, MA, USA: MIT Press.

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 (2016). "A flexible state space model for learning nonlinear dynamical systems". In: *Automatica*. Provisionally accepted.

Digital identity

http://arxiv.org/abs/1603.05486

Parts of the content in this paper has 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)*. Cadiz, 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)*. Cancun, 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[1] $x_t \in \mathbb{R}^{n_x}$, 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 $\{u_{1:T}, y_{1:T}\}$.

[1] v_t and e_t are iid with respect to t, and x_t is thus Markov.

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 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; Kocijan et al. 2005; 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 weights $\{w^{(j)}\}_{j=0}^{m}$ will be learned from data. By introducing certain *priors* $p(w^{(j)})$ *on the basis function weights* 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-the-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 weights $w^{(j)}$, we tailor a learning algorithm using recent sequential Monte Carlo/particle filter methods (Kantas et al. 2015; Schön et al. 2015). The learning algorithm infers the posterior distribution of the sought parameters from data, and come with theoretical guarantees. We will pay extra attention to the problem of finding the maximum mode of the posterior, i.e., 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.

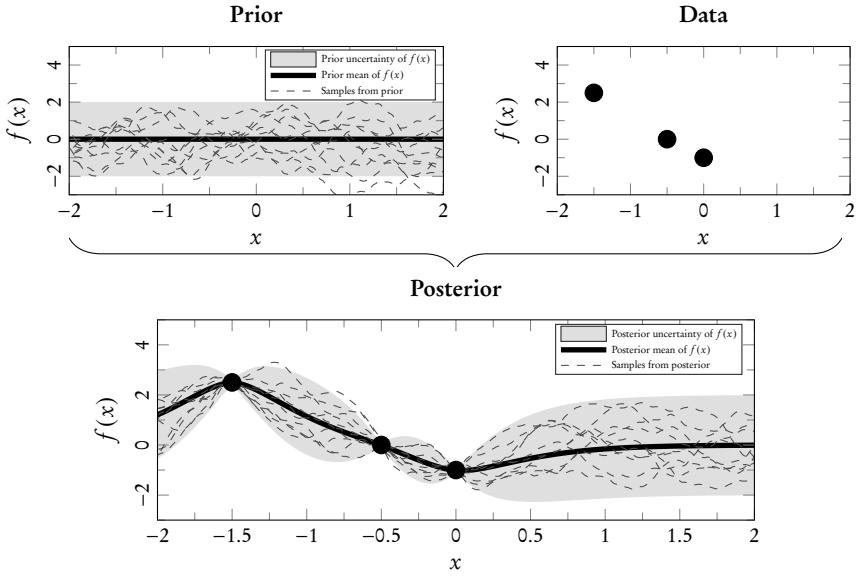


Figure 1. The Gaussian process as a modeling tool for an one-dimensional function $f : \mathbb{R} \mapsto \mathbb{R}$. The prior (upper left plot) is shown in terms of its mean and 2 standard deviations, as well as 10 samples drawn from it. By combining the prior and the data (upper right plot), the posterior (lower plot) is obtained. The posterior mean basically interpolates between the data points, and adhere 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, here plotted as 2 standard deviations, high in data-scarce regions and low where the data provides knowledge about $f(\cdot)$.

2 Related work

Important work using the GP in system identification includes impulse response estimation (Chen et al. 2012; Pillonetto et al. 2011; Pillonetto and De Nicolao 2010), nonlinear ARX models (Bijl et al. 2016; 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 et al. (2013), using a Monte Carlo approach (similar to this paper) for learning. The practical use of Frigola 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 et al. (2014) (and somewhat generalized by Mattos et al. 2016), using inducing points and a variational inference scheme. Another competitive approach is Svensson 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 models 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, referred to as reproducing kernels by Tobar et al. (2015). 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 by Tobar et al. 2015), and – perhaps most important – (iii) pursue systematic encoding of prior assumptions further than Tobar et al. (2015), who instead assume $g(\cdot)$ is 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) prevents the model from overfitting to the training data not by regularization, but

manually choosing a low enough polynomial order and terminate the learning procedure prematurely (early stopping). Paduart et al. are, in contrast to us, focused on the frequency properties of the model and relying 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, in the case of a close-to-linear behavior, use a nonlinear model only to describe deviations from the linear model.

Furthermore, there are also strong connections to our previous work Svensson 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 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 weights, 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 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 generalizing 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 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[2] of the model. The learning algorithm will, however, be applicable to any choice of basis functions.

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

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 to consider, 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 tune 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 other hand, if L is too big, an unnecessarily large m might also be needed, wasting computational power. To chose L in the magnitude 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, 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 weights $w^{(j)}$, 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 data likelihood. To use a prior $w^{(j)} \sim \mathcal{N}\left(0, \alpha^{-1}\right)$ 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[3] covariance functions, i.e., $\kappa(x, x') = \kappa(|x - x'|)$. In particular,

[3]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.

if the basis functions are chosen as (4), then

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

with[4]

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

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 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).

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

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 ($\mathcal{MN}$, Appendix B of the thesis) 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 $\mathrm{vec}(A) \sim \mathcal{N}(\mathrm{vec}(M), V \otimes U)$, where $\otimes$ is the Kronecker product. Its density can be written as

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

By letting $M = 0$ and V 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 scaled not 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 ($\mathcal{IW}$, Appendix B of the thesis) 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.

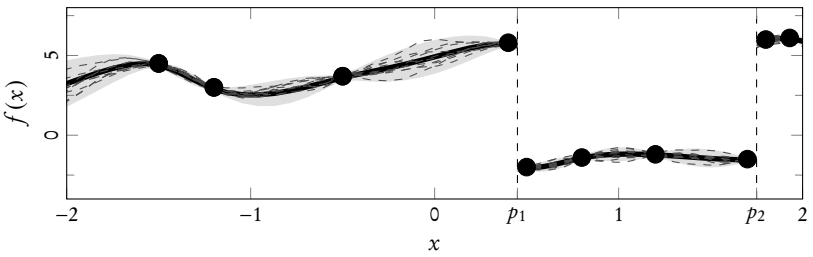


Figure 2. The idea of a piecewise GP: the interval $[-2, -2]$ is divided by $n_p = 2$ discontinuity points p_1 and p_2, and a GP is used to model a function on each of these segments, independently of the other segments. For practical use, the learning algorithm have to be able to infer also the discontinuity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

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 segment, 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).

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):

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

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

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 decent

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 Example IV.B in Svensson et al. (2015).

4 Learning

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

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

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 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 et al. 2015). At the very core in 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 $\widehat{p}_N(x_t \mid y_{1:t}) = \sum_{i=1}^N \omega_t^i \delta_{x_t^i}(x_t)$ for each t. Algorithmically, it amounts to iteratively weigh 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 has been studied extensively (see references in Doucet and Johansen 2011).

Algorithm 1: Particle Gibbs 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 $x_{1:T}[k+1] = x_{1:T}^J$.

The key of one set of SMC methods for learning parameters, 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 particle Gibbs Markov kernel derived by Lindsten et al. (2014) to handle the states in Step I of (17).

The particle Gibbs 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.

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

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

with the (sufficient) statistics

$$\Phi = \sum_{t=1}^{T} x_{t+1} x_{t+1}^{\mathsf{T}}, \quad (20a)$$

$$\Psi = \sum_{t=1}^{T} x_{t+1} \bar{\varphi}(x_t, u_t)^{\mathsf{T}}, \text{ and} \quad (20b)$$

$$\Sigma = \sum_{t=1}^{T} \bar{\varphi}(x_t, u_t) \bar{\varphi}(x_t, u_t)^{\mathsf{T}}. \quad (20c)$$

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

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

the posterior

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

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 $p(x_{t+1} \mid x_t, u_t, \theta) = \mathcal{N}\left(x_{t+1} - h(x_t) \mid A\bar{\varphi}(x_t, u_t), Q\right)$, 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 Bayesian learning. We thus resort to a numerical approximation by drawing samples from $p(\theta, x_{1:T} \mid y_{1:T})$ using MCMC. (An alternative is to use variational methods, akin to Frigola 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 (Appendix B of the thesis). By identifying components in (22), we conclude that

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

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

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**

expression (see Appendix A.1 of this paper)

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

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.

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 more Metropolis-within-Gibbs step (see Svensson 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)

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

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

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

In the standard formulation, $\mathcal{Q}$ 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 $\mathcal{Q}$. 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 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 $\mathcal{Q}$-function (25a) with a Robbins-Monro stochastic approximation of $\mathcal{Q}$,

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

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,

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

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**

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

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

Now, we note that if keeping track of the statistics $\gamma_k t[k] + \gamma_{k\text{-}1} t[k\text{-}1] + \dots$, 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 (20a), 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 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 their influence is minor in many cases. If needed, they can be included in Algorithm 2 by an additional Metropolis-Hastings 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 from two cascaded water tanks. Matlab code for all examples is available via the first authors homepage[5].

5.1 A first toy example

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

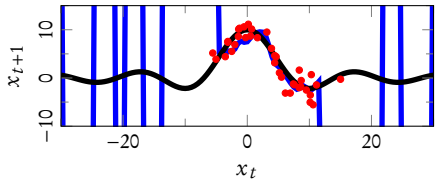
$$x_{t+1} = 10\mathrm{sinc}\left(\frac{x_t}{7}\right) + v_t, \qquad v_t \sim \mathcal{N}(0, 4), \tag{31a}$$

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

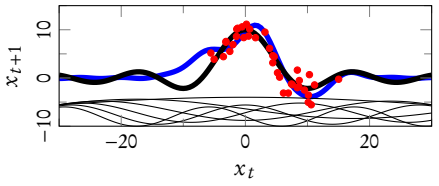
We generate $T = 40$ observations, and the challenge is to learn $f(\cdot)$, when $g(\cdot)$ and the noise variances are known. Note that even though $g(\cdot)$ is known, y is still corrupted by a non-negligible amount of noise.

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 a posteriori credibility interval), in particular in regions where no data is present.

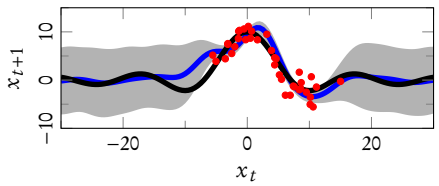
[5] http://www.it.uu.se/katalog/andsv164



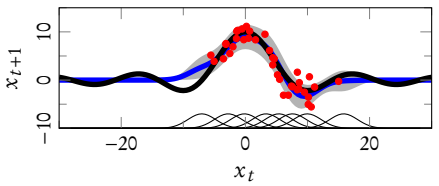
(a). Maximum likelihood estimation of our proposed model, *without* regularization; a useless model.



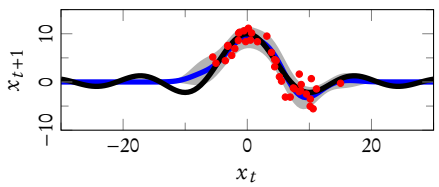
(b). Maximum likelihood estimation of our proposed model, *with* regularization. A subset of the $m = 40$ basis functions are used sketched at the bottom. Computation time: 12 s.



(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.

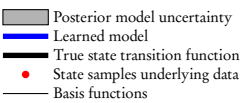


Figure 3. True function (black), states underlying the data (red) and learned model (blue, gray) for the different settings of the example in Section 5.1.

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 weights 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 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 weights $w^{(j)}$ 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).

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 the training data).

5.3 Water tank data

We consider the data set 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, 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:

$$x_{t+1}^1 = f^1(x_t^1, u_t) + v_t^1, \tag{33a}$$
$$x_{t+1}^2 = f^2(x_t^1, x_t^2, u_t) + v_t^2, \tag{33b}$$
$$y_t = x_t^2 + e_t. \tag{33c}$$

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 weights $w^{(j)}$ 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

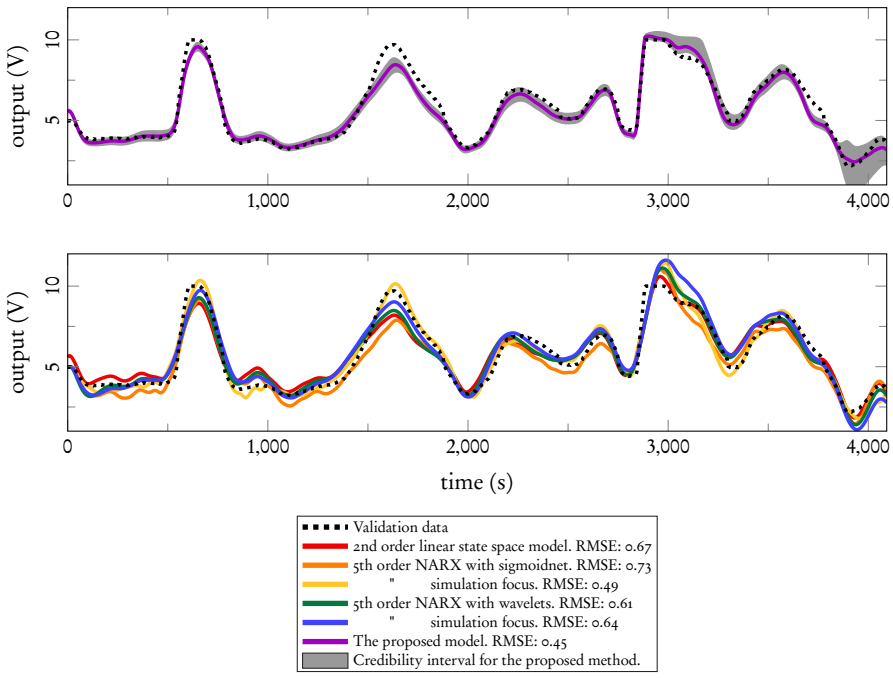


Figure 4. The simulated and true output for the test data in the water tank experiment (Section 5.3). The order of the NARX models refers to the number of regressors in u and y.

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[6]. 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]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 question for further research to figure out how to implement learning with 'simulation focus' for our model.

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 more complex state space model 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). 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 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 provides performance 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

As pointed out by Dyk and Jiao (2014), the combination of Metropolis-within-Gibbs and partially collapsed Gibbs might obstruct the invariant distribution of a sampler. In short, the reason is that a Metropolis-Hastings (MH) step is conditioned on the previous sample, and the combination with a partially collapsed Gibbs sampler can therefore be problematic, which becomes clear if we write the MH procedure as the operator $\mathcal{MH}$ in the following simple example from Dyk and Jiao (2014) of a sampler for finding the distribution $p(a, b)$:

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.

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, 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 par-

ticle MCMC". In: *Advances in Neural Information Processing Systems 26 (NIPS)*. Lake Tahoe, NV, USA, pp. 3156–3164.

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

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, 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 Mattson, 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)*. Cancun, 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)*. Cadiz, 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.

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 Wågber, 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

Comparing two recent particle filter implementations of Bayesian system identification

Authors

Andreas Svensson and Thomas B. Schön

Edited version of

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.

Digital identity

Department of Information Technology, Uppsala University: 2016-008

Financial support

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

Comparing two recent particle filter implementations of Bayesian system identification

Abstract

Bayesian system identification is a theoretically well-founded and currently emerging area. We describe and evaluate two recent state-of-the-art sample-based methods for Bayesian parameter inference from the statistics literature, particle Metropolis-Hastings (PMH) and SMC^2, and apply them to a non-trivial real world system identification problem with large uncertainty present. We discuss their different properties from a user perspective, and conclude that they show similar performance in practice, while PMH is significantly easier to implement than SMC^2.

1 Introduction

In this paper, we are concerned with methods for learning unknown parameters in nonlinear state space models, i.e., gray box identification. We write the state space model as

$$x_{t+1}|x_t \sim f_\theta(x_{t+1}|x_t), \tag{1a}$$

$$y_t|x_t \sim g_\theta(y_t|x_t), \tag{1b}$$

where $y_t \in \mathbb{R}^{n_y}$ is the observed output, $x_t \in \mathbb{R}^{n_x}$ is the state, and an exogenous input $u_t \in \mathbb{R}^{n_u}$ may be included in f. We let f and g denote probability densities (i.e., including stochastic noise), depending on an (unknown) parameter vector $\theta \in \mathbb{R}^{n_\theta}$, and by $\sim$ we mean distributed according to the density. We write $p_\theta(y_{1:T})$ to denote the likelihood of the data $y_{1:T} \triangleq \{y_1, \ldots, y_T\}$ under model (1) and the parameter θ.

Traditionally, the system identification literature (Ljung 1999; Söderström and Stoica 1989) has mostly been focused on maximum likelihood (ML) *point estimates* $\widehat{\theta}_{\mathrm{ML}}$ of the unknown parameter θ,

$$\widehat{\theta}_{\mathrm{ML}} = \arg\max_\theta p_\theta(y_{1:T}) \tag{2}$$

rather than inferring its posterior *distribution* using Bayes' theorem as

$$p(\theta|y_{1:T}) \propto p_\theta(y_{1:T})p(\theta), \tag{3}$$

where $p(\theta)$ denotes the prior belief on θ. An excellent introduction to Bayesian system identification is given by Peterka (1981).

The traditional focus on the ML problem is indeed for a good reason; it is often computationally easier (and more natural) to handle a single parameter value than a full distribution. When the parameters are far from being uniquely determined by the data, however, the posterior distribution automatically gives a quantification of the amount of *uncertainty* present. Such situations may occur when, e.g., there is a model mismatch, the system was not excited enough during data collection, or the data record is very short.

The development of methods for Bayesian identification has made big advances in recent years, in particular methods based on sequential Monte Carlo (SMC) (Kantas et al. 2015; Schön et al. 2015). An important part of the advances has been published in the statistics literature, with a distinct focus on theoretical properties, such as consistency and convergence (Chopin 2004; Del Moral 2004). We believe these methods have a potential of being of great use also in practice. This paper compares two different state-of-the-art algorithms, and evaluate their properties from an engineering perspective.

We will focus on two popular methods, Particle Metropolis-Hastings (PMH, Andrieu et al. 2010) and SMC^2 (Chopin et al. 2013; Fulop and Li 2013). Both methods are developed for being off-the-shelf methods for parameter learning problems, requiring nothing more than the ability to

(a) *simulate* from $f_\theta(x_{t+1}|x_t)$,

(b) *evaluate* $g_\theta(y_t|x_t)$.

The requirements (a) and (b) are fulfilled in most engineering applications. We will give a brief introduction (complete enough so the user can implement the methods on her/his own), and evaluate their performance on two examples. A more extensive introduction can be found in any of the recent tutorials by Schön et al. (2015), Kantas et al. (2015) or Dahlin and Schön (2016).

2 The PMH and SMC^2 algorithms

We assume the reader has some familiarity with particle filters (Doucet and Johansen 2011), and summarize the bootstrap particle filter as Algorithm 1. The particle filter is perhaps most known in the signal processing literature as a tool for state space filtering, but it can also be used as an unbiased (stochastic) estimator of $p_\theta(y_{1:T})$, i.e., the likelihood of the parameters θ. The likelihood is estimated by the weights from Algorithm 1 as

$$\widehat{p_\theta}(y_{1:T}) = \prod_{t=1}^{T} \left(\frac{1}{N_x} \sum_{i=1}^{N_x} w_t^{(i)} \right). \tag{4}$$

It can be shown (Appendix A) that $\mathbb{E}[\widehat{p_\theta}(y_{1:T})] = p_\theta(y_{1:T})$.

Algorithm 1: The basic (bootstrap) particle filter

1 Draw $x_0^{(i)} \sim p(x_0)$ and set $w_0^{(i)} = 1$
2 **for** $t = 1$ **to** T **do**
3 Draw $a_{t-1}^{(i)}$ with $\mathbb{P}\left(a_{t-1}^{(i)} = j\right) \propto w_{t-1}^{(j)}$ *(resampling)*
4 Draw $x_t^{(i)} \sim f_\theta(x_t | x_{t-1}^{a_{t-1}^{(i)}})$ *(propagation)*
5 Set $w_t^{(i)} = g_\theta(y_t | x_t^{(i)})$ *(weighting)*
6 **end**

All statements with (i) *are for* $i = 1, \ldots, N_x$

2.1 Particle Metropolis-Hastings

The PMH algorithm (Andrieu et al. 2010) uses a Metropolis-Hastings sampler, a Markov chain Monte Carlo (MCMC) method, to sample from the posterior distribution $p(\theta|y_{1:T})$ (3).

The idea behind a Metropolis-Hastings sampler is to randomly 'walk around' in the parameter space and thus produce samples from the posterior. This is achieved as follows: While 'standing' at $\theta[k]$, propose a new parameter value θ' from a proposal $q(\theta'|\theta[k])$ (e.g., a random walk). Then, accept the proposed value, i.e. set $\theta[k+1] = \theta'$, with probability

$$\alpha = \min\left(1, \frac{\widehat{p}_{\theta'}(y_{1:T})p(\theta')}{\widehat{p}_{\theta[k]}(y_{1:T})p(\theta[k])}\right), \tag{5}$$

otherwise 'stay' and set $\theta[k+1] = \theta[k]$. To do this, a particle filter has to be run to evaluate (4) for each new proposed value θ'. After repeating this procedure for sufficiently many iterations, a series of samples $\{\theta[k]\}_{k=1}^K$ is obtained, consisting of (correlated) samples of the sought distribution $p(\theta|y_{1:T})$. The PMH algorithm is outlined as Algorithm 2.

There are in general two typical pitfalls with Metropolis-Hastings: It is hard to a priori tell how long the initial transient, the *burn-in period*, will be. A typical behavior is also that the chain gets 'stuck' for long periods of time, i.e., all proposals are rejected. One can intuitively understand SMC^2, which we will detail in the next section, as addressing the first issue by 'warm-starting' the chain by sequentially adding more and more data.

2.2 SMC^2

SMC^2 was proposed independently by Chopin et al. (2013) and Fulop and Li (2013). In SMC^2, an SMC sampler (Del Moral et al. 2006) is used instead of Metropolis-Hastings. The SMC sampler is inspired by the particle filter, but can be applied to sample from a general sequence of distributions, not necessarily originating from state space models. The SMC sampler is applied to the sequence

Algorithm 2: Particle Metropolis Hastings

Input: K (# steps), q (proposal), $\theta[0]$ initial parameter
Output: $\theta[1], \ldots, \theta[K]$ (samples from the posterior, including burn-in)

for $k = 1$ **to** K **do**
  Draw $\theta' \sim q(\theta'|\theta[k-1])$.
  Estimate $\widehat{p}_{\theta'}(y_{1:T})$ by a particle filter, Algorithm 1.
  Compute α (5).
  Draw $d \sim \mathcal{U}(d \mid 0, 1)$.
  if $d < \alpha$ **then**
    Set $\theta[k] = \theta'$ *(accept θ')*
  else
    Set $\theta[k] = \theta[k-1]$ *(reject θ')*
end

$\{p(\theta), p(\theta|y_1), p(\theta|y_{1:2}), \ldots, p(\theta|y_{1:T})\}$, a *data-tempered* sequence, evolving from the prior to the posterior.

The SMC sampler works in a similar fashion to the particle filter, with an iteration of weighting – resampling – propagation. The particles, $\{\theta_t^{(m)}\}_{m=1}^{N_\theta}$, 'live' in θ-space. However, the weighting for particle $\theta_t^{(m)}$ should be done with respect to the data likelihood, so a standard particle filter has to be 'attached' to every single $\theta^{(m)}$ particle to estimate its data likelihood, hence the name SMC^2. Thus, the weighting is with respect to

$$\widehat{p}_{\theta_t^{(m)}}(y_t|y_{1:t-1}) = \frac{1}{N_x}\sum_{i=1}^{N_x} w_t^{(i)}. \tag{6}$$

The resampling in the SMC sampler is identical to the particle filter, but the propagation has to be performed differently, as there is no equivalent to the function f for propagating the particles from $p(\theta|y_{1:t})$ to $p(\theta|y_{1:t+1})$. Instead PMH, Algorithm 2, is used to propagate the particles. To reduce the computational load, it is proposed by Chopin et al. (2013) to apply PMH only when some particle degeneracy criterion (e.g.the effective sample size, ESS) is fulfilled. This is summarized in Algorithm 3.

By construction, SMC^2 is an 'online algorithm' in the sense that it evolves along the time index, and if stopped prematurely at t', samples from the posterior $p(\theta|y_{1:t'})$ are obtained. Similarly, if another data point y_{T+1} is added, the algorithm does not have to start over from scratch, as opposed to the PMH. However, the computational load of SMC^2 is increasing with t, prohibiting use in online implementations with real-time requirements.

Algorithm 3: SMC^2

Input: N_θ (# θ particles), q (proposal)
Output: $\{\omega_T^{(m)}, \theta_T^{(m)}\}_{m=1}^{N_\theta}$ (samples from posterior)

1 Draw $\theta_0^{(m)} \sim p(\theta)$ and set $\omega_0^{(m)} = 1$.
2 Run Step 1 of Algorithm 1 for each $\theta_0^{(m)}$.
3 **for** $t = 1$ **to** T **do**
4 One iteration of the loop in Alg. 1 for each $\theta_{t-1}^{(m)}$.
5 Compute $\widehat{p}_{\theta_t^{(m)}}(y_t|y_{1:t-1})$ (6).
6 Set $\omega_t^{(m)} = \omega_{t-1}^{(m)} \widehat{p}_{\theta_t^{(m)}}(y_t|y_{1:t-1})$.
7 **if** *ESS is too low* **then**
8 Draw $b_t^{(m)}$ with $\mathbb{P}\left(b_t^{(m)} = n\right) \propto \omega_t^{(n)}$.
9 Run PMH, Alg. 2, for each $\theta_{t-1}^{b_t^{(m)}}$ to obtain $\theta_t^{(m)}$.
10 Set $\omega_t^{(m)} = 1$.
11 **else**
12 Set $\theta_t^{(m)} = \theta_{t-1}^{(m)}$
13 **end**

All statements with (m) *are for* $m = 1, \ldots, N_\theta$.

3 Numerical comparison

In this section, we will first apply PMH and SMC^2 to a small simulated example, and then to the problem of learning parameters in a water tank model from real data. We will also discuss their different properties from a user perspective. The Matlab code for all examples can be found on the first authors homepage[1].

3.1 A simulated example

We will start the comparison by a simulated numerical example. Consider the one-dimensional state space model

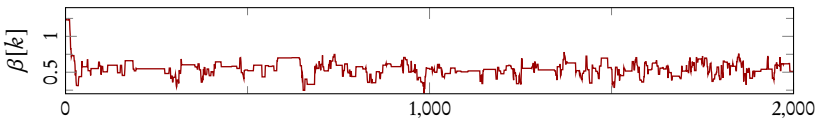
$$x_{t+1} = |x_t|^\beta + u_t + w_t, \qquad w_t \sim \mathcal{N}(0, 1) \tag{7a}$$

$$y_t = x_t + e_t \qquad e_t \sim \mathcal{N}(0, 1), \tag{7b}$$

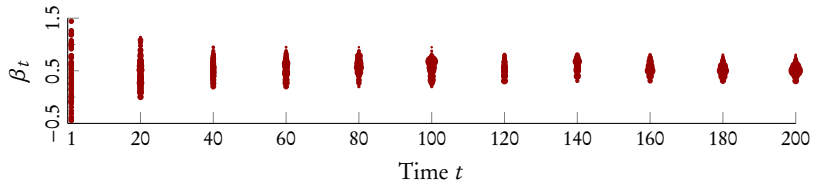
where u_t is a known input signal, drawn from a Gaussian. We want to learn the parameter β, which we believe is drawn from $\mathcal{N}(0, 1)$. This is thus our prior $p(\beta)$.

We simulate $T = 200$ data points $y_{1:T}$ from the model with $\beta = 0.4$. As the data record is relatively short in this example, we expect the posterior distribution to contain a non-negligible amount of uncertainty.

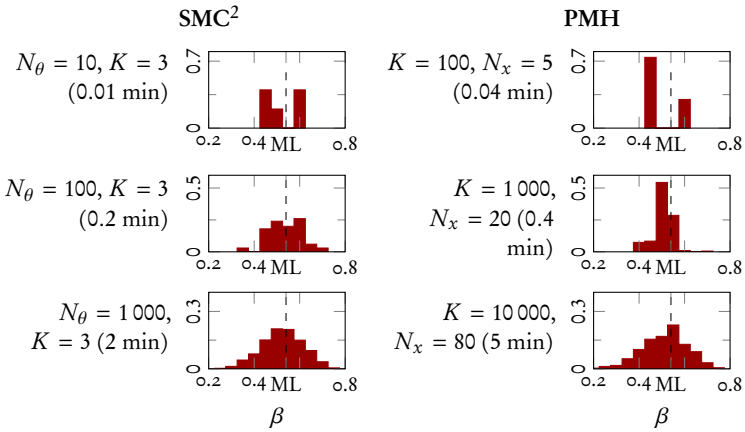
[1] http://www.it.uu.se/katalog/andsv164



(a). Trace plot for 2000 iterations of the PMH.



(b). Trace plot for 50 θ-particles in the SMC^2, at certain time points. The particles are plotted with a diameter proportional to their importance weights. Note how the posterior evolves along t as more measurements are added, from a quite non-informative prior to a distribution in the region around the true value.



(c). The final samples for the simulated numerical example, for different settings of SMC^2 and PMH. A ML estimate is also indicated. Studying the samples, it is clear that the methods suffers from too few particles/iterations, except for the settings in the last row.

Figure 1. Trace plots for PMH (a) and SMC^2 (b), and final samples (c) for Example 3.1, for different settings of the methods.

We apply the PMH algorithm to the problem, with proposal $q(\theta'|\theta)$ taken as the random walk $\theta' \sim \mathcal{N}(\theta'|\theta, 0.01)$. The trace plot for 2 000 iterations is shown in Figure 1a. Note the burn-in period; the Markov chain starts at 1.5, but moves eventually to the relevant part of the parameter space. In more complicated problems, the burn-in period is typically much longer. Also note the sequences of some hundred consecutive samples, where all proposals are rejected, a typical Metropolis-Hastings behavior.

Also SMC^2 was applied to the problem. A resulting trace plot is shown as Figure 1b. To evaluate the methods, we explore three different settings (in PMH the number of iterations K and particles N_x in each particle filter, for SMC^2 the number of θ-particles N_θ and PMH step K) and plot the samples as histograms in Figure 1c. The reported time are for a standard desktop computer[2].

The methods only show similar results in the last row of Figure 1c, and the poor results in the first and second row are due to shortcomings of the methods when not run with sufficiently many particles/iterations. In this example, SMC^2 seems to have a slight advantage as it runs faster. For comparison, we also apply the ML method proposed by Schön et al. (2011) to the same data, and obtain the ML estimate $\widehat{\beta}_{ML} = 0.54$. Note that while the ML method only gives a point estimate of 0.54 (approximately corresponding to the peak of the posterior distribution), the full posterior also indicates the true value 0.4 to be quite likely, and also quantifies the uncertainty present.

3.2 A real world example

We also evaluate the methods in a real-world scenario. We use the first $T = 40$ data samples from the two water tank data presented by Wigren and Schoukens (2013), to learn a discretized version of the model with 6 unknown parameters

$$\begin{pmatrix} \dot{x}_t^{(1)} \\ \dot{x}_t^{(2)} \end{pmatrix} = \begin{pmatrix} -k_1\sqrt{x_t^{(1)}} \\ k_2\sqrt{x_t^{(1)}} - k_3\sqrt{x_t^{(2)}} \end{pmatrix} + \begin{pmatrix} k_4 u_t \\ 0 \end{pmatrix} + w_t, \tag{8a}$$

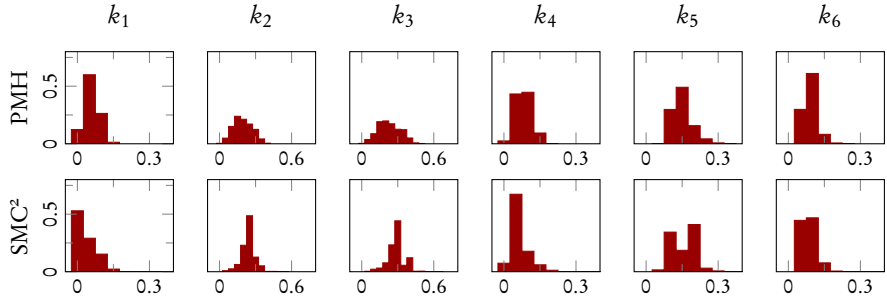
$$y_t = x_t + e_t, \quad w_t \sim \mathcal{N}(0, k_5 I_2), \quad e_t \sim \mathcal{N}(0, k_6 I_2). \tag{8b}$$

(I_2 is the 2-dimensional unit matrix, and k_5 and k_6 are scalar.) From the physical interpretation of the parameters (Wigren and Schoukens 2013), our prior assumptions on k_1, k_2 and k_3 are a uniform distribution on $[0, 1]$, and k_4 is $\mathcal{N}(0, 1)$. As the noise parameters k_5 and k_6 are strictly positive, we formulate priors on their logarithms as $\log k_5 \sim \mathcal{N}(0, 0.1)$ and $\log k_6 \sim \mathcal{N}(-1, 0.1)$ respectively.

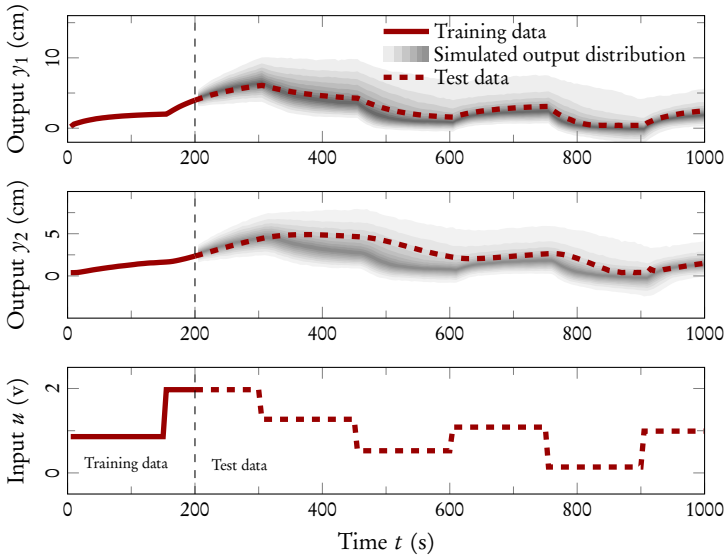
We run the PMH sampler for $K = 50\,000$ iterations with $N = 40$ particles, and SMC^2 with $N_\theta = 1\,000$ and $K = 30$ (requiring 12 and 27 minutes on a standard desktop computer, respectively). The samples obtained by the algorithms are plotted in Figure 2a. The results are quite similar, indicating similar ranges of possible values, but one can guess (by comparing the results for, e.g., k_3) that SMC^2 would benefit from an even bigger number N_θ.

In this example, PMH seems to be the preferred option, as it requires less computational time, still not suffering from the particle degeneracy problem as SMC^2 with, e.g., k_3.

[2]Intel i7-4600 2.1 GHz CPU



(a). The marginals of the posterior distributions for the six unknown parameters in the watertank model, inferred using PMH (top) and SMC^2 (bottom) from the data in (a). The data record is indeed short, but the posterior distribution provides useful information on possible ranges, much more informative than the prior $k_1 \in [0, 1]$, etc.



(b). The input-output data in the watertank data. The first 40 samples (200 s) are used as the training data, for inferring the posterior distribution of the parameters. The model learned with PMH (including the uncertainties) was here used to simulate the behavior of the tanks for another 800 s. (SMC^2 yields similar results.) The more intensive gray color, the more likely simulated output.

Figure 2. The inferred posterior distribution of the parameters (a) are used to simulate the output in (b).

3.3 Computational load and tuning

The computational load pf PMH is governed by $\mathcal{O}(KTN_x)$, i.e., proportional to the number of iterations K, the number of data points T and the number of particles N_x. The PMH is an anytime algorithm, in the sense that K does not have to be determined a priori, but can be run as long as the time permits. It has been shown

that the choice of N_x in PMH affects the acceptance rate, and that N_x should be chosen $\propto T$ to keep an acceptance rate not varying with T (Chopin et al. 2013). Following this rule of thumb, the computational load of PMH becomes $\mathcal{O}(KT^2)$.

SMC^2 has the computational load $\mathcal{O}(KT^2N_xN_\theta)$. The rule of thumb for N_x for PMH also applies to the PMH within SMC^2, giving the load $\mathcal{O}(KT^3N_\theta)$. Although SMC^2 provides the user with $p(\theta|y_{1:t})$ for all t from 0 to T, it is not suited for online problems with real-time requirements, as it grows forbiddingly fast with increasing T.

The time for implementation does usually not count as a part of the computational load, but is nevertheless of big relevance in practice. Our experience from implementing the two examples is that SMC^2 is significantly harder to implement, due to the quite involved interaction between the two nested levels of SMC. PMH, on the other hand, only requires a particle filter algorithm, computation of the acceptance probability α and some storage. The implementation of PMH therefore also has the advantage of being easier to debug than SMC^2.

As an alternative for the user to implement the methods on her/his own, there are currently a few software packages designed for off-the-shelf use of PMH, SMC^2 and related methods available, such as LibBi[3] (Murray 2013) and Biips[4] (Todeschini et al. 2014).

The tuning of both algorithms is indeed crucial for their performance. A perhaps subtle, but important, design choice is the proposal q inside the PMH. Suggestions on how to improve the proposal have been proposed in the literature, e.g Dahlin et al. (2015) for PMH, and Fearnhead and Taylor (2013) for SMC samplers. The number of particles, N_x and N_θ, are of course also of great importance, and a recent work on automatic adaption of N_θ in SMC^2 is presented by Chopin et al. (2015). Further, the rule of thumb $N_x \propto T$ applies, and a recent work with more extensive guidelines is Doucet et al. (2015).

4 Conclusions

PMH has a clear computational advantage over SMC^2 for large T. For small T, as illustrated in the numerical examples, the methods are more comparable. In the examples, there are no clear advantage neither for PMH nor SMC^2: they both perform similarly, with a computational time of the same magnitude. SMC^2 does indeed offer more flexibility in the tuning, and might thus have a greater potential in adaption to certain problems. If all sequential posteriors $p(\theta|y_1), \ldots, p(\theta|y_{1:T})$ are of interest, SMC^2 is indeed preferable. However, as discussed, SMC^2 is considerably more challenging to implement and debug than PMH.

[3] http://libbi.org

[4] http://alea.bordeaux.inria.fr/biips/

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.

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*.

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.

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, Michael K. Pitt, George Deligiannidis, and Robert Kohn (2015). "Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator". In: *Biometrika* 102.2, pp. 295–313.

Paul Fearnhead and Benjamin M. Taylor (2013). "An adaptive sequential Monte Carlo sampler". In: *Bayesian Analysis* 8.2, pp. 411–438.

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

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.

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

Lawrence M. Murray (2013). "Bayesian state-space modelling on high-performance hardware using LibBi". In: *arXiv:1306.3277*.

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

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.

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

Adrien Todeschini, François Caron, Marc Fuentes, Pierrick Legrand, and Pierre Del Moral (2014). "Biips: software for Bayesian inference with interacting particle systems". In: *arXiv:1412.3779*.

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.

Paper III

Title

Nonlinear state space smoothing using the conditional particle filter

Authors

Andreas Svensson, Thomas B. Schön and Manon Kok

Edited version of

Andreas Svensson, Thomas B. Schön, and Manon Kok (2015a). "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

and

Andreas Svensson, Thomas B Schön, and Manon Kok (2015b). *Some details on state space smoothing using the conditional particle filter*. Tech. rep. 2015-019. Department of Information Technology, Uppsala University.

Digital identity

doi:10.1016/j.ifacol.2015.12.257

Financial support

The Swedish Research Council (VR) via the project *Probabilistic modeling of dynamical systems* (contract number: 621-2013-5524) and CADICS, a Linnaeus Center.

Thanks to

Dr. Jeroen Hol and Dr. Henk Luinge at Xsens Technologies for providing the indoor positioning data.

Nonlinear state space smoothing using the conditional particle filter

Abstract

To estimate the smoothing distribution in a nonlinear state space model, we apply the conditional particle filter with ancestor sampling. This gives an iterative algorithm in a Markov chain Monte Carlo fashion, with asymptotic convergence results. The computational complexity is analyzed, and our proposed algorithm is successfully applied to the challenging problem of sensor fusion between ultrawideband and accelerometer/gyroscope measurements for indoor positioning. It appears to be a competitive alternative to existing nonlinear smoothing algorithms, in particular the forward filtering-backward simulation smoother.

1 Introduction

Consider the (time-varying, nonlinear, non-Gaussian) state space model (SSM)

$$x_{t+1} \mid x_t \sim f_t(x_{t+1}|x_t), \tag{1a}$$
$$y_t \mid x_t \sim g_t(y_t|x_t), \tag{1b}$$

with $x_1 \sim \mu(x_1)$. We use a probabilistic notation, with $\sim$ meaning distributed according to. The index variable $t = 1, 2, \ldots, T$ is referred to as time. The variable $x_t \in \mathbb{R}^{n_x}$ is referred to as state, and an exogenous input u_t is possible to include in f_t and g_t. To ease the notation, the possible time dependence of f and g will be suppressed.

For some applications, e.g., system identification, the distribution of the states for given model and measurements,

$$p(x_{1:T}|y_{1:T}), \tag{2}$$

is of interest. We will refer to (2) as the *smoothing distribution*. The smoothing distribution is not available on closed form for the general model (1), and approximations are necessary. In this paper, we will present a method generating Monte Carlo samples, *particles*, from the smoothing distribution, akin to a particle filter. The idea is to iterate a *conditional particle filter*, which generates samples from the smoothing distribution after sufficiently many iterations, as illustrated in Figure 1.

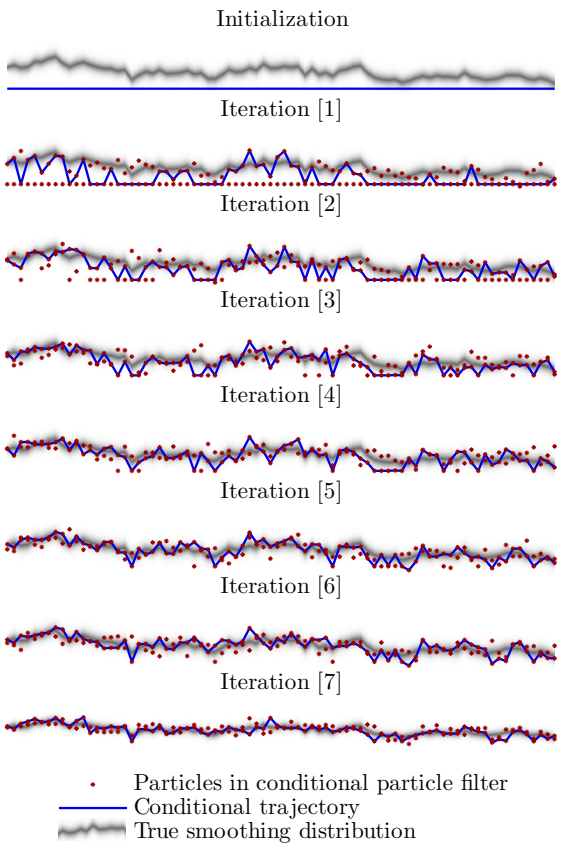


Figure 1. How to use the conditional particle filter to sample from the smoothing distribution $p(x_{1:T}|y_{1:T})$. The distribution $p(x_{1:T}|y_{1:T})$ is shown in gray, with time t at the horizontal axis and state x on the vertical axis. The conditional particle filter (with only 2 particles) is run iteratively, starting with the its arbitrary initialized trajectory (blue line) in the top plot, eventually converging to provide samples from the smoothing distribution.

An overview of existing particle smoothers addressing the problem of generating samples from (2) is provided by Lindsten and Schön (2013). In this work we will in particular compare and relate our developments to the so-called forward filtering-backward simulation (FFBSi) smoother introduced by Douc et al. (2011). The work by Kitagawa (1996) and Briers et al. (2010) are both of interest in that they approach a similar problem using particle filters, the latter also taking inspiration from the two-filter formula. The work of Pillonetti and Bell (2008) is closely related in that they also employ a Markov chain Monte Carlo (MCMC) construction, but they consider the special case of Gaussian noise (f and g can still be nonlinear though).

An alternative to using samples to represent (2) is to solve a linearized version of

the problem, by combining the extended Kalman filter (Schmidt 1966; Smith et al. 1962) with the RTS-smoother (Rauch et al. 1965) to solve the linearized problem analytically. The work of Särkkä (2008) on the unscented RTS-smoother is along the same line.

It should also be possible to generalise the ideas presented in this paper to probabilistic graphical models along the lines of the work by Naesseth et al. 2014.

Source code for simulated examples are available via the first author's homepage, and details on the second simulated example and the indoor positioning problem are available in Appendix A and B of this paper, respectively.

2 Particle methods

We assume the reader has some basic familiarity with particle filters (see, e.g., Doucet and Johansen 2011 for an introduction), but to set the notation we will start by a brief summary of particle filters and particle smoothers.

2.1 Particle filters

In the search for a numerical approximation to $p(x_{1:T}|y_{1:T})$, the following factorization is useful

$$p(x_{1:T}, y_{1:T}) = \mu(x_1) \prod_{t=1}^{T} g(y_t|x_t) \prod_{t=1}^{T-1} f(x_{t+1}|x_t), \tag{3}$$

as it allows for the following recursion to be derived (using Bayes' rule and $p(y_{1:T}) = p(y_1) \prod_{t=1}^{T-1} p(y_{t+1}|y_{1:t})$)

$$p(x_{1:t}|y_{1:t}) = \underbrace{\frac{g(y_t|x_t) f(x_t|x_{t-1})}{p(y_t|y_{1:t-1})}}_{(4\star)} p(x_{1:t-1}|y_{1:t-1}). \tag{4}$$

This factorization can be used to motivate the particle filter. Starting with a particle (Monte Carlo) approximation of $p(x_1|y_1)$ as N particles, $(4\star)$ can be applied to obtain a particle approximation of $p(x_{1:2}|y_{1:2})$. Repeating this $T-1$ times, a particle approximation of $p(x_{1:T}|y_{1:T})$ as N weighted particles $\{x^i_{1:T}, w^i_T\}_{i=1}^N$ is found. This is detailed in Algorithm 1, the *particle filter*, a Sequential Monte Carlo method.

The notation used in Algorithm 1 is

$$W_1(x_1) \triangleq g(y_1|x_1)\mu(x_1)/q_1(x_1|y_1) \tag{5a}$$

$$W(x_{1:t}) \triangleq g(y_t|x_t) f(x_t|x_{t-1})/q(x_t|x_{t-1}, y_t). \tag{5b}$$

Here, q denotes the proposal distribution, which is used to propagate the particles from time t to time $t+1$. If the proposal q is chosen as f, then (5b) simplifies to $W(x_{1:t}) = g(y_t|x_t)$ resulting in the so-called bootstrap particle filter.

Algorithm 1: Particle Filter

Output: A weighted particle system $\{x_{1:t}^i, w_t^i\}_{i=1}^N$ approximating $p(x_{1:t}|y_{1:t})$ for $t = 1, 2, \ldots, T$.

1 Draw $x_1^i \sim q_1(x_i)$ for $i = 1, \ldots N$.
2 Set $w_1^i = W_1(x_1^i)$ for $i = 1, \ldots N$.
3 **for** $t = 2$ **to** T **do**
4 Draw a_t^i with $\mathbb{P}\left(a_t^i = j\right) \propto w_{t-1}^j$ for $i = 1, \ldots N$.
5 Draw $x_t^i \sim q_t(x_t|x_{t-1}^{a_t^i}, y_t)$ for $i = 1, \ldots N$.
6 Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$ for $i = 1, \ldots N$.
7 Set $w_t^i = W(x_{1:t}^i)$ for $i = 1, \ldots N$.
8 **end**

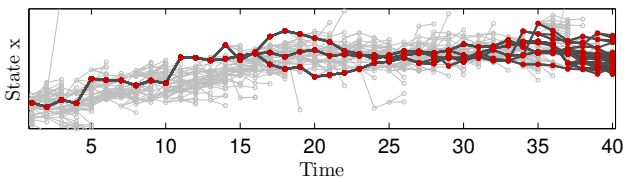


Figure 2. Path degeneracy: The particles in a particle filter are shown as dots, propagated as the lines indicate. The red dots are particles that have 'survived' the resampling steps, whereas the grey dots have not 'survived' the resampling steps. All trajectories $x_{1:40}^i$ have the part $x_{1:13}$ in common. This phenomenon occurs in particle filters and is the reason why a particle filter does not provide a good numerical approximation of $p(x_{1:T}|y_{1:T})$ for a finite N.

The main steps in Algorithm 1, namely 4, 5 and 7 are often referred to as *resampling*, *propagation* and *weighting*, respectively. Step 6 is merely bookkeeping. In Algorithm 1, a notation using *ancestor indices* a_t^i has been used for the resampling step, to prepare for the expansion to conditional particle filter with ancestor sampling.

In theory, a particle filter directly gives a numerical approximation of $p(x_{1:T}|y_{1:T})$. However, in practice with a finite N, the approximation tends to be rather poor (unless T is very small), as it typically suffers from *path degeneracy* as illustrated in Figure 2.

2.2 Forward – backward particle smoothers

A natural way to find the smoothing distribution for SSMs is to first apply a (forward) filter, and then add a backward pass, adjusting for the 'new' information about the state x_t at time t obtained from the later measurements $y_{t+1}, \ldots, y_T$. Such an

example is the RTS smoother for the linear Gaussian case (Rauch et al. 1965), but also the more recent particle-based FFBSi algorithm, see, e.g., Lindsten and Schön (2013) for a recent overview.

The algorithm for FFBSi is not repeated here, but we note that the it relies on the two step

1. A particle filter with N particles.
2. A backward simulation drawing M (uncorrelated) samples from $p(x_{1:T}|y_{1:T})$ using the N particles from Step (1).

The computational complexity of FFBSi is basically $\mathcal{O}(NM)$, although some improvements can be achieved, see Lindsten and Schön (2013, Section 3.3).

To prepare for the upcoming discussions on convergence, let us briefly comment on the convergence properties of FFBSi.

How well can a function $h(x_{1:T})$ be approximated as $\bar{h}$ using samples from FFBSi? Let $\bar{h}^N_{\text{FFBSi}} = \frac{1}{N}\sum_{i=1}^N h(x^i_{1:T})$ denote an approximation of $h(x_{1:T})$ based on $M = N$ backward trajectories. Under some fairly mild assumptions, it has been shown Douc et al. (2011, Corollary 9) that there exists a $\sigma_{\text{FFBSi}} < \infty$ such that

$$\sqrt{N}\left(\bar{h}^N_{\text{FFBSi}} - \mathbb{E}\left[h(x_{1:T})|y_{1:T}\right]\right) \tag{6}$$

converges weakly to $\mathcal{N}(0, \sigma^2_{\text{FFBSi}})$. To summarize, the convergence rate for FFBSi is $\sqrt{N}$, subject to a computational complexity of $\mathcal{O}(N^2)$.

3 Smoothing using the Conditional Particle Filter

The smoothing methodology discussed in Section 2.2 builds on a forward-backward strategy. The MCMC idea offers a fundamentally different way to construct a smoother, *without* explicitly running a backward pass, but *iteratively* running a so-called *conditional particle filter* as illustrated in Figure 1. As we will see, this opens up for a reduced computational complexity. The origin of the method dates back to the introduction of the PMCMC methods by Andrieu et al. (2010), with important recent contributions from Lindsten et al. (2014).

First, the conditional particle filter with ancestor sampling (CPF-AS) will be introduced (Section 3.1), followed by a brief introduction to MCMC (Section 3.2), and they will in the next step (Section 3.3) be combined to form a particle smoother. The convergence properties and the computational complexity of the smoother are then examined in Section 3.4 and Section 3.5, respectively.

3.1 Conditional particle filter with ancestor sampling

The CPF-AS is thoroughly described by Lindsten et al. (2014), and here presented as Algorithm 2. The CPF-AS is similar to a regular particle filter, Algorithm 1, in

Algorithm 2: Conditional particle filter with ancestor sampling (CPF-AS)

Input: Trajectory $x_{1:T}[k]$.
Output: Trajectory $x_{1:T}[k+1]$.

Draw $x_1^i \sim q_1(x_1^i)$ for $i = 1, \ldots, N-1$.
Set $x_1^N = x_1[k]$.
Set $w_1^i = W_1(x_1^i)$ for $i = 1, \ldots, N$.
for $t = 2$ **to** T **do**
  Draw a_t^i with $\mathbb{P}\left(a_t^i = j\right) \propto w_{t-1}^j$ for $i = 1, \ldots, N-1$.
  Draw $x_t^i \sim q(x_t | x_{t-1}^{a_t^i}, y_t)$ for $i = 1, \ldots, N-1$.
  Set $x_t^N = x_t[k]$.
  Draw a_t^N with $\mathbb{P}\left(a_t^N = j\right) \propto w_{t-1}^j f(x_t^N | x_{t-1}^j)$.
  Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$ for $i = 1, \ldots, N$.
  Set $w_t^i = W(x_{1:t}^i)$ for $i = 1, \ldots, N$.
end
Draw J with $\mathbb{P}(i = J) \propto w_T^i$ and set $x_{1:T}[k+1] = x_{1:T}^J$.

many aspects, but with one particle trajectory $x_{1:T}[k]$ specified a priori (trajectory number N in Algorithm 2).

The CPF-AS generates N weighted particle trajectories $\{x_{1:T}^i, w_T^i\}_{i=1}^N$. With the original formulation of the conditional particle filter in Andrieu et al. (2010), one of these trajectories is predestined to be $x_{1:T}[k]$. Extending this with ancestor sampling, the CPF-AS is obtained and the resulting trajectories $\{x_{1:T}^i, w_T^i\}_{i=1}^N$ are still influenced by $x_{1:T}[k]$, but in a somewhat more involved way, as the conditional trajectory may be 'partly' replaced by a new trajectory; see Algorithm 2 for details.

By sampling one of the trajectories $x_{1:T}[k+1] = x_{1:T}^J$ obtained from the CPF-AS with $\mathbb{P}(i = J) \propto w_T^i$, the CPF-AS can be seen as a procedure to stochastically map $x_{1:T}[k]$ onto another trajectory $x_{1:T}[k+1]$.

A Rao-Blackwellized formulation of the CPF-AS for mixed linear/nonlinear models is also possible, see Paper V for details.

3.2 Markov chain Monte Carlo

MCMC offers a strategy for sampling from a complicated probability distribution π on the space $\mathcal{Z}$, using an iterative scheme.

A Markov chain on $\mathcal{Z}$ is a sequence of the random variables $\{\zeta[1], \zeta[2], \zeta[3], \dots\}$, $\zeta[k] \in \mathcal{Z}$. The chain is defined by a *kernel* $\mathcal{K}$, stochastically mapping one element $\zeta[k]$ onto another element $\zeta[k+1]$. That is, the distribution of the random variable $\zeta[k]$ depends on the previous element as $\zeta[k+1] \sim \mathcal{K}(\cdot|\zeta[k])$.

If the kernel $\mathcal{K}$ is *ergodic* with a unique stationary distribution π, the marginal distribution of the chain will approach π in the limit. Let $\zeta[0]$ be an arbitrary initial state with $\pi(\zeta[0]) > 0$, then by the ergodic theorem (Robert and Casella 2004)

$$\frac{1}{K}\sum_{k=1}^{K} h(\zeta[k]) \to \mathbb{E}_\pi\left[h(\zeta)\right], \tag{7}$$

as $K \to \infty$ for any function $h : \{\mathbb{R}^n\}^T \mapsto \mathbb{R}$, with $\mathbb{E}_\pi[\cdot]$ denoting expectation w.r.t. ζ under the distribution π.

That is, for sufficient large k, the realization of $\{\zeta[k], \zeta[k+1], \dots\}$ is (possibly correlated) samples from π. This summarizes the idea of the MCMC methodology; if π is of interest, construct a kernel $\mathcal{K}$ with stationary distribution π and simulate a Markov chain to obtain samples of π.

Note that any *finite* realization of the chain $\{\zeta[1], \dots, \zeta[K]\}$ may be an arbitrarily bad approximation of π. This typically depends on the initialization $\zeta[0]$ and on how well the kernel $\mathcal{K}$ manages to explore $\mathcal{Z}$, referred to as the *mixing*.

3.3 Smoothing using MCMC

Take the general space $\mathcal{Z}$ as the more concrete space $\{\mathbb{R}^{n_x}\}^T$ (where $x_{1:T}$ lives). Note that CPF-AS in Algorithm 2 maps one element in $\{\mathbb{R}^{n_x}\}^T$ onto another element in $\{\mathbb{R}^{n_x}\}^T$, and can therefore be interpreted as an MCMC kernel. The unique stationary distribution for CPF-AS is $p(x_{1:T}|y_{1:T})$ (which is far from obvious, but shown by Lindsten et al. (2014)). Now, by constructing a Markov chain, Algorithm 3 is obtained, generating samples from the distribution $p(x_{1:T}|y_{1:T})$ (i.e., a smoother).

An illustration of Algorithm 3 was provided already by Figure 1; The initial trajectory is obviously not a sample from $\pi = p(x_{1:T}|y_{1:T})$, and artifacts from the initializations appear to be present also in iteration [1], [2], and possibly [3]. However, iterations [5], [6], [7] appear to be (correlated) samples from the distribution π, which is what was sought.

Algorithm 3: MCMC smoother

Input: $x_{1:T}[0]$ (Initial (arbitrary) state trajectory).
Output: $x_{1:T}[1], \ldots, x_{1:T}[K]$ (K samples from the Markov chain).
1 **for** $k = 1$ **to** K **do**
2 Run the CPF-AS (Algorithm 2) conditional on $x_{1:T}[k-1]$ to obtain $x_{1:T}[k]$.
3 **end**

3.4 Convergence

The convergence analysis of Algorithm 3 can, similar to the FFBSi in Section 2.2, be posed as the question of how well $h(x_{1:T})$ can be approximated by $\bar{h}^K_{\text{CPF-AS}} = \frac{1}{K}\sum_{k=1}^{K} h(x_{1:T}[k])$, where $x_{1:T}[k]$ comes from Algorithm 3. Before stating the theorem, let us make the following two rather technical assumptions

A1. The proposal q is designed such that given any x_{t-1} with non-zero probability (given the measurements $y_{1:t-1}$), any x_t with non-zero probability (given $y_{1:t}$) should be contained in the support of q.

A2. There exists a constant $\kappa < \infty$ such that $\|W\|_\infty < \kappa$.

Theorem 1 (Convergence for Algorithm 3)**.** *Under the assumptions A1 and A2, for any number of particles $N > 1$, and for any bounded function $h : \{\mathbb{R}^{n_x}\}^T \mapsto \mathbb{R}$, there exists a $\sigma_h < \infty$ such that*

$$\sqrt{K}\left(\bar{h}^K_{CPF\text{-}AS} - \mathbb{E}\left[h(x_{1:T})|y_{1:T}\right]\right) \tag{8}$$

converges weakly to $\mathcal{N}(0, \sigma_h^2)$.

Proof. The CPF-AS is uniformly ergodic for $N > 1$, Lindsten et al. (2014, Theorem 3). Therefore Theorem 1.5.4 in Liang et al. (2010) is applicable. □

Note: The convergence of Algorithm 3 to the smoothing distribution is by Theorem 1 not dependent of the number of particles $N \to \infty$, but is *only* relying on the number of iterations $K \to \infty$.

3.5 Computational complexity

The computational complexity of Algorithm 3 is of order $\mathcal{O}(KN)$, where N is the number of particles in the CPF-AS and K the number of iterations. However, in some programming languages, e.g., Matlab, vectorized implementations are prefered. The sequential nature of Algorithm 3 in k does not allow such a vectorized implementation, which is a clear drawback. On the other hand, K does not have to be

specified a priori, but Algorithm 3 can be run repeatedly until satisfactory results are obtained, or a given computational time limit is violated.

The short message here is: The convergence rate for Algorithm 3 is $\sqrt{K}$, obtained at a computational cost of $\mathcal{O}(K)$ (for a fixed number of particles N). This can be compared to the convergence rate $\sqrt{N}$ to the less beneficial cost of $\mathcal{O}(N^2)$ for FFBSi. However, one should remember that the samples obtained from FFBSi are uncorrelated, which is typically not the case for Algorithm 3.

4 Simulated examples

4.1 Scalar linear Gaussian SSM

As a first example, consider the scalar linear Gaussian SSM

$$x_{t+1} = 0.2x_t + u_t + w_t, \qquad w_t \sim \mathcal{N}(0, 0.3), \tag{9a}$$

$$y_t = x_t + e_t, \qquad e_t \sim \mathcal{N}(0, 1), \tag{9b}$$

with $\mathbb{E}[x_1] = 0$ and $\mathbb{E}[x_1^2] = 0.1$. Implementing Algorithm 3 with $N = 2$ (with $T = 80$ and u_t being low-pass filtered white noise), the result shown in Figure 1 is obtained. As the system is linear and Gaussian, analytical expressions for $p(x_{1:T}|y_{1:T})$ can be found using the RTS smoother, shown in gray in Figure 1.

4.2 Nonlinear, multi-modal example

We will now turn to a more challenging problem, pinpointing some interesting differences between the forward-backward smoother (FFBSi) and our MCMC-based smoother in Algorithm 3. We will start with a discussion using intuitive arguments, to motivate the example.

The FFBSi smoother handles the path degeneracy problem in the particle filter discussed in Section 2. However, the support for the backward simulation is still limited to the particles sampled by the particle filter. As those particles, for $t < T$, are sampled from the *filtering* distribution $p(x_t|y_{1:t})$ (and not the smoothing distribution $p(x_t|y_{1:T})$, due to the factorization (4)), only few of the particles may be useful if the difference between the filtering and smoothing distribution is 'large'. This might cause a problem for the FFBSi smoother, since there might exist cases where the particles do not explore the relevant part of the state space. An interesting question is now if Algorithm 3 can be expected to explore the relevant part of the state space better than the FFBSi smoother?

One way to understand the effect of the conditional trajectory in CPF-AS is as follows: If a proposal distribution $q \neq f$ is used in a regular particle filter (Step 5 in Algorithm 1), it is compensated for in the update of the weights, Step 7 and (5b), so that $\{x_{1:t}^i, w_t^i\}_{i=1}^N$ are still an approximation of $p(x_{1:t}|y_{1:t})$, even if $q \neq f$.

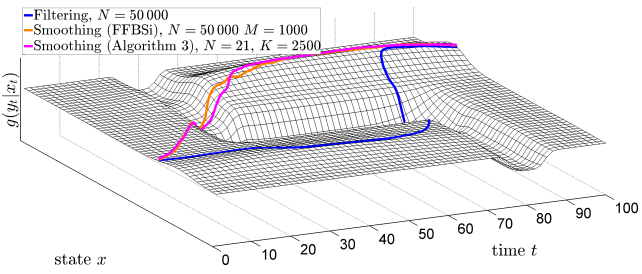


Figure 3. The 'landscape' for the Example in 4.2. The surface is proportional to $g(y_t|x_t)$, and the axis are time t and state x, respectively. The trajectories are the mean of the results of a particle filter (Algorithm 1, filtering), FFBSi (smoothing) and Algorithm 3 (smoothing). Note that for both smoothing methods, a total of 50 000 particles were sampled, so the comparison is 'fair' in that sense.

The CPF-AS can be thought of as a regular particle filter, but with a 'proposal' $q(x_t)$ that deterministically sets $x_t^N = x_t[k]$ (Step 7 of Algorithm 2) and 'artificially' assigns an ancestor to it (Step 8). However, there is no compensation for this 'proposal' in Step 10. Therefore, the samples $\{x_{1:t}^i, w_t^i\}_{i=1}^N$ from the CPF-AS can be expected to be biased towards the conditional trajectory $x_{1:T}[k]$.

On the other hand, we know from Lindsten et al. (2014) that the conditional trajectories in the limit $k \to \infty$ are samples of $p(x_{1:T}|y_{1:T})$. The bias towards $x_{1:T}[k]$ in the CPF-AS can therefore be thought of as 'forcing' the CPF-AS to explore areas of the state space relevant for the smoothing distribution $p(x_{1:T}|y_{1:T})$ (rather than the filtering distribution $p(x_{1:t}|y_{1:t})$) for large k.

A simulated example, appealing to this discussion, is now given. The problem is to sample from the smoothing distribution for a one-dimensional SSM with multimodal properties of $g(x_t|y_t)$. The state space model is $f(x_{t+1}|x_t) = \mathcal{N}(x_{t+1}|x_t, \sigma^2)$ and $g(y_t|x_t)$ is implicitly defined through the surface in Figure 3, where the surface level in point (x, t) defines $g(y_t|x_t)$, for a given y_t (not shown).

Given x_0, finding the maximum a posteriori estimate of the smoothing distribution $p(x_{1:T}|y_{1:T})$ amounts to finding the path $x_{1:T}$ maximizing $p(x_{1:T}|y_{1:T}) \propto \prod_{t=1}^T f(x_t|x_{t-1}) \prod_{t=1}^T g(y_t|x_t)$, where $g(y_t|x_t)$ is defined through the surface in Figure 3. Intuitively, this can be thought of as going from left ($t = 0$) to right ($t = 100$) in Figure 3, playing the children's game 'the floor is hot lava' with the cost f for moving sideways.

The mean of the filtering distribution $p(x_t|y_{1:t})$ for $t = 1, \ldots, T$ (obtained by Algorithm 1) is shown in Figure 3, together with the mean from two different smoothers; FFBSi (Section 2.2) and Algorithm 3, respectively.

The two smoothing approximations can indeed be expected to approach each other in the limit $N \to \infty$ / $K \to \infty$. The problem is interesting because the

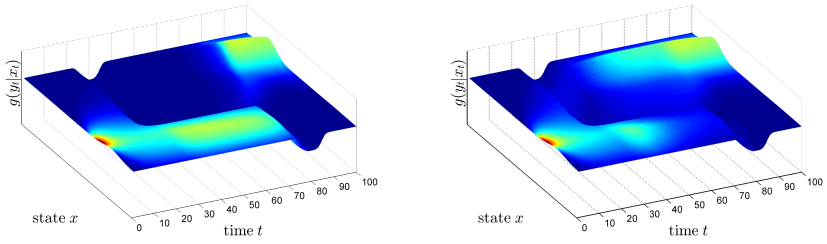


Figure 4. Particle densities (on a blue-yellow-red scale, from low to high) for FFBSi (left) and Algorithm 3 (right). In both cases, 50 000 particles are sampled for each t. They are, however, centered along the *filtering* distribution for the FFBSi, but biased toward the *smoothing* distribution for Algorithm 3. That is, Algorithm 3 explores (at least in this example) the relevant part of the state space (i.e., the left 'shoulder') better.

'likelihood landscape' in Figure 3 contains a 'trap'. The filtering distribution (and hence the particle filter in FFBSi) will follow the right 'shoulder' and discover 'too late' (the valley at $t \approx 70$) that it 'should' have walked along the left. The smoothing distribution, however, walks along the left shoulder earlier, as it 'knows' that the valley at the right hand side will come.

To quantify this discussion on how well the particles explore the state space for the two smoothers, the densities of the sampled particles for both smoothers are plotted in Figure 4. This suggest that Algorithm 3 is able to give a better approximation of the smoothing distribution, as a larger proportion of the particles are sampled in a relevant part of the state space.

5 Indoor positioning application

In this section, the presented algorithm is applied to a real-world sensor fusion problem; indoor positioning using ultrawideband (UWB), gyroscope and accelerometer measurements. We apply the model from Kok et al. (2015), but rather than using the optimization-based approach in that paper, we employ Algorithm 3. Instead of obtaining the maximum a posteriori (MAP) estimate as a point (as in Kok et al. 2015), we will obtain samples from the posterior distribution, which can be used to estimate the MAP, mean, credibility intervals, etc.

5.1 Problem setup

We take the problem as presented by Kok et al. (2015), a 10-dimensional nonlinear non-Gaussian problem. The goal is to estimate the position, velocity and orientation of the sensor board with the UWB transmitter, accelerometer and gyroscope, placed on the foot of a human. The UWB transmitter sends out pulses at (unknown) times

τ_t, and the time of arrival at the 10 receivers (indexed by m) are measured. The setup is calibrated using the algorithm in Kok et al. (2015), making sure that the receiver positions r_m^n are known and that their clocks are synchronized.

In the model, the state vector is $x_t^T = [p_t^T v_t^T q_t^T]$, p_t is the (3D) position, v_t the velocity and q_t the orientation (parametrized using unit quaternions). The SSM is given by

$$p_{t+1}^n = p_t^n + T_s v_t^n + \tfrac{T_s^2}{2} a_t^n, \tag{10a}$$
$$v_{t+1}^n = v_t^n + T_s a_t^n, \tag{10b}$$
$$q_{t+1}^{nb} = q_t^{nb} \odot \exp \tfrac{T_s}{2} \omega_t, \tag{10c}$$
$$y_{m,t} = \tau_t + \tfrac{1}{c} \| r_m^n - p_t^n \|_2 + e_{m,t}, \tag{10d}$$

where (10a) – (10c) are the dynamics and (10d) is the measurement equation. The superscripts n and b denote coordinate frames (n is the navigation frame aligned with gravity, and b is the body frame, aligned with the sensor axes of the accelerometers), c denotes the speed of light, $\odot$ denotes the quaternion product and exp denotes the vector exponential (Hol 2011). T_s is the time between two data samples from the accelerometer and gyroscope, sampled with 120 Hz. However, the UWB samples are sampled at approximately 10 Hz. Due to the nature of UWB measurements, $e_{m,t}$ is modeled as

$$e_{m,t} \sim \begin{cases} (2-\alpha)\mathcal{N}(0, \sigma^2), & e_{m,t} < 0, \\ \alpha \text{Cauchy}(0, \gamma), & e_{m,t} \geq 0, \end{cases} \tag{11}$$

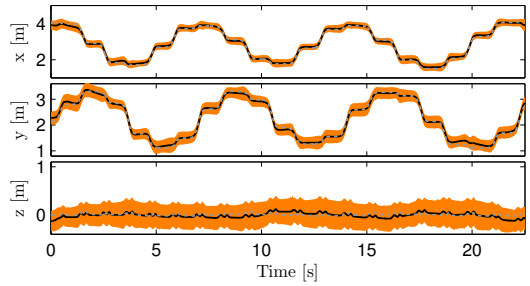
because measurements can only arrive later (and not earlier) in case of multipath and non-line-of-sight propagation. The acceleration a_t^n is found via accelerometer measurements $y_{a,t}$, modeled as

$$y_{a,t} = R_t^{bn}(a_t^n - g^n) + \delta_a + e_{a,t}, \tag{12}$$

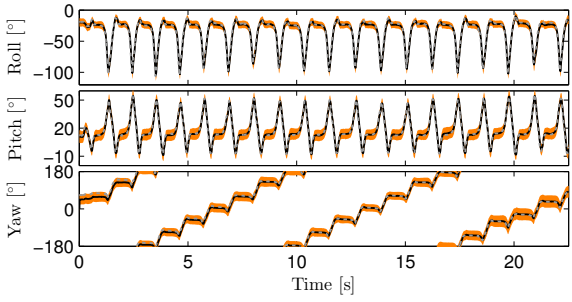
with g^n denoting gravity, R_t^{nb} is a rotation matrix representation of q_t^{nb}, and $R_t^{bn} = (R_t^{nb})^T$. The angular velocity ω_t is obtained from the gyroscope measurements $y_{\omega,t}$ as

$$y_{\omega,t} = \omega_t + \delta_\omega + e_{\omega,t}. \tag{13}$$

The noise $e_{a,t}$ and $e_{\omega,t}$ are modeled as $\mathcal{N}(0, \sigma_a^2)$ and $\mathcal{N}(0, \sigma_\omega^2)$, respectively. δ_a and δ_ω are sensor biases. Note that the accelerometer and gyroscope measurements are not treated as outputs in (10), but rather as inputs to the dynamics, implicitly introducing an uncertainty in $f(x_{t+1}|x_t)$ through the measurement noise.



(a). Smoothing distribution for position.



(b). Smoothing distribution for orientation.

Figure 5. The mean (black line) and 99% credibility intervals (orange fields) of $p(x_{1:T}|y_{1:T})$ for the position and orientation states, and ground truth (dashed gray) from an optical reference system.

5.2 Results

Algorithm 3 was applied to data presented by Kok et al. (2015) with (10) – (13). The results for $K = 1000$ iterations and $N = 500$ particles are summarized in Figure 5 in terms of the mean and credibility intervals (cf. Figure 13 and 14 in Kok et al. 2015). For reference, the ground truth (obtained by an optical reference system) is also shown in the plot. In terms of computational load, the presented results took about 1 day to obtain on a standard desktop computer.

Note the credibility intervals, which are the gain of using this method producing samples (as opposed to a method based on point estimates). The credibility intervals are varying over time and are different for different states, which indeed adds information to the results.

6 Conclusions

We have shown how the CPF-AS can be used to solve the nonlinear state smoothing problem in a disparate way compared to the currently available particle smoothers. The asymptotic convergence of our smoother was established, and we also illustrated the use of the smoother on two simulated examples and one challenging real-world application.

Based on the results of Theorem 1 and the numerical examples we conclude that Algorithm 3 is indeed a competitive alternative to the existing state-of-the-art smoothers. The present development opens up for interesting future work, such as hybrid versions of FFBSi and Algorithm 3, where FFBSi is used to initialize Algorithm 3. Further studies on how to tackle the trade-off between the number of particles N and the number of iterations K in Algorithm 3 for optimal performance (given a computational limit) would also be interesting.

A Appendix: Simulated nonlinear, multimodal example

This appendix gives an extended introduction to the problem considered in Section 4.2. The essential details are in 4.2, but we will provide an alternative introduction. The state space model considered in the problem is

$$x_{t+1}|x_t \sim \mathcal{N}(x_t, 0.5), \tag{14a}$$

$$y_t|x_t \sim g(y_t|x_t), \tag{14b}$$

$$x_1 = 3. \tag{14c}$$

The problem concerns smoothing for a given output sequence $y_{1:t}$. Thus we write $G(x_t) \triangleq g(y_t|x_t)$, the likelihood of x_t, omitting the dependence on y_t. $G(x_t)$ is defined by the surface of Figure 6, which is parametrized as a sum of six Gaussian-like functions. The specific parametrization is found in the provided source code.

The state space filtering/smoothing problem arising from this problem is solved using three approaches. For filtering, the particle filter in Algorithm 1, with $q_t = f$, is used. For smoothing, the FFBSi with rejection sampling, as presented in Lindsten and Schön (2013, Algorithm 5), is used, together with the MCMC smoother in Algorithm 3, and the results reported by in Section 4.2 are obtained.

B Appendix: Indoor positioning

This appendix focuses on the indoor positioning application in Algorithm 5. The problem is discussed in Kok et al. (2015), and concerns a sensor fusion problem involving accelerometer, gyroscope, and ultrawideband (UWB) measurements. There are several details on the problem not covered in the main text; the non-uniform

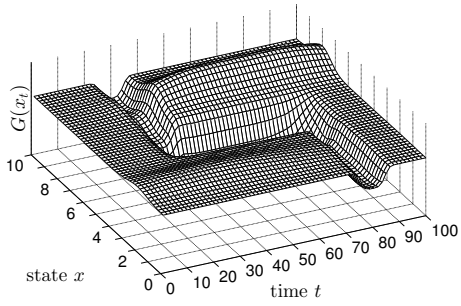


Figure 6. Definition of $G(x_t)$.

sampling intervals of the UWB measurements, the unknown transmission times, the sensor biases, some issues on the ancestor sampling, and the initialization of the first conditional trajectory.

We repeat the state space model here:

$$\left.\begin{aligned} p_{t+1}^n &= p_t^n + T_s v_t^n + \tfrac{T_s^2}{2} a_t^n, \\ v_{t+1}^n &= v_t^n + T_s a_t^n, \\ q_{t+1}^{nb} &= q_t^{nb} \odot \exp \tfrac{T_s}{2} \omega_t, \end{aligned}\right\} f(x_{t+1}|x_t, a_t^n, \omega_t) \tag{15a}$$

$$\left.\begin{aligned} y_{m,t} = \tau_t + \|r_m^n - p_t^n\|_2 + e_{m,t} \end{aligned}\right\} g(y_t|x_t). \tag{15b}$$

Most important, $x_t^\mathsf{T} = [(p_t^n)^\mathsf{T} \; (v_t^n)^\mathsf{T} \; (q_t^{nb})^\mathsf{T}]$ forms the state vector, and a_t^n are the acceleration, ω_t the angular velocity, and $y_{m,t}$ are the UWB measurements (for receiver number m). The sampling frequency for the accelerometers and the gyroscopes is 120 Hz.

The acceleration and angular velocity are modeled to be measured as

$$y_{a,t} = R_t^{bn}(a_t^n - g^n) + \delta_a + e_{a,t}, \tag{16}$$

$$y_{\omega,t} = \omega_t + \delta_\omega + e_{\omega,t}. \tag{17}$$

Through the measurement noise in these models, stochastic noise enters the state space.

B.1 Non-uniform sampling interval

For every sampling step t, the accelerometer and gyroscope data $y_{a,t}$ and $y_{\omega,t}$ are available. However, the UWB measurements are recorded at a lower sampling rate, and are available only for approximately every tenth sample. A high level algorithm illustrating how this is handled is shown in Algorithm 4.

Algorithm 4: MCMC smoother

for $k = 1$ **to** K **do**
  Sample x_1^i from $q_1(x_1)$.
  Draw $x_1^i \sim p(x_1^i), i \in [1, N-1]$.
  Set $x_{1:T}^N = x_{1:T}[k]$.
  for $t = 1$ **to** $T-1$ **do**
    if *UWB measurement* $\{y_{m,t}\}_{m=1}^M$ *available* **then**
      Set $w_t^i = g(y_t|x_t^i), i \in [1, N]$.
      Draw a_t^i with $\mathbb{P}(a_t^i = j) \propto w_{t-1}^j, i \in [1, N-1]$.
      Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}, i \in [1, N-1]$.
    end
    Draw x_{t+1}^i from $f(x_{t+1}^i|x_t^i, a_t, \omega_t), i \in [1, N-1]$.
    Draw a_{t+1}^N with $\mathbb{P}(a_{t+1}^N = j) \propto f(x_{t+1}^N|x_t^j)$.
    Set $x_{1:t+1}^N = \{x_{1:t}^{a_{t+1}^N}, x_{t+1}^N\}$.
  end
  Draw J with $\mathbb{P}(i = J) \propto \frac{1}{N}$ and set $x_{1:T}[k+1] = x_{1:T}^J$.
end

(Note that the notation for the ancestor sampling variable is b, instead of a, not to confuse it with the accelerometer data.)

B.2 Unknown transmission times

When evaluating (10d), the transmission time τ_t is unknown. It is approximately handled by Monte Carlo integration:

$$\begin{aligned} g(y_t|x_t) &= p_e(y_{m,t} - \tau_t - \|r_m^n - p_t^n\|_2 \,|\, \tau_t) \\ &\tilde{\propto} \sum_j p_e(y_{m,t} - \tau_t^j - \|r_m^n - p_t^n\|_2), \end{aligned} \tag{18}$$

where p_e is the pdf defined by (11), and τ_t^j are samples with possible values of τ. All particles (indexed with i) are evaluated with the same set of samples τ_t^j.

B.3 Sensor bias

The numerical values of the sensor biases δ_a and δ_ω are small, compared to the noise levels. They were therefore approximated ad hoc. However, a more thorough system identification approach can be applied within the CPF-AS framework, as proposed by Lindsten (2013).

B.4 Evaluation of $f(x_{t+1}|x_t, a_t^n, \omega_t)$

For the ancestor sampling in Step 11 in Algorithm 4, $f(x_{t+1}|x_t, a_t^n, \omega_t)$ needs to be evaluated. This involves the quaternion product and vector exponential in (10c), which can be manipulated as follows:

$$\begin{aligned} q_{t+1}^{nb} &= q_t^{nb} \odot \exp \tfrac{T_s}{2}(\omega_t + e_{\omega,t}) \Leftrightarrow \\ e_{\omega,t} &= \tfrac{2}{T_s} \log\left((q_t^{nb})^{-1} \odot q_{t+1}^{nb}\right) - \omega_t \end{aligned} \tag{19}$$

where log denotes the logarithm for unit quaternions (Hol 2011). An approximation of this expression was used in the computations.

B.5 Low chance of 'new ancestor'

One challenge is the small uncertainty in the position (10c), because of the (physically reasonable) factor $\frac{T_s^2}{2}$ ($\approx 10^{-5}$) in front of the noise term. To handle this, the uncertainty was artificially increased in the ancestor sampling step. This causes a substantial increase in the mixing, but also a non-feasible 'jump' in the smoothing trajectories. These artificial 'jumps' appears, however, to be rare and can also be expected to 'even out' as $K \to \infty$, but might indeed cause an overestimate of the variance.

A more thorough treatment would be to apply the recent development by Lindsten et al. (2015).

B.6 Initialization

The model has a state space of 9 dimensions (parametrized using 10 variables). The structure of the model and the uncertainties cause, according to our experience, the particle filter to diverge quickly if N is not sufficiently large. To speed up the initialization phase, a more 'loose' model was used in the first run of the CPF-AS to find one reasonable trajectory. This non-diverging trajectory was then used as the conditional trajectory for the subsequent run with the correct model.

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.

Mark Briers, Arnaud Doucet, and Simon Maskell (2010). "Smoothing algorithms for state–space models". In: *Annals of the Institute of Statistical Mathematics* 62.1, pp. 61–89.

Randal Douc, Aurélien Garivier, Eric Moulines, and Jimmy Olsson (2011). "Sequential Monte Carlo smoothing for general state space hidden Markov models". In: *The Annals of Applied Probability* 21.6, pp. 2109–2145.

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: Oxford University Press, pp. 656–704.

Jeroen D. Hol (2011). "Sensor fusion and calibration of inertial sensors, vision, ultrawideband and GPS". PhD thesis. Sweden: Linköping University.

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.

Manon Kok, Jeroen D. Hol, and Thomas B. Schön (2015). "Indoor positioning using ultrawideband and inertial measurements". In: *IEEE Transactions on Vehicular Technology* 64.4, pp. 1293–1303.

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

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, Pete Bunch, Sumeetpal S. Singh, and Thomas B. Schön (2015). "Particle ancestor sampling for near-degenerate or intractable state transition models". In: *arXiv:1505.06356*.

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.

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.

Gianluigi Pillonetti and Bradley M. Bell (2008). "Optimal smoothing of non-linear dynamic systems via Monte Carlo Markov chains". In: *Automatica* 44.7, pp. 1676–1685.

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.

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

Simo Särkkä (2008). "Unscented Rauch–Tung–Striebel smoother". In: *IEEE Transactions on Automatic Control* 53.3, pp. 845–849.

Stanley F. Schmidt (1966). "Application of state-space methods to navigation problems". In: *Advances in Control Systems: Theory and Applications*. Vol. 3. New York, NY, USA: Academic Press, pp. 293–340.

Gerald L. Smith, Stanley F. Schmidt, and Leonard A. McGee (1962). *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. Tech. rep. TR R-135. NASA.

Paper IV

Title

Marginalizing Gaussian process hyperparameters using sequential Monte Carlo

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

$$p(y^*|x^*,y,x) = \int p(y^*|x^*,y,x,\theta)p(\theta|y,x)\mathrm{d}\theta, \tag{2}$$

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.

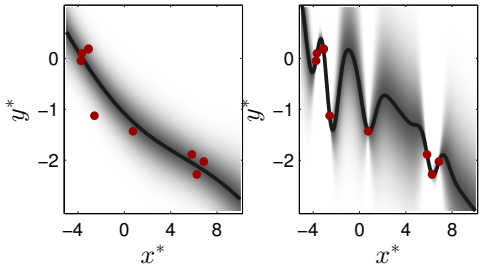
Our contribution is a method for sampling from the hyperparameter posterior distribution $p(\theta|y,x)$, based on sequential Monte Carlo (SMC) samplers. SMC samplers and their convergence properties are well studied (Whiteley 2012).

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 2010; 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.

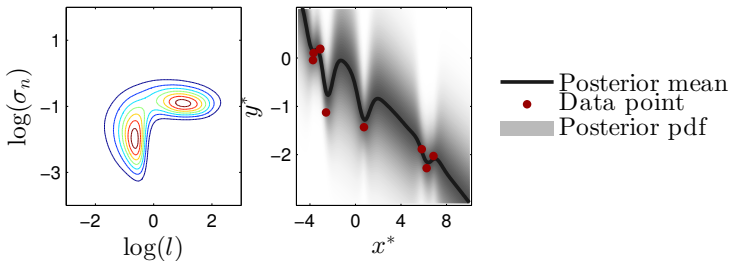
In this paper, we compare our proposed method to some of these methods, and apply it to two real-data problems: the first demonstrates that marginalization does not have to be more computationally demanding than finding point estimates. The second example, which deals with a fault detection problem from 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.

[1] http://www.it.uu.se/katalog/andsv164



(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.

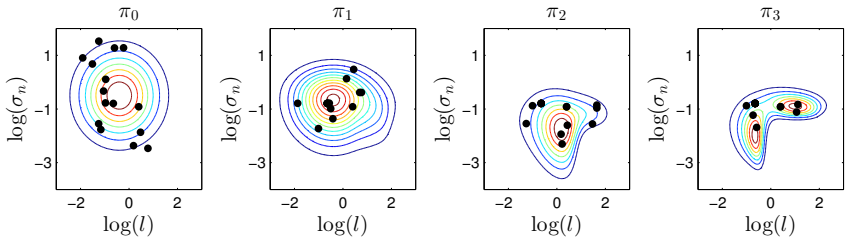
2 Sampling hyperparameters using SMC

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.

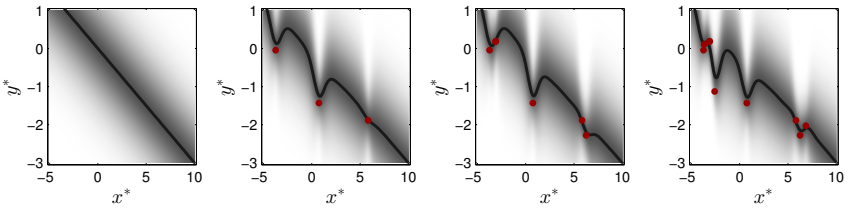
To construct a sequence $\{\pi_0, \ldots, \pi_P\}$, we use the fact that $p(\theta|y, x)$ depends on the data (y, x), by partitioning the data points into P disjoint batches $\{B_n\}_{n=1}^P$ and adding them sequentially as $\pi_n(\theta) \propto p(y_{B_{1:n}}|x_{B_{1:n}}, \theta)p(\theta)$.

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)



(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.

$$w_n^{(i)} = \frac{\pi_n(\theta_{n-1}^{(i)})}{\pi_{n-1}(\theta_{n-1}^{(i)})} w_{n-1}^{(i)}. \tag{4}$$

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: (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

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

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.

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.

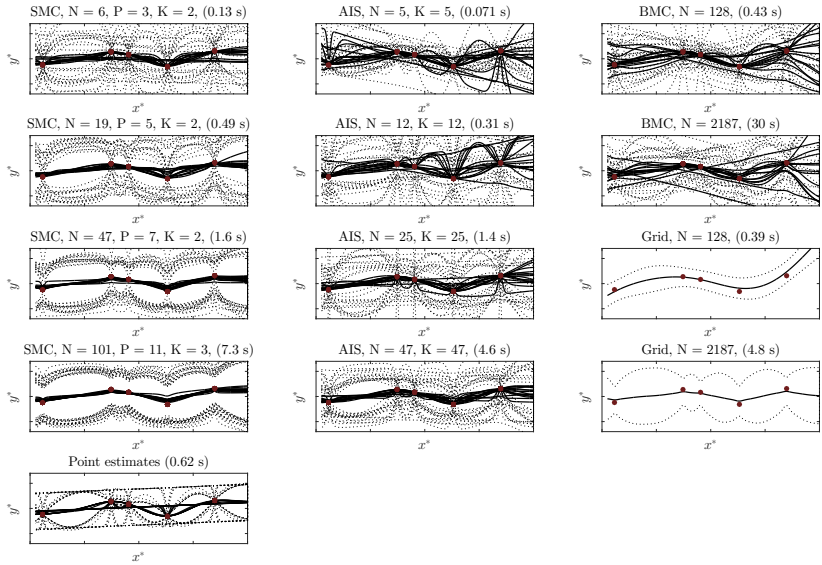


Figure 3. Comparison between 15 runs of SMC (Algorithm 1), BMC, AIS, and griding, as well optimized point estimates. The predictions (mean, solid, and 3 standard deviations, dashed) are shown, together with the red data points. The number of particles/samples/grid points is denoted by N, while K and 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.

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.

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 10.1. For Algorithm 1, $N = 15$, $P = 20$ and $K = 5$ 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 10.1 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 10.1 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

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 10.1. Results for the SARCOS example in Section 3.2.

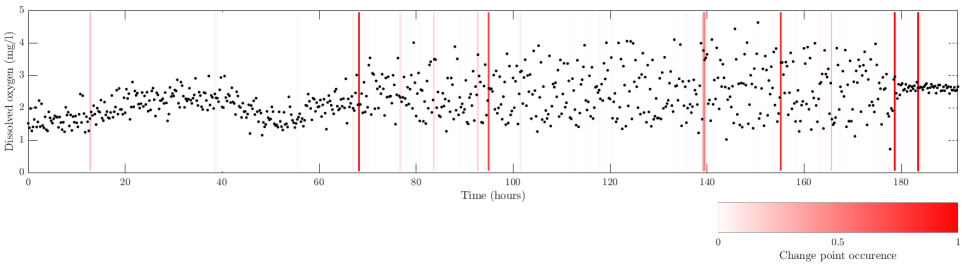
apply the GP-based online change point detection algorithm by Saatçi et al. (2010), where the hyperparameters are marginalized using our proposed method.

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 (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.

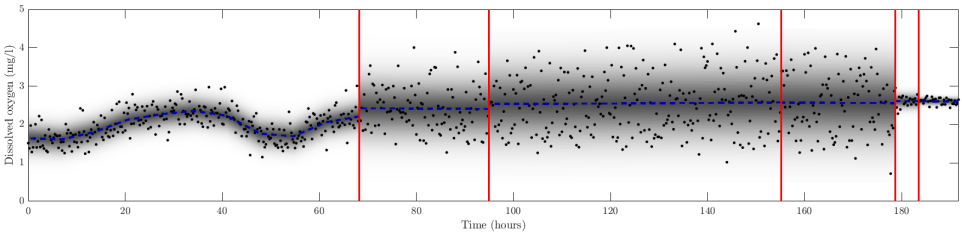
To make predictions using a GP model, the hyperparameters either have to be fixed across all data segments, or marginalized. As it is not relevant to use fixed hyperparameters, an efficient sampling algorithm is a key enabler in solving this problem. The consecutive predictions $p(y_t|r_{t-1}, y_{r_t:t-1})$ 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.



(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.

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 (2010). "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo*. Ed. by S. Brooks, A. Gelman, G. Jones, and X-L. Meng. Boca Raton, FL, USA: 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.

Paper V

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 et al. 2010; Lindsten 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*

$$\mathcal{Q}(\theta, \theta') \triangleq \int \log p_\theta(x_{1:T}, y_{1:T}) p_{\theta'}(x_{1:T} \mid y_{1:T}) \mathrm{d}x_{1:T}. \tag{3}$$

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,

$$\begin{aligned} &\text{(E)} \quad \text{Compute } \mathcal{Q}(\theta, \theta_{k-1}). \\ &\text{(M)} \quad \text{Compute } \theta_k = \arg\max_{\theta \in \Theta} \mathcal{Q}(\theta, \theta_{k-1}). \end{aligned}$$

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 $\mathcal{Q}(\theta, \theta')$ is intractable and we are forced to approximate solutions; see e.g. Cappé et al. (2005), Lindsten (2013), Olsson et al. (2008), and Schön 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 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 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 $\mathcal{Q}$ in EM with

$$\widehat{\mathcal{Q}}_k(\theta) = (1 - \gamma_k)\widehat{\mathcal{Q}}_{k-1}(\theta) + \gamma_k \log p_\theta(y_{1:T}, x_{1:T}[k]), \tag{5}$$

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 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 $p_\theta(z_t|s_{1:T}, y_{1:T}) = \mathcal{N}(z_t|\widehat{z}_{s;t}, P_{s;t})$ 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 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 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

Algorithm 1: MCMC smoother

1 Initialize $s_{1:T}[0]$ arbitrarily
2 **for** $k \geq 1$ **do**
3 | Generate $s_{1:T}[k] \sim \mathcal{K}_\theta(\cdot|s_{1:T}[k-1])$
4 **end**

the *ancestor indices* $\{a_t^i\}_{i=1}^N$. To generate s_t^i, the ancestor index is sampled according to $\mathbb{P}\left(a_t^i = j\right) \propto w_{t-1}^j$, and s_t^i as $s_t^i \sim p_\theta(s_t|s_{t-1}^{a_t^i})$. The trajectories are then augmented as $s_{1:t}^i = \{s_{1:t-1}^{a_t^i}, s_t^i\}$.

This is repeated for $i = 1, \ldots, N-1$, whereas s_t^N is set as $s_t^N = s_t'$. To 'find' the history for s_t^N, the ancestor index a_t^N is drawn with probability

$$\mathbb{P}\left(a_t^N = i\right) \propto p_\theta(s_{1:t-1}^i|s_{t:T}', y_{1:T}). \tag{8}$$

The probability density in (8) is proportional to

$$p_\theta(y_{t:T}, s_{t:T}'|s_{1:t-1}^i, y_{1:t-1}) p_\theta(s_{1:t-1}^i|y_{1:t-1}), \tag{9}$$

where the last factor is the importance weight w_{t-1}^i.

By sampling $s_{1:T}[k+1] = s_{1:T}^J$ from the rendered set of trajectories $\{s_{1:T}^i, w_T^i\}_{i=1}^N$ with $\mathbb{P}\left(J = j\right) = w_T^j$, 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 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 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 $w_{t-1}^i p(x_t'|x_{t-1}^i)$ (Lindsten et al. 2014). This does not hold in the Rao-Blackwellized case.

To handle this, (8) can be rewritten as

$$w_{t-1}^i p(y_{t:T}, s_{t:T}'|s_{1:t-1}^i, y_{1:t-1}). \tag{10}$$

Using the results from Section 4.4 in Lindsten and Schön (2013) (adapted to model (1)), this can be written (omitting w_{t-1}^i, and with the notation $\|z\|_\Omega^2 \triangleq z^T\Omega z$, $P \triangleq \Gamma\Gamma^T$, i.e. the Cholesky factorization, $Q_t \triangleq F_tF_t^T$ and $A_t \triangleq A_{s'_t}$ etc.)

$$p(y_{t:T}, s'_{t:T}|s_{1:t-1}^i, y_{1:t-1}) \propto Z_{t-1}|\Lambda_{t-1}|^{-1/2}\exp(-\frac{1}{2}\eta_{t-1}),$$

with

$$\Lambda_t = \Gamma_{f;t}^{i,T}\Omega_t\Gamma_{f;t}^i + I, \tag{11a}$$

$$\eta_t = \|\widehat{z}_{f;t}^i\|_{\Omega_t}^2 - 2\lambda_t^T\widehat{z}_{f;t}^{i,T} - \|\Gamma_{f;t}^n(\lambda_t - \Omega_t\widehat{z}_{f;t}^n)\|_{M_t^{-1}}^2, \tag{11b}$$

where

$$\Omega_t = A_{t+1}^T\left(I - \widehat{\Omega}_{t+1}F_{t+1}M_{t+1}^{-1}F_{t+1}^T\right)\widehat{\Omega}_{t+1}A_{t+1}, \tag{11c}$$

$$\widehat{\Omega}_t = \Omega_t + C_t^TR_t^{-1}C_t,\ M_t = F_t^T\widehat{\Omega}F_t + I, \tag{11d}$$

$$\lambda_t = A_{t+1}^T\left(I - \widehat{\Omega}_{t+1}F_{t+1}M_{t+1}^{-1}F_{t+1}^T\right)m_t, \tag{11e}$$

$$\widehat{\lambda}_t = \lambda_t + C_t^TR_t^{-1}(y_t - D_tu_t), \tag{11f}$$

$$m_t = (\widehat{\lambda}_{t+1} - \widehat{\Omega}_{t+1}B_{t+1}u_{t+1}). \tag{11g}$$

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

$$p_\theta(y_t|s_{1:t}^i, y_{1:t-1}) = \mathcal{N}(y_t; C_{s_t^i}\widehat{z}_{p;t}^n + D_{s_t^i}u_t, C_{s_t^i}P_{p;t}C_{s_t^i}^T + R_{s_t^i})$$

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 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.)

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 $\mathbb{P}(a^i_t = j) = w^j_{t-1}$ for $i = 1, \ldots, N-1$.

  Draw s^i_t with $\mathbb{P}(s^i_t = n) = \pi_{s^i_{t-1},n}$ for $i = 1, \ldots, N-1$.

  Compute $\{\Lambda^i_{t-1}, \eta^i_t\}$ according to (11a)-(11b).

  Draw a^N_t with $\mathbb{P}(a^N_t = i) \propto w^i_{t-1}\pi_{s^i_{t-1},s^N_t}|\Lambda^i_{t-1}|^{-1/2}\exp(-\frac{1}{2}\eta^i_{t-1})$.

  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 $\mathbb{P}(J = j) = w^j_T$.

This leads to the approximation (cf. (5))

$$\widehat{\mathcal{Q}}_k(\theta) = (1-\gamma_k)\widehat{\mathcal{Q}}_{k-1}(\theta) + \gamma_k \sum_{i=1}^N w^i_T \mathbb{E}_{\theta_k}\left[\log p_\theta(y_{1:T}, z_{1:T}, s^i_{1:T})|s^i_{1:T}, y_{1:T}\right], \quad (12)$$

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

Algorithm 3: Rao-Blackwellized PSAEM

1 Initialize $\widehat{\theta}_0$ and $s_{1:T}[0]$, and $\widehat{\mathcal{Q}}_0(\theta) \equiv 0$.
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 $\widehat{\mathcal{Q}}_k(\theta)$ according to (12).
6 Compute $\widehat{\theta}_k = \arg\max_{\theta\in\Theta} \widehat{\mathcal{Q}}_k(\theta)$.
7 **end**

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 $\widehat{\mathcal{Q}}_k(\theta)$. 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

$$\sum_{i=1}^{N} w^i_T \mathbb{E}_{\theta_k}\left[\log p_\theta(y_{1:T}, z_{1:T}, s^i_{1:T}) | s^i_{1:T}, y_{1:T}\right] = \langle S^k, \eta(\theta)\rangle, \tag{13}$$

for a sufficient statistics S and corresponding natural parameter $\eta(\theta)$. Hence $\widehat{\mathcal{Q}}_k$ can be written as

$$\widehat{\mathcal{Q}}_k(\theta) = (1-\gamma_k)\widehat{\mathcal{Q}}_{k-1}(\theta) + \gamma_k\langle S^k, \eta(\theta)\rangle = \langle \mathbb{S}^k, \eta(\theta)\rangle \tag{14}$$

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

$$M_{t|T}^i = \begin{pmatrix} P_{s;t}^i & P_{s;t,t-1}^i & 0 & 0 & P_{s;t}^i & 0 \\ P_{s;t,t-1}^i & P_{s;t-1}^i & 0 & 0 & P_{s;t,t-1}^i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ P_{s;t}^i & P_{s;t,t-1}^i & 0 & 0 & P_{s;t-1}^i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{16g}$$

For computing this, the RTS-smoother in step 17 in Algorithm 2 has to be extended by calculation of $P_{s;t+1,t} \triangleq \mathrm{cov}\left[\widehat{z}_{s,t+1}\widehat{z}_{s;t}^T\right]$, which can be done as follows (Shumway and Stoffer 2006, Property P6.2)

$$P_{s;t,t-1} = P_{f;t} J_{t-1}^T + J_t(P_{s;t+1,t} - A_{t+1}P_{f;t})J_{t-1}^T, \tag{17}$$

initialized with $P_{T,T-1|T} = (I - K_T C_T) A_T P_{f;t-1}$.

For notational convenience, we will partition $S_n^{(3)}$ as

$$S_n^{(3)} = \begin{pmatrix} \Phi_n & \Psi_n & & \\ \Psi_n^T & \Sigma_n & & \\ & & \Omega_n & \Lambda_n \\ & & \Lambda_n^T & \Xi_n \end{pmatrix}. \tag{18}$$

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 $\widehat{\mathcal{Q}}_k(\theta)$ for the jump Markov linear model* (1) *are then given by*

$$\pi_{n,m}^j = \frac{\mathbb{S}_{n,m}^{(1),k}}{\sum_l \mathbb{S}_{n,l}^{(1),k}}, \tag{19a}$$

$$\begin{bmatrix} A_n & B_n \end{bmatrix} = \Psi_n \Sigma_n^{-1}, \tag{19b}$$

$$\begin{bmatrix} C_n & D_n \end{bmatrix} = \Lambda_n \Xi_n^{-1}, \tag{19c}$$

$$\begin{bmatrix} Q_n \end{bmatrix} = (\mathbb{S}_n^{(2),k})^{-1} \left(\Phi_n - \Psi_n \Sigma_n^{-1} \Psi_n^T \right), \tag{19d}$$

$$\begin{bmatrix} R_n \end{bmatrix} = (\mathbb{S}_n^{(2),k})^{-1} \left(\Omega_n - \Lambda_n \Xi_n^{-1} \Lambda_n^T \right), \tag{19e}$$

for $n, m = 1, \ldots, K$.

$\Phi_n, \Psi_n, \ldots$ are the partitions of $\mathbb{S}_n^{(3),k}$ 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 $\left[\widehat{z}_{s;t-1}^{i,T}\right]$, and (19b) becomes $\begin{bmatrix} A_n \end{bmatrix} = \Psi_n \Sigma_n^{-1}$. 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

$$\pi_{n,m} = \frac{\mathbb{S}_{n,m}^{(1),k}}{\sum_l \mathbb{S}_{n,l}^{(1),k}}. \tag{20}$$

□

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.

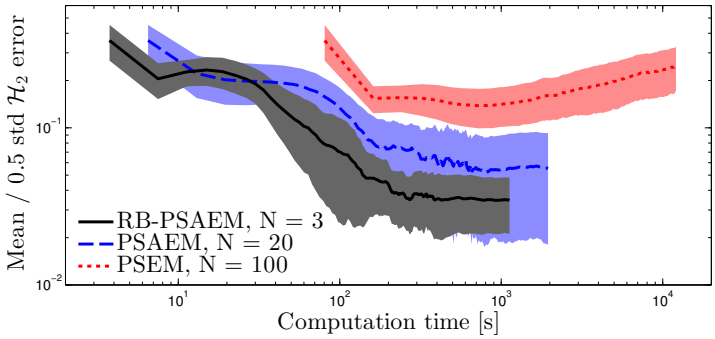


Figure 1. Numerical example 1. Mean (lines) and 0.5 standard deviation (fields) $\mathcal{H}_2$ error for 7 runs of our RB-PSAEM using $N = 3$ particles (black) PSAEM (Lindsten 2013) using $N = 20$ particles (blue) and PSEM (Schön et al. 2011) using $N = 100$ particles and $M = 20$ backward trajectories (red).

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

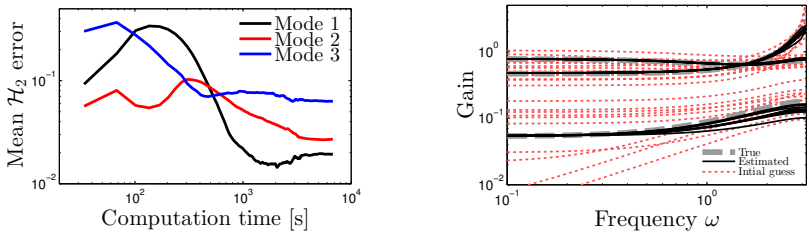
The first example concerns identification using simulated data ($T = 3\,000$) for a one-dimensional ($n_z = 1$) jump Markov linear model with 2 modes ($K = 2$) (with parameters randomly generated according to $A_n \sim U_{[-1,1]}$, $B_n \sim U_{[-5,5]}$, $C_n \sim U_{[-5,5]}$, $D_n \equiv 0, Q_n \sim U_{[0.01,0.1]}, R_n \sim U_{[0.01,0.1]}$) 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 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

[1] http://www.it.uu.se/katalog/andsv164



(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.

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.

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 investiga-

tion. 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.

Recent licentiate theses from the Department of Information Technology

2016-010 Aleksandar Zeljić: *Approximations and Abstractions for Reasoning about Machine Arithmetic*

2016-009 Timofey Mukha: *Inflow Generation for Scale-Resolving Simulations of Turbulent Boundary Layers*

2016-008 Simon Sticko: *Towards Higher Order Immersed Finite Elements for the Wave Equation*

2016-007 Volkan Cambazoglou: *Protocol, Mobility and Adversary Models for the Verification of Security*

2016-006 Anton Axelsson: *Context: The Abstract Term for the Concrete*

2016-005 Ida Bodin: *Cognitive Work Analysis in Practice: Adaptation to Project Scope and Industrial Context*

2016-004 Kasun Hewage: *Towards a Secure Synchronous Communication Architecture for Low-power Wireless Networks*

2016-003 Sven-Erik Ekström: *A Vertex-Centered Discontinuous Galerkin Method for Flow Problems*

2016-002 Rubén Cubo: *Mathematical Modeling for Optimization of Deep Brain Stimulation*

2016-001 Victor Shcherbakov: *Radial Basis Function Methods for Pricing Multi-Asset Options*

2015-006 Hanna Holmgren: *Towards Accurate Modeling of Moving Contact Lines*

2015-005 Siyang Wang: *Analysis of Boundary and Interface Closures for Finite Difference Methods for the Wave Equation*

UPPSALA
UNIVERSITET

Department of Information Technology, Uppsala University, Sweden